dietall2.sas: Explanation of code

Goals of code:

- More on reading character values
- Fitting ANOVA models
- "After the ANOVA"
  - estimating group means and standard errors
  - estimating linear combinations of means

More on reading character values: `proc import ...; guessingrows=200;`
SAS looks at the first 20 or so rows of data to decide how to read a .csv file. Sometimes that isn't enough. In particular, for the diet data, only looking at the first 20 rows doesn't read all the treatment names, so not enough information is stored. guessingrows=200; tells proc import to look at 200 rows, which is enough.

Further explanation and tricks for reading character values is in readcharacter.sas.

Fitting an ANOVA model: `proc glm;`
proc glm is a very flexible proc that fits many different sorts of models. The code for two groups (last week) is identical to the code for many groups (ANOVA, this week). You can't use proc ttest; for more than 2 groups; you can use proc glm.

The model to be fit is specified by the class and model statements:
`class trt;`: identifies variables that specify groups
`model lifetime = trt;`: identifies the response variable (left hand side of =) and the desired model. Here, the model is a function of the trt value. Because of the class statement, trt identifies groups. Hence, glm will fit a model with a separate mean for each trt group.

Important pieces of the output are:

- Error SS, error df, and MSE: In the first box of output, line labelled Error
- Pooled sd: In the second box of output, box labelled Root MSE
- F test of H0: no difference in means: In the last box of output, with the column labeled Type III SS, line labeled trt.

There is also a box with a Type I SS. For this model, the type I and type III SS and tests are exactly the same. Lecture will discuss the differences between these two. Except in special circumstances, US practice is to use type III SS.

After the ANOVA: The last `proc glm;`
I argued in lecture that fitting an ANOVA model and calculating the F statistic is really just the start of a data analysis. All of what I call "After the ANOVA" is specified by additional statements in the proc glm. All come **after** the model statement. The example includes a lot of different things. None depends on any other statement, so each can be used in isolation or combination.

group means and se's: `lsmeans trt /stderr;`
These se's use the pooled sd.

estimating linear combinations of means: `estimate 'Diff: R/R50 - N/R50' trt 0 0 -1 0 1 0;`
The code includes many different estimate statements, because many different specific combinations are of interest. My code illustrates how to do some things that are not part of the book's analysis of the diet data. The purpose of these results will be explained in lecture.

Each estimate statement has the same structure: the word estimate followed by:
a title in quotes (single or double): This can be anything, only used to label the output
a variable name: the variable identifying the groups to be compared
a string of numbers separated by spaces: Gives coefficients for your desired linear combination.

The coefficients are specified **in the same order** that SAS uses for the group names. You see this order in two places:

- The output from the class statement. Look for the Values list.
- The output from any lsmeans statement.

The order is alphanumeric (ASCII sort sequence, if you recognize/care about that), with lower case letters following upper case. I find it easiest to run a proc glm to figure out the order, then write the estimate statements and rerun the proc glm.

Fractions can be written as decimal numbers (e.g., 0.5) if the fraction is a nice number. They can also be written as integers followed by a `/divisor =` option. The two lines:
`estimate 'N/N85 - ave(N/R50, N/R40)' trt 1 -0.5 -0.5 0 0 0;`, and
`estimate 'same thing, diff. way' trt 2 -1 -1 0 0 0 /divisor = 2;` are identical. You need to use `/divisor =` when the fraction is not "nice", e.g. 1/3 or 1/7. That is because the linear combination only makes sense when the coefficients sum to 0. Approximating 1/3 doesn't always work unless you use enough digits to trigger roundoff. For example -0.333 -0.333 -0.333 1 sums to 0.001, which is not 0. If you have various not nice fractions, you have to find the least common multiple, so all fractions have the same divisor.

Other things you should or could want to do:

Storing and plotting residuals and predicted values: `output out=resids r=resid p=yhat;`
We used the output command to store residuals last week. Now, we want to plot residuals (Y axis) against predicted values (X axis). That means we need to store both residuals and predicted values

for each observations. `r=` names the variable to store the residual value for each observation. `p=` names the variable to store the predicted value.

These values are then plotted using `proc sgplot; scatter x=yhat y=resid;`
The scatter statement draws a scatter plot. `x=` identifies the variable for the X axis; `y=` identifies the variable to go on the Y axis.

Linear trend coefficients: `26.66 -18.33 -8.33 0 0 0`
This compares the three fixed kcal post-weaning diets: N/N85, N/R50 and N/R40. It specifically looks for a linear trend in the kcal. The average of 40, 50, and 85 is 58.3333, so the coefficients are 40-58.33 = -18.33, 50-58.33=-8.33 and 85-58.33 = 26.66. The groups are in the order N/N85, N/R40, N/R50, then the other three diets, so the coefficients need to be in the order 26.66 -18.33 -8.33 0 0 0.
Note: You don't need to worry about round off here (too much) because 0.66 - 0.33 - 0.33 = 0. You could also write 80 -55 -25 0 0 0 /divisor = 3.
If you have coefficients -1 0.33 0.33 0.33 (for one group compared to the average of 3, you do need to worry about round off because 0.33 + 0.33 + 0.33 is only 0.99 so the sum of coefficients is -0.01, which is not 0. Easier to use `/divisor = 3` here.

The actual value of this linear contrast isn't very meaningful. It is proportional to the regression slope (next week), but it isn't the slope. The test of contrast = 0 is meaningful. Here, $p < 0.0001$ so there is strong evidence of a difference when you look for a linear trend.

all pairwise differences, with multiple comparisons adjustments:
`lsmeans trt /stderr pdiff adjust = tukey cl;`
The cl option requests 95% confidence intervals for each lsmean. The pdiff options requests all pairwise differences. The default for pdiff is just the p-value for the test. pdiff and cl give you the estimated difference and 95% confidence intervals for each pair. Adding alpha = gives you other coverages. E.g., alpha = 0.10 gives you 90% intervals. adjust= specifies which multiple comparisons adjustment to use. There are many choices. The most commonly used are tukey, t, and bon (Bonferroni). `t` gives you no adjustment. I also mentioned scheffe (in the book), sidak (Dunn-Sidak), and dunnett.

By default, you get two plots: the lsmeans for each treatment and a plot of differences.
The differences plot is a compact graphical summary of all the pairwise differences. The X and Y axes are the mean longevity (same scale for both axes). Horizontal and vertical bars are drawn at the lsmean for each group. Each diagonal line has length of the simultaneous confidence interval for the difference between a pair of groups. The long dashed line is the line of equality. If a pair of groups intersect on the dashed line, the estimated difference is 0. If the colored diagonal line crosses the dashed line, the confidence interval for the difference includes 0. These are colored red. If the colored diagonal line does not cross the line of equality, the confidence interval does not include 0 and the difference is statistically significant. These are colored blue. For these data, you see that comparisons involving NP and N/N85 are significantly different from 0, while comparisons not involving those two groups are often no evidence of a difference.