Development of a practice and assessment tool for learning data science concepts in R

Garrett Dancik, PhD
Eastern Connecticut State University
http://gdancik.github.io/bioinformatics
March 11, 2018

swirl courses

https://github.com/swirldev/swirl_courses

Here are our current offerings, organized by level of difficulty:

Beginner

- **R Programming**: The basics of programming in R
- **R Programming Alt**: Same as the original, but modified slightly for in-class use (see below ***)
- **Data Analysis**: Basic ideas in statistics and data visualization
- **Mathematical Biostatistics Boot Camp**: One- and two-sample t-tests, power, and sample size
- **Open Intro**: A very basic introduction to statistics, data analysis, and data visualization

*** *R Programming Alt is identical to R Programming, except we've eliminated the prompts for Coursera credentials at the end of each lesson and instead give students the option to send an email to their instructor notifying them of completion. Admittedly, it's sort of a hack until we come up with a more robust solution for in-class use (i.e. an instructor "dashboard").*

Intermediate

- **Regression Models**: The basics of regression modeling in R
- **Getting and Cleaning Data**: dplyr, tidyr, lubridate, oh my!

Advanced

- **Statistical Inference**: This intermediate to advanced level course closely follows the Statistical Inference course of the Johns Hopkins Data Science Specialization on Coursera. It introduces the student to basic concepts of statistical inference including probability, hypothesis testing, confidence intervals and p-values. It concludes with an initiation to topics of particular relevance to big data, issues of multiple testing and resampling.

Current swirl package:

| To assign the result of **5** + **7** to a new variable called **x**, you type **x** <- **5** + **7**.
| This can be read as '**x** gets **5** plus **7**'. Give it a try now.

swirl-tbp (https://github.com/gdancik/swirl-tbp/bioinformatics/swirl-tbp)

Current swirl package:

| To assign the result of **5** + **7** to a new variable called **x**, you type **x** <- **5** + **7**.
| This can be read as '**x** gets **5** plus **7**'. Give it a try now.

- A swirl extension allowing for *template-based practice problems*.

swirl-tbp: question templates are specified, using 'tokens' whose values are determined at run-time

Output: To assign the result of **<x>** + **<y>** to a new variable called **<var>**, you type **<var>** <- **<x>** + **<y>**. This can be read as '**<var>** gets **<x>** plus **<y>**'. Give it a try now.

swirl-tbp (https://github.com/gdancik/swirl-tbp/bioinformatics/swirl-tbp)

- Non-template version for learning

```
- Class: cmd_question
  Output: To assign the result of 5 + 7 to a new variable called x, you type x <-
    5 + 7. This can be read as 'x gets 5 plus 7'. Give it a try now.
  CorrectAnswer: x <- 5 + 7
  AnswerTests: any_of_exprs('x <- 5 + 7', 'x <- 7 + 5')
```

- Template version for practice

```
- Class: cmd_question
  Token: X = sample(1:10,1); Y =  sample(1:10,1); var=sample(c('sum', 'res', 'ans'),1)
  Output: Add the numbers <X> and <Y> together and store the result in a variable called <var>.
  CorrectAnswer: <var> <- <X>+<Y>
  AnswerTests: any_of_exprs('<var> <- <X> + <Y>','<var> <- <Y> + <X>)
```

- Token values generated using R code
- Tokens (e.g., <X>, <Y>, and <var>) will be replaced by their corresponding values when the lesson is run
- Once a token is defined, it can be specified in any subsequent section of the YAML lesson

swirl-tbp example output

```
| Add the numbers 3 and 4 together and store the result in a variable
| called ans
```

```
| Add the numbers 1 and 9 together and store the result in a variable
| called sum
```

With templates, students are provided with effectively an endless supply of problems

swirl-*tbp* example applications

- Tokens will be used to dynamically generate all values in **red**.
  - **Add** the numbers **1 – 9** together and store the result in a variable called **ans**
  - Create a vector named **x** that stores the numbers **1**-**10**.
  - Find the **mean** of the **GPA** column from a data frame (table) called '**survey**'
  - Suppose that X is normally distributed with mean **25** and standard deviation of **3.1**. Find the probability that X is **greater** than **22**.

