Tutorial 3: Graphics and Exploratory Data Analysis in R

Jason Pienaar and Tom Miller

Getting to know the data

An important first step before performing any kind of statistical analysis is to familiarize oneself with the data at hand (this is often called exploratory data analysis). This usually involves graphing the variables in various distributional displays (histograms, box plots etc.) and plotting the relationships between variables (scatter plots, box plots etc). The focus of this tutorial is therefore on how to use some graphical and other tools in R that are generally useful in preliminary graphic data analyses. Exploratory analysis also helps us decide on whether or not to transform a variable to meet some statistical requirements. In this tutorial we will cover some of the basic plotting functions that are built into the R system. In particular, we will cover histograms, boxplots, and scatterplots (with linear regression).

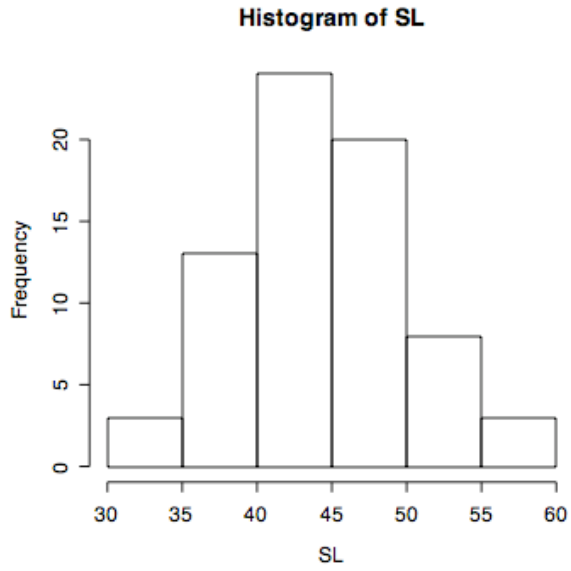
Basic plotting functions in R

Before starting, create a data frame named *fish* of the data contained in eg1.txt (this file should be available to you along with this tutorial). The data in this example (kindly provided by Joe Travis) are derived from a larger study of character variation in sailfin mollies (*Poecilia latipinna*). Specifically, the data consists of selected measurements on females from three populations (identified by the POP variable, populations 1, 2 and 5). We can ignore the column headed IDNO: this matches the line in the example data set to a record in a lab notebook where the data was originally recorded. The actual data recorded includes several measurements taken on individual fish: RAYNO, a discrete variable for the number of rays (finger-like bones) in the dorsal fin; SL, a quantitative variable of the body lengths of the females in millimeters; FINAREA, a quantitative variable containing measurements for the area of the dorsal fin in square millimeters (a function of the fin's length, height and shape), and TAREA, a quantitative variable of the tail-fin area in square millimeters (a function of the tail's length height and shape).

The Histogram in R

The most common way to examine the distribution of a **quantitative (continuous) variable** is by means of a histogram. A histogram dissects the range of the variable into equal-width class intervals called bins and then plots the number of observations falling in each bin as a bar chart (i.e. the height of the bar represents the number, proportion or percentage of observations in that class). To plot histograms of the variables SL, FINAREA and TAREA, use the *hist* command as follows (remember to *attach* the fish data frame first to make the variables visible to R):

```
>hist(SL)
```



For histograms of the other two variables simply replace the input in parenthesis by the variable names (note that R is case-sensitive). These data come from three different populations; can we plot histograms separately for each population? Recall that POP is a **factor** identifying which population each measurement came from. Therefore we can simply specify which variable and from which population to plot using the population variable as follows:

>*hist*(SL[POP==1])
>*hist*(FINAREA[POP==5])

Note the double equals sign is used for specifying the population: a single equals sign is used to change the actual values stored in POP, so don't do this. In addition, there cannot be a space between the equal signs. Often a **factor** such as the POP variable will be specified by a name (i.e., a text string) rather than a number. In this case, if we want to specify the **level** of the factor we place it inside inverted commas (Note: a factor is just another name for a **categorical variable** where the different classes of the variable are known as the levels of the factor). For example if the POP factor contained the entries U.S.A, Mexico, Canada etc, we could specify the USA level with:

