Tested Studies for Laboratory Teaching
Proceedings of the Association for Biology Laboratory Education
Vol. 37, Article 2, 2016

Utilizing the Software Package "R" to Generate Graphs and Perform Statistical Analyses in Undergraduate Laboratory Courses

Michael S. Berger

Washington State University Vancouver, School of the Environment, 14204 NE Salmon Creek Ave., Vancouver WA 98686 USA
(**msberger@wsu.edu**)

Courses in life sciences require students to graphically present data and utilize statistical tests. Students frequently lack basic skills regarding visual representation of data and performing statistical tests. I have developed a lab exercise that utilizes the software "R" as a learning tool to improve graphing skills and statistically analyze data. Learning goals were focused on, competency in the visual display of data, performing statistical analyses, and using the software package "R". A script based software removes the "black-box" effect that can limit conceptual understanding of the components involved in the process of visually representing data or performing statistical analyses.

Keywords: graphing, statistical analysis, introductory biology, R, software

Introduction

Many courses in the life sciences require students to graphically present data and use statistical tests to interpret results. Students frequently lack basic skills regarding visual representation of data and processes involved in performing statistical tests. Classroom software is often expensive and can have limitations on graph quality, complexity of graphs generated, or power to perform statistical analyses. This series of lab exercises utilizes the software package "R" as a learning tool to improve graphing skills and statistically analyze data. Learning goals are focused on: (1) competency in the visual display of data; (2) using the appropriate statistical analysis based on the question addressed; (3) learning how to use the software package "R". The software package "R" is an open source script-based software that can be used with a graphical user interface, such as "R Commander". Students can use this free software to generate high quality graphs and perform numerous statistical tests. This software is highly flexible and can be adapted to fit diverse course objectives. The use of a script-based software removes the "black-box" effect that can limit conceptual understanding of the components involved in the process of visually representing data or performing statistical analyses. Students should be introduced to "R" with step-by-step tutorials that guide them through the process of generating graphs or performing statistical analyses.

Student Outline

Graphically presenting data and using statistical tests to interpret results are important skills that are frequently used by scientists. We will use the software package "R" in this lab as a learning tool to help you improve graphing and data analysis skills. In the process, you will master a very powerful open source software package. The software package "R" is an open source script-based software that can be used with a graphical user interface (i.e., point and click menu), such as "R Commander". You can download and use this free software to generate high quality graphs and perform numerous statistical tests. Just in-case you missed the last sentence, "R" is free! You are not required to use "R" outside of this lab, but we do encourage you to try. We recommend that you take advantage of the opportunity to learn how to use a new data analysis tool. Outside of lab, you should use a graphing and statistical software package that you are comfortable with. Please note that Microsoft Excel is not acceptable for performing statistical analyses.

Graphs are used in science as a visual mechanism to efficiently and concisely communicate complex information. A good graph must have all of the necessary components, so the target audience can understand the material presented. Data must be presented in a manner that is not distorted, that is clearly visible, and that communicates the appropriate information.

Statistical tests, such as t-tests or ANOVA's, are tools to help interpret data in a standardized and objective manner. These tests compare differences between mean (i.e., average) values, taking variation within a mean into account. You can use a statistical test to indicate whether mean values differ, which will help you present the results of your experimental work. For example, if you performed an experiment that examined the effect of pH on enzyme activity, you could use a statistical test to support your observation that enzyme activity was significantly higher at pH of 7 compared to pH of 9 and the activity was the same at pH of 7 compared to pH 8. Always graph your data and visually examine it, before using a statistical tool to help you interpret your data.

Learning Goals

The goals of this lab are: (1) to understand components used to construct a graph; (2) to improve graphing skills and produce graphs for a lab report; (3) to interpret your data, using a statistical test to support your observations; (4) to learn how to use the software package "R".

The following material provides line-by-line instructions to construct graphs or perform statistical analyses. When "R" script (i.e., coding) is included, an explanation of the script terms is provided with bulleted points. A text file that contains necessary script, which can be copied and pasted into "R", has been provided.

1. Write down three specific components that should be included in an informative graph.

2. How can you represent variation in a graph? In other words, the data you collected has a mean value, but each data point is not exactly on the mean. Describe how you could visually display the variation that your data points have around a mean value.

3. What does the term sample size refer to? What is an appropriate sample size (i.e., number of replicates) to use when performing an experiment and analyzing data?

