PairedT.sas: Explanation of code

Goals of code:

- Some useful tricks for working in SAS
  - Clear results viewer window
  - Clear log window
  - Executing a piece of code / shortcuts for submit
  - Suggestions for getting help
  - Readable SAS code

- Inference on difference in means from paired data
  - Using proc means
  - Using proc univariate

- Transforming variables, especially log

**Clearing the results**: `ods html close; ods html;`
This line (2 commands) is optional. It tells SAS you are done storing results (`ods html close;`) then you want to store results again (`ods html`). The result is that the **NEXT** time SAS produces any output, the results viewer window is cleared before storing anything. The result is that you get rid of all the “old” contents.

Note: SAS does not clear the window immediately. It is cleared only when SAS tries to store new results. So don’t worry if you execute the `ods` commands and nothing happens to the results window. It will clear the next time a proc step runs.

**Clearing the log window**: mouse/menu or keyboard shortcut, not code
It is easier to clear the log window. Click in the log window to make it active. Either:
- Edit / Clear all, using the main menu
- Click the Black script X icon on the main menu bar (next to the running person)
- ctrl-e is the keyboard shortcut.
The Log window is cleared immediately.

If you accidentally clear the editor window (i.e. the window with your code), type ctrl-Z (the undo hot key) to restore the contents of the editor window.

**Executing a piece of code**: mouse/menu or keyboard shortcut, not code
You already know that clicking Run / Submit or clicking on the running person icon will submit all the code in the active editor window.

The keyboard shortcut for “submit” is the F3 key. If you have a laptop with a small keyboard, you will have to figure out what F3 is on your keyboard (probably some combination of special+key).
You can run a block of code by highlighting it (mouse over it or shift-click), then clicking the running person, hitting F3, or Run / Submit to submit it. If you select all lines that are part of a data or proc step, this will submit (or resubmit) just that step.

If you don’t include the `run;` in your selection, SAS won’t know to run your block of code. Select just the `run;` and submit that. The second submit gets appended to the first and your code is run.

**How to get help with code**
We are very willing to figure out why your code isn’t working. The easiest way is for us to see a clean copy of the SAS log window. We suggest:

- Clear the log window (clear all or ctrl-e when the log window is active)
- Run your entire program, from reading the data file through doing the analysis/analyses.
- Copy the contents of the log window to the clipboard (make the log active, then mouse over all the text or type ctrl-a **twice**, then click Edit / Copy or type ctrl-c.
- Paste the clipboard into **body** of an e-mail message.
- Add a short message about what you’re trying to do at the start of the e-mail message.
- Send the e-mail message.

There are various options for e-mailing contents of windows from SAS and various ways to send the log file as an attachment. All of those make it a lot more difficult to read your e-mail. Looking at the log file in the body of an e-mail message is much easier.

**Readable SAS code**
The following are three identical SAS programs:
```
data schiz; infile 'case0202.txt'; input unaff aff; diff = unaff - aff; run;
data schiz;
infile 'case0202.txt';
input unaff aff;
diff = unaff - aff;
run;
data schiz;
infile 'case0202.txt';
input unaff aff;
diff = unaff - aff;
run;
```
The three examples differ in what is on one line and whether commands are indented. I find the last is the easiest to read. I put one command per line and have indented the commands that are part of the data or proc steps by a couple of spaces. If there is a very long command, it is easiest to split into two or more lines, with extra indenting on all but the first line.

**PairedT.sas**: Data step

As we will discuss in lecture, the paired t analysis starts by calculating the difference within each paired set of observations. The easiest way to enter the paired data is as one row of data for each pair. The two groups are two columns, i.e. two variables. This format makes it easy to calculate the difference for each pair. That is done by the \texttt{diff = unaff - aff;} line. The name of the new variable to be computed is on the left of the \texttt{=} symbol. You can use any name (with some restrictions like can’t have a space in the name), but I recommend you use something that reminds you of the content (like diff to store the difference). The formula is on the right-hand side.

