Romi Burks - Swirl Template

Adapted from: Greg Wilson (ed.): *Teaching Tech Together*. Lulu.com, 2018, 978-0-9881137-0-1, http://teachtogether.tech/.

Step 1: Brainstorming

Provide answers to the questions below that are relevant to your course.

1. **What problem(s) will students learn how to solve?**

The students will learn how to use the data manipulation functions within dplyr to alter the structure of their data set so that they can run non-parametric tests when they fail to find a normal distribution. Depending on how much I can edit or create, I would like to get them to create two new variable columns (nptreat, data1). The challenge will be in combining names of columns.

2. **What concepts and techniques will students learn?**

Statistically, they will learn/reinforce their understanding of parametric versus nonparametric statistics and the aspects of experimental design with an emphasis on factors and comparisons between treatments.

In terms of swirl/R, they will learn that it can be important to set up your data in multiple ways initially if possible.

My goal will be to create a swirl tutorial with a similarly-structured dataset.

3. **What technologies, packages, or functions will students use?**

Students usually conduct their work in R Studio.
They will use dplyr as a package and I think the Dunn's test takes carr as well
Functions: naming an object, read.csv, data manipulation function (select, filter, arrange, mutate, summarize), Shapiro.test. KW-test, Dunn's test

4. **What terms or jargon will you define?**

Parametric, non-parametric, power, normality, null hypothesis, reject/fail to reject, statistical significance

5. What analogies will you use to explain concepts?

I use the concept of the statistical highway - parametric statistics are “free” and non-parametric statistics require an automatic “toll” (loss of degrees of freedom). Thus, you arrive at your destination with less $$$ (or “power). Either way, every time you stop to compare distances, you also lose time (additionally akin to loss of degrees of freedom).

6. What heuristics will help students understand things?

They will do my practice swirl module.
I may have them fill out a worksheet regarding the structure of the dataset I create and then provide a space for them to “input” their equivalent.
I will have them write out in words what they need to do in order to structure the data in association with a one factor column with multiple treatments (i.e., reducing any hierarchical structure in the experimental design).

7. What approach will you use to help students implement the lesson?

Repetition.
Low stakes assignment
Partner assistance - I’ve paired “R_experienced” students with “R_naive” students to help decrease the anxiety and increase the problem solving ability.

8. How will you scaffold the lesson in to your course?

I asked the students to complete Lessons 1 - 5 of “R_Programming” prior to my lab on 2/19. During lab on 2/19, we will review these 5 lessons and then give them the rest of the time to work on Lessons 6-11. We will compile a Google Doc of R_notes that all of the students can use *ad libetum*.

I will then likely have them complete the “Manipulating_Data_with_dplyr” lesson within the “Getting_and_Cleaning_Data” swirl course - https://github.com/swirldev/swirl_courses/tree/master/Getting_and_Cleaning_Data

9. What mistakes or misconceptions do you expect?

Conceptual misconceptions - confusing factors, levels and treatments
Conceptual misconception - incorrectly interpreting a significant p-value from an assumption test
Operational misconception - keeping in mind order of operations
Silly mistakes - typos

10. What datasets will you use?

I created a dataset for this purpose.

11. How much time will be devoted to this lesson?

Total “R” time = 2, three hour laboratory periods; one of those dedicated to data re-arranging and anticipating analysis from their experiment

Step 2: Who Is This Course For?

Provide a summary describing who your learners are. In your summary, address the following: what class you are implementing this lesson in, the level of the students (e.g. upper division biology students, students meeting a general education requirement, etc.), whether students have any prior programming and/or statistics experience. Following your summary, address the following questions:

1. How will this lesson help these students?
2. What specific prerequisite skills or knowledge do you expect students to have?

This swirl lesson that can be implemented in my Methods in Ecology and Evolution class (the pre-req to Ecology; usually sophomores and juniors) or my upper-level courses with labs where we do experiments.

