Supplemental SAS Material for *Longitudinal Analysis: Modeling Within-Person Fluctuation and Change* by Lesa Hoffman (2015, Routledge Taylor & Francis)

General Notes about SAS Syntax

- What follows should be helpful in understanding the syntax files provided for each chapter, but more general and comprehensive help with SAS syntax for data management and analysis can be found in my notes for a previous summer course at UNL called *Make Friends with SAS*: http://www.lesahoffman.com/930SAS/index.html

- **Basic rules about SAS syntax and output:**
  - SAS uses three kinds of files: **.sas** is for syntax, **.log** is for the log that records what happened and any error messages in doing so, and output, of which there are many kinds (html is the current default). The syntax and log files are just text files (so you can open and edit them in notepad or Word).
  - By default SAS output goes to an internal output window. To save the output to an external file using syntax, you can use the Output Delivery System (abbreviated ODS in SAS syntax), whose options include HTML (viewable in a web browser), RTF (rich text file viewable in a text editor), PDF (viewable using Adobe Acrobat), and others.
  - For example, the first `ODS HTML` statement below opens a file in the location and file name given to save the subsequent output (using the default `HTMLBLUE` style), and the second statement closes the file:
    - **Save as .html:**
```
ODS HTML FILE="C:\MyFolder\SAS_Output.html" STYLE=HTMLBLUE;
ODS HTML CLOSE;
```
  - However, if you change the name of the file extension from .html to .xls, it will open as an Excel worksheet instead. The style `MINIMAL` will strip out unnecessary formatting for easier use in Excel:
    - **Save as .xls:**
```
ODS HTML FILE="C:\MyFolder\SAS_Output.xls" STYLE=MINIMAL;
ODS HTML CLOSE;
```
  - This code below will create output in a rich text format that is editable in any text editor (e.g., Microsoft Word). I like the style `STATDOC` for RTF files but there are several others to choose from as well (ask the google for others). The additional `STARTPAGE=NO` and `BODYTITLE` options prevent page breaks between each output table and direct any titles (see below) to appear in the document rather than in the header, respectively:
    - **Save as .rtf:**
```
ODS RTF FILE="C:\MyFolder\SAS_Output.rtf" STYLE=STATDOC
                  STARTPAGE=NO BODYTITLE;
ODS RTF CLOSE;
```
  - The `OPTIONS` statement at the beginning of all my SAS syntax files controls several aspects of the format of SAS output, including to stop the printing of page numbers (`NOnumber`), dates (`NOdate`), centering of output (`NOcenter`), and automatic page breaks between output tables (`FormDlim=' '`). In addition, the maximum length and width of output are allowed via `PageSize=MAX` and `LineSize=MAX`, respectively. Finally, the blank `TITLE`; statement shuts off the unnecessary "The SAS System" title printed by default, and `ODS TRACE OFF` shuts off the names of the output tables printed to the log (replace with `ODS TRACE ON` to show them instead).
  - The `TITLE` command can be used to list titles to appear in your output, in which the text to be printed is shown within single or double quotes. Up to 10 `TITLE` commands (with an optional number appearing at the end, such as `TITLE1` and `TITLE2`) can be used and will continue printing until they are changed or removed. To shut off a title, list the `TITLE` command with no text after it followed by a semi-colon. For example:
```
TITLE1 'This is my first title,  which will print until the blank TITLE1 below';
TITLE2 'This is my second title, which will print until the blank TITLE2 below';
TITLE1; TITLE2;
```

- In general, SAS syntax is not case-sensitive or space-sensitive, and it will continue reading syntax written to any line length. However, in my syntax, I write SAS keywords in CAPS and data-specific or model-specific syntax in lower-case to make it easier to understand and modify.
- In SAS syntax, SAS keywords appear in **BLUE** or **BLUE**, and each command ends in a **semi-colon**. Titles, labels, and file paths appear in **PURPLE** and **are case-sensitive and space-sensitive**. Numbers, programming variable references (see below), and variable formats all appear in **TEAL**. If you see **RED**, that usually means something is wrong (e.g., **RED** text after a forgotten semi-colon); alternatively, new beta commands sometimes shown in **RED**.
- Comments are indicated with * at the beginning of each so that the text that follows will turn **GREEN.** Several lines can be commented out by starting the first line with /* and ending the last line with */
- You can select just parts of code to run by commenting out the irrelevant lines of syntax within /* and */
- You can abbreviate lists of variables using a single dash for contiguous names that *end* in ordered numbers (e.g., **var1-var10** will abbreviate var1, var2, var3…. var10, but **var1x-var10x** will not work). Alternatively, you can abbreviate lists of variables in contiguous positions using a double dash (e.g., **var1x--var10x** will refer to **var1x**, whatever variable is next in the file, next, next… **var10x**).

- **Getting data into SAS:**
  - In my syntax, I use a short-cut programming technique so that any file locations only need to be indicated once in a syntax file. Specifically, the location from which files should come and go is indicated by a SAS programming variable I've named **filesave** whose value is defined using the keyword %LET, for example:
    - `%LET filesave=C:\Dropbox;`
  - Multiple such variables can be created for different locations. As with any SAS programming variable, this location abbreviation is then invoked by referring to it beginning with an ampersand (&) and ending with a period.
  - For example, this LIBNAME statement defines a SAS data library also called **filesave** whose location is indicated by the path referred to by **filesave** defined above: `LIBNAME filesave "&filesave.";`
  - For example, this ODS HTML statement (in which ODS = Output Delivery System) opens an .html file in which to save all resulting output. The file is saved to the location indicated by **filesave** above with the name and extension given: `ODS HTML FILE="&filesave.\SAS_Output.html";`
  - Once their libraries are defined, existing SAS datasets can be imported into SAS for further modification or use via a "data step" combination of **DATA**, **SET**, and **RUN** commands, each of which ends in a semi-colon. **DATA** lists the new SAS data file being created, **SET** lists the SAS data file it is coming from, and the **RUN** command executes the command (otherwise nothing will happen). SAS data file names have two parts, separated by a period: the first indicates the library location, and the second indicates the file name (no extension needed).
  - For example below: the SAS data file called **myfile** currently in the location indicated by **filesave** will be copied into a new SAS data file to be called **mySAsfile** in the **work** library, a temporary directory. Then, any subsequent data modifications (e.g., adding or removing new variables, removing cases) can be done to the temporary copy in the work library, leaving the original data safely intact in the **filesave** location.
    - `DATA work.mySASfile; SET filesave.myfile; RUN;`
  - Data files stored in other formats (e.g., SPSS, STATA, Excel) can be read into SAS through **PROC IMPORT**. For example, each of the following imports a file called **myfile** from the location indicated by **filesave** and stores it as a SAS data file called **mySASfile** in the **work** temporary library. The **REPLACE** option allows the file to be re-created if there is already a SAS file in the work library by that name (so that you can re-create your data copy as needed in case something goes wrong later on):

- **SPSS:**
```
PROC IMPORT DATAFILE="&filesave.\myfile.sav" OUT=work.mySASfile DBMS=SAV
            REPLACE; RUN;
```

- **STATA:**
```
PROC IMPORT DATAFILE="&filesave.\myfile.dta" OUT=work.mySASfile DBMS=DTA
            REPLACE; RUN;
```

- **Excel with .xls extension** from sheet `mysheet` in which the first row has the variable names:
```
PROC IMPORT DATAFILE="&filesave.\myfile.xls" OUT=work.mySASfile DBMS=EXCEL
            REPLACE; SHEET="mysheet"; GETNAMES=YES; RUN;
```

- **Excel with .xlsx extension** from sheet `mysheet` in which the first row does NOT have the variable names:
```
PROC IMPORT DATAFILE="&filesave.\myfile.xlsx" OUT=work.mySASfile DBMS=XLSX
            REPLACE; SHEET="mysheet"; GETNAMES=NO; RUN;
```

