Class Exercise 9

This exercise is based upon Chapter 5 of Delwiche and Slaughter's "The Little SAS book". We will learn a little bit more about enhancing output using `ODS` and `PROC FORMAT` statements.

We have grown used to using the HTML format in the `Results` window to cut-and-paste tables to our class assignments. Sections 5.6, 5.7, 5.10 and 5.11 include some ideas for further enhancements of our HTML output. For this exercise, we will be making modifications to `PROC PRINT` and inspecting the modifications in our `HTML Results` window. Let's read in some stock data to get started. Import the Excel spreadsheet into SAS (name it `Stocks`) and then run the following commands for an initial look at it:

```
data stocks1;
set stocks;
percent_change=percent_change/100;
run;
proc print data=stocks1;
run;
```

Note: As noted before, **PROC IMPORT** may preserve blanks in the original Excel column names rather than replacing the blanks with underscores. If you want to reference a SAS variable with blanks in its name, embed the name in *single* quotes followed immediately by the letter *n*. E.g., if SAS imported `share_price` as `Share Price`, then the label statement below would start: `label 'Share Price'n="Share Price"`. Alternatively, you can use the same notation to immediately rename the variable as `share_price` or simply rename the variables in Excel before importing the data. I placed the following code after `set stocks;` to change variable names:

```
percent_change='Percent Change'n;
share_price='Share Price'n;
drop 'Percent Change'n 'Share Price'n;
```

We can first use some commands from previous chapters to improve the appearance of two of the variable names, remove observation numbering, and reformat `percent_change`. I can't say I'm entirely happy with the format I used for `percent_change`, but it will do:

```
proc print data=stocks1 label noobs;
label share_price="Share Price" percent_change="Percent Change";
format Percent_Change percent8.2;
run;
```

Similar to the Chart windows in Excel, SAS's default HTML window may have a gray background; I often prefer a white background. SAS can use the `style` option to change many elements of printed texts and cells (see pp. 164-165 for options). I first changed the table (or data) background to white, but then decided I wanted to change the header background to white as well. After inspecting these changes, I changed the header font to green. In order to change both header and data backgrounds to white, you could list them within the `style` command and apply the same style to them, but you would need to re-set the header background to white when you change the header font color to green, so it's actually inefficient to try to change both backgrounds in one step. Instead, I used one style statement to change the data's background, and I used a second `style` command to change the font and background for the header. *Note:* Some students may need to use `foreground=green` rather than `color=green` to change the header font. While I was at it, I also included a title with a green font (see section 5.6). Run the commands below and comment.

```
proc print data=stocks1 label noobs style(data)={background=white}
style(header)={color=green background=white};
label share_price="Share Price" percent_change="Percent Change";
title color=green "Active stocks, Biggest Gains and Losses";
format Percent_Change percent8.2;
run;
```

Section 5.10 includes some options for "traffic-lighting" cells. I don't think these tricks can be applied to the text, but they do work for the background. This requires the use of `proc format`. Note that we can use ranges within the `value` statement (the ranges are indicated by a "-"), and that `Low` and `High` are built-in values that seem to operate similar to `-Inf` and `Inf` in **R**. I wanted to use red for losses and black for gains, but the black background would hide the text, so I used a red/yellow/green color scheme here. Note that I use an equality operator on 0 (no gain), which isn't really appropriate for the continuous variable `percent_change`.

```
proc format;
value winlose Low - <0   = "red"
              0          = "yellow"
              >0  - High = "green";
proc print data=stocks1 label noobs style(data)={background=white}
style(header)={color=green background=white};
label share_price="Share Price" percent_change="Percent Change";
title color=green "Active stocks, Biggest Gains and Losses";
var stock share_price;
var percent_change/style={background=winlose.};
format Percent_Change percent8.2;
run;
```

We want to change the background of the `percent_change` column and no others, so we use a separate `var` statement for it. We still need a `var` statement for the other two variables, though, so that they will be printed.

As I said, I really wanted to use a black background for stocks posting gains, so I tried changing the font color for `percent_change` to white; note that we have to re-set the header for `percent_change` again in our last style statement so that it agrees with the header for the other variables. I had to change the color for the stock that showed no change from yellow to green, because white text on a yellow background didn't show up. It would be nice if I could use the conditional formats on the text as well as the background–I'm sure there are SAS commands out there somewhere that would do that.

I also changed the `name` variable to an italic font for a little variety; it required me to create a separate `var` statement for each of my three variables since they all have different formats now.

```
proc format;
value winlose Low - <0   = "red"
              0          = "green"
              >0  - High = "black";
run;
proc print data=stocks1 label noobs style(data)={background=white}
style(header)={color=green background=white};
label share_price="Share Price" percent_change="Percent Change";
title color=green "Active stocks, Biggest Gains and Losses";
var stock/style(date)={font_style=italic};
var share_price;
var percent_change/style={background=winlose. foreground=white}
style(header)={color=green background=white};
format Percent_Change percent8.2;
run;
```