ESSAY

# Creating Figures in R that Meet the AFS Style Guide: Standardization and Supporting Script

**Hayley C. Glassic** | Montana Cooperative Fishery Research Unit, Department of Ecology, Montana State University, Bozeman, MT 59717. E-mail: hcg0509@gmail.com

**Kurt C. Heim** | Department of Ecology, Montana State University, Bozeman, MT

**Christopher S. Guy** | U.S. Geological Survey, Montana Cooperative Fishery Research Unit, Department of Ecology, Montana State University, Bozeman, MT

Visual display of information in scientific and nonscientific literature is the most efficient way to summarize large amounts data, focus the readers' attention on patterns, and substantiate the message in the narrative. Figures often represent years of data collection and substantial monetary investment, and it is worth repeating the cliché "a [figure] is worth a thousand words." Well-designed figures are usually simple, yet their ability to relate complex information through simplicity makes them powerful (Tufte 2001). Figures are often the focal point of articles when scientists, science communicators, and policy makers are quickly searching for scientific information. Scientists in academia ranked figures and tables as the most important component of research articles (Hubbard and Dunbar 2017); moreover, figures quickly become the focus when discussing articles among colleagues or in a classroom setting. If created effectively in combination with a well-articulated figure caption, figures convey complex information readily for the reader to make a conclusion about results without reading details presented in the narrative. This is especially important as the quantity of research articles continues to increase exponentially in the 21st century, as does the number of journals with specific figure guidelines (Jinha 2010).

Scientific journals have their own guidelines for constructing figures, and in this article we focus on the American Fisheries Society (AFS) style guide (Table 1). The AFS figure guidelines for publications are simple and allow for some creativity. In addition to journal-specific guidelines, generally suggested practices exist for figure design such as ensuring the figure provides the correct message for the audience (Rougier et al. 2014; Table 2), making legends or captions clear and succinct (Moon 2012), and limiting the messages communicated by a figure to facilitate easy and efficient interpretation (Moon 2012; Table 2).

Many programs are available for scientists when creating publication-quality figures including those with graphical user interfaces (GUIs) such as Microsoft Excel (Microsoft Corporation) and SigmaPlot (Systat Software Inc.), and those without including Matlab (MathWorks), SAS (SAS Institute), and R (R Core Team). Additionally, integrated development environments such as RStudio (Rstudio, Inc.) and SAS Studio (SAS Institute) provide many tools for source code editing, debugging, saving work and projects, and making graphics. All environments have advantages and disadvantages with regards to ease of use, flexibility, and software expense that can vary from free to thousands of dollars. The popularity of R, in particular, has increased rapidly in the past decade because it is free and includes many contributed packages for powerful analyses applicable to many fields of research. Also, because R is one of the primary analytical tools used in statistics courses at universities (Carson and Basiliko 2016), it has seen growing popularity among recently trained ecologists. Where users of GUIs may know how to execute simple analysis or create a visually appealing figure, GUIs are generally limited in scope compared to opportunities provided by integrated environments for programming languages (Baker 2017).

R can be used to quickly visualize data, which is highly recommended before any statistical analyses are performed (Hilborn and Mangel 1997). In addition, R allows the user to analyze data and create publication-quality figures in one environment, thus there is no moving of data and analyses among software packages that can introduce mistakes. Finally, the figures created with R are easily reproducible because the instructions are saved in a script, which allows collaborators and peer reviewers to track the exact steps and data used in figure creation.

Using R for publication-quality figures appears to be increasing in popularity among fisheries professionals, particularly with early-career scientists. We conducted a simple poll of 80 recent fisheries graduate students at Montana State University, University of Alaska Fairbanks, and Utah State University, asking them "What type of graphing tool do you most frequently use for publication-quality figures?" including a list with R, Excel, SigmaPlot, SAS, and write-in response. The survey participants reported using R to create publication-quality figures for 80% of recent manuscript submissions.