Visual Representation of Data Part 1

Line plot with error bars

1. Use "R" to construct a line plot that displays the reaction rate (i.e., time of starch disappearance) as a function of pH. You will want to use a sample size of five (n=5), so you can display a measure of variation in the form of error bars. Use a standard error of the mean (SEM) as your measure of variation.

2. Which variable will be used for the x-axis and y-axis of each graph?

3. **Examine the formatting of your data set.** A spreadsheet template has been provided for you (Enzyme_data.csv). **Do not change the column headers.** You will have three columns in your spreadsheet (Table 1). The spreadsheet is organized with two columns for your treatment level (i.e., pH) and one column for your response variable (i.e., time of starch disappearance). The file should be formatted as a "CSV (Comma delimited) (*.CSV)" format.

Table 1. Example of spreadsheet formatting with n=1 hypothetical data. "Acidbase" is a categorical independent variable, "ph" is a continuous independent variable, and "time" is a response (i.e., dependent) variable.

acidbase	ph	time
four	4	12
five	5	8
six	6	2
seven	7	4
eight	8	9
nine	9	11

4. **Open R and load required libraries.** Double click on the "R" desktop icon. Using the text file provided, **paste in #1 script to open the required libraries** at the prompt and press enter. **"R" is syntax specific.** For example, if you use a lower case "r" in the command "library(rcmdr)", the command will not work.

The script used in "R" is listed in this handout in bold. Each line of script is followed by an explanation. Use the text file to copy script into "R". The "#" symbol is used in "R" to denote text that is not part of the script. Text following the "#" symbol is not used by "R" and is included to explain the script components.

1.Script to Open Required Libraries

library(bear) # loads bear package that is used to summarize data
library(ggplot2) # loads ggplot2 package that is used to generate graphs
library(Rcmdr) # loads the graphical user interface (GUI) R Commander package

5. **Import the data file into R.** Using "R Commander" as a menu based graphical user interface, import your data file by using the menus: Data > Import data > "from text file, clipboard, or URL".
   A. Enter the name **enzyme** for the data set; remember, "R" is case-sensitive, so be sure to use lower-case letters.

   B. Click on "Commas" in the "Field separator" section.

   C. Click OK. Open your saved Enzyme_data.csv file in the dialog box.

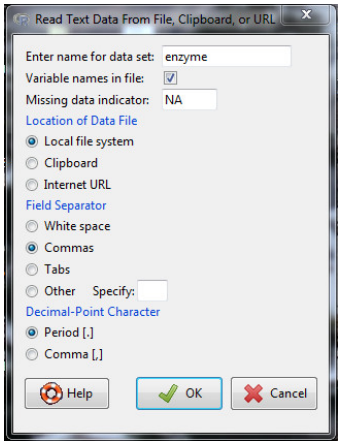

Figure 1. Screenshot of data import dialog box in "R".

6. **Construct a line plot.** You will construct a line plot for the experiment testing the effect of pH on enzyme activity. This is where you will enter your code into the "Script Window" in "R commander". The script can be copied and pasted in from the text file provided.

7. The first step is to use a command that summarizes your data into a mean and standard error of the mean for each treatment level (i.e., each pH). "R" will do all of the calculations and remember the summary for you.

   A. Paste in #2 script to summarize data from the text file into the "Script Window", highlight the script, and click on the "Submit" button.

#2. Script to Summarize Data

summary <- summarySE(enzyme, measurevar="time", groupvars=c("ph"))
summary

- # the term *summary* is used to identify and refer to the summary you generated
- # summarySE is the command to summarize the data set enzyme
- # "*enzyme*" is the name you assigned to the data set, "*measurevar*" defines your response variable, and "*groupvars*" defines your independent variable
- # graph will display your summary in the output window
- # In "R" speak, "c" means to concatenate, which is a complex way to say combine. In this instance, you are using "c" to combine all of the levels in your independent variable into one term, which is labeled "ph". The arrow symbol "<-" refers to the object receiving the value of the expression.

8. Use your data summary to construct a graph for reaction rate (i.e., time) as a function of pH.

   A. Paste in #3 **script to construct a line plot with error bars** from the text file into the "R Commander" script window.

   B. Make the following changes to the script lines that adjust the y-axis range and tick marks.
   scale_y_reverse(limits=c(60, 0),
   - **Change the number 60 to a value slightly greater than the greatest response variable value in your data set**

   breaks=seq(0,60,5))
   - **Change the number 60 to the same value you changed in the previous line**

   C. Highlight all of the script you pasted in and click on "Submit". **Your graph will appear in the original "R" window you opened up.** Look over the script to examine each line of code and the explanation that follows.