- Data files can only be modified if they are in a SAS format; data files in other formats will not be modified.
- When importing data from other programs, if missing data has been indicated by a recognized missing data value code within that program, it will appear as a dot (period) in SAS data files for numeric variables or as a blank for string (text) variables.
- To enter data into a brand-new data file (such as for creating "fake people"), `DATA` indicates the location and name of the new file, `INPUT` lists the variables it should contain (in order), and `DATALINES` indicates that raw data values are coming next (with spaces or tabs separating the values into columns matching those on `INPUT`). Entered data will be shaded in yellow, terminated by a semi-colon, and then `RUN` makes it go. Below, we create a new data file in the work library called `fakepeople` with two rows of data for two fake people (IDvar = −99) and give them values of 0 or 1 for the `sexMW` variable and values of 80 or 90 for the `age` variable:
```
DATA work.fakepeople; INPUT IDvar sexMW age; DATALINES;
-99  0  80
-99  1  90
; RUN;
```
- To merge two SAS data files by one or more ID variables (such as `IDvar` here), each should be sorted by those ID variables first. For example below, `DATA` creates a new file in the work library called `myfilemerged` from existing `myfile` and `fakepeople` data files from the work library as listed on the `MERGE` statement, as used instead of `SET`). The option `BY` indicates that rows and columns should be matched based on `IDvar`:
```
PROC SORT DATA=work.myfile; BY IDvar; RUN;
PROC SORT DATA=work.fakepeople; BY IDvar; RUN;
DATA work.myfilemerged; MERGE work.myfile work.fakepeople; BY IDvar; RUN;
```
- Note that the `MERGE` statement will work to combine multiple data files that have the same variables but different cases, the same cases but different variables, or both different cases and different variables. In the latter case, however, if the data files to be merged have a different structure, then the order in which the data files are listed is critical. For instance, to merge multivariate data into stacked data (i.e., merging data with one row per person into data with more than one row per person), the stacked data file should be listed first on `MERGE`.

- **Making new variables, labeling new variables, and using value labels in SAS syntax:**
  - SAS variable names can be up to 32 characters. SAS variable labels (as created by the `LABEL` command within a data step, as demonstrated below) can provide even more description (up to 256 characters).
  - In addition, SAS value labels can be created using `PROC FORMAT` to assign descriptive labels to specific values of a variable. Then, using a subsequent `FORMAT` statement, these value labels can be applied just in a data file, just in an analysis, or both. For example below, `PROC FORMAT` is used to create two value labels: `Fgender` in which 0 = women and 1 = men, and `Fdementia` in which 1 = none, 2 = future, and 3 = current. However, I also include the original value first in the label because SAS re-orders them alphabetically otherwise (which would change the reference group for any variables treated as categorical on a `CLASS` statement, see below):

```
PROC FORMAT;
VALUE Fgender 0 ="0women" 1 ="1men";
VALUE Fdementia 1 ="1none" 2 ="2future" 3 ="3current";
RUN;
```

- Data modifications (e.g., creating new variables, removing variables) can only happen inside a "data step" (the combination of **DATA/SET/RUN** commands) so that SAS knows which data file should be modified. As before, **DATA** lists the new data file to be created, **SET** lists the data file to start from, and **RUN** executes the command.
- Here is an example of how to create a new variable through mathematical operations on existing variables. Below we create a new variable called **age80** (and accompanying variable label) in which 0 values represent age 80 from the old variable called **age**. The **DATA/SET/RUN** commands save a copy of the modified data in the same work library using the same data file name (essentially saving the data file as itself):

```
DATA work.myfilemerged; SET work.myfilemerged;
age = age - 80; LABEL age80 = "age80: Age in Years (0=80)"; RUN;
```

- In contrast, these **DATA/SET/RUN** commands save a copy of the modified data in the same work library using the a different data file name **myfilemergednew** ("save as" a new data file instead):

```
DATA work.myfilemergednew; SET work.myfilemerged;
age = age - 80; LABEL age80 = "age80: Age in Years (0=80)"; RUN;
```

- Another way of creating new variables is through conditional logic: a combination of **IF**, **THEN**, and **ELSE IF**. **IF** lists the first condition; if that condition is not met, **ELSE IF** lists the others. For example, here an existing variable called **sexMW** in which men = 0 and women = 1 is to be recoded into a new variable within the same data file called **sexWM** in which men = 1 and women = 0. The **FORMAT** statement tells SAS to use the value label **Fgender.** as defined in **PROC FORMAT** for the new variable in the data file (and in any analysis as well):

```
DATA work.myfilemerged; SET work.myfilemerged;
IF sexMW=0 THEN sexWM=1; ELSE IF sexMW=1 THEN sexWM=0;
LABEL sexWM = "sexMW: Sex (0=Women, 1=Men)";
FORMAT sexWM Fgender.; RUN;
```

- If multiple new variables are to be created based on a condition being true, then commands **DO** and **END** are needed to segment the steps to be followed. For example, if an existing variable called **demgroup** = 1, then four new variables are to be created within the same data file as given next (starting with **DO** and ending with **END**). Otherwise (**ELSE IF**) **demgroup** = 2, the new variables take on different values, otherwise (**ELSE IF**)… and so on. The **FORMAT** statement tells SAS to use the value label **Fdemgroup.** as defined in **PROC FORMAT** for the original **demgroup** variable, as well as variable labels for the new variables via the **LABEL** command:

```
DATA work.myfilemerged; SET work.myfilemerged;
     IF demgroup=1 THEN DO; demNF=0; demNC=0; demFN=1; demFC=0; END;
ELSE IF demgroup=2 THEN DO; demNF=1; demNC=0; demFN=0; demFC=0; END;
ELSE IF demgroup=3 THEN DO; demNF=0; demNC=1; demFN=0; demFC=1; END;
FORMAT demgroup Fdemgroup.;
LABEL demNF = "demNF: Dementia Contrast for None=0 vs Future=1"
      demNC = "demNC: Dementia Contrast for None=0 vs Current=1"
      demFN = "demFN: Dementia Contrast for Future=0 vs None=1"
      demFC = "demFC: Dementia Contrast for Future=0 vs Current=1";
RUN;
```

- The **KEEP** and **DROP** commands (mutually exclusive) can be used to keep or remove (respectively) only certain variables in a data file. All variables are kept by default otherwise. For example below, these **DATA/SET/RUN** commands save a copy of the modified data in the same work library using the same data file name (essentially saving the data file as itself). In the first version, **DROP** removes a series of contiguous unwanted variables (**oldvar**); in the second version the same is accomplished by listing the variables to remain on **KEEP** instead:

```
DATA work.myfilemerged; SET work.myfilemerged; DROP oldvar1-oldvar10; RUN;
DATA work.myfilemerged; SET work.myfilemerged; KEEP IDvar newvar1-newvar10; RUN;
```

- Certain cases (rows) can be kept or removed based on the same conditional logic. For example below, the **WHERE** statement indicates that only cases in which **demgroup** = 1 are to be transferred from the old data file **myfilemerged** to a new data file called **demgroup1only** (both in the **work** library):

```
DATA work.demgroup1; SET work.myfilemerged; WHERE demgroup=1; RUN;
```

- The example below has the same result, but from the opposite direction: each **IF** statement indicates that cases in which **demgroup** = 2 or 3 are NOT to be transferred from the old data file **myfilemerged** to a new data file called **demgroup1only** (both in the **work** library). The first version writes this explicitly, whereas the second version uses the **IN** function to do the same thing (which allows many values to be selected upon at once):

```
DATA work.demgroup1; SET work.myfilemerged;
IF demgroup=2 OR demgroup=3 THEN DELETE; RUN;
```

```
DATA work.demgroup1; SET work.myfilemerged;
IF demgroup IN(2,3) THEN DELETE; RUN;
```

- Here is how to select cases to be kept based on whether they have complete data for a series of variables. The **NMISS** function counts the number of missing values across the numeric variables listed in its parentheses. Here, we tell SAS that if the number missing is > 0, the case should be removed from the new data file **completers**:

```
DATA work.completers; SET work.myfilemerged;
IF NMISS(pred1, dv1, pred2, dv2)>0 THEN DELETE; RUN;
```

- **Summarizing data into new variables:**
  - Summarizing data into new variables proceeds differently depending on whether one is summarizing across columns or across rows. Here are some examples of each type that are used in the chapter examples.
  - One can summarize across columns into new variables using many built-in functions. More generally in each statement, the new variable appears on the left of the equals, the function appears on the right, and what variables the function is to use appear in parentheses (along with the keyword **OF** to prevent SAS to misinterpreting any single-dash abbreviations as a minus sign). Sometimes commas are needed to separate variables (but not always).
  - Here are some examples:

```
DATA work.myfilemerged; SET work.myfilemerged;
meanvar = MEAN(OF var1-var10); * Save the mean of variables as a new variable;
maxvar = MAX(OF var1-var10); * Save the maximum of variables as a new variable;
RUN;
```

