hamburger.sas: Explanation of code

Goals of code:

- Read an excel sheet (.xls or .xlsx file)
- Wilcoxon rank sum test
- Diagnostics
- Transformations

Reading an excel sheet:
Use proc import to read the contents of an .xls or .xlsx file. The default is to load the first worksheet, but you can provide options (see help for proc import) to read specific sheets.

Wilcoxon rank sum test: `proc npar1way wilcoxon;`
With the wilcoxon keyword (instead of scores=data), proc npar1way does the wilcoxon rank sum test. The syntax is exactly like proc ttest or doing a randomization test in proc npar1way. The class statement names the groups; the var statement names the response variable. The exact statement requests a permutation test or a randomization test (when you add /mc n=10000, as with a randomization test).

The hardest thing is finding the appropriate results. The exact two-sided p-value is in the Exact Test / Two-Sided Pr row. This includes the continuity correction; you see the note about that just below the p-value. I prefer to not use a continuity correction. The p-value without the continuity correction is in the Kruskal-Wallis Test/ Pr¿Chi-square box.

Diagnostics: equal variance:
Numerical: `proc means mean std; class treatment;` : You want to compare the sd's for each group. Remember `proc means` can compute many different summary statistics. The `std` keyword requests the sd.

Graphical diagnostics for equal variance and normality
Our workhorse procedure for many different analyses will be `proc glm;`. This fits a linear model. Linear models include t-tests, linear regressions, and many types of analysis of variance.
`proc glm plots=(diagnostics(unpack));` runs proc glm and requests a large number of diagnostic plots. If you use `proc glm plots=(diagnostics);`, you get all the plots in one multi-panel figure. Adding `(unpack)` puts each plot in a separate figure, so it is easier to grab the ones you want. We will eventually talk about most of these plots. For now, we only need the residual vs predicted values and the QQ plot. If you unpack the figures, look at the plot titles to find the ones you want:
Residual vs predicted values: Residuals by Predicted for XX
normal QQ plot: Q-Q Plot of Residuals for XX
In both titles, XX is the name of the response variable.

SAS decides where to draw its line different than does R. When the data are normally distributed, the two lines are similar.

Using proc glm to fit a t-test:
Because proc glm can fit a huge number of models, its syntax is slightly different from proc ttest or proc npar1way.
Class statement: `class treatment;` Names variables that define groups.
Model statement: `model cfu=treatment;` Names the linear model to fit. The response variable goes on the left-hand side of the = sign; the model terms go on the right. For a t-test, the model only has the variable that defines the groups.
Remember: for a t-test (or a one-way ANOVA), the variable that defines the groups needs to go in two statements: the class statement to tell SAS that it defines groups, and the model statement.

Transformations
All manipulations of data need to be done in data steps. When you read data using a data step (i.e. with input), you can add statements defining the transformations below the input statement. You can not add manipulations to proc import. All proc import does is create a data set.

The SAS strategy for additional manipulations of data is to create a new data set, read observations from a pre-existing SAS data set (not a raw data file), then do the manipulations you need. This is what the data step that creates the hb2 data set does.

Reading observations from a pre-existing SAS data set `set hb;`
The set command reads observations from a pre-existing SAS data set. Unlike input, you don't need to specify variable names because those are already defined in the pre-existing data set. Once you read observations, you can specify any manipulation you need.

log transformations `log(cfu)`
The natural transformation is the log() function. These values are assigned to a new variable name. I prefer informative variable names, such as logcfu.

Other transformation functions:
square root: sqrt()
log base 10: log10()
reciprocal: 1/

Managing multiple data sets
When you have created more than one data set, which one does SAS use in a proc step. Its rule is: Use the last created data set, unless directed otherwise.
So, after you create the hb data set, that will be used in all proc steps (until you create a new data set). After you create the hb2 data set, that will be used in all subsequent proc steps.

explicitly specifying the data set `data=`: You can explicitly indicate the data set you use (by its name) by adding data= to any proc step. That overrides the "last created" for that proc step. It does not change the default, so if you write another proc step without data=, that will go back

to the default last created data set.

Backtransformation of results: In SAS, it is easier to grab the log-transformed results from the output and use a calculator to back transform to results on the original scale. If you need to do these calculations many times, I can show you how to store the results, extract the ones you want, and have SAS do the back transformation.