# Walk-through Example

To help cement some of the aspects of RStudio usage and concepts outlined in this chapter, we have provided a short, editable R script (R script for walk-through.R) and a data set (called seed\_data.csv) to allow us to walk you through an example of graph-drawing using RStudio. Before we start, you’ll need to download and save both of these files to your computer. The data we are looking at in this example is adapted from an experiment into how different amounts of water affect the germination of seeds (Chatfield 1982). Five different levels of watering were coded as 1–5, with 1 corresponding to the least water and 5 corresponding to the most water. For each amount of water, four boxes were sown with 100 seeds each and left uncovered, and the number of seeds that had germinated after two weeks was recorded.

***Step 1*:** To start with, we need to load the R script. Once you have saved the script to your computer, you do this by going to ‘File’ in the main task bar of RStudio, selecting ‘Open File’ from the drop-down options, and finding the R script (R script for walk-through.R) wherever you have saved it. If you open it, the R script will appear as a window in the top left-hand corner of RStudio. You should see our annotations as green text preceded with # symbols, as well as code in black (for text) and blue (for numbers) ready to run (see the supplementary material Additional Guide 1.1 R Basics for a recap of Rstudio and R scripts).

***Step 2*:** We’ll start by entering some simple data to the R Console. You can either copy and paste the following code from the R script into the Console or highlight it on your script and press the ‘Run’ button near the top right of the script window.

**one<- c(22,25,27,23)**

**two<- c(41,46,59,38)**

**three<- c(66,72,51,78)**

**four<- c(82,73,73,84)**

**five<- c(79,68,74,70)**

These lines of code create lists for each level of watering (1–5) of how many seeds germinated per box (remember, there were four boxes for each watering level). R will now know that the name ‘one’ refers to the list of numbers 22, 25, 27, and 23, for example (see the Additional Guide 1.1 R Basics for a recap of inputting simple data). Note that we named the lists using text rather than the numbers themselves. This is because R gets confused if numbers are names instead of values (see Bigger Picture 1.1 for some additional general tips for working in R).

***Step 3*:** We can now quickly visualize our entered data as a boxplot by running the next chunk of code in the R Script:

**boxplot(one,two,three,four,five, ylab = "Seeds germinating per box",**

 **xlab = "Amount of water",ylim=c(0,100), yaxs = "i",**

 **names=c(1,2,3,4,5))**

Do not worry about all the unfamiliar code here—the important components of any figure type will be covered in their specific chapters (see chapters 4 and 5 for single- and multiple-sample boxplots respectively). What we are interested in showing you here is, firstly, how little code is needed to produce a figure quickly in R and, secondly, how reproducibleR code is across computers. The figure you have just produced (Figure S1.1) is identical to the figure we produced on our computers when first drafting this chapter, and you didn’t need to follow any complicated sequence of steps or button-pushing to replicate it exactly. Base R and associated code are consistent across versions and systems.



Figure S1.1: The number of seeds that germinated per uncovered box under each level of watering (n=4 for each watering treatment). Adapted from data from Chatfield (1982).

You might have noticed that the figure produced (Figure S1.1) is a little drab. This is because the code we have run so far does not include any instructions for R concerning colour. We’ll address this next.

***Step 4*:** As discussed in section 1.6, colour can add a lot to a figure’s visual appeal, but we need to be careful that our use of colour does not confuse our figure’s key message. With our seed germination boxplot (Figure S1.1), we can easily add colour with a tiny bit of extra code. Try running the next section of the script, but feel free to swap out the colour ‘green’ from the **col** (i.e. **col**our) argument to any colour you like from [the online list](http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf):

**boxplot(one,two,three,four,five, ylab = "Seeds germinating per box",**

 **xlab = "Amount of water",ylim=c(0,100), yaxs = "i",**

 **names=c(1,2,3,4,5),**

 **col="green")**



Figure S1.2: As Figure S1.1, but with boxes filled with a single colour.

In Figure S1.2, the boxes of the plot are all filled with the same colour and the figure is looking more attractive for it. While the recorded number of germinated seeds is a discrete continuous variable, the amount of watering the boxes received is an ordinal qualitative variable (see section 1.3 and the Additional Guide 1.4 Data types for details on different types of data). The categories of watering treatment are already shown on the x-axis, though, so there is no real need to use different colours across the boxes to visualize the difference between water levels further.