>*hist*(SL[POP=="U.S.A"])

Note also how R automatically creates the axes labels. To modify an axis label we set the properties *xlab* or *ylab* (or both) within the histogram command, separated from the other function arguments by commas. The axis title is given between inverted commas. Thus

>*hist*(SL[POP==1], *xlab* = " body length for population 1")

produces a histogram of SL for population 1, where the X axis is labeled with the text

between the inverted commas. The *main* property is used to give a main heading to your graph. So to give the graph above the heading "Size" we would type:

> *hist*(SL[POP == 1], *xlab* = " body length for population 1", main="Size")

Type ?*hist* or help(*hist*) to find out about more options on how to customize your histogram. Note that there are quite a few options concerning bar color, number of bins to use etc. The available colors are encoded by numbers or you can try some color names in quotes (e.g., col = "blue"). Try different colors for your histogram by setting the *col* property. Eg:

> *hist*(SL[POP == 1], *xlab* = " body length for population 1", main="Size", col=2)

should change your bars to red. If you prefer shading lines rather then a full color for you histogram bars then set the density function:

> *hist*(SL[POP == 1], *xlab* = " body length for population 1", main="Size", col=2,
density = 5)

Try some different density values to see what happens (you can go up to pretty high numbers (50+) to get a shading effect).

The Box Plot in R

The box plot is another useful way to examine the distribution of a variable. A box plot consists of a box on a set of axes where the top and bottom lines of the box represent the upper and lower **quartiles** respectively (see the example in the figure below). A horizontal line within the box shows the **median**. The length of the box is known as the **inter-quartile range**. Box plots also include vertical lines (called whiskers), extending from the top and bottom of the box up to the largest and smallest observations respectively, that fall within 1.5 inter-quartile ranges. All observations that fall beyond these limits are plotted individually as points.

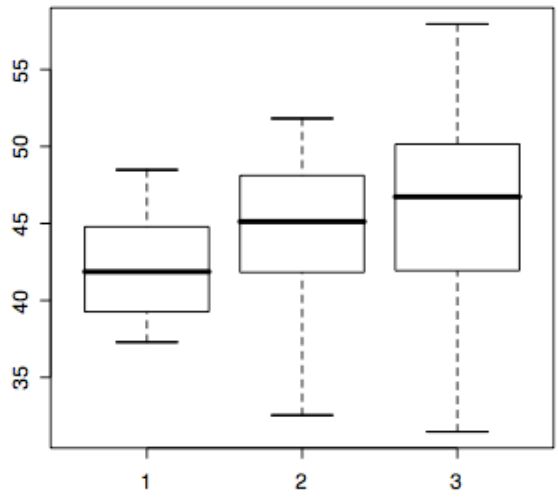
Statisticians really like boxplots because a number of distributional properties are easily visualized. Box plots show a measure of **location** (the median line), **dispersion** (the length of the box and distance between the upper and lower whiskers), **skewness** (asymmetry of the upper and lower portions of the box and asymmetry of the whisker lengths), and long drawn out tails (for example whisker length in relation to the length of the box). Box plots are particularly useful as a quick visual comparison between 2 or more samples. To create a box plot in R use the command *boxplot*(variable), where "variable" is the name of the variable you want to plot. Adding labels to the axes is done in exactly the same way as for *hist*, as is changing the color (use *?boxplot* to check all the options, some differ from the *hist* function). A very useful feature of the *boxplot* function is the ability to plot a number of box plots of different variables, parallel to each other on

the same plot. Thus

>*boxplot*(SL[POP==1])

Creates a box plot of the SL measurements from population 1 whereas:

>*boxplot*(SL[POP==1], SL[POP== 2], SL[POP==5])



creates parallel box plots of the SL variable from all three populations. Note here that you can see that both the median (solid bar in box) and range (whiskers) increase as we go from population 1 to 5.