  - Summarizing across rows can be done in a variety of ways, but I have used a special case of **PROC MEANS** in my examples using *stacked* data (in which there is one row per occasion per person) to do so. The general procedure is to save the mean across occasions for the desired variables for each person to a new data file in which each person has only one row. The new person mean data file is then merged back into the original stacked data file.
  - In the example below, **PROC SORT** first sorts the **work.stacked** data file by the variable to be the unit of summary across rows (**IDvar**). **PROC MEANS** then calculates summary information (by default, the N, mean, SD, minimum, and maximum) per unique value of **IDvar** for the variables on the **VAR** statement (**pred1 pred2**). Rather than printing these per-person summaries to the output (suppressed by the **NOPRINT** option), the **OUTPUT OUT=** indicates they are to be saved a new data file called **work.IDvarMeans**. However, to limit the saved information to only the person means as desired, the **MEAN** function indicates that the mean per unique **IDvar** for the existing variables (**pred1 pred2**) should be saved to new variables (**meanpred1 meanpred2**). The new data file of means per unique IDvar called **work.IDvarMeans** is then merged back into the **work.stacked** data file, matching on **IDvar**, and the **DROP** removes some unwanted variables created during **PROC MEANS** from the **work.stacked** data file:

```
PROC SORT DATA=work.stacked; BY IDvar; RUN;
PROC MEANS NOPRINT DATA=work.stacked; BY IDvar;
VAR pred1 pred2;
OUTPUT OUT=work.IDvarMeans MEAN(pred1 pred2)= meanpred1 meanpred2;
RUN;
DATA work.stacked; MERGE work.stacked work.IDvarMeans; BY IDvar;
DROP _TYPE_ _FREQ_; RUN;
```

- **Describing data:**
  - **PROC MEANS** is more commonly used to get summary statistics for continuous (quantitative) variables. By default, this includes N, mean, SD, minimum, and maximum (other statistics can be added or removed by listing them on the **PROC MEANS** statement). Relative to the previous example, without the **OUTPUT OUT=** statement, no new data files are created; without the **BY** statement, the entire data file will be summarized.
  - For example below, **DATA** tells SAS which data file to use, **VAR** lists the variables to be summarized, and **RUN** executes the command:

    ```
    PROC MEANS DATA=work.myfilemerged; VAR age grip cognition; RUN;
    ```

  - Summaries can be provided for the entire sample or for subsets of the sample. As an example of the latter below, **CLASS** lists the variables for which summaries should be provided for each unique value. One the **WAYS** statement, **0** requests an overall summary, and **1** requests a summary per level of the **CLASS** variables:

    ```
    PROC MEANS DATA=work.myfilemerged;
    CLASS demgroup;
    WAYS 0 1;
    VAR age grip cognition; RUN;
    ```

  - **PROC FREQ** can be used to get frequencies and cross-tabulations for categorical (grouping) variables. A variable listed alone will be summarized marginally, and sets of variables linked by an asterisk will be summarized jointly. By default, this includes frequencies and percentages per cell, per row, and per column (others can be added or removed by listing them as options after the **/** on the **TABLE** statement). For example below, **DATA** tells SAS which data file to use, **TABLE** lists the variables to be cross-tabulated, and **RUN** executes the command:

    ```
    PROC FREQ DATA=work.myfilemerged; TABLE sexMW*demgroup; RUN;
    ```

  - **PROC CORR** can be used to get descriptive statistics and correlations for continuous (quantitative) variables. Pearson correlations are printed by default; other versions (e.g., Spearman, Pearson covariance) are available by listing them on the **PROC CORR** statement. For example below, **DATA** tells SAS which data file to use, **VAR** lists the variables to be correlated, and **RUN** executes the command:

    ```
    PROC CORR DATA=work.myfilemerged; VAR age grip cognition; RUN;
    ```

- **Restructuring data:**
  - SAS does not have an automatic way to restructure data from wide (multivariate) to long (stacked), but code to do so is provided in several examples (e.g., chapters 7b and 9). In the example below, **DATA** creates a new long (stacked) data file, **SET** indicates the previous wide (multivariate) file, and then each line of code creates new variables per occasion, such that all variables for the same occasion should be included prior to the **OUTPUT** statement, which writes a new line of data. New variables can be created from constant values (e.g., occasion below) or from existing variables (e.g., age). The previous multivariate variables will still be included by default.

    ```
    DATA work.stackeddata; SET work.multivariatedata;
    occasion=12; age=age12; DV=DV12; IV=IV12; OUTPUT;
    occasion=13; age=age13; DV=DV13; IV=IV13; OUTPUT;
    occasion=14; age=age14; DV=DV14; IV=IV14; OUTPUT; RUN;
    ```

Understanding the SAS PROC MIXED commands used for Longitudinal and Multilevel Modeling (mostly in order of appearance in my example models)

- **`PROC MIXED` statement options used in my example code (note that most options turn blue, but some do not):**
  - **`DATA=`** Data file to use (default is last data file referred to in your syntax)
  - **`COVTEST`** Print SEs and *p*-values for significance test of variance estimates (not printed by default)
  - **`NOCLPRINT`** Do not print class variable values (otherwise are printed by default)
  - **`NAMELEN=`** # characters printed in fixed effects tables (default=20, I use 100 in my examples)
  - **`METHOD=`** select REML or ML estimator (default is REML)
  - **`IC`** Print other information criteria and model degrees of freedom (not printed by default)
    - Note that the model degrees of freedom printed in the **`IC`** table when using ML will refer to ALL model parameters (fixed effects in the model for the means plus all variances, covariances, and other variance model parameters), but will exclude the fixed effects when using REML. Although this may seem arbitrary, it makes sense if you remember that models that differ in fixed effects cannot have their fit compared using −2LL, AIC, or BIC when using REML (and thus the number of fixed effects is irrelevant for those statistics).
  - For example (all options should be listed before the semi-colon terminating the **`PROC MIXED`** statement):

```
PROC MIXED DATA=work.datafile COVTEST NOCLPRINT NAMELEN=100 IC METHOD=REML;
```

- **Other `PROC MIXED` statement options that can be useful:**
  - **`NOITPRINT`** Do not print iteration history (otherwise is printed by default)
  - **`NOINFO`** Do not print the model information table (otherwise is printed by default)
  - **`MAXITER=`** specify # iterations (default = 50)
  - **`EMPIRICAL`** adjust SEs for non-normality of residuals (i.e., the "sandwich" estimator)
  - Using the **`EMPIRICAL`** option is analogous to **`ESTIMATOR = MLR`** for robust maximum likelihood in Mplus with respect to the standard errors for the fixed effects. But unlike Mplus, SAS does not provide the necessary scaling factors with which to compute properly rescaled likelihood ratio tests.

- **`CLASS` statement:**
  - The **`CLASS`** statement is used to list model predictors to be treated as *categorical*, which here means that SAS will automatically create all necessary pairwise contrasts by which to estimate all possible mean differences across groups (unique values) of these variables. By default, SAS uses the highest number or last string alphabetically as the reference group and creates contrasts for each other possible value relative to that reference group. Although it is possible to change this default by using a data sorting option, it will likely be more efficient to re-code your predictor variable so that your desired reference group is indeed the highest numeric or last alphabetic value. Note that *alphabetic* also refers to the use of any value labels through a **`FORMAT`** statement (which is why I repeated the number for the value label in my **`PROC FORMAT`** examples earlier, to keep the groups in the same original order).
  - As discussed in the chapter 2 appendix, predictor variables listed on the **`CLASS`** statement are marginalized over any interacting variables when estimating the interacting variables' main effects for the Type 3 Tests of Fixed Effects. For example, consider an interaction of **`demgroup*agegroup`**, in which **`demgroup`** is listed on the **`CLASS`** statement but **`agegroup`** is NOT on the **`CLASS`** statement and is thus being treated as a continuous predictor. In the Type 3 Tests of Fixed Effects, the main effect of **`agegroup`** will refer specifically to its slope *averaged across all levels of* ***`demgroup`*** (i.e., not just for the reference **`demgroup`**), but the main effect of **`demgroup`** will refer to the omnibus group mean difference just for the specific reference **`agegroup`** = 0. In

contrast, if **agegroup** was also listed on the **CLASS** statement and was thus also treated as a categorical predictor, then the main effect of **demgroup** will refer to the omnibus group mean difference *averaged across all levels of* ***agegroup*** (i.e., not just for the reference **agegroup** = 0). However, in the Solution for Fixed Effects, all main effects for predictors that are part of an interaction are their simple effects at the reference group/value of their interacting predictors, regardless of whether the predictors are on the **CLASS** statement or not. Thus, the tests of main effects provided in the Solution for Fixed Effects and in the Type 3 Tests of Fixed Effects may not agree when interaction terms are included in the model because they can refer to different main effects (simple or marginal) depending on which predictors are on the **CLASS** statement.

