# Doing Graphics for Assignment 1

In assignment 1, your goal is to do some simple descriptive statistics about the dataset (to characterize some of its main features) which should include some simple graphics. You should limit graphics really to barplots, piecharts and histograms. As a rule of thumb, you can say:

|  |  |
| --- | --- |
| Chart Type | Best For |
| Histogram | Shape of Distributions |
| Barplot | Magnitudes |
| Piechart | Proportions |

When you run any of the functions for such graphics in R Studio ( hist(), piechart(), barplot() ), the graphics are immediately shown in the display area of R Studio. If however you want to save them as graphics files (png for instance) for incorporation in your assignment document, you should enclose your graph commands between

|  |
| --- |
| png("yourfilename.png") #this opens the file for writing # here put the commands used to generate the graph  dev.off()# this closes the file after writing |

I wont mention these in again in the rest of the tutorial to avoid repetition.

Lets look again at the student survey results again. Place the spreadsheet and the file practical\_3\_starter.R into a new folder, then in R Studio, choose new project, and create it from an existing directory (the folder you created).

In this script, all the column renaming, and numeric forcing has already been done. Make sure you run it first by clicking on the "source" button when you have loaded it into the editor.

## Basic Histogram

A basic histogram is good for distributions. Its main use is on numerical data (its not so good with counts). An easy one, which shouldnt cause any surprises is that of height. Since peoples height is known to be normally distributed it shouldnt cause any surprises

Run this command:

|  |  |
| --- | --- |
| hist(df$height\_cm)  #As you see, it is informative but very basic. It does not have a proper title, nor proper labels. Moreover, the division into bands (10cm) is a bit broad – and maybe smaller bands would be better. Also, we might wish to filter the data, for example showing the heights of women only, or men only, or both together |  |

The principal options you can add to achieve these ends are:

|  |  |
| --- | --- |
| number of bands | breaks=10 |
| title | main="Histogram of student heights" |
| Label for the x axis | xlab="Height in bands of x cm" |
| Label for the y axis | ylab="Frequency" |
| Filtering the data (e.g for women only) | df$height\_cm[df$gender=="Woman"] |
| Colouring the bands | col = "steelblue" |
| Setting limits (optional) (be careful here, this will hide anyone below 140 – this might be useful to remove outliers) but you should mention this in the graphic description | Xlim=c(140,180) |

|  |  |
| --- | --- |
| hist(df$height\_cm[df$gender=="Woman"],breaks=10,  xlab="Height in bands of 5cm",  ylab="Frequency",  main="Female height on 7COM1079",  col = "steelblue"  ) |  |

Now you know the basic modifications you can do, look in your own dataset and see what you can do.

However, you might find that you have distributions that are a bit meaningless because of massive outliers.

For instance

|  |  |
| --- | --- |
| hist( df$miles\_from\_campus,  breaks = 20,  main = "Histogram of Miles from Campus ( < 100 miles )",  xlab = "Miles from campus",  ylab = "Frequency",  col = "lightblue"  ) |  |

Clearly, because some strange values were added (maybe distance from Hatfield from locations in India), it makes the histogram less meaningful. To resolve this, just filter the data (including only responses where the number was less than 100):

|  |  |
| --- | --- |
| hist( df$miles\_from\_campus[df$miles\_from\_campus<100],  breaks = 20,  main = "Histogram of Miles from Campus ( < 100 miles )",  xlab = "Miles from campus",  ylab = "Frequency",  col = "lightblue"  ) |  |

If you do such filterings, make sure it is clear in the graphic that filters have been applied. The filtering techniques shown above can be used also in pie charts and barplots, but I will assume you know enough to be able to do that.