By the way, there is a short-cut that you can use to see all boxplots for all the subclasses of a variable:

>boxplot(SL~POP)

Neat, eh? Just use the tilda (~). This works for a lot of the graphics, but not generally for other functions.

If you are the nonvisual type and prefer text output rather than a graph the function *summary* could help you out. *Summary* returns the minimum and maximum values, the 1st and 3rd quartiles, the mean and the median of a given vector. For example, to return these values for RAYNO from population 5 type use:

>*summary*(RAYNO[POP= =5])

which returns:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
14.00	15.00	15.00	15.04	15.25	16.00

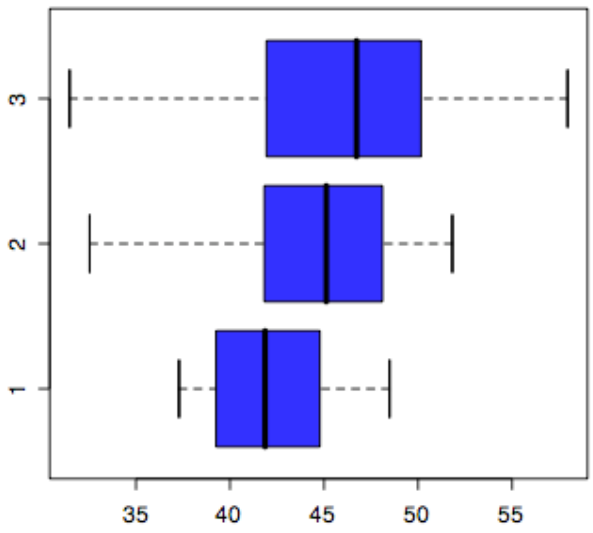
Whereas:

>*summary*(fish)

would return the same statistics for all the variables in the data frame "fish" (not by population however).

There are many other ways of customizing your boxplot (as with any plot in R), but that is something you can check for yourself using the help functions or user manuals. As a final example, let's say we want to plot some blue boxplots of the SL variable on top of each other rather than next to each other, then we would type:

>*boxplot*(SL [POP==1], SL[POP==2], SL[POP==5], *col*=4, *horizontal* = TRUE)



All these parameters can be found in the help(boxplot) information.

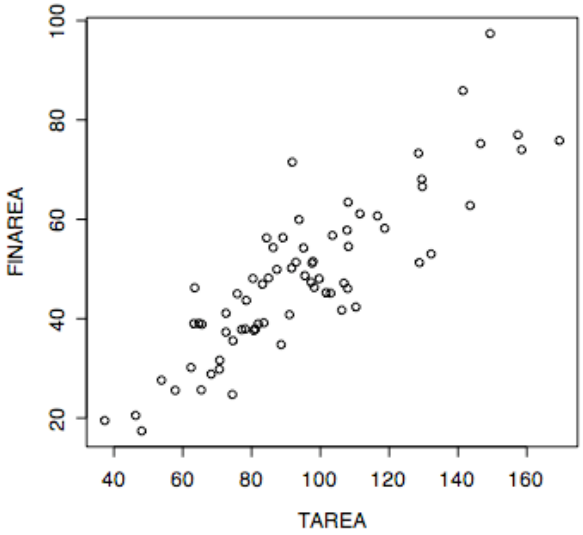
The Scatter Plot in R

The scatter plot is the standard graph for examining the relationship between two quantitative variables. The *plot* function is used to create scatter plots (amongst other things), where two numeric vectors are required as arguments to the function. For example, to create a simple scatter plot of FINAREA against TAREA type:

>*plot*(FINAREA, TAREA)

R interprets the first vector as the horizontal coordinates and the second as the vertical coordinates. An alternative means to plotting the relationship is by specifying a simple linear model in the function arguments. This is done with the tilde symbol (~):

>*plot*(FINAREA~TAREA)



Note that using the tilda switches the axes: the **response or dependent** variable (FINAREA) is now plotted on the vertical (y) axis whereas the **predictor or independent** variable (TAREA) is on the horizontal (x) axis.