All students took Methods in Ecology and Evolution so that they have been exposed to experimental design, hypothesis testing and data interpretation. Half of the students took this class with me in which I used R; the other half took a section from my colleague who used SPSS.

Students in my courses often do relatively simple factorial design experiments that lend themselves to analysis with ANOVA (1 x 4, 2 x 2, 2 x 3). However, in many cases, an abundance of low values or zeros will skew the data to the point of violating assumptions of normality and forcing a more appropriate non-parametric test. It’s not obvious that the datasheet needs to be in a different structure for R to perform such a test.

They will already have:

- Created a data sheet in Excel/Google Sheets
- Saved it as a CSV file
- Inputted it into R
- Examined it with functions like head, tail, str, dim, summary
- Tested their response/dependent variable for normality with a Shapiro.test
- Interpreted that p-value
- Attempted to transform
- Re-run the Shapiro
- Gotten to the point that they accept that they need nonparametric statistics

Step 3: What Will Learners Do Along the Way?

The best way to make the goals in Step 1 firmer is to write full descriptions of a couple of specific activities that students will do during the lesson. These activities may include an introduction to the swirl lesson, an in-class activity to demonstrate concepts that will be addressed by the swirl lesson, specific analyses to be performed within the swirl lesson, an assessment following the swirl lesson, etc. Use the table below to provide a brief description of each activity and explain which component of the brainstorming step each activity addresses (the "Goal" column, or what the activity will accomplish). Add additional rows to the table as necessary

Activity	**My Goals (not written exactly as student learning outcomes)**
Get everyone to have R access on their computers	Prior to R Week 1: Troubleshooting installation and access
Review swirl lessons 1 - 5 in R_Programming	R Week 1 (2/19): Assess comfort level with R for naive users and opinion of the interface with experienced users
Demonstrate note-taking using group Google Doc for lesson 6-8 in R_Programming	R Week 1 (2/19): Show how to use swirl as more than a "click through" program
Lecture/Recap Methods 101	R Week 1 (2/19): Get everyone on the same page regarding experimental design and statistical approaches
Self-paced learning for R_Programming, lessons 9 -15	R Week 1 (2/19): Be available for helping students through more R content.
Swirl "homework" - Lesson "Manipulating_Data_with_dplyr" lesson within the "Getting_and_Cleaning_Data"	Due before R Week 2 (current set for 3/11 which is Wednesday before Spring Break- maybe better to move to 3/27): Award completion points and get students to reflect
Apply/conduct new swirl lesson	Will create some worksheet/questions that maybe can be done in R Markdown? Following exercise: Assess students understanding of order of operations and the context.

Step 4: How Are Concepts Connected?

In this stage, put the activities in a logical order then derive a point-form lesson outline from them. You can accomplish this by re-organizing the activities outlined above into a numbered list (below) that corresponds with the planned sequence of activities.

Lesson sequence: - see above

1. _
2. _
3. _
4. _
5. _

Step 5: Lesson Overview

You can now write a lesson overview consisting of:

- a one-paragraph description
- learning objectives (taken from steps 1 and 3) - really later steps

Include your summary and learning objectives below.

This swirl lesson will show you how to rearrange your data within R from one that shows a multiple factor design to a single factor to allow for data analyses that require nonparametric statistics. I anticipate that this skill will be useful when it comes to analyzing data from the wildflower experiment. In addition, a lot of ecological data often violates assumptions of normality, cannot be successfully transformed and thus requires nonparametric approaches. The swirl lesson I create will build off of the existing swirl lesson “Manipulating_Data_with_dplyr” within the “Getting_and_Cleaning_Data” swirl course. Primarily, this swirl lesson will focus on the commands select(), filter() and summarize(). By completing a whole swirl course first, all students will learn about the basic functions in R and how to work within R_Studio. R-experienced students will help guide the learning of R-naive students. At the end of the two data manipulation lessons (existing and modified), you will have gained the skills necessary to arrange your future data in the right form for easy analysis in R. In addition, you will reinforce your understanding of the process associated with proper statistical analysis.