Hints in swirl

```
- Class: cmd_question
  Output: The easiest way to create a vector is with the c() function, which stands
    for 'concatenate' or 'combine'. To create a vector containing the numbers 1.1,
    9, and 3.14, type c(1.1, 9, 3.14). Try it now and store the result in a variable
    called z.
  CorrectAnswer: z <- c(1.1, 9, 3.14)
  AnswerTests: omnitest(correctExpr='z <- c(1.1, 9, 3.14)')
  Hint: Inputting z <- c(1.1, 9, 3.14) will assign the vector (1.1, 9, 3.14) to a
    new variable called z. Including single spaces after the commas in the vector
    is not required, but helps make your code less cluttered and more readable.
```

Customized hints using a Hint Function

```
- Class: cmd_question
  NumTimes: 2
  Token: |
    num1 = sample(1:10,1)
    num2 = sample(11:20,1)
  Output: Create a vector named 'values' that holds the values <num1> and <num2>.
  CorrectAnswer: values <- c(<num1>,<num2>)
  AnswerTests: omnitest(correctExpr='values <- c(<num1>,<num2>)')
  HintFunction: createVectorHint
```

Customized hints using a Hint Function

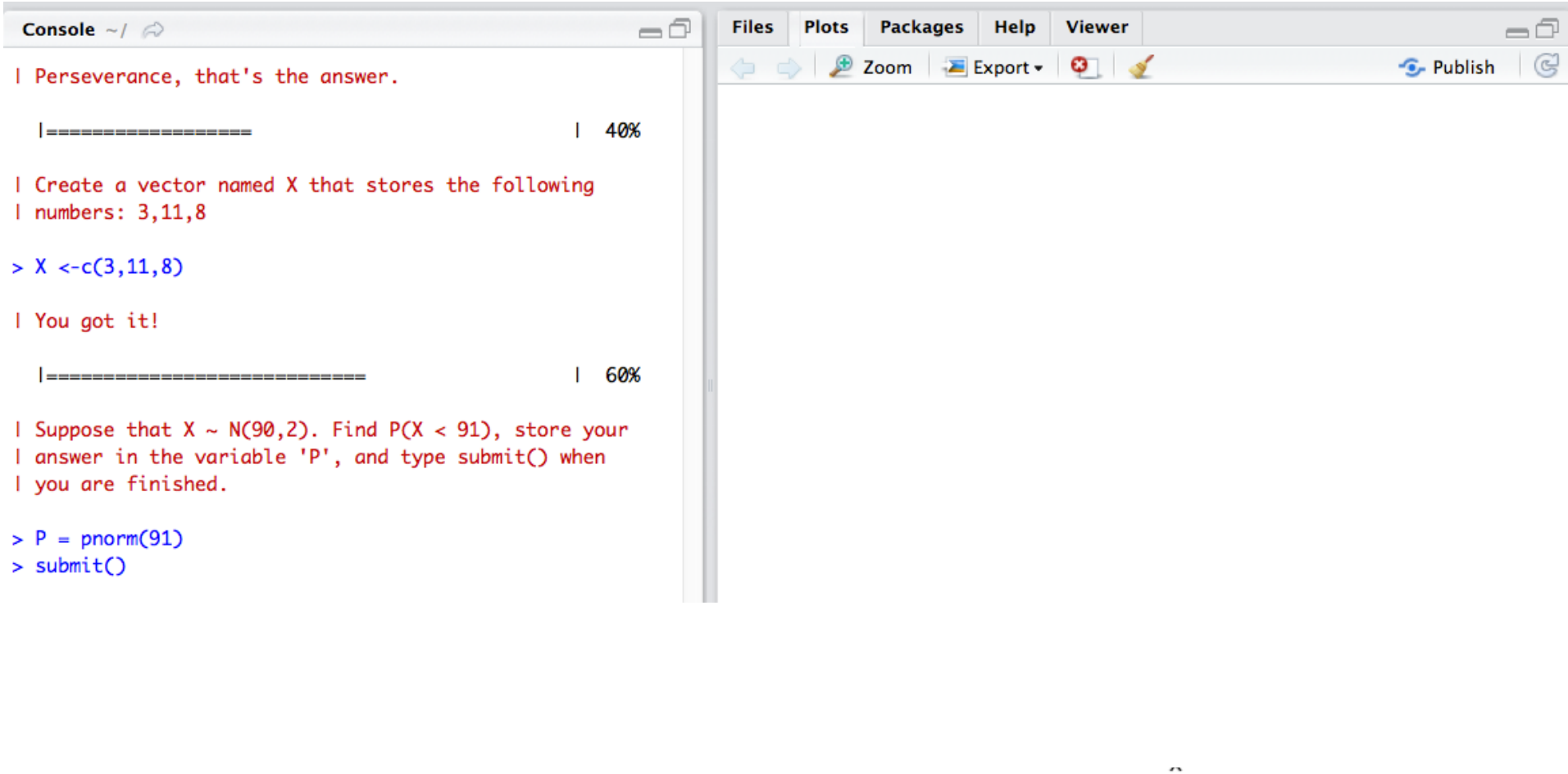
```
createVectorHint <- function(){
  e <- get("e", parent.frame())
  ans = c(e$token.list$num1, e$token.list$num2)

  # if user response is not numeric, return NA and give default hint
  if (!is.numeric(e$val)) return(NA)

  # Check that user enters correct number of values
  if (length(ans) != length(e$val)) {
        return ("The vector should contain 2 numbers only.")
  }

  # if the user and correct answer vectors are identical, only
  # error would be that the vector name is not correct
  if (all(e$val== ans)) {
        return ("Make sure to use the correct vector name.")
  }
  return (NA)
}
```