Once again *xlab* and *ylab* can be used to add labels to the axes (most of the plotting functions use roughly same arguments to modify their attributes). Interpretation of a scatter plot is often assisted by enhancing the plot with least squares or non-parametric regression lines. As such lines are often dependent on the particular statistical model we are applying we won't go into too much depth here, but just to see what I mean, let's add a simple **linear regression** line to the plot above. The function *lm* can be used to specify a **linear model** and *abline* to add a line of the slope-intercept form. So to add a simple linear regression line to the plot above we type:

>*abline*(*lm*(FINAREA~TAREA))

Note: the graph has to already be there, so if you have closed the graphics panel, first replot the variables. As we are busy with graphics, we might as well discuss some ways to modify the line (i.e. width, type and color). The attributes *lwd*, *lty*, and *col* are used to specify the line width, line type and line color respectively. First, close the graphics window and plot FINAREA against TAREA again (this is because we are going to add a line to the graph but at the moment, we are adding a new line, not modifying the existing

one, so it will be difficult to see against the one already plotted). Next, type:

> *abline*(*lm*(FINAREA~TAREA), *lwd* = 10, *lty* = 2, *col*=3)

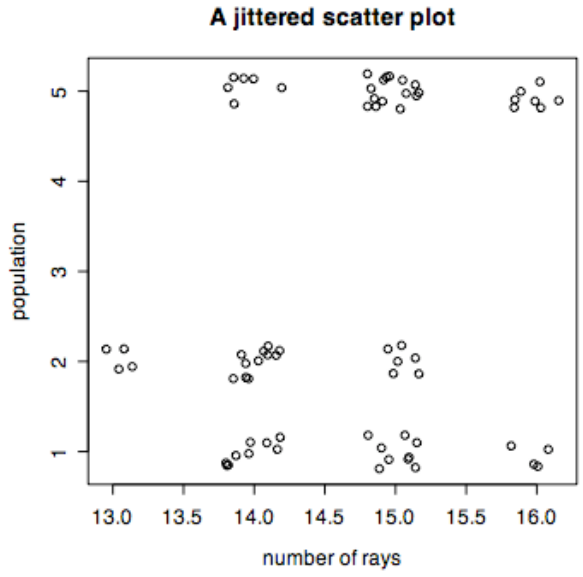
Wow, didn't expect that, did you? Repeat this a few times with different values for the line parameters to see what's available.

Often scatter plots include a number of overlapping points (this is especially true when we are plotting **discrete** variables, as they can only take so many values). For example create a scatter plot of RAYNO against POP. You will notice that only nine points are shown on the plot. This is because a number of these points are overlapping (i.e. POP can only take on three different values and RAYNO 4, so there is bound to be a lot of overlap). In these situations, the *jitter* function comes in handy. The *jitter* function adds a small random quantity to the data coordinates thus serving to separate the overplotted points. Try the following:

>*plot*(*jitter*(RAYNO), POP)
>*plot*(*jitter*(RAYNO), *jitter*(POP))

And, finally:

>*plot*(*jitter*(RAYNO), *jitter*(POP), *xlab*="number of rays", *ylab*="population", *main*="A jittered scatter plot")



Multiple Figures on One Plot

You will have noticed by now that each time a plotting function is used, the previous graph is replaced by the new one. Although you can copy and paste the graphs into some

other program in order to compare them, it might be useful to plot them simultaneously in R in one panel. One way of doing this is to use the *split.screen* function (as per usual, there are many other ways, but we will start with this one). Let's say that we want to plot histograms and box plots of the SL variable for the three different populations, all on one screen. We could begin by using *split.screen* to create 6 separate areas in the graphics panel as follows:

```
>split.screen(figs=c(2, 3))
```