Reminder This process is described as a sequence, but in practice you will loop back repeatedly as each stage informs you of something you overlooked.

Here is the Yaml File Text:

```
- Class: meta
  Course: Burks Course
  Lesson: Setting_Up_Nonparametric_Test
  Author: your name goes here
  Type: Standard
  Organization: your organization's name goes here
  Version: 2.4.5

- Class: text
  Output: Welcome to first swirl lesson by Dr. Burks!

- Class: text
```

```
  Output: Ecologists often collect data that does not meet a normal distribution even after
transformation

- Class: text
  Output: The goal of this lesson will be to teach you how to test for normality, try a
transformation, re-arrange the data and then conduct a non-parametric test and post-hoc.

- Class: text
  Output: It's a tall order but I know you can do it!

- Class: text
  Output: Today we will analyze some data from a experiment that looked at plant consumption
by snails.

- Class: cmd_question
  Output: We'll start by getting the data. Swirl already loaded it (as well as several packages) so
take a look at the data "snailexp" using str()
  CorrectAnswer: str(snailexp)
  AnswerTests: omnitest(correctExpr='str(snailexp)')
  Hint: This is a command for you to look at the structure of the data

- Class: mult_question
  Output: Based on the str output, how many factors occur in this experiment?
  AnswerChoices: 1;2;3;4;
  CorrectAnswer: 2
  AnswerTests: omnitest(correctVal='2')
  Hint: Remember a factor represents an independent variable.

- Class: mult_question
  Output: Just to double check your understanding, a couple more questions. How many
treatments in this design?
  AnswerChoices: 2;3;4;20
  CorrectAnswer: 4
  AnswerTests: omnitest(correctVal='4')
  Hint: Remember treatments represent the combination of factors and levels.

- Class: text
  Output: Great! So, now you know that you have a 2 x 2 design with 2 factors each with 2 levels
and we measured the mass eaten by the snails. We set up this experiment intending to use
parametric statistics.

- Class: mult_question
  Output: What appropriate test could we use if the data meets a normal distribution?
  AnswerChoices: t-test;1-way ANOVA;Krustal-Wallis; 2-way ANOVA
  CorrectAnswer: 2-way ANOVA
  AnswerTests: omnitest(correctVal='2-way ANOVA')
  Hint: Think about the number of factors

- Class: cmd_question
  Output: Let's check if our dependent variable (masseaten) meets a normal distribution.
Remember that the name of the test we should use rhymes with DeNiro. If you need to further
jog your memory, hit play() and go find out! And then nxt() to return to the lesson. Remember to
identify the location of the object using the $ symbol!
  CorrectAnswer: shapiro.test(snailexp$masseaten)
  AnswerTests: omnitest(correctExpr='shapiro.test(snailexp$masseaten)')
  Hint: You've done this before in Methods - if you get an error, remind R the location of the data
with a $

- Class: text
  Output: Great job.  Now to interpret!

- Class: mult_question
  Output: Does the data meet the assumptions of a normal distribution?
  AnswerChoices: NO; YES;
  CorrectAnswer: NO
  AnswerTests: omnitest(correctVal='NO')
  Hint: Remember that you fail to reject the null hypothesis when p > 0.05

- Class: mult_question
  Output: Okay, not normally distributed, so now what?
  AnswerChoices: Try to transform;Give up;Do the ANOVA anyway;Move to nonparametric tests
  CorrectAnswer: Try to transform
  AnswerTests: omnitest(correctVal='Try to transform')
  Hint: Can't give up or move on just yet...something to do first.

- Class: cmd_question
  Output: Yes! Transform the data. Take a shot at doing that with a log transformation and
naming that new dependent variable "logmasseaten" and appending it to your data that we
called snailexp! Also remember to always add 1 when you try and log transform to account for
any zero values (cannot take the log of zero!)
  CorrectAnswer: snailexp$logmasseaten <- log10(snailexp$masseaten + 1)
