Class Exercise 5

We will be working with the graphing commands we studied in Exercise 4. This exercise will work with a user-specified function, advanced graphics, and iteration.

This is how we produced a pdf file of our graph in Exercise 4 (don't run these commands):

```
pdf("Barium.pdf")
Time.seq=1:36
smooth.Ba=loess(Ba.540~Time.seq,na.action=na.omit)
plot(Ba.540,xlab="Period",ylab="Concentration",axes=F)
title("Standardized Barium Concentrations (ppm)"
lines(1:36,predict(smooth.Ba,1:36))
axis(2)
axis(1,at=1:36,labels=F,tck=-0.02)
axis(1,at=seq(6,36,6),labels=names(Ba.517)[seq(6,36,6)],tck=-0.04,cex.axis=0.7)
dev.off()
```

The original study had 6 wells (labelled 1, 7, 8, 9, 10 and 11). Suppose we wanted to generate graphs for all 6 wells using a 2-column, 2-row format; this would print Wells 1, 7, 8 and 9 on one page, and Wells 10 and 11 on a second page of the same file. To do this, we need a few more variables in our workspace. Download the R workspace from the website again and examine the variables `well, dat, labdate, Ba`; you can see that observations from all 6 wells have been concatenated well-by-well. The variable `dat` takes the place of our simple sequence `1:36` in `lines`.

Take an especially close look at the function `tplot`–you can either type `tplot` or look at it using `fix(tplot)`. Note that it only has a single argument, the name of the analyte (e.g., `"Ba"`). The rest of the function is pretty confusing at first, but notice that there is a `for` loop– for Wells 1, 7, 8 and 9, 10, and 11.

I would like the program to use the input variable as both a character variable and a continuous variable, so the function initially uses the analyte name as character variable input. The `get` function then converts a character variable so that it can also be used as a variable name; that variable name can then be used in functions, such as `length`, `loess`, and `plot`. Type the following commands to see how `get` works.

```
"Ba"
Ba
get("Ba")
plot(get("Ba"))
```

The first set of commands in `tplot` creates the pdf file `ts.BA.1-11.pdf` for storing the graph by using the `paste` command. This command simply pastes together character variables–I don't want spaces, so I use `sep=""` to jam the character strings together. To understand

why we do not want spaces, type the following commands for an unsuccessful and a successful demonstration of `paste` to create a file name.

```
paste("ts.","Ba",".1-11.pdf")
paste("ts.","Ba",".1-11.pdf",sep="")
```

The filename doesn't specify the drive and directory–the pdf file will be stored wherever your placed the R workspace when you downloaded it. If you want to verify the directory to which the file will be sent, type:

```
getwd()
```

Alternatively, you can select **Get Working Directory** from one of the toolbar tabs in **R**. This will output your current work directory–change it if necessary (in some cases, you may not be permitted to print to the default work directory).

The remainder of the loop should look familiar; we cycle through Wells 1, 7, 8, 9, 10 and 11 in turn, computing a smoothed curve (`span=0.9` controls the amount of smoothing), plotting the data points, then over-laying the smoother, and then fixing the axes.

Run the function by typing `tplot("Ba")`; **R** will print a line in the GUI window indicating the active window upon completion of the commands in `tplot`. Note that the variables `well, dat, labdate` are used in the function, but not defined in the function; **R** will find them in the workspace. Leave **R** and confirm that the pdf file is in the appropriate directory. How do the smoothed curves look? Can you think of any other improvements to the plots?