- Only predictor variables that are listed on the **CLASS** statement can have their model-predicted means per group requested via **LSMEANS** statements (see below).
- When used in **ESTIMATE** or **CONTRAST** statements, predictor variables listed on the **CLASS** statement must have a value listed for each possible level of their variable (i.e., each distinct group; see examples below).
- No options should be necessary for most uses of the **CLASS** statement. All variables should be listed individually before the semi-colon that terminates the **CLASS** statement. For example: **CLASS demgroup age;**

- **MODEL statement:**
  - The **MODEL** statement is used to list the outcome variable to be predicted (immediately following **MODEL**), and then after an equal sign, all fixed effects of model predictors. The fixed intercept is included in the model by default to create at least an empty means model, and should not be listed with the other predictors (but including the **INT** option after the **/** explicitly lists the fixed intercept; see the **NOINT** option below).
  - Any predictors also listed on the **CLASS** statement will be treated as *categorical*, such that all possible contrasts between a reference group and each other group will be estimated, whereas any predictors not listed on the **CLASS** statement will be treated as *continuous*, such that their slopes will be estimated instead. This distinction (*categorical* means on the **CLASS** statement, *continuous* means not on the **CLASS** statement) is also maintained when predictors are listed on the **CONTRAST**, **LSMEANS**, and **ESTIMATE** statements described below.
  - Listing predictor variables individually requests their main effects only. Interaction effects can also be requested by listing the variables to be part of an interaction, each joined by an asterisk. For example:
  - To predict the outcome **cognition** from main effects of **age85** and **grip9**:

```
MODEL cognition = age85 grip9 / ;
```

  - To predict the outcome **cognition** from main effects of **age85**, **grip9**, and their two-way interaction:

```
MODEL cognition = age85 grip9 age85*grip9 / ;
```

  - To predict the outcome **cognition** from main effects of **age85**, **grip9**, **sexMW**, as well as their three possible two-way interactions:

```
MODEL cognition = age85 grip9 sexMW age85*grip9 age85*sexMW grip9*sexMW / ;
```

  - To add their three-way interaction to the previous model:

```
MODEL cognition = age85 grip9 sexMW age85*grip9 age85*sexMW grip9*sexMW
                  age85*grip9*sexMW / ;
```

  - To add a main effect and interaction of **sexMW** with **demgroup** to the previous model:

```
MODEL cognition = age85 grip9 sexMW age85*grip9 age85*sexMW grip9*sexMW
                  age85*grip9*sexMW demgroup demgroup*sexMW / ;
```

  - Alternatively, main effects and interactions can be specified more efficiently using the **|** operator combined with an **@x** to indicate the maximum order **x** of their effects to be estimated. I did not use this specification in my examples gives its lesser transparency, but it can be more convenient because the **|** operator automatically includes any lower-order main effects and interactions nested in a higher-order interaction. For example:

- To predict the outcome **cognition** from main effects of **age85** and **grip9**:
```
MODEL cognition = age85|grip9@1 / ;
```
- To predict the outcome **cognition** from main effects of **age85**, **grip9**, and their two-way interaction:
```
MODEL cognition = age85|grip9@2 / ;
```
- To predict the outcome **cognition** from main effects of **age85**, **grip9**, **sexMW**, as well as their three possible two-way interactions:
```
MODEL cognition = age85|grip9|sexMW@2 / ;
```
- To add their three-way interaction to the previous model:
```
MODEL cognition = age85|grip9|sexMW@3 / ;
```
- If you include more than one series of joined effects, it will omit any redundant effects across the multiple series. For example, to add a main effect and interaction with **sexMW** for **demgroup** to the previous model:
```
MODEL cognition = age85|grip9|sexMW@3 sexMW|demgroup@2 / ;
```
- Here are the **MODEL** statement options that have been used in my examples, as indicated by these keywords after the **/** on **MODEL** statement (all should be listed before the semi-colon terminating the **MODEL** statement):
  - **NOINT** Omit the global fixed intercept in multivariate models (or **INT** is included by default)
  - **SOLUTION** Print fixed effects solution (not printed by default)
  - **CL** Print upper and lower confidence intervals for fixed effects (not printed by default) Change from 95% by modifying default option **ALPHA=.05** to another desired value
  - **CHISQ** Print χ^2-value in Type 3 Tests of Fixed Effects (prints only *F*-value by default) Note that numerator degrees of freedom * *F*-value = χ^2-value in Type 3 Tests
  - **DDFM=** Method for calculating denominator degrees of freedom (see chapter 5) Examples use **Satterthwaite** for multilevel models or **BW** for single-level models (**BW** is irrelevant then, but corresponds to an ANOVA-based decomposition)
  - **COVB** Print asymptotic covariance matrix of the fixed effects (used for regions of significance)
  - **OUTPM=** Lists the name of data file to which to save new columns for each case's model-predicted outcome based on the fixed effects only (called **Pred**), its standard error (called **StdErrPred**), and the difference from actual outcome (called **Resid**); the squared correlation between the **Pred** and actual outcome column is then the model total R^2
  - **OUTP=** Lists the name of data file to which to save new columns for each case's model-predicted outcome based on the fixed AND random effects (called **Pred**), its standard error (called **StdErrPred**), and the difference from actual outcome (called **Resid**); this would be used to compute level-1 residuals for each case, but NOT to compute model total R^2
- For example, a complete **MODEL** statement with options after the **/** before the terminating semi-colon:
```
MODEL cognition = age85|grip9|sexMW@3 sexMW|demgroup@2
                  / SOLUTION CL CHISQ DDFM=Satterthwaite;
```

- **CONTRAST statement:**
  - The **CONTRAST** statement is used to request custom hypothesis tests of many different kinds. In my examples, I tend to use **CONTRAST** statements for multivariate Wald tests of the significance of multiple fixed effects at once or for custom interaction contrasts. Explanations and examples of each will be shown below. Note that the **ESTIMATE** statement (see below) can also be used to test a contrast with one degree of freedom.
  - Each **CONTRAST** statement needs a user-defined label of up to 200 characters included in single or double quotes (which is helpful for documentation more generally).

- Although others exist, I have only used two options (i.e., SAS keywords after a **/**) for the **CONTRAST** statement (all options should be listed before the semi-colon terminating the **CONTRAST** statement):
  - **CHISQ** requests that the χ^2-value is printed in addition to the default *F*-value (in which numerator degrees of freedom * *F*-value = χ^2-value, so this is only relevant for multiple degree-of-freedom contrasts)
  - **E** requests that the mapping of values to groups for categorical predictors be printed (see below)
- An example of a multivariate Wald test is the traditional test of the significance for the variance accounted for by a regression model (i.e., the *F*-test of the model R^2). For *continuous* predictors (those not on the **CLASS** statement), each fixed effect to be included is listed followed by a **1** (i.e., to "turn it on" because you want to test 1 fixed effect for it). Commas are used to separate each effect in order to indicator the degrees of freedom for the test.
- For example, given the following model with main effects of two *continuous* predictors (i.e., that are not listed on the **CLASS** statement, and for which slopes are estimated), **CONTRAST** can provide a test of their joint significance (i.e., test of the significance of the model R^2) with two degrees of freedom as follows:

```
MODEL cognition = age85 grip9 / SOLUTION CL CHISQ DDFM=Satterthwaite;
CONTRAST 'Model R2 Test' age85 1, grip9 1 / CHISQ;
```