***Step 5*:** Despite the fact that we have just concluded that one single colour across all the boxes makes the most sense, we will quickly consider how we could make the boxes different colours if (for a reason guided by our data) we were so inclined. We will imagine here that it is through the box colours, rather than the x-axis, that we need to communicate the ordinal levels of watering. To do this, we would want to use a sequential colour palette, colouring lower levels a lighter shade of a hue and increasing the hue’s darkness as the level of watering increases (see section 1.6 and Scientific Approach 1.3 for more on colour palettes). As our variable concerns water, it might also be intuitive for viewers if we were to use the colour blue. Run the next chunk of code in the R script (changing the list of five colours to a different five shades of the same hue that increase in darkness if you like):

**boxplot(one,two,three,four,five, ylab = "Seeds germinating per box",**

 **xlab = "Amount of water",ylim=c(0,100), yaxs = "i",**

 **names=c(1,2,3,4,5),**

  **col=c("lightskyblue1","cadetblue2","cornflowerblue","dodgerblue4","darkblue"))**



Figure S1.3: As Figure S1.1, but with boxes filled with a sequential colour palette, increasing in the shade of darkness with the increasing amount of water added to boxes.

***Step 6*:** The seed experiment was also repeated, but with the boxes covered in order to slow evaporation (Chatfield 1982). We’re now going to bring in this extra data, but this time, rather than entering our data in the more laborious way of typing it all out, we’re simply going to load it in as a data set. The data set we have provided (seed\_data.csv) has been prepared in Excel and saved as a .csv file (see the Additional Guide 1.1 R Basics for tips on setting up and loading in Excel spreadsheets for use in R). To open it in RStudio, you can run the following line of code from the script in your Console and select the file from your computer’s folders:

**seeds <- read.table(file.choose(), header = T, sep = ",")**

Alternatively, you can follow the route of using the ‘Import Dataset’ button (see the Additional Guide 1.1 R Basics for a recap of this) but if you do so, you may want to subsequently rename the loaded data set as ‘seeds’ (to keep your code briefer) using the line of code below:

**seeds<-seed\_data**

Once your data set is loaded, have a look at it by running the following code from the script in the Console:

**View(seeds)**

You should now see the data in a tab that has opened where your Rscript was, in the top left-hand window of RStudio. There are three columns of data: first, the **treatment** (that is, whether boxes of seeds were uncovered or covered); second, the **water** level experienced by each box; and third, the number of seeds that **germinated** in a box. If you are particularly eagle-eyed, you might notice that there are only results for three boxes in the covered treatment with water level 5. This is because no result was recorded for the fourth box in the original data. This is not a problem for our boxplots, but we might want to mention this in our figure caption (see section 1.2 and Scientific Approach 1.2 for our advice on writing effective captions).

***Step 7*:** If we now open our script again (by selecting its tab at the top of the left-hand window in RStudio) we can now run the following sections of code that subset the two treatments and then plot the uncovered and covered seed data onto a grouped boxplot:

**uncovered <- subset(seeds, treatment=="uncovered")**

**covered <- subset(seeds, treatment=="covered")**

**boxplot(uncovered$germinated~uncovered$water, ylab = "Seeds germinating per box",**

 **xlab = "Amount of water",ylim=c(0,100), yaxs = "i",boxwex=0.27,at = 1:5 - 0.15,**

 **xlim=c(0.5,5.5),xaxt="n",col="cornflowerblue")**

**par(new=TRUE)**

**boxplot(covered$germinated~covered$water, ylab="",xlab="",**

 **ylim=c(0,100), yaxs = "i", boxwex=0.27,at = 1:5 + 0.15,xlim=c(0.5,5.5),**

 **xaxt="n",yaxt="n",col="forestgreen")**

**axis(side=1, at=1:5, labels=c(1:5))**

**legend("topleft", inset=.02, bty="n", c("Uncovered","Covered"),**

 **fill=c("cornflowerblue","forestgreen"))**

Again, do not worry about the details of the code for these boxplots (though see chapter 5 if you are interested in producing grouped multiple-samples boxplots). What we want to highlight here is the addition of a legend in the resulting figure (Figure S1.4). Because we are now comparing data from the two experiments (uncovered and covered boxes), colour is necessary to distinguish the separate results and a legend is an essential component of the figure (i.e. it is not just ‘chart junk’—see section 1.7 for further advice on keeping figures uncluttered).



Figure S1.4: The number of seeds that germinated in uncovered (blue) and covered (green) boxes experiencing different levels of watering (n=4 for all except covered boxes with a watering level of 5, for which n=3). Adapted from data from Chatfield (1982).

Thinking about colour again (see section 1.6 for further discussion on colour choices): Figure S1.4 has distinct colours (green and blue) to distinguish the results of the two experiments because this difference concerns a qualitative feature (whether boxes were uncovered or covered). However, it is possible that green and blue might be quite difficult for some viewers to distinguish. Therefore, we might want to think about how we could make our colour choices more accessible. One way we could do this would be to use a package that contains colour palettes that are colour-blind friendly.