   D. If tick mark spacing is not appropriate, make the following change to the script "**breaks=seq(0,60,5))**".
   - **Change the number 5 to a lower or higher value.** This number refers to the tick interval. For example, "5" will result in a tick along the y-axis every 5 seconds.

#3. Script to Construct a Line Plot with Error Bars

ggplot(data=summary, aes(x=ph, y=time)) +
- # data= indicates the data set used, x=ph is the x-variable and y=time is the y-variable. We associated the summarized data (i.e., means and SEM) as the data set *summary*.

geom_line(size=0.8) +
- # command to construct a line plot with line size=0.8

geom_point(size=2.5) +
- # command to add a symbol with a symbol size of 2.5

geom_errorbar(aes(ymin=time-se, ymax=time+se), size=1, width=.1) +
- # command to add error bars that represent plus and minus 1 standard error of the mean

xlab("pH") + ylab("Reaction rate (seconds)") +
- # command to add x-axis and y-axis labels

scale_y_reverse(limits=c(60, 0),
- # scale_y_reverse sets a reversed range of the y-axis between the values indicated in the parenthesis

breaks=seq(0,60,5)) +
- # Adjusts the y-axis ticks from 0 to 60 with 5 unit intervals.

theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
- # removes default x-axis and y-axis grids

panel.background = element_blank(), axis.line = element_line(color = "black"), axis.text.y = element_text(size = 12, color="black"),
- # sets y-axis text size and color

axis.text.x = element_text(size = 12, color="black"),
- # sets x-axis text size and color

axis.title.y=element_text(size=14,face="bold", vjust=1.5),
- # sets y-axis label text size and position

axis.title.x=element_text(size=14,face="bold", vjust=-.5)) +
- # sets x-axis label text size and position

ggtitle("pH")
- # Adds a plot title

9. Review the script you used to construct your graph. Make changes to some of the commands, so the graph is customized to your needs or style. You can always undo any changes, if you do not like the results. The following website, http://www.cookbook-r.com/, is a decent resource for "R" script. If you are feeling very adventurous, try looking up how to change an attribute of your graph, such as line color.

10. Save your graph and tape a copy on your refrigerator. In the "R" window, where your graph is located, click on your graph. In the upper left of the "R" window, click on File > Save as. Choose a file type and save your graph. I suggest saving your graph as a JPEG at 100% quality, which is an image that is similar to a photo. A JPEG file can easily be pasted into a word document and used in a lab report.

II. Statistical Analysis of Data

One-way ANOVA

1. After visually exploring your results by using the graph you generated, you will want to statistically analyze your results by performing a one-way ANOVA with a post hoc test. The statistical output that you generate can be used to help you interpret the effects of pH on amylase reaction rates. A one-way ANOVA can be performed in "R Commander" using menu commands. Script can be used, but is not needed for a basic analysis.

2. Test for the assumption of normality by running a Shapiro-Wilk normality test. **Statistics > Summaries > Shapiro-Wilk Test for Normality**.

   A. In the "Variable" box, choose "**time**", your response variable.

   B. Click OK. Results can be viewed in the output window.

Results from Shapiro-Wilk test for normality should be interpreted before proceeding. If the *p-value* is less than 0.05, the distribution is non-normal. In cases where data is not normally distributed, ANOVA's are robust to departures from normality (Quinn and Keough, 2002; Underwood, 1997).

3. Test for the assumption of equal variance. **Statistics > Variances > Levene's Test**.

   A. In the "Factors" box, choose your categorical dependent variable, "**acidbase**".

   B. In the "Response Variable" box, choose your response variable, "**time**".

   C. Click OK. Results can be viewed in the output window.

Results from Levene's test for equal variance should be interpreted before proceeding. If the *p-value* is less than 0.05, the variance between mean values is not equal. Violations of homogeneity of variance should be addressed before proceeding. ANOVA results can be affected by unequal variances (Quinn and Keough, 2002; Underwood, 1997).

4. Perform a one-way ANOVA with a post-hoc test. In R Commander, use the menu**: Statistics > Means > One-way ANOVA**. A one-way analysis of variance window will open.

   A. Change the name of the model to pH.

   B. In the "Groups" box on the left, click on "**acidbase**", which is the group you want to analyze.

   C. In the "Response Variable" box on the right, click on the response variable **"time**".

   D. Check the box labeled "Pairwise comparison of means". This setting will run a post-hoc test, so you can determine whether mean values in an experiment were statistically different from each other.