This line must go after the \texttt{input} line (because the variables \texttt{unaff} and \texttt{aff} aren’t defined until they are read) and before the \texttt{run;} (because the \texttt{run;} ends the block of code).

**PairedT.sas**: Proc means

The \texttt{var} statement names the variable to use. The data step stores three variables (\texttt{unaff}, \texttt{aff}, and \texttt{diff}). You only care about the difference, \texttt{diff}. If you care about two variables, name them both in the \texttt{var} statement, e.g. \texttt{var diff unaff;}

The title is optional, as always, but it helps you locate results in the output.

There is no class statement in this code because I want results that summarize all observations in the data set.

The keywords after \texttt{proc means} request specific pieces of output. Some are part of the default output; most are only provided when specifically requested. The keywords give you:

- \texttt{mean}: the sample average
- \texttt{std}: the sample standard deviation
- \texttt{stderr}: the sample standard error
- \texttt{t}: T statistic for test of mean = 0
- \texttt{prt}: two-sided p-value for T test of mean = 0
- \texttt{clm}: 95% confidence interval for the mean. You can change the coverage by adding \texttt{alpha=NN} to get a 1-NN interval. \texttt{alpha=0.10} gives you the 90% interval; \texttt{alpha=0.01} gives you the 99% interval.

\textbf{Note}: All of these keywords are options for the \texttt{proc means} statement. They go between the \texttt{proc means} and the ending ;.
**PairedT.sas**: Proc univariate

Proc univariate and proc means both calculate summary statistics for data. Proc univariate calculates many different summaries and gives you all of them by default. The var statement names the variable(s) you want summaries of. You could add a **class** statement if you want summaries for groups of observations.

Guide to the output from proc univariate. The following are the pieces of information we will talk about by the end of this week (with some items duplicated from the explanation of creativity.sas). All describe the difference between treatments, because they are calculated on the **diff** variable.

- In the box of results labeled Moments:
  - **N**: number of observations
  - **Mean**: sample average
  - **Std Deviation**: sample standard deviation
  - **Std Error Mean**: standard error of the mean

- In the box of results labeled Tests for location: **Mu0=0**:
  - row labeled **Student’s t**: The t-statistic and its two-sided p-value.
  - row labeled **Signed Rank**: The Wilcoxon signed rank statistic and two-sided p-value.

  We’ll talk about this test in a week or so.

**Data step to transform variables**

All manipulations of data values are done in a data step. You can either do these operations as part of the data step that reads the data set or in a new data step. The first is illustrated by calculating the difference when the Schizophrenia study are read. The second is illustrated in the second data step. That code is explained line by line.

**PairedT.sas**: data schiz2;

Create a new data set named **schiz2**. You can have many SAS data sets, but each needs a unique name. I prefer to use related names for related data sets. The code to create the data set is all the lines between the data line and the **run**;

**PairedT.sas**: set schiz;

This is the key command in this data step. **set** tells SAS to read observations from a previously created data set. **schiz** is arbitrary, but it has to match the name of the original data set (from **data schiz;**) at the top of the file.

**PairedT.sas**: logaff = log(aff);

This command does the actual work: computing the log of the value in aff and storing it in the variable logaff. **log()** is the function that computes the natural log (base e), which is the commonly used log transformation. The next two lines repeat that computation for the unaff variable, then calculate the difference in the log values. You can choose any name you like to store the result, but it helps to choose a name that you can remember and that gives some idea of the contents.
PairedT.sas: /* stuff */
Any text between /* and */ is a comment. All that text is ignored by SAS. The comment can span multiple lines. The program editor window changes the color of commented text so you see what is and is not a comment.

PairedT.sas: log10aff = log10(aff); sqrtaff=sqrt(aff);
These two lines give you the functions for the log base 10 transformation (e.g. pH) and the square-root transformation. Again, you can choose any variable name you like.