**Exercise 1**

We are presented with an output file of an individual-based simulation of which we want to make a graph. This process should be easy to scale up and process many output files.

Making a bar plot of one simulation output using polygon.

Remember that polygon takes a vector for x-values and their y-values polygon(c(x1, x2, x3 ….), c(y1, y2, y3, ….), ….)

1. read the file randomMating.csv into R. We start with the 20 columns with “BDM\_..” and we are interested in the last row of these columns.
2. Make a vector containing these 20 BDM values of interest.
3. Make an empty plotting region. Think how long the x-axis (xlim) and y-axis (ylim) should be (how do you get the max value from an vector?).
4. Use a for loop to go through all 20 categories and plot a bar for each. Think ahead as we further on may want to offset these bar a little to add a second set of bars.
5. Fill the bars with a semitransparent colour.

There is another category of interest, namely hybrids. Select the last row of the columns with header ‘hybrid…’ and assign it to a vector

1. Add the hybrids to the bar plot. Some method, but now slightly offset to the right of the BDM bars.
2. Add a horizontal line at y = 0.1 of a colour of your choice. (Hint: abline())
3. Save the figure as a pdf to a folder names “figures” in your working dir.

OPTIONAL: Let’s scale this up.

1. There are more files, similar set-up in terms of variables, different file name (and actual numbers hopefully). Adjust the code so you can loop through all five files and make the above barplot for each. (Function tips: list.files(), paste())

**Exercise 2**

Let’s embrace some multi-panel action. The files fitness.landscape.csv contains a matrix where each column contains a value for  (to become y variable) and each row for (to become x variable).

1. Make a heat plot of the matrix using image(). Note that you will need to make your own vectors for the x and y (Hint: seq()). Suppress the x and y labels for now.
2. Use heat.colors() for the colours.
3. Make the x and y labels. We like fancy Greek letters so use expression() to make them. And don’t forget a proper subscript.

This is all nice looking but let’s help the reader to visualize the pattern by adding two panels to the side, each showing a line graph at = 0.1 and 0.3.

1. Use fig() to make the side panels. Use the right axis suppression to make the figures look clean.
2. Add panel numbers to all three panels. Think of a way that it will remain in the same relative location if you change the length of the axis.

OPTIONAL: if you feel up for the challenge, make a custom legend. You can use a polygon to do this and fill this with the heat.colors gradient. As you want to plot this outside the normal plot region, use xpd = TRUE to allow this. Don't forget to add the min and max value to the legend.