fdrfull.sas: Explanation of code

Goals of code:

- FDR computation, starting from raw data
- Storing results in a SAS data set
- Analyzing a subset of the data
  - Creating a subset with a data set
  - Subsetting the data “on the fly”
- False discovery rate computations (repeat of fdr.sas)

**False discovery rate control:**
The general concept of what follows is:
- analyze many response variables one at a time
- save the p-values from that analysis to a SAS data set
- calculate adjusted p-values
- interpret the results using your desired false positive rate.

Here are the details on each step:

**Analyzing many response variables**

```sas
model y1 -- y30 =
```
We have used `proc glm ; model Y =` with one Y variable to fit ANOVA models to a single response variable. You can analyze more than one response in one `proc glm` by specifying the list of response variables on the left hand side of the `=`. The two commented lines and one non-commented line illustrate three ways to specify multiple variables.

```sas
model y1 y2 y3 =
```
You can explicitly name all the variables you want to analyze.

```sas
model y1 - y30 =
```
Or, you can specify a list of numbered variables. When the variables have the same prefix, you can specify the first and last, separated by a hyphen. `y1 - y30` means “y1 y2 y3 · · · y29 y30”.

```sas
model y1 -- y30 =
```
Or, you can specify a list of variables. SAS stores the variables in the order they were created. You can specify a list of variables by giving the first variable in the list, two hyphens (no space in between), and the last variable in the last. That refers to all variables between the first and the last, including the ends. Here `y1 -- y30` is equivalent to `y1 - y30`, but when the variables are in the order: A1 A2 B CO2 A3 funny A4 games, then A2-A4 means “A2 A3 A4” while A2 -- A4 means “A2 B CO2 A3 funny A4 games”. SAS objects if you try funny - games because funny and games don’t have the same root, but funny -- games means “funny A4 games”.

1
How do you find the order that SAS has stored the variables? print the data set and look at the order (left to right) of the variable names. When you print a data set, you see that variables are listed in the order they were created. The left-most variable name is the first one created (e.g., the first variable on the input line). Any variables created by statements below an input statement are shown to the right of all the variables created by the input statement.

**Saving results into a SAS data set ods output**

SAS allows you to store any printed result into a SAS data set. Each piece of output has a name. The `ods output` command saves the named piece of output into a SAS data set. You provide a name for the data set; you need to use the correct name for the piece of output. For example, the command used here is: `ods output modelANOVA = tests; modelANOVA` is the name of the piece of output. Here, that is the block of Type I and Type III tests. That will be stored in a data set named tests. You could have written `ods output modelANOVA = dolly;`, which will create a data set named dolly. But, the word `modelANOVA` has to be correct. My code examples will give you names you need. Others can be found in the SAS documentation. See me if you want to find out how to find those names.

The subsequent proc print prints the data set. You see there are two lines for each response variable. The printed results have three lines with similar information. For reasons to be discussed later, we want the type III tests. Those are the lines with HypothesisType of 3. We don’t want both tests (because then subsequent analysis will treat that as 60 tests, not 30 tests with a duplicate).

**Subsetting the data to extract the type 3 values:**

We have seen data steps with if / then commands before. These execute code only when some condition is true. The second form of the if statement is called a “subsetting if”. This has no then clause. This stores the observations only when the condition is true. This statement, in a data step, will keep only the observations for which hypothesistype = 3.

The result is that the data set tests2 has only 30 observations, one for each response.

**Calculating adjusted p-values:**

```
proc multtest inpvalues(ProbF)=tests2 fdr out=fdr;
proc multtest calculates many different sorts of adjusted p-values. You can use multtest to do simple tests and then make adjustments. I have found it easier to do the analysis I want separately, save the p-values, then use multtest only to adjust the p-values. This statement calculates FDR adjusted p-values.
```

The `inpvalues(ProbF)=tests2` tells SAS where to find the p-values. The assumption is that this is a data set with one row for each test. The `=tests2` piece names the SAS data set containing the p-values. The `inpvalues` piece tells multtest that you are providing p-values as input. The `(ProbF)` specifies the name of the variable containing the p-values. If you look at the printout of the contents of tests2, you see that SAS used ProbF as the name for the variable containing the p-values (remember, SAS created this data set, by the ods output command, so you couldn’t choose names).
The **fdr** option requests fdr adjustment. There are many others. See the documentation for the full list.

The **out=fdr** tells SAS to store the results in a new data set. You provide the name (just like all other out= commands). The example code includes saving the data set so we can print the subset of observations with small adjusted p-values.

The output from proc multtest is explained in the false discovery rate section, above.

You can also print the data set that stored the fdr results, which shows you more information about each test.

**printing a subset of observations:** \texttt{where fdr\_p < 0.2;}

To print only observations that have a small value of \texttt{fdr\_p}, you could use a subsetting if statement to create a new data set with only the subset (just like we created tests2 above). You can also subset the data set “on the fly” using a where command. The where command does the subsetting in one step without requiring you to create a new data set.

The second proc print illustrates this. The where command only uses observations that satisfy the specified condition. \texttt{where fdr\_p < 0.2;} will print only those observations with values of \texttt{fdr\_p} less than 0.2. If you want to use the subset multiple times, you probably want to create a data set with that subset. If you want to use the subset only once, you probably want to use where.

You can use a where command in any proc step. It does the same thing everywhere: restricts the action of the proc step to those observations that satisfy the where condition.