Using scripts to perform analyses and create figures in R has a relatively steep learning curve compared to analogous methods using GUIs, but investing time in coding can improve work efficiency and quality in the long term (Baker 2017). RStudio is an integrated development environment for R that helps edit scripts by displaying icons marking lines of script where mistakes occur. In addition, RStudio provides panes that make viewing figures, file management, and project development easier. One of the shortcomings of R, and the focus of this paper, is that default plots generated by the program do not adhere to AFS standards required for publications. Most figures need editing, regardless of the software used to create them; thus, a good rule-of-thumb is to never assume the defaults provide the necessary components to meet the journal

DOI: 10.1002/fsh.10272

Table 1. Abbreviated guidelines for figures published in American Fisheries Society (AFS) journals. The complete guidelines are available at: https://fisheries.org/books-journals/writing-tools/style-guide/.

| AFS Guideline |
|---|
| Ratio: As a rule, figures should be rectangular with a height : width ratio of 2 : 3 or 3 : 4 (unless such a ratio distorts the data). |
| Borders: Do not use borders around figures. In figures portraying the (x, y) plane, show only those two axes unless data are also arrayed along a *y*-axis on the right. |
| Tick marks: Place tick marks on the outside of the axes (i.e., to the left of the *y*-axis and below the *x*-axis). |
| Font: Use the font Times Roman for all axis labels; the font size should be 6–9 points after reduction and not bold or italic. Other fonts may be used for other items as long as (1) the same font is used for analogous items and (2) the font size is no larger than that used for the axis labels (but at least 6 points). |
| Font emphasis: A bold sans serif font (e.g., Arial) may be used for the letters that distinguish different panels (A, B, etc.). Otherwise avoid bold type, as it tends to fill in when reduced. Also avoid italic type, as it tends to wash out. |
| Capitalization: Capitalize the first word in all labels as well as any proper nouns and adjectives; do not capitalize other words. |

Table 2. Suggested practices for figure design.

| Moon (2012) | Rougier et al. (2014) |
|---|---|
| Orient reader with a clear legend | Captions are not optional |
| Organize each graph sensibly | Identify your message |
| Show data clearly and efficiently | Know your audience |
| Scale the data frame to best show patterns in the data | Adapt the figure to the support medium |
| Help readers see the patterns in the data | Avoid "Chartjunk" |
| Make the data the most prominent part of the graph | Use color effectively |
| | Do not mislead the reader |
| | Do not trust the defaults |
| | Message trumps beauty |
| | Get the right tool |

style guides (Table 1) or meet the guidelines for outstanding figures (Rougier et al. 2014; Table 2).

Here we provide templates for creating figures in R that adhere to the AFS guidelines, which should help address the problems outlined above. Nearly every aspect of figure appearance can be controlled in R by using syntax, functions, or packages. Two main figure creation methods in R include base R graphics (hereafter referred to as base R) and ggplot2 (ggplot2 is part of the tidyverse package; Wickham 2017). Both have unique syntax and defaults, which may deviate from AFS standards. Here we will focus on base R and ggplot2 in relation to creating publication-quality figures adhering to AFS standards using R version 3.5.1 (2018), RStudio version 1.1.456, and package 'tidyverse' (version 1.2.1, 2017; ggplot2 version 3.1.0). A publication-quality figure can be made using either base R or ggplot2; often, the method used is simply a matter of personal preference.

Learning R can be challenging, and once learned, can be easily forgotten. There are many ways to get help and one of the most useful features of R is the question mark (?). Using ? will bring the user immediately to the help page for any given function. For example, if you are confused about the specifics of making a .tiff file, type ?tiff() (in the console) and press enter. A searchable help page providing guidance will appear on the screen. Secondly, there are many sources of help online (e.g., Stack Overflow, Cookbook for R, R-bloggers), and typing R questions directly into Google (e.g., "how to make a stacked barplot in R") usually yields good results. Lastly, one of the benefits of coding (rather than "clicking") to make your figures is that each step in the process is saved, reproducible, and transferable. Even if a particular function or complex string of dedicated syntax are forgotten between uses, if you have learned it once (and saved your work) then it can be retrieved and copied to use in another plot.