Customized and graphical hints

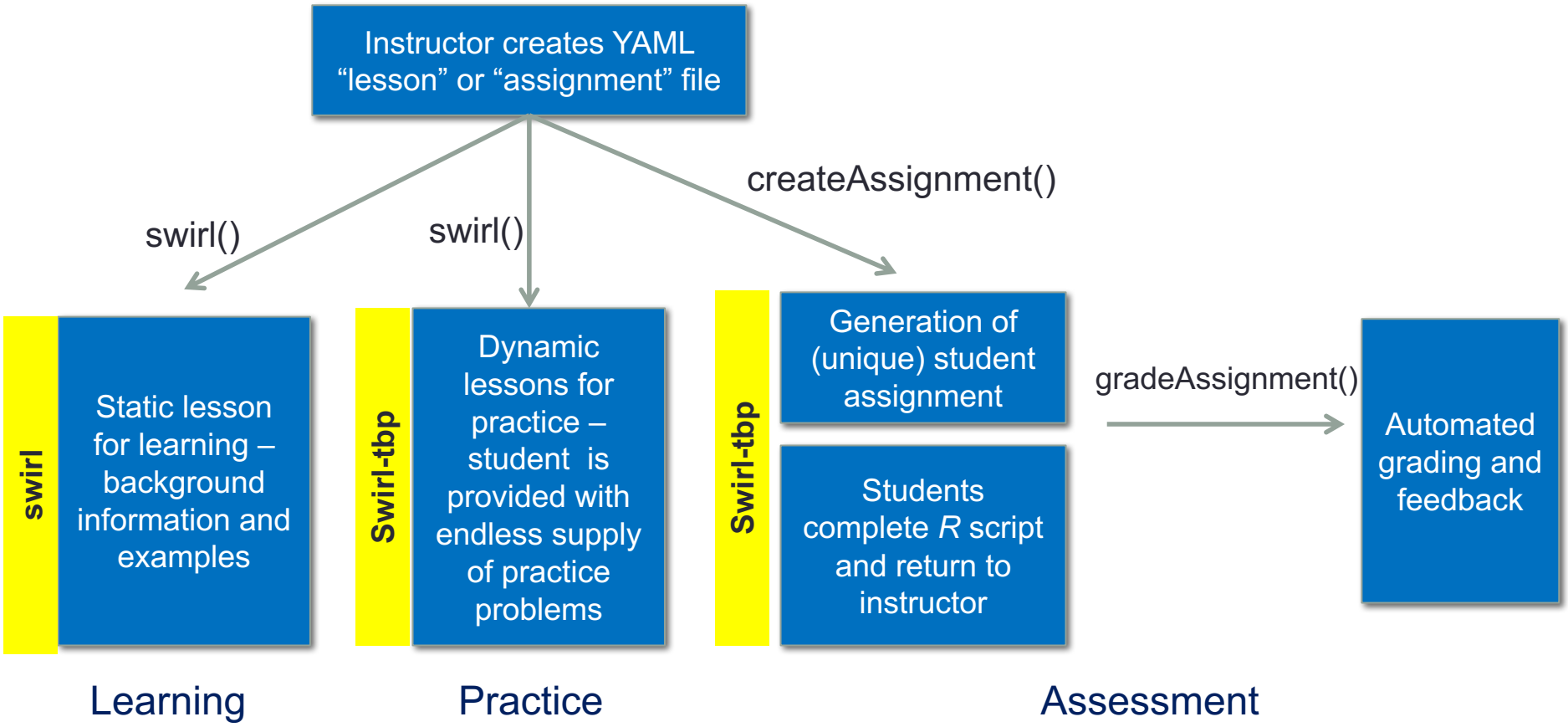
| Suppose that X ~ N(**mu**,**sigma**). Find P(X < **val**), store your answer in the
| variable 'P', and type submit() when you are finished.

```
- Class: multi_cmd_question
  Token: mu = sample(20:100,1); sigma = sample(2:5,1); val = round(mu+sigma*runif(1,.2,3))
  Output: Suppose that X ~ N(<mu>,<sigma>). Find P(X < <val>), store your answer in the variable 'P',
 and type submit() when you are finished.
  CorrectAnswer: P <- pnorm(<val>,<mu>,<sigma>)
  AnswerTests: var_has_value('P', pnorm(<val>,<mu>,<sigma>), 1e-10)
  HintFunction: calcNormProbHintLT
```