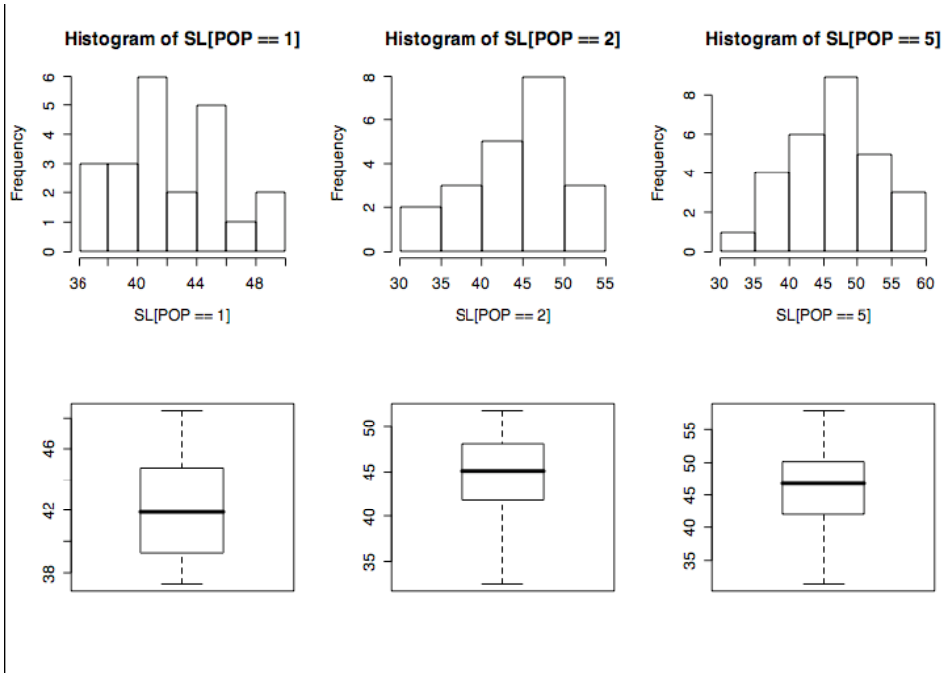
The function above splits the graphics panel into a matrix of 2 rows and 3 columns where the screens are numbered 1-6 by rows (the original graphics surface is now called screen 0). Next we specify which screen we want to plot in. for example to plot a histogram of the SL variable from population 1 in the first screen we use the following two commands:

```
>screen(1)
>hist(SL[POP= =1])
```

We can now plot the rest of the graphs on the screen by specifying which screen to use each time, then typing the actual plotting function. E.g.:

```
>screen(2)
>hist(SL[POP==2])
….
>screen(4)
>boxplot(SL[POP== 1])
```

etc.



Split.screen mode is terminated by using the function *close.screen*(all=TRUE) or simply by closing the graphics window. One consideration when plotting multiple plots on the same screen is the size of the text labels and headings. You might have noticed that the text is a little too big and ungainly and that no journal would accept such figures for publication. To change the size of the axis labels use the *cex.axis* attribute (this is just one of those things that is not so intuitive and often needs to be looked up in some reference manual). Try:

```
>plot(FINAREA, TAREA, cex.axis=0.5)
```

In a similar fashion, *cex.lab* and *cex.main* change the text size of your axes labels and graph heading respectively.

Finally, another way to get multiple graphs on the same figure is to first use the *par* function that allows you to set graphical parameters. If we simply say:

```
> par(mfrow=c(1,3))
> hist(SL[POP==1])
> hist(SL[POP==2])
> hist(SL[POP==5])
```

You will get the histograms found at the top of the last graph. Using the par(mfrow=c(1,3)), we told R to set up a graphics space for three figures in one row. Then each time we asked for a graph, it created the graph in the next available graphic space. This can be a little bit easier way to draw things, but doesn't use as precise control over where each figure is created.