Our goal is to provide general guidelines and annotated code for making publication-quality figures that adhere to the AFS style guidelines and best practices for figure creation as outlined by Moon (2012) and Rougier et al. (2014; Table 2) while maintaining creative freedom for authors. Although we provide a demonstration using a scatterplot, the script we provide will also be a useful starting point for constructing other types of plots (e.g., barplots, boxplots, lineplots) commonly used to communicate fisheries data. We also believe this will be useful for subject and technical editors as a document that they can reference to authors for finding helpful information for creating figures in R.

## PLOTTING IN BASE R

The code below (Box 1; also available: https://github.com/CGuyMSU/AFS-figures) will generate length and weight data for Largemouth Bass *Micropterus salmoides* and Channel Catfish *Ictalurus punctatus* using coefficients from standard–weight equations found in Neumann et al. (2012). Then, the code creates a default plot (Figure 1, panel A) and a user-customized plot (Figure 1, panel B). We discuss some of the important functions used to adjust default values that enhance readability and meet AFS guidelines below. In the following text, R functions are bolded (e.g., **plot()**) and arguments used within functions are bolded and italicized (e.g., ***bty* = *"n"***). Although this code creates a scatterplot, the commands to adjust style (e.g., all commands within **par()**) and axes are compatible with other base R plotting functions like **barplot()**, **hist()**, and **boxplot()**.

*Aspect ratio and figure quality* – One of the most frustrating issues is the creation of figures with poor resolution—pixilated as if it were a long-lost photograph of bigfoot. This can be avoided by using any one of the following commands to export a high-quality image from R— **tiff()**, **png()**, **jpeg()**, or **bmp()**. These functions allow one to control aspect ratio (***height* =**, ***width* =**), resolution (***res* =**), and file format explicitly. Here we use **tiff()** at line 15, which routes what would normally appear in the plotting window to a generated file in the working directory (line 2). We recommend setting resolution to 300 dpi at a minimum (***res* =** *300*). The figure is saved by running the **dev.off()** function (line 45). Usually, some trial and error is involved in creating a final figure and once completed, the code can be nested within **tiff()** and **dev.off()** to save. The figures can be exported with the aspect ratio of 2:3 or 3:4 by changing values associated with ***width* =** and ***height* =** (line 15) and printed to ensure readability prior to journal submission. As noted in the AFS guidelines, other aspect ratios may be needed to best represent the data. Figure exporting functions are also useful when making figures for presentations projected onto large screens where poor image quality is magnified.

*Remove the box* – Figure 1 panel A displays a secondary *y*-axis without tick marks, which violates AFS standards. This "box" is a default but can be adjusted using the graphical parameter ***bty* = *"n"*** or ***bty* = *"l"*** within the **plot()** function. A more flexible alternative is to suppress drawing all default axes with ***axes* = *FALSE*** (line 29), and then build the axes separately with the **axis()** function (lines 33 and 35). While building custom axes does add an extra step, it may help the user explicitly consider both the message a figure conveys and its appearance (i.e., scale, tick-mark parameters, etc.).

*Customizing axes* – American Fisheries Society journals require tick marks be placed on the outside of axes (Table 1), and fortunately R will do this by default. Carefully consider the scale required to convey your message and use an appropriate number of tick marks to minimize difficulty of interpretation by the reader. Figures display trends, thus, if exact values are critical to understanding, consider using a table instead (Moon 2012). If the minimum value in the dataset is near zero, start the axis at zero. This will avoid potential misinterpretation of the figure, especially if the other axis does start at zero. Axes that end on a tick mark complete the figure and make it easier for the reader to determine the range of the data (Figure 1, panel B). Furthermore, the "hanging" line (e.g., Figure 1 A, *x*-axis) can suggest something is missing. These design tips can be accomplished with the ***at* =** command within the **axis()** function (lines 33 and 35). With these suggestions, it is possible that the user-built axes can crop data from the figure. We suggest viewing the default margin values before manipulating axes yourself to avoid unintentional cropping out data.

