PairedT.sas: Explanation of code

Goals of code:

- Some useful tips and hints
  - Keyboard shortcut keys
  - Getting help (from SAS documentation or from the instructor)
  - Writing readable SAS code
- Inference on difference in means from paired data
  - Using proc means
  - Using proc ttest
- Transforming variables, especially log

Keyboard shortcut keys
Run code (i.e. submit): 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). Hitting F3 when the program editor window is selected will run its entire contents. Hitting F3 when some other window is selected will select the program editor window.

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.

Clear a window: Select the window and hit ctrl-E.

All the usual Windows shortcuts (e.g., ctrl-C, ctrl-X, ctrl-V, ctrl-A) work in SAS.

How to get help with code
Start with the "companion" documents like this one.

If you need more, SAS has extensive documentation. You can go to the online manuals (link on the lab / SAS page). Or, SAS 9.4 now supports context sensitive help. Put the cursor on the name of the procedure you want help with and hit F1. If you get a message about a firewall, ignore it and close the message window. A window will open with the documentation for that procedure (and you can click on one of the component commands to see about that). You also get a list of the other places that phrase is found. SAS documentation is so extensive, it can sometimes be very hard to find the answer to a simple question.

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. Although SAS has a Send Mail option, it is much easier for us to read if you just paste the log window into the body of an e-mail message.

We suggest:

- Clear the log window (clear all or ctrl-e when the log window is active, or use a dm command (last week))
- 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, 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 discussed 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 `diff = unaff - aff;` line. The name of the new variable to be computed is on the left of the `=`. 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 `input` line (because the variables `unaff` and `aff` aren't defined until they are read) and before the `run;` (because the `run;` ends the block of code).

PairedT.sas: Proc means
Proc means calculates various summary statistics for one or more variables individually. We can use it with the difference calculated in the data step to conduct a paired-t test. The var statement names the variable to use. The data step stores three variables (`unaff`, `aff`, and `diff`). You only care about the difference, `diff`. If you care about two (or more) variables, name them both in the var statement.

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 `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:

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

Note: All of these keywords are options for the proc means statement. They go between the `proc means` and the ending `;`.

PairedT.sas: `proc ttest; paired unaff*aff;` proc ttest demonstrated last week had a class statement and a var statement to identify a grouping variable and a response variable. With those, you get a two-sample t-test.

In 9.4 (and perhaps 9.3), you can use proc ttest to compute a paired t-test. The command is `paired`. This is followed by the name of the two variables separated by `*`.

PairedT.sas: `data schiz2;` All manipulations of data values are done in data steps. You can either do these operations as part of the data step that reads a data set or do them in a new data step. The first is illustrated by calculating the difference when the Schizophrenia study are read. The second is illustrated here and described line-by-line.

If you use `proc import` or the Import Wizard to read a data file, you don't have a data step to which you can add the calculation of the difference. Both proc import and the Wizard directly create the SAS data set. To calculate additional variables, we need to write a "follow on" data step. That data step will read observations from an already existing SAS data set (instead of from a data file) and do the desired computations.

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 schiz2 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: `diff = unaff-aff;`
This calculates the difference. It needs to be placed after the set command (so you have read a line of data).

To do a paired t-test, you would follow this data step with the commands to compute the paired t-test.

The rest of the data step illustrates transforming variables.

Transforming variables:

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.