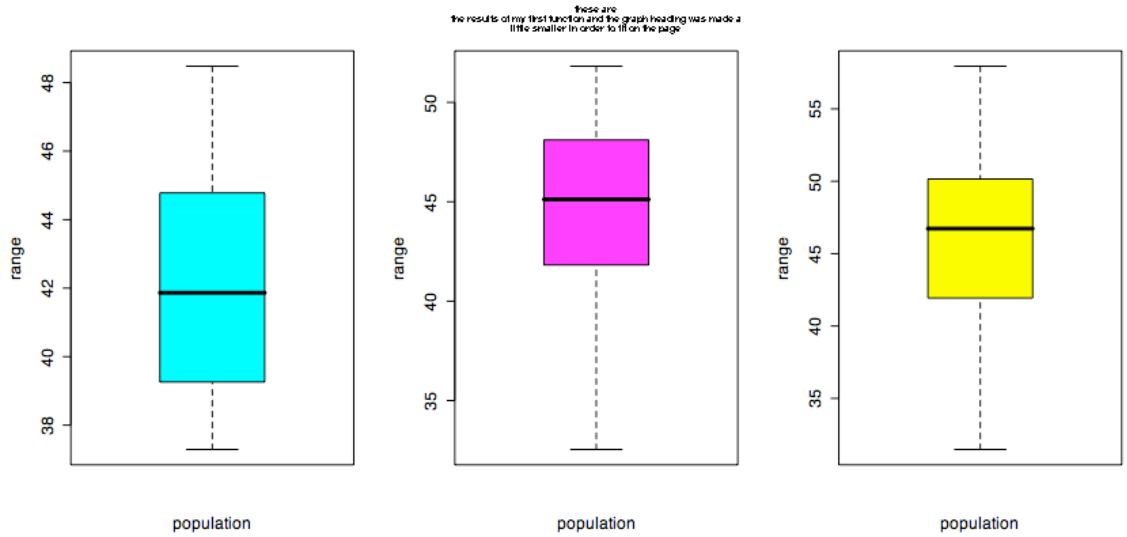
Exercises

1. Create parallel box plots of a given variable for the three different populations in the fish data set. Add the labels "population" to the x axes and "range" to the y axes. Give the boxes in each plot different colors. In addition give the graph a title named "these are the results of my first function and the graph heading was made a little smaller in order to fit on the page". In other words, change the size of the heading so that it fits in the graphics panel.

2. Now, this time use the par(mfrow=(c(2,3)) format. Plot the graph above, as well as 3 different scatter plots of TAREA vs. FINAREA (for the three different populations) with appropriate axes labels and headings, all on the same graphics panel. Add a linear regression line showing the relationship between the points in each graph and use a different color for the points and lines in each of your three scatter plots.

Answers:

1.
```
>split.screen(fig=c(1,3));
>screen(1);
>boxplot(data1,xlab="population",ylab="range",col=5);
>screen(2);
>boxplot(data2,xlab="population",ylab="range",main="these are the
results of my first function and the graph heading was made a
little smaller in order to fit on the page",cex.main=0.6,col=6);
>screen(3);
>boxplot(data3,xlab="population",ylab="range",col=7);
>close.screen(all=T);
```



2.
```
>par(mfrow=c(2,3));
>boxplot(SL[POP==1],xlab="population",ylab="range",col=4);
>boxplot(SL[POP==2],xlab="population",ylab="range",col=5);
>boxplot(SL[POP==5],xlab="population",ylab="range",col=6);
>plot(TAREA[POP==1],FINAREA[POP==1],col=4);
>abline(lm(FINAREA[POP==1]~TAREA[POP==1]),col=4)
>plot(TAREA[POP==2],FINAREA[POP==2],col=5);
>abline(lm(FINAREA[POP==2]~TAREA[POP==2]),col=5)
>plot(TAREA[POP==5],FINAREA[POP==5],col=6);
>abline(lm(FINAREA[POP==5]~TAREA[POP==5]),col=6)
>close.screen(all=T);
```