*Use space wisely* – The graphic displayed in panel A of Figure 1 has excessively wide outer margins. The **par()** function is worth understanding if using base R for plotting; **par()** provides flexibility to alter individual components of the figure appearance. This was remedied in Figure 1 panel B by adjusting the margins with the **par(*mar* =)** function **(**line 27). Default R plots also place the axis labels quite far away from the axis tick marks, which can be adjusted with **par(*mgp* =)** (line 27).

*Annotate figures* – Using **mtext()** provides excellent flexibility to add axis labels and panel lettering (e.g., A, B, C) exactly where they are needed. We used this to add panel labels ("A" and "B" in lines 24 and 41), as well as axis labels (line 37 and 39).

*Use Times New Roman* – Changing the font used in an R plotting device can be adjusted with the ***family* =** command in the **par()** function, or directly within other functions like **mtext()** or **plot()**. Depending on the plotting device used and the operating system, different options for this will be available. Generally, typing ***family* = *"Times"*, *family* = *"Times New Roman"*, *or family* = *"serif"*** will change the font. As recommended by AFS, use bold and italics sparingly.

*Adding a legend* – Adding a legend is easy, and often leads to more rapid figure interpretation than limiting details to the figure caption. The function **legend()** will add a legend to the plot, and the location of the legend can be set with the first two arguments (we used x = 125 and y = 2,200 in line 43) where the positional coordinates use the units of the data.

**BOX 1. BASE R**

```
 1 #set working directory, you select this on your computer
 2 setwd("your/directory/here")
 3
 4 #generate length and weight data for Channel Catfish and Largemouth Bass
 5 length <- seq(from = 200, to = 500, by = 10)
 6 #a and b values from Fisheries Techniques Ws equations
 7 a_lmb <- (-5.528)
 8 b_lmb <- 3.273
 9 a_cat <- (-5.800)
10 b_cat <- (3.294)
11 weight_lmb <- 10^(a_lmb + b_lmb * log10(length))
12 weight_cat <- 10^(a_cat + b_cat * log10(length))
13
14 #begin .tiff file of the following two panel figure, finished and saved at line 45
15 tiff("baseR_figure.tiff", width = 20.32, height = 7.62, units = "cm", res = 300)
16 #one row, two panels
17 par(mfrow = c(1,2))
18
19 #make default plot
20 plot(length, weight_lmb)
21 #add points to default plot
22 points(length, weight_cat)
23 #add panel label A to default plot
24 mtext("A", at = min(length), adj = 0, line = 2, cex = 1.5)
25
26 #make custom figure that meets AFS style guidelines, first set plotting parameters (e.g., a serif font)
27 par(family = "Times New Roman", mar = c(3,4,2,2), mgp = c(3,.6,0))
28 #make custom plot
29 plot(length, weight_lmb, axes = FALSE, xlab = NA, ylab = NA, ylim = c(0,2400), xlim = c(100,600), pch = 19)
30 #add points for catfish
31 points(length, weight_cat, pch = 21, bg = "white")
32 #build x-axis
33 axis(1, pos = 0, at = seq(100,600, by = 100), lwd = 1.5)
34 #build y-axis
35 axis(2, pos = 100, at = seq(0,2400, by = 400), las = 1, lwd = 1.5)
36 #add x-axis label
```

(continues)

**BOX 1 (CONTINUED)**

```
37 mtext("Length (mm)", side = 1, line = 1.5)
38 #add y-axis label
39 mtext("Weight (g)", side = 2, line = 2)
40 #add panel label B to custom plot
41 mtext(expression(bold("B")), at = 100, adj = 0, line = 0, cex = 1.5, family = "Arial")
42 #add legend
43 legend(125, 2200, c("Largemouth Bass", "Channel Catfish"), pch = c(19,1), bty = "n")
44 #finish plot and save to disk
45 dev.off()
```