## Basic PieChart

Pie charts are great for proportions. You break a circle into different segments based on the percentages of the whole each item constitutes.

|  |  |
| --- | --- |
| pie(  table(df$award),  main = "Distribution of Awards",  col = rainbow(length(unique(df$award)))  )  #here we have just a simple table as the data to be rendered. A title (*main*), and then a parameter *col* – meaning colors, which is basically an array of colors, generated by the function *rainbow*() |  |

To be honest, this is quite nice, and tells us alot. However as well as giving us the award, it might be good to also see the number of students each slice of the pie represents. To do that we will have to include a *labels* parameter to override the default

|  |  |
| --- | --- |
| pie(  table(df$award),  main = "Distribution of Awards",  col = rainbow(length(unique(df$award))),  labels = paste(names(table(df$award)), "\n", table(df$award)),  ) |  |

Lets look at that labels line a bit more clearly

|  |  |
| --- | --- |
| labels = paste(names(table(df$award)), "\n", table(df$award)) | |
| paste | A string concatenation command which takes an array of items and joins them with a space between each item. The equivalent in most languages is .join() |
| names | This gets the respective *name* from the table |
| \n | A newline |
| table | This gets the respective *value* from the table |
| And when used in this context, it means that for every "name" in the table (in this case names of awards), append to it a newline, and then the number of times it came up in the column | |

To show percentages instead, like last week we have to use the prop.table() function. The code below is really the same as the above except as well as concatenating a number for each slice, we are also multiplying that number by 100, adding the character "%" at the end and specifying separator is an empty quote (meaning no separator – by default it is a space)

|  |  |
| --- | --- |
| pie(  table(df$award),  main = "Distribution of Awards",  col = rainbow(length(unique(df$award))),  labels = paste(  names(table(df$award)),  "\n",  round(prop.table(table(df$award)) \* 100, 1), "%",  sep = ""  ),  ) |  |

## Basic Barplot

Sometimes when the difference between proportions are not so big, it might make it easier to read as a barplot

|  |  |
| --- | --- |
| If we do just a basic barplot barplot(table(df$award)) we get something almost unreadable. The labels for the bars are not aligned, some of them are missing completely. Also this should require a title.  We can easily add a title, and we should probably force the y axis to go to 120, but how do we make the labels visible, because the amount of text will always be outside the space given by the bars? |  |

There are two ways we could solve this. Firstly, lets just make the text flow vertically for the labels, and also place numbers above the bars. This is slightly more involved.

|  |  |
| --- | --- |
| award\_counts <- table(df$award)  cols <- rainbow(length(award\_counts))  bp <- barplot(  award\_counts,  col = cols,  main = "Distribution of Awards",  xlab = "Award",  ylab = "Count",  **las = 2,**  cex.names = 0.8,  ylim = c(0, 120)  )  # place counts above bars, ensure labels stay within plot area  text(bp, award\_counts, labels = award\_counts, pos = 3, cex = 0.8) |  |

This is an improvement, even though the labels are not fully visible.

Note the changes made here

|  |  |
| --- | --- |
| **las = 2** | Makes the text vertical |
| cex.names = 0.8 | scales the label font to 80% |
| ylim = c(0, 120) | Make the minimum value 0 and the maximum value on the y axis 120 (higher than any of the individual values being plotted) |
| text(bp, award\_counts, labels = award\_counts, pos = 3, cex = 0.8) | This adds the values for each category, and positions them across the top of their respective bar pos=3 makes the text flow horizontally |

To get round the problem with the labels being too big, we could instead use a legend like this

|  |  |
| --- | --- |
| award\_counts <- table(df$award)  cols <- rainbow(length(award\_counts))  bp <- barplot(  award\_counts,  names.arg = rep("", length(award\_counts)),  col = cols,  main = "Distribution of Awards",  xlab = "Award",  ylab = "Count",  ylim = c(0, 120)  )  text(bp, award\_counts, labels = award\_counts, pos = 3, cex = 0.8)  legend(  "topright",  legend = names(award\_counts),  fill = cols,  bty = "o",  bg = "white",  box.col = "black",  cex = 0.9  ) |  |

|  |  |
| --- | --- |
| names.arg = rep("", length(award\_counts)) | This basically replaces all the labels with empty strings |
| legend = names(award\_counts), | This is what appears as words in the legend |
| fill = cols, | These are the colors (from the variable *cols* previously used to specify the color for each bar) |
|  |  |

Barplots are not only possible with counts. You also do them on numerical values such as averages.

As you can see, there are lots of customization options in R Base Graphics. My advice at this point is to find something interesting in the dataset you were allocated, and try to make a graphic from it. If you are having any problems ask the tutors, or ask your question on Slack