   E. Click "OK". Your results will appear in the "Output" window in "R Commander. Save the output as a text file by using the menu: **"File" > "Save output as"**.

5. The results of the ANOVA can be found in the "Output" window, below the text "summary(model name)". Note that model name is the name you assigned in step 4A. See Table 2.

Table 2. Example of the ANOVA output in "R".

Summary(pH)

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
pH	5	140	28	19	1.3e-07***
Residuals	24	36	1.5		

Based on this output, you can report in your results section that a change in pH resulted in an increase in the reaction rate of amylase ($F_{5,24}$ = 140, p<0.000). This tells us that pH had an effect on enzyme reaction rate, but it does not tell us which specific pH were statistically different from each other.

6. The output for the post-hoc test can be found below the text reading "Linear Hypotheses:", in the table with reported p-values (Pr(>|t|). Be sure you are looking at the correct output.

Table 3. Example of a partial post-hoc text output in "R".

	Estimate	Std. Error	T value	Pr(>|t|)
four - five == 0	0.42	0.7704	0.55	0.99
nine - five == 0	0.06	0.7704	0.08	1.0
seven - five == 0	-3.08	0.7704	-4.0	<0.006**
six - five == 0	-5.64	0.7704	-7.32	<0.001***

A post-hoc test compares all mean values to all mean values, taking variation into account. Based on this partial output, you can conclude that enzyme rate activity is the same at pH 4 and pH 9, compared to pH 5. Enzyme rate activity is significantly faster at pH 6 and pH 7 compared to pH 5.

III. Visual Representation of Data Part 2
Constructing a Line Plot with Multiple Lines

You will construct a multiple line plot in "R" using absorption spectra data for the pigments: chlorophyll-a, chlorophyll-b, xanthophyll, carotene, and total pigment.

1. Load your data into "R", through R Commander. You will have previously saved your data in a "CSV (Comma delimited) (*.csv)" format. A data file labeled **Absorption_spectra_data.csv** has been provided. **Data > Import data > "from text file, clipboard, or URL"**.

   A. Enter the name **pigment** for the data set; remember, "R" is case-sensitive, so be sure to use lower-case letters.

   B. Click on "Commas" in the "Field separator" section.

   C. Click OK. Open your saved Absorption_spectra_data.csv file in the dialog box.

2. Graph the data as a line plot.

   A. Begin with data for the absorption spectra of chlorophyll-a as a function of wavelength. Use the text file provided to paste in #4 script to construct an X-Y line plot with one line into the "R Script" window in R Commander.

#4. Script to Construct an X-Y Line Plot with One Line

plot(chlorophylla~ wavelength, data=pigment, type="l", lty=1, lwd=2, col="green", xlab="x-axis label", ylab="y-axis label", ylim=c(-0.1,2))

- # plot is the command to make a line or symbol plot based on an x-y coordinate(s)
- # *chlorophylla~wavelength* variable tells "R" which variables to plot as Y-axis by X-axis variables, respectfully
- # type="l" specifies a line plot without symbols
- # col="green" results in a green line
- # lty=1 specifies a solid line
- # lwd=2 specifies the line width
- # xlab and ylab code for x-axis and y-axis labels
- # ylim=c(-0.1,2) codes for the y-axis limits to range between -0.1 and 2

   B. Use #5 general script to add additional lines for chlorophyll-b, xanthophyll, carotene, and total pigment. Each line will require a new script.

   C. Copy and paste the #5 script from the text file to add an additional line for chlorophyll-b. Change the Y-axis variable to match the pigment data you want to visually display. **Make sure the variable name exactly matches the name in the data file.** Click on "View data set" in R Commander to view names of data headers, if needed.

   D. Change the line color. For example, you may want to represent chlorophyll-b as blue, xanthophyll as yellow, carotene as orange, and total pigment as black

   E. Repeat the previous steps C through E for the remaining pigment data.

   F. Highlight all of the script and click on "Submit".

#5. General Script to Add Additional Lines.

lines(Y-axis variable~X-axis variable, data=pigment, type="l", lty=1, lwd=2, col="blue")

- # This script adds another line to your plot. You can use this command to plot the remaining dependent variables. In other words, repeat this script for each additional dependent variable. Make sure you change the y-axis variable name to match the data you want to plot. You can change the color (col="blue") to the color of the pigment being graphed.