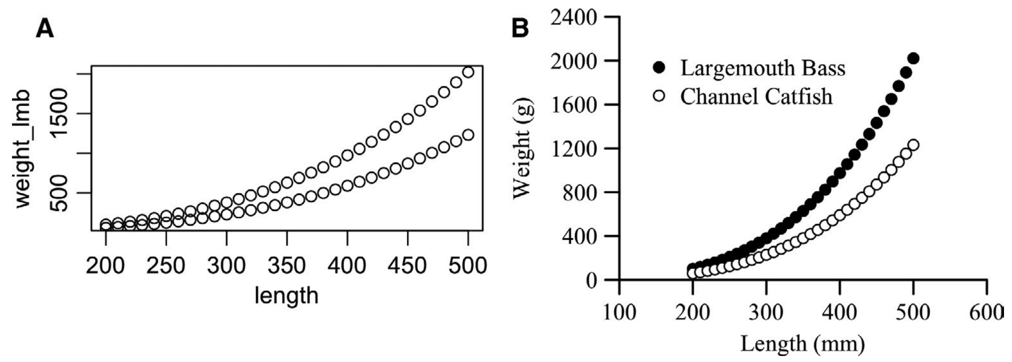

Figure 1. Figures made with base R including one with (A) default values and (B) a user-customized figure that adheres to American Fisheries Society guidelines for authors.

## PLOTTING IN GGPLOT2

The code below (Box 2; also available here: https://github.com/CGuyMSU/AFS-figures) will generate a figure using ggplot2 with the same data as shown in Box 1. We will not repeat the step-by-step style process as in Box 1 because they are the same for any plot submitted to AFS for publication. Here we use the guidelines in Box 1 and show code used to create a default ggplot2 figure (Figure 2, panel A) and a custom ggplot2 figure (Figure 2, panel B). We illustrate some important functions used to adjust a default ggplot2 figure to enhance readability and meet AFS guidelines. There are many outstanding references on using ggplot2 (e.g., Burchell and Vargas 2017; Wickham and Grolemund 2017; http://serialmentor.com/dataviz/index.html; https://www.datacamp.com/home; https://www.rstudio.com/resources/cheatsheets/), and we suggest reading Wickham and Grolemund (2017) to better understand the syntax grammar of graphics using ggplot2.

*Default ggplot* – The code on lines 22–23 will generate a default figure using ggplot2. The default ggplot2 has a grey background, white gridlines, and black tick marks. This coloration does not adhere to AFS guidelines, the *x*-axis and *y*-axis often do not end on values—leaving hanging axes, which can make it difficult to interpret data in the figure. The default figures in ggplot2 do not adhere to the AFS style guidelines (Table 1) nor those recommended by Moon (2012) and Rougier et al. (2014) (Table 2).

*Adjusted ggplot that meets AFS style guidelines* – Although the amount of code required to adjust a ggplot2 figure that meets the AFS style guide may seem substantial, once a template has been created (especially lines 51-76) the code can be reused for a variety of plots. This recycling of code can be accomplished by simply changing the data (line 28), the **geom** function (line 30), and the scale limits (lines 32 and 34). The ***expand* = c**(0,0) on lines 32 and 34 ensures that there will be no hanging axes. The **theme_classic()** function is useful for removing the grey background with white grid marks (line 47). Once the grey background has been removed, syntax within the **theme()** function can be used to modify individual figure components such as axis titles, font type, legend, tick mark, tick mark titles, and axis type. The **theme()** (lines 51–76), which allows the user to control all the individual components of the figure, is similar to the **par()** function in base R. Keep in mind that the aforementioned descriptions apply to basic ggplot2 usage and that the flexibility of R graphics will allow for nearly unlimited options for alteration of figure appearance.

As mentioned for Box 1, the possibility exists to save a high-resolution figure for AFS. Using **ggsave** and specifying the file type (e.g., .tiff) and the dots per inch (dpi) will ensure that AFS is using a high-resolution image and it will not be pixelated in the printed or online version (line 85).