Customized and graphical hints



Learning, practice, and assessment



Grade Report for dancik_assignment1.R

Number correct = 6/9

```
# The questions that follow cover the normal distribution.
```

Question 1: ✓

```
# <Q1> Suppose that X ~ N(65,3). Find P(X < 68), store your answer in the
# variable 'P'.

P <- pnorm(68, 65,3)
```

Question 2: ✘

```
# <Q2> Suppose again that X ~ N(65,3). Find P(X > 68), store your answer in the
# variable 'P'.

P <- pnorm(68, 65,3)
```

Question 3: ✓

```
# <Q3> What is the probability that an observation from the standard normal
# distribution is greater than -1.83? Store your answer in the variable 'P'.

P <- 1-pnorm(-1.83)
```

Summary

- *swirl-tbp* extends the *swirl* package to provide *template-based* practice problems for learning *R* and data science
- Templates provide students with effectively an endless supply of practice problems
- Tokens are specified using R code, so that all features of *R* may be used in generating questions
- Instructors can use Hint Functions to provide customized and graphical hints based on a student's response
- Auto-gradable assignments can be created

- Source code and examples are available on Github: https://github.com/gdancik/swirl-tbp/

THANK YOU!

Acknowledgements:

Keith Cafiero, CS major, Eastern Connecticut State University

Contact:

Garrett Dancik, PhD

Eastern Connecticut State University

E-mail: dancikg@easternct.edu

http://gdancik.github.io/bioinformatics

CSC 315: Bioinformatics programming and analysis

1. Programming in R
2. Statistical analysis using R and related theory
3. Analysis of gene expression data

Presentation Tips

- Presentation is written out and practiced ahead of time
- You do NOT read off of the page
- Additional slides are included at the end
  - For results or background not presented do to time
  - To answer possible questions

Customized hints using a Hint Function

```
##############################################################################
# Functions to provide graphical hints for questions of the form
# find P(X < val) or P(X>val) when X is normally distributed
##############################################################################
calcNormProbHintLT <-function(question) calcNormProbHint(question,FALSE)
calcNormProbHintGT <-function(question) calcNormProbHint(question,TRUE)

calcNormProbHint <- function(question,gt) {
  e <- get("e", parent.frame())
  hint = "Use the 'pnorm' function to find the desired probability, as indicated by the graph, a
  hint = paste0(hint, "\n", question)
  if (!gt) {
     shade.norm(-Inf,e$token.list$val, mean = e$token.list$mu, sd = e$token.list$sigma)
  } else {
     shade.norm(e$token.list$val, Inf, mean = e$token.list$mu, sd = e$token.list$sigma)
  }
  return (hint)
}
```

Presentation Tips

- You are presenting your paper:
  - background, significance, objective, methods, results
- Almost every slide is a picture (or table)
  - From the internet (with reference)
  - From another publication (with reference)
  - From original research