- If the main effect of a *categorical* predictor (i.e., as listed on the **CLASS** statement, such that mean differences across unique levels or groups are estimated) with two groups is added to the model (such as **sexMW** below), then the group difference must be indicated through the use of the contrast **-1** and **1** instead. One value for each possible group must be listed after the predictor. The original values of the categorical predictor do not matter, as SAS simply refers to their numeric or alphabetic order (i.e., given **sexMW** values of **0** or **1**, **0** is first and **1** is second; given value labels of **M** and **W**, M is first and W is second). The **CONTRAST** statement below now provides a joint test of significance of the model predictors with three degrees of freedom as follows:

```
CLASS sexMW;
MODEL cognition = age85 grip9 sexMW / SOLUTION CL CHISQ DDFM=Satterthwaite;
CONTRAST 'Model R2 Test' age85 1, grip9 1, sexMW -1 1 / CHISQ;
```

- If the main effect of a *categorical* predictor (i.e., as listed on the **CLASS** statement) with three groups is added to the model (such as **demgroup** below), then the differences among those three groups must be indicated through the use of two separate contrasts involving **-1** and **1** instead. One value for each group must be listed after the predictor in both contrasts, but which specific groups get the **-1** and **1** doesn't matter, so long as the two contrasts are indeed different. More generally, the number of separate contrasts needed is equal to one fewer than the number of groups; this set of group contrasts is otherwise known as an "omnibus" test of mean differences in an ANOVA context. Given the interchangeability of the contrast values, either **CONTRAST** statement below provides an equivalent joint test of significance of the model predictors with five degrees of freedom as follows:

```
CLASS sexMW demgroup;
MODEL cognition = age85 grip9 sexMW demgroup
                  / SOLUTION CL CHISQ DDFM=Satterthwaite;
CONTRAST 'Model R2 Test version 1' age85 1, grip9 1, sexMW -1 1,
          demgroup -1 1 0, demgroup -1 0 1 / CHISQ E;
CONTRAST 'Model R2 Test version 2' age85 1, grip9 1, sexMW 1 -1,
          demgroup 0 -1 1, demgroup 1 0 -1 / CHISQ E;
```

- The **CONTRAST** statement can also be used to test custom contrasts within an interaction involving one or more *categorical* predictors (i.e., as listed on the **CLASS** statement). In the example below, the interaction of the two-group **sexMW** predictor and the three-group **demgroup** predictor is represented by the six values after their interaction term, in which the order of the values is determined by the order of these predictors on the **CLASS** statement. The first three contrasts request the simple main effect of **sexMW** within each level of **demgroup**. The next three contrasts then request how those simple main effects of **sexMW** differ across levels of **demgroup** (i.e., single degree-of-freedom interactions within the omnibus two degree-of-freedom **sexMW*demgroup**

interaction), which can be found by literally subtracting the contrast values between the relevant lines. Note that the option **/ E** is used to print the mapping of values to groups in the output so you can be sure it is correct:

```
CLASS sexMW demgroup;
MODEL cognition = age85 grip9 sexMW demgroup
                  / SOLUTION CL CHISQ DDFM=Satterthwaite;
* Value order for sexMW (s) and demgroup (d): s1d1 s1d2 s1d3 s2d1 s2d2 s2d3;
* First, sexMW simple effects within each demgroup;
CONTRAST 'Sex diff in demgroup 1' sexMW*demgroup -1  0  0  1  0  0 / E;
CONTRAST 'Sex diff in demgroup 2' sexMW*demgroup  0 -1  0  0  1  0 / E;
CONTRAST 'Sex diff in demgroup 3' sexMW*demgroup  0  0 -1  0  0  1 / E;
* Next, sexMW simple effect differences across demgroups;
CONTRAST 'Sex diff: demgroup 1v2' sexMW*demgroup -1  1  0  1 -1  0 / E;
CONTRAST 'Sex diff: demgroup 1v3' sexMW*demgroup -1  0  1  1  0 -1 / E;
CONTRAST 'Sex diff: demgroup 2v3' sexMW*demgroup  0 -1  1  0  1 -1 / E;
```

- **ESTIMATE statement:**
  - The **ESTIMATE** statement is used to obtain estimates, standard errors, and *p*-values for significance tests against a null hypothesis of 0 for model-implied effects of a single degree of freedom, such as for model-predicted outcomes (intercepts), comparisons of coefficients, or simple effects within higher-order interaction terms (the logic of which is presented in more detail in chapter 2).
  - Each **ESTIMATE** statement needs a user-defined label of up to 200 characters included in single or double quotes (which is helpful for documentation more generally).
  - Here are the options that have been used in my examples, as indicated by these keywords after the **/** on the **ESTIMATE** statement (all options should be listed before the semi-colon terminating the **ESTIMATE** statement):
    - **CL** requests printing of upper and lower confidence intervals for estimated means (not printed by default). You can change the default level of 95% by modifying the option **ALPHA=.05** to another desired value.
    - **E** requests that the mapping of values to groups for categorical predictors be printed (see example below).
    - **DIVISOR** allows for the specification of fractional values, such that whole numbers can be given in the **ESTIMATE** statement as the numerator, and the denominator can then be given using **DIVISOR=** . This is usually only relevant for categorical predictors (i.e., as listed on the **CLASS** statement; see examples below).
  - Similar to their listing on the **CONTRAST** statement, only one value should be given in an **ESTIMATE** statement for each *continuous* predictor (i.e., those not listed on the **CLASS** statement for which slopes are estimated), whereas one value must be given for each unique level of a *categorical* predictor (i.e., those listed on the **CLASS** statement for which mean differences across groups are estimated instead).

  - Below is an example of how to request model-predicted outcomes (i.e., intercepts), in which the values given should indicate the corresponding values for the predictor variables in the model equation:

```
CLASS sexMW demgroup;
MODEL cognition = age85 grip9 sexMW demgroup
                  / SOLUTION CL CHISQ DDFM=Satterthwaite;
ESTIMATE 'Cognition for age85=-5, grip9=3, sexMW=M, demgroup=2'
          intercept 1 age85 -5 grip9 3 sexMW 1 0 demgroup 0 1 0 / E;
ESTIMATE 'Cognition for age85=-5, grip9=3, sexMW=W, demgroup=3'
          intercept 1 age85 -5 grip9 3 sexMW 0 1 demgroup 0 0 1 / E;
```

  - Note that in requesting model-predicted outcomes, the intercept should always be included with a value of **1** (because the fixed intercept always contributes once to any predicted outcome), and values should be specified for every other predictor in the model. Otherwise, any *continuous* predictor without a value given is assumed to be

held to `0` (i.e., held to its reference value), but the same is not true for any *categorical* predictors (i.e., those listed on the `CLASS` statement), for which the mean across groups is then assumed instead. For instance, the following would result in an estimate averaged across levels of `demgroup` instead of for the reference `demgroup`:

```
CLASS sexMW demgroup;
MODEL cognition = age85 grip9 sexMW demgroup
                  / SOLUTION CL CHISQ DDFM=Satterthwaite;
ESTIMATE 'Cognition for age85=-5, grip9=3, sexMW=W, demgroup=Mean'
          intercept 1 age85 -5 grip9 3 sexMW 0 1 / E;
```

- This practice of averaging across levels (groups) of a *categorical* predictor is made more explicit below, in which the `DIVISOR` option is used to assign values of `1/3` to each possible level of `demgroup` (and thus all other values given must be multiplied by `3` to counter-act the divisor):

```
CLASS sexMW demgroup;
MODEL cognition = age85 grip9 sexMW demgroup
                  / SOLUTION CL CHISQ DDFM=Satterthwaite;
ESTIMATE 'Cognition for age85=-5, grip9=3, sexMW=W, demgroup=Mean'
          intercept 3 age85 -15 grip9 9 sexMW 0 3 demgroup 1 1 1 / E DIVISOR=3;
```

- Below is an example of how to compare coefficients using an `ESTIMATE` statement. Note that in order for such comparisons to make sense, the coefficients should be on the same scale (i.e., a one-unit change means the same thing in both predictors). The `ESTIMATE` statement below provides the test of the difference in the slope of one additional year since birth (via `YearsSinceBirth`) and the slope of one additional year further away from death (via `YearsUntilDeath`), in which the common unit of prediction is one year:

```
MODEL cognition = YearsSinceBirth YearsUntilDeath
                  / SOLUTION CL CHISQ DDFM=Satterthwaite;
ESTIMATE 'Difference in year coefficients' YearsSinceBirth -1 YearsUntilDeath 1;
```