**BOX 2. GGPLOT2**

```
1 #load ggplot2 and other useful packages
2 library(tidyverse)
3 #load package for combining figures
4 library(gridExtra)
5
6 #set working directory, you select this on your computer
7 setwd("your/directory/here")
8
9 #generate length and weight data for Channel Catfish and Largemouth Bass, and put in dataframe
```

(continues)

**BOX 2 (CONTINUED)**

```
length <- seq(from = 200, to = 500, by = 10)
species <- c(rep("lmb", 31), rep("cat", 31))
12  a_lmb <- (-5.528)
b_lmb <- 3.273
a_cat <- (-5.800)
b_cat <- (3.294)
weight_lmb <- 10^(a_lmb + b_lmb * log10(length))
weight_cat <- 10^(a_cat + b_cat * log10(length))
weight <- c(weight_lmb, weight_cat)
length_weight_data <- data.frame(species, length, weight)

#make default ggplot figure with a legend and annotated label
len_wt_default <- ggplot(data = length_weight_data, aes(x = length, y = weight, fill = species)) + geom_point() +
labs(title = "A")
#view the plot, will appear in R plotting window
len_wt_default

#make ggplot figure that meets AFS style guidelines
len_wt_afs <- ggplot(data = length_weight_data, aes(x = length, y = weight, fill = species)) +
#set symbol shape and size
geom_point(shape = 21, size = 2) +
#set the limits and tick breaks for the y-axis
scale_y_continuous (limits = c(0,2400), expand = c(0,0), breaks = seq(0,2400,400)) +
#set the limits and tick spacing for the x-axis
scale_x_continuous(limits = c(100,600), expand = c(0,0), breaks = seq(100,600,100)) +
#adjust the order of the legend, make new labels, and select the symbol colors
scale_fill_manual(limits = c("lmb", "cat"), labels = c("Largemouth Bass", "Channel Catfish"),
          values = c("black", "white")) +
#add B to figure
ggtitle ("B") +
#label the y-axis
ylab("Weight (g)") +
#label the x-axis
xlab("Length (mm)") +
#add legend title, but left blank here because we want a legend but no title
labs(fill = "") +
#makes the figure background white without grid lines
theme_classic() +

#below are theme settings that provide unlimited control of your figure and can be a template for other figures
#set the size, spacing, and color for the y-axis and x-axis titles
theme (axis.title.y = element_text(size = 14, margin = margin(t = 0, r = 10, b = 0, l = 0), colour = "black"),
axis.title.x = element_text(size = 14, margin = margin(t = 10, r = 0, b = 0, l = 0), colour = "black"),
#set the font type
text = element_text(family = "Times New Roman"),
#modify plot title, the B in this case
plot.title = element_text(face = "bold", family = "Arial"),
#position the legend on the figure
legend.position = c(0.3,0.85),
#adjust size of text for legend
legend.text = element_text(size = 12),
#margin for the plot
plot.margin = unit(c(0.5, 0.5, 0.5, 0.5), "cm"),
#set size of the tick marks for y-axis
axis.ticks.y = element_line(size = 0.5),
#set size of the tick marks for x-axis
axis.ticks.x = element_line(size = 0.5),
#adjust length of the tick marks
axis.ticks.length = unit(0.2,"cm"),
#set size and location of the tick labels for the y axis
axis.text.y = element_text(colour = "black", size = 14, angle = 0, vjust = 0.5, hjust = 1,
margin = margin(t = 0, r = 5, b = 0, l = 0)),
#set size and location of the tick labels for the x axis
axis.text.x = element_text(colour = "black", size = 14, angle = 0, vjust = 0, hjust = 0.5,
margin = margin(t = 5, r = 0, b = 0, l = 0)),
#set the axis size, color, and end shape
axis.line = element_line(colour = "black", size = 0.5, lineend = "square"))

#view the plot, will appear in R plotting window
len_wt_afs

#arragne the two plots side by side using the gridExtra package
ggplot_figure <- grid.arrange(len_wt_default, len_wt_afs, ncol = 2)

#save the plot as a .tiff as a very large file, which is publication quality
ggsave(ggplot_figure, file = "ggplot_figure.tiff", width = 20.32, height = 7.62, units = "cm", dpi = 300)
```