  AnswerTests: omnitest(correctExpr='snailexp$logmasseaten <- log10(snailexp$masseaten +
1)')
  Hint: Remember to tell R where the data occurs or goes. Remember to add the "1" in case of
zeros in the data set. Remember the command is not ln10.

- Class: cmd_question
  Output: Recheck the structure of your data set to see that your transformation worked
  CorrectAnswer: str(snailexp)
  AnswerTests: omnitest(correctExpr='str(snailexp)')
  Hint: Should look different than before!

- Class: cmd_question
  Output: Now re-check logmasseaten for normality. Remember to tell R Studio where the data occurs using the $ symbol
  CorrectAnswer: shapiro.test(snailexp$logmasseaten)
  AnswerTests: omnitest(correctExpr='shapiro.test(snailexp$logmasseaten)')
  Hint: Remember you can pull up past code by using the up arrow.

- Class: mult_question
  Output: Does the transformed data meet the assumptions of a normal distribution?
  AnswerChoices: NO; YES;
  CorrectAnswer: NO
  AnswerTests: omnitest(correctVal='NO')
  Hint: You reject the null hypothesis that the data meets a normal distribution when p < 0.05

- Class: text
  Output: Okay, so bummer. When data still fail to meet the assumptions after a log transformation, there are just a couple courses of action. You could try another transformation (square, inverse, power, arcsine) depending on the structure of the data...OR you could accept that you'll lose some power and proceed with a non-parametric approach. Let's go with Option 2.

- Class: mult_question
  Output: What do you need to do to make that happen?
  AnswerChoices: Make a column for one factor instead of two; Add a dependent variable; Run an ANOVA; Start the experiment over;
  CorrectAnswer: Make a column for one factor instead of two
  AnswerTests: omnitest(correctVal='Make a column for one factor instead of two')
  Hint: Nonparametric statistics do not handle factorial designs well.

- Class: text
  Output: So, you could just go back to your .csv file and create a new column called "treat" and then put in the combinations of factors and levels. That's easy for small data sets. However, you can also tell R to create a new variable for you and assign it the values you want based on replication.

- Class: text
  Output: Now the tricky part - creating a column and assigning those treatments to the rows.

- Class: text
  Output: This is a complex series of steps so pay attention!!
```

```
- Class: text
  Output: Swirl & R Studio use four packages to do this work including tidyr (to add the column), FSA and Dunn.test (to run a post-hoc for a significant nonparametric test), and ggplot2 to make a graph. Swirl already loaded these packages for you so you can work to append a new column to your dataset. Let us call that dataset snailexp1 versus the original snailexp and the new column we will call treat

- Class: mult_question
  Output: What command from tidyr do you think creates a new column for you?
  AnswerChoices: unite;divide;filter;select;
  CorrectAnswer: unite
  AnswerTests: omnitest(correctVal='unite')
  Hint: Consider that you want to put two pieces of information (our snail factor and our nativeplant factor) together

- Class: text
  Output: Okay, so you want a new dataset called snailexp1 to contain a new column called treat and to do that with the command unite which looks generically like this...newdata <- unite(originaldata, treat, c(columnA, columnB), remove = FALSE)]

- Class: cmd_question
  Output: So try and make that happen now! Substitute your column names for columnA and columnB in the generic formula newdata <- unite(originaldata, treat, c(columnA, columnB), remove = FALSE)
  CorrectAnswer: snailexp1 <- unite(snailexp, treat, c(snail, nativeplant), remove = FALSE)
  AnswerTests: omnitest(correctExpr='snailexp1 <- unite(snailexp, treat, c(snail, nativeplant), remove = FALSE)')
  Hint: Every space and name matters and take out the #

- Class: cmd_question