- Here is another example of comparing coefficients but in a different context. In this example, the three-group `demgroup` categorical predictor that was on the `CLASS` statement has instead been represented by two manually-created dummy-coded continuous predictors of `demNF` and `demNC` (for none vs. future and none vs. current, as described in chapter 2). The `ESTIMATE` statement below provides the missing comparison of future vs. current, which is the difference of `demNF` and `demNC` coefficients created by the subtraction of their values:

```
CLASS sexMW;
MODEL cognition = age85 grip9 sexMW demNF demNC
                  / SOLUTION CL CHISQ DDFM=Satterthwaite;
ESTIMATE 'Future vs. Current when demgroup is not categorical' demNF -1 demNC 1;
```

- Below is an example of how to request model-implied simple effects of a two-way interaction among *continuous* predictors (i.e., those not listed on the `CLASS` statement). Each `ESTIMATE` statement below contains the simple effect to be estimated and how it is modified by any interacting predictors as specified in the `MODEL` statement. The interaction terms in the `ESTIMATE` statements should be read in pieces—whichever is the simple effect of interest is given a value of `1`, and the value given for the interacting predictor creates the new conditional simple effect, in which positive values create addition and negative values create subtraction. In the example below, the first two `ESTIMATE` statements build an `age85` slope by adding its simple effect given directly by the model (when `grip9`=0) to the interaction value for how the `age85` slope changes per unit of `grip9`, whereas the second two `ESTIMATE` statements build a `grip9` slope by adding its simple effect given directly by the model (when `age85`=0) to the interaction value for how the `grip9` slope changes per unit of `age85`. Note that because `sexMW` and `demgroup` do not interact with `age85` or `grip9` in the `MODEL` statement, they should not be included in their simple effect `ESTIMATE` statements:

```
CLASS sexMW demgroup;
MODEL cognition = age85 grip9 sexMW demgroup age85*grip9
                  / SOLUTION CL CHISQ DDFM=Satterthwaite;
* Simple effects of age85 conditional on grip9;
ESTIMATE 'Age85 Slope if Grip9=-3' age85 1 age85*grip9 -3;
ESTIMATE 'Age85 Slope if Grip9= 3' age85 1 age85*grip9  3;
* Simple effects of grip9 conditional on age85;
ESTIMATE 'Grip9 Slope if Age85=-5' grip9 1 age85*grip9 -5;
ESTIMATE 'Grip9 Slope if Age85= 5' grip9 1 age85*grip9  5;
```

- Below is an example of how to request model-implied simple effects of a two-way interaction of a *continuous* predictor with a *categorical* predictor. The logic is the same, but the categorical predictor must have values given for each of its unique levels (and thus the **/ E** option is added to print the mapping of values to groups in the output). The first set of **ESTIMATE** statements provide the simple effect of **age85** for each level of **demgroup** (via the placement of the **1** for **age85*demgroup**), whereas the second set provides the differences in the simple effects of **age85** between levels of **demgroup** (via the placement of the **–1** and **1** for **age85*demgroup**). The third set provides the simple effects of **demgroup** for specific levels of **age85**, in which the contrast values of **–1** and **1** are multiplied by the values for **age85** (note that when **age85**= **–5**, the signs reverse accordingly):

```
CLASS sexMW demgroup;
MODEL cognition = age85 grip9 sexMW demgroup age85*demgroup
                  / SOLUTION CL CHISQ DDFM=Satterthwaite;
* Simple effects of age85 conditional on demgroup;
ESTIMATE 'Age85 Slope if DemGroup=1' age85 1 age85*demgroup 1 0 0 / E;
ESTIMATE 'Age85 Slope if DemGroup=2' age85 1 age85*demgroup 0 1 0 / E;
ESTIMATE 'Age85 Slope if DemGroup=3' age85 1 age85*demgroup 0 0 1 / E;
* Differences of simple effects of age85 between levels of demgroup;
ESTIMATE 'Age85 Slope: DemGroup=1vs2' age85*demgroup -1  1  0 / E;
ESTIMATE 'Age85 Slope: DemGroup=1vs3' age85*demgroup -1  0  1 / E;
ESTIMATE 'Age85 Slope: DemGroup=2vs3' age85*demgroup  0 -1  1 / E;
* Simple differences for demgroup conditional on age85;
ESTIMATE 'Demgroup 1vs2 if Age85=5'  demgroup -1  1 0 age85*demgroup -5  5 0 /E;
ESTIMATE 'Demgroup 1vs3 if Age85=5'  demgroup -1  0 1 age85*demgroup -5  0 5 /E;
ESTIMATE 'Demgroup 2vs3 if Age85=5'  demgroup  0 -1 1 age85*demgroup  0 -5 5 /E;
ESTIMATE 'Demgroup 1vs2 if Age85=-5' demgroup -1  1 0 age85*demgroup  5 -5 0 /E;
ESTIMATE 'Demgroup 1vs3 if Age85=-5' demgroup -1  0 1 age85*demgroup  5 0 -5 /E;
ESTIMATE 'Demgroup 2vs3 if Age85=-5' demgroup  0 -1 1 age85*demgroup  0 5 -5 /E;
```

- Given the number of distinct values to be specified in an **ESTIMATE** statement to represent all possible unique group combinations, simple effects within interactions among *categorical* predictors (i.e., those included on the **CLASS** statement) are much easier to specify using **LSMEANS**, as shown next.

- **LSMEANS statement:**
  - The **LSMEANS** statement is used to request model-predicted means for *categorical* predictors only (i.e., those listed on the **CLASS** statement). Similar to their listing on the **MODEL** statement, listing a categorical predictor by itself on the **LSMEANS** statement requests its marginal means, whereas listing its interaction with another categorical predictor requests the mean for each unique combination of groups implied by the interaction (i.e., cell means or simple means). For example:
  - To request only the marginal means for **sexMW** and for **demgroup**:

```
CLASS sexMW demgroup;
MODEL sexMW demgroup / SOLUTION CL CHISQ DDFM=Satterthwaite;
LSMEANS sexMW demgroup / ;
```

  - To request only the cell means for each unique combination of **sexMW** by **demgroup**:

```
CLASS sexMW demgroup;
```

```
MODEL sexMW demgroup sexMW*demgroup / SOLUTION CL CHISQ DDFM=Satterthwaite;
LSMEANS sexMW*demgroup / ;
```

- To request both marginal means and cell means for each unique combination of **sexMW** by **demgroup**:

```
CLASS sexMW demgroup;
MODEL sexMW demgroup sexMW*demgroup / SOLUTION CL CHISQ DDFM=Satterthwaite;
LSMEANS sexMW demgroup sexMW*demgroup / ;
```

- To request both marginal means and cell means for each unique combination of **sexMW** by **demgroup**:

```
CLASS sexMW demgroup;
MODEL sexMW|demgroup@2 / SOLUTION CL CHISQ DDFM=Satterthwaite;
LSMEANS sexMW|demgroup@2 / ;
```

- The **LSMEANS** statement has several useful options for decomposing interactions. Here are the options that have been used in my examples, as indicated by these keywords after the **/** on the **LSMEANS** statement (all options should be listed before the semi-colon terminating the **LSMEANS** statement):
  - **CL** requests printing of upper and lower confidence intervals for estimated means (not printed by default). You can change the default level of 95% by modifying the option **ALPHA=.05** to another desired value.
  - **DIFF=ALL** requests all possible pairwise comparisons of the requested means. Unadjusted *p*-values are printed by default, but more conservative *p*-values can be obtained by adding the **ADJUST=** option.
  - **SLICE** is used to request conditional main effects for interacting predictors for each unique level of the predictor listed on **SLICE**. That is, **SLICE** means "by" or "per". For example, the first **SLICE** will provide the omnibus main effect of **sexMW** for each level of **demgroup**, and the second **SLICE** does the opposite:

```
CLASS sexMW demgroup;
MODEL sexMW|demgroup@2 / SOLUTION CL CHISQ DDFM=Satterthwaite;
LSMEANS sexMW|demgroup@2 / SLICE=demgroup SLICE=sexMW;
```