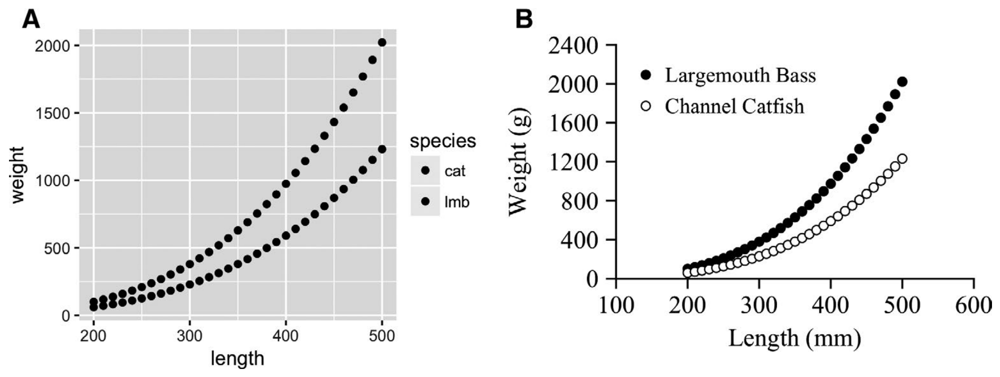

Figure 2. Figures made with ggplot2 including one with (A) default values and (B) a custom figure that adheres to American Fisheries Society guidelines for authors.

## ACKNOWLEDGMENTS

The Montana Cooperative Fishery Research Unit is jointly sponsored by the U.S. Geological Survey (USGS); Montana Fish, Wildlife & Parks; and Montana State University. Any use of trade, product, or firm names is for descriptive purposes only and does not imply endorsement by the U.S. Government. Although this code has been processed successfully on a computer system at the USGS, no warranty expressed or implied is made regarding the display or utility of the code for other purposes, nor on all computer systems, nor shall the act of distribution constitute any such warranty. The USGS or the U.S. Government shall not be held liable for improper or incorrect use of the code described and/or contained herein.

## REFERENCES

Baker, M. 2017. Scientific computing: code alert. Nature 541:563–565. Nature Research.

Burchell, J., and M. Vargas. 2017. The hitchhikers guide to ggplot2. Leanpub, Victoria, British Columbia, Canada.

Carson, M. A., and N. Basiliko. 2016. Approaches to R education in Canadian universities. F1000Research 5:2802. Faculty of 1000 Ltd.

Hilborn, R., and M. Mangel. 1997. The ecological detective: confronting models with data. Princeton University Press, Princeton, New Jersey.

Hubbard, K. E., and S. D. Dunbar. 2017. Perceptions of scientific research literature and strategies for reading papers depend on academic career stage. PLoS ONE 12:e0189753.

Jinha, A. 2010. Article 50 million: An estimate of the number of scholarly articles in existence. Learned Publishing 23:258–263.

Moon, R. D. 2012. Design of tables and figures for display of scientific data. Pages 89–109 *in* D. Mason, C. A. Jennings, T. E. Lauer and B. Vondracek, editors. Scientific communication for natural resource professionals, 1st edition. American Fisheries Society, Bethesda, Maryland.

Neumann, R. M., C. S. Guy, and D. W. Willis. 2012. Length, weight, and associated indices. Pages 637–676 *in* A. V. Zale, D. L. Parrish and T. M. Sutton, editors. Fisheries techniques, 3rd edition. American Fisheries Society, Bethesda, Maryland.

R Core Team. 2018. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. Available: https://www.R-project.org/.

Rougier, N. P., M. Droettboom, and P. E. Bourne. 2014. Ten simple rules for better figures. PLoS Computational Biology 10:e1003833.

Tufte, E. R. 2001. The visual display of quantitative information. Graphics Press, Cheshire, Connecticut.

Wickham, H. 2017. tidyverse: Easily install and load the 'Tidyverse'. R package version 1.2.1. Available: https://CRAN.R-project.org/package=tidyverse

Wickham, H., and G. Grolemund. 2017. R for data science. O'Reilly Media Inc., Sebastopol, California. AFS