***Step 8*:** To make the colour choices of our boxplot more accessible, we are going to use the package ‘viridis’. To use viridis, we also need to have the package ‘viridisLite’ installed (Garnier et al. 2021). To install packages, we can substitute the package names into the code from the script:

**install.packages("viridisLite")**

**install.packages("viridis")**

Alternatively, you can select the ‘Packages’ tab of the bottom right-hand window in RStudio and follow the steps from the ‘Install’ button (see the Additional Guide 1.1 R Basics for a recap of both approaches). Whichever method you follow, once the packages are installed, they can be easily activated by running the code below in the Console:

**library(viridisLite)**

**library(viridis)**

***Step 9*:** Now that we have viridis up and running, we can use it to generate colours for our figure that will be easy for viewers to distinguish. To do this, we’ll create a list called ‘**colourchoice**’ that will consist of two distinct colours generated by viridis. Run the code from the script:

**colourchoice<-viridis(2)**

This tells R to take the first two colours (because we have specified ‘2’) viridis provides in its colour-blind accessible palette and collect them as a list called ‘**colourchoice**’. If we ask R to show us

**colourchoice**

two values preceded by # symbols are listed—these are codes for the two colours.

***Step 10*:** To get R to use these colour codes in our boxplot (where we previously typed the names of colours we selected), we could now either paste in those colour codes or tell R to extract the relevant colours from our list ‘**colourchoice**’. We do this latter option by using [1] and [2] respectively to identify the first and second colours listed. See how this is done in the following code from the R script and have a look at how distinct the colours viridis has provided are (in hue and shade) in the resulting figure:

 **boxplot(uncovered$germinated~uncovered$water, ylab = "Seeds germinating per box",**

 **xlab = "Amount of water",ylim=c(0,100), yaxs = "i",boxwex=0.27,at = 1:5 - 0.15,**

 **xlim=c(0.5,5.5),xaxt="n",**

 **col=colourchoice[1])**

**par(new=TRUE)**

**boxplot(covered$germinated~covered$water, ylab="",xlab="",**

 **ylim=c(0,100), yaxs = "i", boxwex=0.27,at = 1:5 + 0.15,xlim=c(0.5,5.5),**

 **xaxt="n",yaxt="n",**

 **col=colourchoice[2])**

**axis(side=1, at=1:5, labels=c(1:5))**

**legend("topleft", inset=.02, bty="n", c("Uncovered","Covered"),**

 **fill=colourchoice)**



Figure S1.5: As Figure S1.4, but with a more accessible colour palette.

***Step 11*:** We now have a clear grouped boxplot, displaying the results of both experiments using an effective and accessible colour palette. To save this figure, we just need to click on the ‘Export’ button near the top of the plot window in RStudio, select ‘Save as Image’, choose an image type and give the plot a name, and then choose where in our computer’s directory we want to save it (see the Additional Guide 1.1 R Basics for more on saving plots).

***Step 12*:** We will also want to save the R script, in case we want to come back to any of this code in the future or send it on to someone else to use in R (see the Additional Guide 1.1 R Basics for more on using and saving R scripts). If you have edited the script at all, you can either save the latest version of the script over the original version (by clicking on the floppy disk ‘Save’ icon near the top of the script window) or save your edited script as a separate file to the original (by going to ‘File’ on the main task bar and then selecting ‘Save as…’ from the drop-down menu). If you choose to save a script as a new edited version, remember to give it a new name when you are saving it, to prevent overwriting the original.

And that’s it! You have now used RStudio to: enter simple data, create plots, add colour, edit colour, load in a data set, install and activate packages, save a plot and save an R script. Sticking to the concepts of good colour choice and ‘simple but not too simple’ figure design will help you produce effective figures, no matter what type of data you have, and following some useful R tips (like those we provide in Bigger Picture 1.1) will help the process of coding run smoothly.

We finish with a small disclaimer. Ordinarily we would not recommend making boxplots for only four discrete values (see our advice in sections 4.1 and 5.1 on treating data with seven or fewer different values as qualitative data instead). Here we just wanted to show you a familiar-looking plot, without any danger of you feeling overwhelmed by the volume of data.

**Reference**

CHATFIELD, C. 1982. Teaching a Course in Applied Statistics. *Journal of the Royal Statistical Society. Series C (Applied Statistics),* 31, 272–289.

GARNIER, S., ROSS, N., RUDIS, R., CAMARGO, A. P., SCIANI, M. & SCHERER, C. 2021. Rvision - Colorblind-Friendly Color Maps for R. R package version 0.6.0.