calcmeans.r: Explanation of code

Goals of code:

- Comments in R code
- Compute average response for each experimental unit
- Compute residuals for each observation

Note: the explanation focuses on the new things in this code. Ask if you have any questions about reading in the data file.

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 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. There are a couple of ways to do this in R. The most straightforward is to use functions in the dplyr library.

library(dplyr) To calculate means for groups of observations.
The dplyr library provides powerful tools to manipulate data sets and especially to create summary statistics for subsets of data. This library is automatically included with RStudio. If you are using “vanilla” R, you need to packages / install package from the main menu first.

In both RStudio and R, you need the library(dplyr) statement to load the package before you use either of the next two statements, which use dplyr functions.

pattyGr <- group_by(patty, trt, rep) Define grouping variables
The group_by() function has two or more arguments. The first is the name of a data frame. The rest are the names of one or more variables that identify groups of observations. For the hamburger patties, you need the treatment name and the rep number to identify the three observations from a specific patty. The data frame with additional grouping information is stored in a new object, for each I chose the name pattyGr.

means <- summarize(pattyGr, meancfu=mean(cfu)) Calculate the mean for each group
The summarize() function does the actual work of calculating the means. The first argument is the name of the object with grouping information. The rest of the pieces (one or more) are pairs of variable names and a function applied to a variable in the object. The code here calculates the mean, specified by mean(), of the cfu variable, because it is mean(cfu), and stores that in the variable
named to the left of the =. That’s meancfu in this example. You can have multiple requests, each
separated by a comma, if you want multiple summary statistics.

Note: If you end up with only one observation in the result, you applied summarize() to the original
data frame (no grouping information) instead of the object with grouping information.

**means** Print the result so you can see what summarize did

**Calculating residuals**
To calculate residuals, we need to use a general purpose model fitting function, lm(). We will use
this extensively over the next few weeks to fit both regression models and ANOVA models. It can
also be applied to fit the t-test model. That model is an ANOVA applied to data with only two
groups.

**factors** Fit an ANOVA model or a regression model
lm() knows whether to fit an ANOVA model or a regression model by looking at whether a variable
is defined as a factor or not. If a variable is defined as a factor, that variable is used to define groups
of observations and lm() will fit an ANOVA model that compares the mean for each group. If a
variable is not a factor, that variable defines a regression line and lm() will estimate the intercept
and slope for that regression.

```r
means$trt.f <- factor(means$trt)
```
Create a factor variable
My habit is to explicitly indicate when I want a variable to be a factor. I also define a new variable
that is the factor version. The factor() function creates the factor version of a variable. Here, we
want to create a factor version of the trt variable. You could store that in the original variable, but
I prefer to create a new variable, with a name related to the original variable. trt.f reminds me that
the variable is the factor version of trt.

The next two statements print the original variable and its factor version. Notice the additional
Levels piece of information for the factor variable. We will talk more later about levels of a factor.

```r
patty.lm <- lm(meancfu ~ trt.f, data=means)
```
Fit the t-test model
The lm() function does the actual fitting of the t-test model. The important piece is the first
argument, which specifies the formula to be fit. The first piece, to the left of the ~, is the name
of the response variable. The piece to the right of the ~ specifies the model to be fit. Here, we
want to fit a mean to each treatment group, that is each level of the factor trt.f. The data= tells
lm() where to find the variables; here, that is in the means data frame. The result is stored in the
patty.lm object.

**resid() and predict()** Calculating residuals and predicted values
The resid() function takes the object created by lm() and extracts the residuals. The predict()
function extracts the predicted values. In both cases, these are being stored back in the original
data frame with variable names that indicate their contents.

These variables can then be manipulated using other functions, e.g. boxplots, as indicated using
the boxplot function.