3. Add a legend to your graph

   A. Copy and paste in script #6 to generate a legend.

   B. Highlight the script you have entered and click on "Submit"

   #6. Script to Generate a Legend

   legend("top", c(Chl-a","Chl-b","Xanthophyll","Carotene", "Total pigment"), lty=c(1,1,1,1,1,1), lwd=c(2.5,2.5,2.5,2.5,2.5), col=c("green","blue","yellow","orange","black"))

   - # legend is the command to add a legend to the plot
   - # "top" indicates the location ("topright" or "topleft" could also be used)
   - # c() indicates the legend text
   - # lty=c(1,1,1,1,1,1) specifies a solid line type for each variable in the plot
   - # lwd=c(2.5,2.5,2.5,2.5,2.5) specifies the line width for each symbol
   - # col=c("green","blue","yellow","orange","black") specifies a line color for each symbol

4. Save your graph. In the "R" window where your graph is located, click on your graph. In the upper left of the "R" window, click on File > Save as. Choose a file type and save your graph.

IV. Visual Representation of Data Part 3

Constructing an X-Y-Z Contour Plot

X-Y-Z contour plots are commonly used to graphically display data across three dimensions, where two independent variables are included. For example, marine scientists frequently measure water temperature as a function of depth and distance. An X-Y-Z plot can be used to display how water temperature varies by depth and across a known distance.

1. Enter your data in a spread sheet with three columns labeled with your X-variable (independent variable 1), Y-variable (independent variable 2), and Z-variable (dependent variable). **Save the file as a text (Tab delimited) file with a .txt extension.** A data file for this exercise has been provided on your computer.

2. Open "R"; you do not need to use "R commander". We will not use the "R commander" interface, so you can learn to write directly into "R"! If you are more comfortable with R commander, you can continue to use it.

3. To open your data, enter the following at the prompt, which is the > sign. Make sure your data is saved on the desktop.

#7. Script to Load Data for X-Y-Z Plot Construction

data<-read.table("c:\\user\\Desktop\\xyz.txt",header=T)

- # This command loads the data set based on the file location and associates it with the name "*data*". You will need to know the location of your data file. File location can be determined by right mouse clicking on the file and clicking on "Properties".
- # The location of "*user*" may need to be changed, depending on how user is defined
- # *header=T* lets "R" know your data has a text header

attach(data)

- # *attach* commands allow you to call the variables by name

names(data)

- # n*ames* will lists the variables you have assigned to the name *data*

4. Load the akima package, which is needed to interpolate the data in our graphs. Interpolate your data using script #8, which will "fill" in data between know data points. This operation is necessary to generate a contour gradient.

#8. Script to Interpolate Data

library(akima)

- # This opens the package akima

graph <-interp(distance,depth,temperature)

- # the *interp* command will interpolate X (distance), Y (temperature), and Z (temperature) to "fill" in data between know data points
- *graph* is the name we are associating with the interpolated data

5. Create a filled contour plot.

A. Paste in script #9 to generate a filled contour plot into the "R" prompt and press enter.

B. Press the up arrow on your keyboard. The last command you entered will appear. Use the left arrow to scroll through the script. Play around with the commands and make changes to customize the graph. For example, you may want to change the color scheme, number of colors in the gradient, axis text, or graph title.

#9. Script to Generate a Filled Contour Plot

filled.contour(graph,col = colorRampPalette(c("blue","yellow", "orange", "red"))(27), main = "Temperature",xlab="Distance (km)",ylab="Depth (m)", font.lab=2, ylim=rev(range(depth)), key.title = title(main=expression(~degree~C)))

- # the command *filled.contour* will produce a contour plot with a scale bar
- # the command colorRampPalette(c("blue", "yellow", "orange", "red")) determines the color scheme
- The number in parenthesis after colorRampPalette determines the number of colors in the gradient. If colors repeat, increase this number.
- xlab = and ylab = indicate the x-axis and y-axis label text
- font.lab = 2 indicates bold font for axis labels
- # The command [ylim=rev(range(Depth))] reverses the y-axis so zero is on top
- #key.title = adds a title to the Z-variable key

6. Save your graph. Click on the graph. File > Save as.

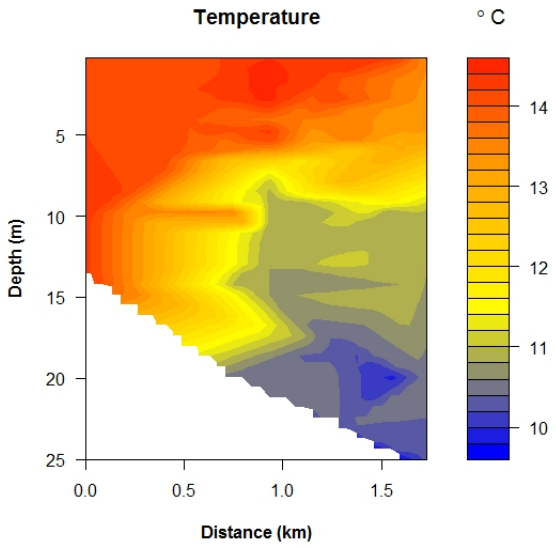

Figure 2. Example of X-Y-Z contour plot of temperature as a function of depth and distance. Plot was generated with "R".

