Class Exercise 4

Don's coverage of graphics capabilities was quite thorough. In this exercise, I am going to demonstrate some additional capabilities of the graphics package. The example is drawn from a well-monitoring project that our statistical consulting center, the Stat Lab, helped analyze. As part of the Exploratory Data Analysis for this project, we plot analyte concentrations (in this case, periodic table elements like Zinc, Lead, Arsenic, etc) over time. New samples are collected every six months. Download the Exercise 4 R workspace from the website and look at the Barium concentrations for Well 1:

```
Ba.540
```

Note that we have a missing value. We will next look at commands to generate a scatterplot of the data for Well 1 and then add a smoothed trend curve; we have to use a smoothing function that can handle missing values, so there are a couple extra steps here. We are using methods here you may not be familiar with; `smooth.Ba` is the output object from the `loess()` function, in which we're "smoothing" the response data `Ba.540` as a function of `Time`. The `lines()` command actually creates the smoothed curve on the already-existing plot. Note that the ∼ symbol may not paste correctly into R–you may have to enter it by hand.

```
plot(Ba.540)
title("Standardized Barium Concentrations (ppm)")
Time.seq=1:36
smooth.Ba=loess(Ba.540∼Time.seq,na.action=na.omit)
lines(1:36,predict(smooth.Ba,1:36))
```

This graph is a little disappointing; e.g., note that it didn't use `names(Ba.540)` to generate an attractive set of x-axis labels. We are going to wipe out the current axes, and then add axis features in separate steps to generate more attractive labels. Be sure to pay attention to your plotting window as you complete each step; it's interesting to observe the features as they are added.

```
plot(Ba.540,xlab="Period",ylab="Concentration",axes=F)
title("Standardized Barium Concentrations (ppm)" )
lines(1:36,predict(smooth.Ba,1:36))
```

This looks a little odd right now. In the next set of commands, we first restore the vertical axis (the argument "2" in `axis(2)` refers to the vertical axis). Then we add minor tick marks to the horizontal axis ("1"), but without labels. Finally, we add labels and major tick marks to the horizontal axis; note that we use a subsetting option (`seq(6,36,6)`) to hand-select the labels for the major tick marks from the vector `names(Ba.540)`.

```
axis(2)
axis(1,at=1:36,labels=F,tck=-0.02)
axis(1,at=seq(6,36,6),labels=names(Ba.540)[seq(6,36,6)],tck=-0.04,cex.axis=0.7)
```

Sometimes we have to play around with the sequencing on the horizontal axis labels. If the labels are "crowded", we may not get the sequencing we asked for; here, we used `cex.axis` to shrink the labels so they would not be crowded. The labels refer to the semiannual data collection times–"1H05" represents data collected in the first half of 2005, for instance.

We can also save our graph as a pdf file. I know you're thinking that we could just right-click on the figure and choose "Save as", but what if we've embedded the graphing commands in a function or a loop? In that case, we will need to know about the following approach to saving files. It's rather straightforward; simply use the `pdf` command with a file specification, then enter your plotting commands. All commands appearing after the `pdf` statement will be executed, though you will observe no changes to your plotting window. Execution can be halted either by opening a new pdf file, or by "closing" the current file with the simple command `dev.off()`. Here is how I would store the graph I created above; you can change the file specification and save a copy in your chosen directory.

```
pdf("z://STAT 540//Barium.pdf")
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.540)[seq(6,36,6)],tck=-0.04,cex.axis=0.7)
dev.off()
```

If you want to see the actual file, you'll need to quit **R**. Inspect the file after leaving **R** and comment; are there any additional changes you might suggest? We can use a similar approach to save postscript files to embed in documents.