calcmeans.sas: Explanation of code

Goals of code:

- Comments in SAS code
- Compute average response for each experimental unit
- Reminders about log transformations and side-by-side box plots

Note: the explanation focuses on the new things in this code. Ask if you have any questions about the data step and the proc sort.

The patty data set has 3 plate counts for each hamburger patty. The bacterial concentration was measured three times for each patty. In the language to be used shortly, the experimental unit is the patty; the observational unit (in the original data) is plate count. This is a violation of independence. A more appropriate analysis is based on the average count per patty. After averaging, the observational unit is the patty (because one row of data per patty), which is the same as the experimental unit (good).

The three observations for a patty are three rows of data with the same value of **trt** and **rep**. trt indicates the treatment and rep indicates the experimental unit (the treated patty).

The strategy is to calculate the averages for each patty and save them in a new data set. Then, we will analyze that new data set. The actual averaging is done by proc means. Doing this for each patty requires that the data be sorted so that all observations from one experimental unit (one patty) are next to each other in the data set. The trt and rep variables uniquely identify each patty. So, proc sort is used before the proc means.

patty.txt is a text file with spaces between columns. It is read using a data step.

/* calcmeans.sas: average ou's within each eu */: Comments in SAS code
SAS ignores anything between the /* and */ symbols. This can span multiple lines of text. The program editor will put comments in green text.

You can also include a short comment by using * as the first character on the line. This style of comment ends at the next ; indicating the end of that command.

proc means noprint: Calculate means for each experimental unit.
The goal is to create a new data set containing the means. We don't need a print out of the means. The noprint option suppresses all printed output.

by trt rep; var cfu;
The by statement specifies how to identify a block of observations to averaged. In this case, we

want to average all values for each combination of treatment and rep. Those two variables uniquely identify each patty. The var statement identifies the response variable to be averaged, i.e., the cfu recorded for that plate count.

output out=means mean=meancfu; Save results in a new data set
This creates a new SAS data set containing summary statistics for each group of observations. The details of what to save are given by the keyword=something pairs.
out= gives the name of the new SAS data set
mean= gives the name of the variable that will contain the mean
The out= is required. After that you can have as many requests as you want. You can store all sorts of pieces of output, e.g., standard deviations, sample sizes, standard errors. The SAS documentation has a list of the keywords for everything SAS can compute and save. I will introduce useful keywords as we need them during the semester. The only requirement is that each request has a unique variable name. SAS includes the by variables in the output data set automatically.

This output statement creates a new SAS data set called means. It will have 5 variables: trt and rep because they identify the by groups, meancfu because that stores the mean, and two variables, `_type_` and `freq_` that SAS creates automatically and we will ignore.

The saved values can be used in subsequent proc steps. SAS doesn't care where a SAS data set came from (a data step, a proc import step, a proc means step). Once created, they are essentially identical.

proc print; (two of them) These print the data set with the means and the original data set.

proc print data= patty(obs=10);, the second proc print. Sometimes, especially when you have a large data set, you want to see a bit of it to check variable names and appropriate treatment of each variable. Explicitly naming the data set, using data=NAME, followed by (obs=10) will print only the first 10 observations. If you want the first 6, use (obs=6).

log transformations. These were described in lab 4 with code in pairedt.sas, but we haven't needed them until now. This is a reminder. More details, including many other functions that might be useful, are in the explanation document for pairedt.sas

Transforming values must done inside a data step. We create a new data set (note the new name) by reading observations (the set statement) from a previously created data set.
`logmeancfu = log(meancfu);` does the work. The right hand side of = has the name of the new variable. `log(meancfu)` is the natural log transformation of the variable inside the ().

This code draws boxplots, but you've seen other things you might do.