V. Comparison between "R" and Excel

1. Choose one of the graphs you made in "R". Make the identical graph in Excel. Decide which software you prefer to use. Take into consideration the time spent constructing the graph and the quality of the end product. You can also take bragging rights into consideration.

2. Try to perform a one-way ANOVA with a post-hoc test in Microsoft Excel. Take into consideration how much time you spend figuring out how to run the statistical test, whether it can be done, and what the output is like.

Notes for the Instructor

When using "R" in the classroom, time should be devoted to teaching students how to use the software in the context of the content being delivered. A tutorial with step-by-step instruction should be provided to the students. The use of "R" can be integrated into a lab for use as a learning tool and as a means to produce a tangible product, such as a graph or statistical test result, from student generated data. If students are using "R" for the first time providing students with script to copy and paste can reduce typographical and syntax errors.

To install "R", visit http://www.r-project.org/. Follow the download instructions. "R" is command based. You will be using a Graphical User Interface called "R Commander", which allows you to point and click compared to typing in commands. To install "R Commander", open "R" and type **install.packages()** and then press enter. Choose a site to download from, such as USA (CA1). Scroll through the packages until you find a series of packages labeled with Rcmdr. Highlight all of these packages and press enter. When you run "R Commander" for the first time, you may be prompted to install more packages; click on yes to install the packages. **You will also want to install the following packages: bear, ggplot2, and akima.** If you receive a warning that a specific package is missing, go through the process of installing the missing package. If a package will not install, try closing and re-opening "R".

This lab is based on "R" version 3.12. If you are using an older version of "R", the following script can be used to update "R" to the latest version.

Script to update "R" to the latest version.

if(!require(installr)) {install.packages("installr"); require(installr)}

- # Installs and loads the package

updateR()

Begins the updating process of R. This script will check for newer versions and if one is available, allow you to update.

Literature Cited

Quinn, G. P., and M. J. Keough. 2002. *Experimental Design and Data Analysis for Biologists*. Cambridge University Press, Cambridge, 553 pages.

Underwood, A. J. 1997. *Experiments in Ecology: Their Logical Design and Interpretation using Analysis of Variance*. Cambridge University Press, Cambridge, 524 pages.

Acknowledgments

I thank all of the Biology 107 teaching assistants who have trialed portions of this laboratory and provided constructive feedback. Drafts of this laboratory exercise were improved by comments from the Association for Biology Laboratory Education conference participants.

About the Author

Michael Berger is a Senior Instructor at Washington State University Vancouver. He received his B.S. in Ecology and Evolutionary Biology from the University of Connecticut and a Ph.D. in Marine Biology from the University of Oregon. His background and area of interest is in marine ecology and invertebrate biology.

Mission, Review Process & Disclaimer

The Association for Biology Laboratory Education (ABLE) was founded in 1979 to promote information exchange among university and college educators actively concerned with teaching biology in a laboratory setting. The focus of ABLE is to improve the undergraduate biology laboratory experience by promoting the development and dissemination of interesting, innovative, and reliable laboratory exercises. For more information about ABLE, please visit **http://www.ableweb.org/.**

Papers published in *Tested Studies for Laboratory Teaching: Peer-Reviewed Proceedings of the Conference of the Association for Biology Laboratory Education* are evaluated and selected by a committee prior to presentation at the conference, peer-reviewed by participants at the conference, and edited by members of the ABLE Editorial Board.

Citing This Article

Berger, M. S. 2016. Utilizing the Software Package "R" to Generate Graphs and Perform Statistical Analyses in Undergraduate Laboratory Courses. Article 2 in *Tested Studies for Laboratory Teaching,* Volume 37 (K. McMahon, Editor). Proceedings of the 37th Conference of the Association for Biology Laboratory Education (ABLE). http://www.ableweb.org/volumes/vol-37/?art=2