  - **AT** is used to list continuous predictors (those not on the **CLASS** statement) and their values at which to hold constant in estimating means for levels of the categorical predictors. For example, to hold **age85** at **5** and **grip9** at **3** in estimating conditional cell means and differences for each combination of **demgroup*sexMW**:

```
CLASS sexMW demgroup;
MODEL sexMW|demgroup@2 age85 grip9 / SOLUTION CL CHISQ DDFM=Satterthwaite;
LSMEANS sexMW|demgroup@2 / DIFF=ALL AT(age85 grip9) = (5 3);
```

  - **AT** can be used to hold all continuous predictors at their individual means instead (the default):

```
CLASS sexMW demgroup;
MODEL sexMW|demgroup@2 age85 grip9 / SOLUTION CL CHISQ DDFM=Satterthwaite;
LSMEANS sexMW|demgroup@2 / DIFF=ALL AT MEANS;
```

- **RANDOM statement:**
  - The **RANDOM** statement is used to indicate the structure of the **G** matrix of random effects variances and covariances. If you omit the **RANDOM** statement, then there is no **G** matrix in your model, such that the total variance and covariance across observations would be given in the **R** matrix only. When using a **RANDOM** statement, then the total variance and covariance across observations will be created in the **V** matrix from **G**, **R**, and **Z**, as $\mathbf{V} = \mathbf{ZGZ}^T + \mathbf{R}$, as described in chapter 5. Thus, without the **RANDOM** statement, $\mathbf{V} = \mathbf{R}$.
  - The following options have been used in my examples, as indicated by these keywords after the **/** on the **RANDOM** statement (all options should be listed before the semi-colon terminating the **REPEATED** statement):
    - **G** Print the **G** matrix of random effects variances and covariances in the output
    - **GCORR** Print the standardized **G** correlation matrix in the output
    - **V** Print the **V** matrix of total combined variances and covariances in the output

- **VCORR** Print the standardized **V** correlation matrix in the output
- **SOLUTION** Print the predicted random effects per higher-level unit in the output (Note: this can greatly increase estimation time)
- **SUBJECT=** Identification variable for how rows of data should be joined together (random per what?) (ID variable is treated as categorical, so it should be listed on the **CLASS** statement) Note that multiple ID variables can be specified as joined by an asterisk to create a unique value for each case (e.g., **SUBJECT=GroupID*PersonID**)
- **TYPE=** Structure of the **G** matrix, which should ALWAYS be **TYPE=UN** for unstructured in order to estimate covariances among random effects at the same level (see chapter 5), even though the default is **TYPE=VC** for uncorrelated random effects instead

- In multilevel models, you must explicitly list any effects for which you want to estimate a random effect variance, including the intercept (which is not included by default). Notably, to prevent estimation problems, all random effects should be for continuous predictors only (i.e., that are not listed on the **CLASS** statement). If you want to estimate a random effect variance for differences across levels of a categorical predictor, those differences should be represented via manual contrasts treated as continuous predictors (not listed on the **CLASS** statement) instead.
  - For example, to allow only the intercept to vary randomly across levels of **PersonID**:

```
CLASS PersonID;
MODEL outcome = time / ;
RANDOM INTERCEPT / G GCORR TYPE=UN SUBJECT=PersonID;
```

  - For example, to allow both the intercept and time slope to vary randomly across levels of **PersonID**:

```
CLASS PersonID;
MODEL outcome = time / ;
RANDOM INTERCEPT time / G GCORR TYPE=UN SUBJECT=PersonID;
```

- **REPEATED statement:**
  - The **REPEATED** statement is used to indicate the structure of the **R** matrix of variances and covariances (which are *residual* if a **RANDOM** statement has been included, but *total* otherwise). If you do not explicitly include a **REPEATED** statement, by default the model still includes a **TYPE=VC** diagonal **R** matrix, in which all observations are assumed to have the same residual/total variance with no covariances.
  - The following options have been used in my examples, as indicated by these keywords after the **/** on **REPEATED** statement (all options should be listed before the semi-colon terminating the **REPEATED** statement):
    - **R=X** Print the **R** matrix of variances and covariances in the output for the order of case given by **X** (otherwise the **R** matrix for the first case is printed by default if **=X** is omitted)
    - **RCORR=X** Print the standardized **R** correlation matrix in the output for the order of case given by **X** (otherwise the **RCORR** matrix for the first case is printed by default if **=X** is omitted)
    - **SUBJECT=** Identification variable for how rows of data should be joined together (ID variable is treated as categorical, so it must also be listed on the **CLASS** statement) Note multiple ID variables can be specified as joined by an asterisk to create a unique value for each case (e.g., **SUBJECT=GroupID*PersonID**)
    - **TYPE=** Structure of **R** matrix (see chapter 4 for details), such as: **VC** for Variance Components, **CS** for Compound Symmetry, **CSH** for Compound Symmetry Heterogeneous, **AR(1)** for First-Order Auto-Regressive, **ARH(1)** for First-Order Auto-Regressive Heterogeneous, **TOEP(X)** for Toeplitz, **TOEPH(X)** for Toeplitz Heterogeneous, and **UN(X)** for Unstructured, in which **X** = # rows/columns (default is all possible)

- In longitudinal data, you may list a time-level identification variable by which to organize the rows and columns of the **R** matrix (this ID variable must also be listed on the `CLASS` statement). This distinction is only relevant in the presence of missing data (in which case you want the correct value to go to the correct row and column). For example, to structure the **R** matrix by the time-level variable `studyday` per unique level of `PersonID`:

```
CLASS studyday PersonID;
REPEATED studyday / R RCORR TYPE=UN SUBJECT=PersonID;
```

- **`ODS OUTPUT` statement:**
  - The `ODS OUTPUT` statement is used to list the SAS output tables to be saved to SAS data file, as can then be utilized to conduct further calculations (such as in my macro programs).
  - For example, to save two different output tables to separate SAS data files using generic names:

```
ODS OUTPUT SASoutputname1 = SASdatafilename1 SASoutputname2 = SASdatafilename2;
```

  - The names of each specific SAS output table can be found in the online SAS documentation for any `PROC`. Alternatively, before estimating a model, include the command `ODS TRACE ON;`. All of that model's output tables will then be listed in the log, and the `Name:` listed under each `Output Added:` entry is how you refer to the SAS output table in the `ODS OUTPUT` statement in order to save it to a SAS data file.
  - Here are the `ODS` tables that are used in my examples, along with my own naming conventions for them:
    - `CovParms = CovModelname` — Covariance Parameter Estimates (model for the variance)
    - `SolutionF = FixModelname` — Solution for Fixed Effects (model for the means)
    - `InfoCrit = FitModelname` — Information Criteria (for fit statistics and number of parameters)
    - `COVB = CovBModelname` — Asymptotic Covariance Matrix for Fixed Effects
    - `SolutionR = VariableURandom` — Predicted random effects per higher-level unit
  - The use of these `ODS` tables in my custom SAS macros is presented next.

- **A general note about my custom SAS macro programs at the beginning of the chapter example files:**
  - Most of my data example SAS syntax files begin with macro programs I have written to automate otherwise manual tedious calculations, as listed below. Each of these works by requesting the necessary user input explicitly as well as saving the relevant output tables into SAS data files, manipulating their content and performing the necessary calculations, and then printing the summary results to the output window. As noted in the comments that start these initial macro sections, nothing needs to be changed to use them with `PROC MIXED`. Instead, the analyst can invoke them for use with specific models by "calling" them—by referring to the name of the macro and inserting its required arguments, as described in more detail for each specific macro below.
- **Required macro arguments (information and corresponding SAS output tables to be saved via `ODS OUTPUT`):**
  - `%ModelR2`: used in chapter 2 example only to compute single-level model R^2 relative to the empty means model (and incremental R^2 between nested models)
    - `CovBase=` Name of `ODS CovParms` table for baseline empty model
    - `CovFewer=` Name of `ODS CovParms` table for nested model
    - `CovMore=` Name of `ODS CovParms` table for comparison model
  - `%Regions`: used in chapter examples 2 and 7a to compute regions of significance for fixed effect interactions
    - `FixData=` Name of `ODS SolutionF` table that stores fixed effects for model
    - `CovBData=` Name of `ODS COVB` table that stores XTX inv matrix for model