  Output: Now check the structure of snailexp1 with a command
  CorrectAnswer: str(snailexp1)
  AnswerTests: omnitest(correctExpr='str(snailexp1)')
  Hint: You should see 5 variables now

- Class: mult_question
  Output: Which one is not a treatment that you see under your new column treat?
  AnswerChoices: native_submerged;native_floating;nonnative_submerged;nonnative_floating;emergent
  CorrectAnswer: emergent
  AnswerTests: omnitest(correctVal='emergent')
  Hint: The correct treatments have a combination of two factors
```

```
- Class: mult_question
  Output: Okay! Exciting! Approaching the finish line, now what do you do?
  AnswerChoices: Perform a Kruskal-Wallis non-parametric test; Check the data for normality again; Run a 2-way ANOVA; Make a graph;
  CorrectAnswer: Perform a Kruskal-Wallis non-parametric test
  AnswerTests: omnitest(correctVal='Perform a Kruskal-Wallis non-parametric test')
  Hint: Remember you already decided that the data did not fit a normal distribution  so you probably want to run the analysis between treatments first

- Class: text
  Output: Now it is time for a Kruskal-Wallis test for your dependent variable of masseaten against your treatment within the dataset snailexp1 - you can always use ? for help and find the formula that is this...kruskal.test(dependentvariable ~ independentvariable, data = datasetname)

- Class: cmd_question
  Output: Write the code for the KW test.
  CorrectAnswer: kruskal.test(masseaten ~ treat, data = snailexp1)
  AnswerTests: omnitest(correctExpr='kruskal.test(masseaten ~ treat, data = snailexp1)')
  Hint: Remember that ~ means "by"

- Class: mult_question
  Output: Did you find a significant difference overall between the treatments?
  AnswerChoices: YES;NO;I DO NOT KNOW
  CorrectAnswer: YES
  AnswerTests: omnitest(correctVal='YES')
  Hint: Remember you reject the null hypothesis of equal ranks in treatments when p < 0.05

- Class: text
  Output: Significant results! That is exciting  Now you need to figure out exactly which treatments differ from each other! Whereas parametric tests such as ANOVA often use Tukeys as a posthoc, non-parametric statistics use a Dunns test

- Class: cmd_question
  Output: Write your Dunns test here - almost identical to a KW but just with dunnTest as the function
  CorrectAnswer: dunnTest(masseaten ~ treat, data = snailexp1)
  AnswerTests: omnitest(correctExpr='dunnTest(masseaten ~ treat, data = snailexp1)')
  Hint: Only change one thing. No period between dunn and Test

- Class: mult_question
  Output: Okay now we can intrepret the comparisons - how many significant differences exist?
  AnswerChoices: 1;2;3;4;5;6;
  CorrectAnswer: 3
  AnswerTests: omnitest(correctVal='3')
  Hint: Check for adjusted p-values less than 0.05

- Class: mult_question
  Output: Which of the following pairs DID NOT significantly differ from each other?
  AnswerChoices: native_submerged and nonnative_submerged;native_floating and
nonnative_floating;native_submerged and nonnative_floating;nonnative_floating and
nonnative_submerged
  CorrectAnswer: native_submerged and nonnative_submerged
  AnswerTests: omnitest(correctVal='native_submerged and nonnative_submerged')
  Hint: The lack of significance can be interpreted with a p-value greater than 0.05

- Class: text
  Output: GREAT JOB! You could go on to graph the results but that's another module. We
made it through our first Burks swirl lesson together in which you did A LOT including testing
normality, transforming the data with a log10 command, testing it again, deciding to opt for
nonparametric statistics and then doing the analysis complete with a posthoc test. Along the
way, you made lots of decisions regarding statistical significance. Now you can use this lesson,
especially the functions of unite, kruskal.test and dunnTest to analyze your own data later in the
course. As practice, go ahead and write a results paragraph about this experiment. Swirl on!!
```