Comments and Hints
Assignment #4

General Comments

Here is something for starters. As mentioned, this may be updated later on.

- When working with *data frames* or *lists*, take advantage of the `with` or `attach` functions, or, at a more basic level, take advantage of the `$` operator.

FAQ's & Hints

You can, but do not have to, use a for-loop in problems 1 and 2.

Problem 1

The key is to extract the relevant data.

You can take advantage of the `attach` function, and then extract gender data using something like `age[gender == "male"]` and `score[gender == "male"]`

Or you can use the `with` function and extract gender specific data using something like

```
males <- with(bubbaStudy,
    data.frame(age = age[gender == "male"], score = score[gender == "male"])
```

This gives one set of data to plot.

Problem 2

When you get the contingency table, quickest way is to use a 3-dimensional table (3-way), extract the 1-dimensional tables for each case (e.g. project males, project females, etc.). Then `sort` can be used when you apply the `barplot` function.

Problem 3

I used the `split.screen` function, but the `layout` function has much more power and seems to work a lot easier. Try them both out if time permits – see pp. 943-946 of the R Book.

For a look at the `layout` function, on p. 943 of the R Book code equivalent to

```
win.graph(width = 6, height = 6)
par(ps = 10, cex = 0.75)
nf <- layout(mat = matrix(data = c(2, 0, 1, 3), nrow = 2, ncol = 2,
    byrow = TRUE), width = c(3, 1), height = c(1, 3), respect = TRUE)
layout.show(nf)
```

configures a plotting window, then displays the configuration. Read through the relevant pages in the R Book and also run `?layout`, then play around with the above code to get a feel for what it does. For example, try running the above code, but with `respect = FALSE`. You might have to look very closely to notice a difference between the two configurations.

Now try playing around with the `mat` argument – note that the numbers in the assigned matrix identify the window cells and the corresponding plots, by order of construction, that are placed in them. For example, try using

```
mat = matrix(data = c(1, 0, 0, 2), nrow = 2, ncol = 2, byrow = TRUE)
```

instead of what is given. Now try,

```
mat = matrix(data = c(1, 1, 2, 3), nrow = 2, ncol = 2, byrow = TRUE)
```

and also

```
mat = matrix(data = c(0, 1, 2, 0), nrow = 2, ncol = 2, byrow = TRUE)
```

As you will notice, the `mat` argument partitions the window into cells, and the numbers placed into the matrix assigned to the `mat` argument identify which plot is to be placed into which screen or sub-window(s).

Similarly, play around with the `width` and `height` arguments – pay close attention to the lengths of the vectors assigned to these arguments, they need to match the number of rows and the number of columns in the matrix assigned to the `mat` argument.

Remember, *these functions* (`split.screen` and `layout`) *are alternatives* to the partitioning accomplished through the graphic parameter settings using `par(mfrow = c(m, n))` or `par(mfcol = c(m, n))`. *Do not use either the* `split.screen` *or* `layout` *functions in combination with the* `mfrow` or `mfcol` *methods.*