- **Pred=** Case-sensitive name of predictor effect regions are for
- **Mod=** Case-sensitive name of moderator effect (for region values)
- **ModCenter=** Centering point of moderator predictor
- **Interact=** Case-sensitive name of interaction effect
- **Order=** Order of entry of interaction in **MODEL** statement

- **%FitTest**: used in chapter example 3b and later to conduct likelihood ratio tests between nested models
  - **FitFewer=** Name of **ODS InfoCrit** table for nested model
  - **FitMore=** Name of **ODS InfoCrit** table for comparison model
- **%PseudoR2**: used in chapter examples 5 and later to compute the proportion reduction in each variance component between nested models
  - **NCov=** Total # of entries in covariance parameter estimates table
  - **CovFewer=** Name of **ODS CovParms** table for nested model
  - **CovMore=** Name of **ODS CovParms** table for comparison model
- **%TotalR2**: used in chapter examples 7a and later to compute multilevel model total R^2 (and incremental R^2 between nested models)
  - **DV=** Case-sensitive name of dependent variable
  - **PredFewer=** Name of **OUTPM=** data file of predicted outcomes for nested model
  - **PredMore=** Name of **OUTPM=** data file of predicted outcomes for comparison model

Model-Specific Notes by Chapter Example

- **A note about model fit, degrees of freedom, and information criteria across programs:**
  - As presented in chapters 3 and 5, **AIC** is computed as AIC = −2LL + 2*df, in which *df* = degrees of freedom. But within most programs (including SPSS and SAS but excluding STATA), the degrees of freedom used in computing AIC are different when using ML versus REML estimation. In ML, degrees of freedom include ALL model parameters (fixed effects in the model for the means plus all variances, covariances, and other variance model parameters), but will exclude the fixed effects when using REML. Although this may seem arbitrary, it makes sense if you remember that models that differ in fixed effects cannot have their fit compared using −2LL, AIC, or BIC when using REML (and thus the number of fixed effects is irrelevant for those statistics).
  - As also presented in chapters 3 and 5, **BIC** is computed as BIC = −2LL + Log(N)*df, in which *df* = degrees of freedom (which differs between ML and REML for the same programs as noted for AIC), and *N* refers to some kind of sample size. But what value of *N* is used differs by program. In SPSS and MPLUS, *N* = the total number of observations, whereas *N* = the sample size at the highest level of the model in SAS. More specifically, in SAS *N* is the number of unique repetitions of the **V** matrix, which means that *N* = 1 in models with crossed random effects. In STATA, the user can tell the program what *N* should be (and in my examples, I follow the SAS convention). Consequently, BIC will differ across these programs for any multilevel model.

- **Chapter 2**

- Because chapter 2 includes single-level models only, no **REPEATED** or **RANDOM** statements were specified. The default **REPEATED** statement if one is not provided is a diagonal **R** matrix (**TYPE=VC**) in which the residual variance is assumed constant over all observations with no covariance across observations.
- The method of fake people was used as a means to create predicted outcomes for use in plotting interactions. In this method (as introduced in chapter 2), a data file of fake cases with desired values of the model predictors is created and merged into the real data, and predicted outcomes are then requested for each case. The fake cases will not contribute to the model given that they do not have values for the outcome variable, but they will receive model-predicted values you can then plot.

- **Chapter 3a**
  - The data to be read in were initially in multivariate (wide) format, in which one row contains all observations for a person. The initial data step restructured the data into stacked (long) format of one row per occasion per person, as is needed for **PROC MIXED**.

- **Chapter 3b**
  - The data to be read in were already in stacked format, so no further data transformation was necessary.
  - Note that the time ID variable, **session**, in the **MODEL** statement also appears in the **CLASS** statement, such that it is being treated as a categorical predictor, thus estimating all possible mean differences across sessions. The **session** ID variable is also used to structure the **R** matrix in the **REPEATED** statement in each model.

- **Chapter 4**
  - The data to be read in were already in stacked format, so no further data transformation was necessary.
  - Multiple likelihood ratio tests were requested after some models via separate calls to the **%FitTest** macro.
  - Note that only the **REPEATED** statement was used for the **R**-only models, whereas both the **RANDOM** and **REPEATED** statements were used for the **G** and **R** combined models (to control **G** and **R**, respectively).

- **Chapter 5**
  - The data to be read in were already in stacked format, so no further data transformation was necessary.
  - Given the use of both the **G** and **R** matrices, both **RANDOM** and **REPEATED** statements were used. However, the **REPEATED** statements in which **TYPE=VC** were technically unnecessary to include, as that is already the default.

- **Chapter 6**
  - The data to be read in were already in stacked format, so no further data transformation was necessary, except for the creation of the specific time predictors needed (e.g., linear and quadratic time, piecewise time slopes).
  - The negative exponential models could not be estimated using **PROC MIXED**, and thus maximum likelihood within **PROC NLMIXED** was used instead, which differs from the usual **PROC MIXED** syntax in several respects:
    - In the **PROC NLMIXED** statement, **METHOD=GAUSS** and **TECH=NEWRAP** was used to request adaptive Gaussian quadrature with Newton-Raphson optimization for any model requiring numeric integration. In addition, **GCONV=1e-12** was used to request tighter convergence criteria than required by default (to ensure the estimated parameters are as close to their most likely values as possible).
    - The **PARMS** statement is used to give all parameters to be estimated user-defined names and to set their starting values for estimation. Any omitted parameters are assumed to start at a default value of **1**. As noted in the comments, I listed fixed effects in the model for the means in the first line (each parameter that starts with

an **f**) and variance model parameters in the second line. However, note that the parameters are estimated in a standard deviation metric instead (and are squared to become variances later in the syntax) to prevent estimation problems resulting for these particularly large variance numbers.

- The level-2 and level-1 equations were then written explicitly in the syntax using the user-defined fixed effects from the **PARMS** statement. Individual random effects were also included, but not a level-1 residual.
- In the **MODEL** statement, the outcome is listed followed by a **~** and its assumed distribution (**normal** in this case), the parameters for which are then given in parentheses. Thus, the conditional mean of the normal distribution for the RT outcome is given per case by **PredY** (defined previously, which includes the random effects), with conditional residual variance given by the squared **sdE** parameter.
- The **RANDOM** statement then identifies the distribution and associated parameters for any user-defined random effects. Accordingly, the random effect U's in the level-2 equations are said to have a multivariate normal distribution (**~ normal**), in which each U random effect has a **0** mean, followed by their **G** matrix of variances and covariances (in the same order as listed in the Covariance Parameter Estimates table in **PROC MIXED**). The **SUBJECT=** option is used to provide an identification variable for how rows of data should be joined together (i.e., random per what unit).
- The **ESTIMATE** statements serve the same purpose as in **PROC MIXED**, but are constructed differently, in that the addition or subtraction of model parameters must be written explicitly. Notably, unlike **PROC MIXED**, these can include variance model parameters, as were requested by squaring the SD estimates here.

- **Chapter 7a**
  - The data to be read in were already in stacked format, so no further data transformation was necessary, except for the creation of the specific centered predictors needed.
  - Because time was unbalanced, the **REPEATED** statement was not used given that it was never anything other than the default **TYPE=VC** (the other types are for balanced data).
  - The heterogeneous variance models were estimated using maximum likelihood within **PROC NLMIXED** in order to create a log-linear model for the prediction of both the random intercept and residual variances. Note that variances were estimated rather than standard deviations (as used in chapter 6) given the scale of the outcome. The **CONTRAST** statements function the same as in **PROC MIXED**, but are written with explicit multiplication of the **1**.

- **Chapter 7b**
  - The data to be read in were initially in multivariate (wide) format, in which one row contains all observations for a person. The initial data step restructured the data into stacked (long) format of one row per occasion per person, as is needed for **PROC MIXED**.
  - The method of fake people was again used as a means to create predicted outcomes for use in plotting interactions.
  - Because time was unbalanced, the **REPEATED** statement was not used given that it was never anything other than the default **TYPE=VC** (the other types are for balanced data).

- **Chapter 8**
  - The data to be read in were already in stacked format, so no further data transformation was necessary, except for the creation of the specific centered predictors needed.
  - All level-1 predictors were treated as observed variables that were not included in the likelihood.
  - **ESTIMATE** statements were used to create predicted outcomes instead of the method of fake people.

- The heterogeneous variance models were estimated using maximum likelihood within **PROC NLMIXED** in order to create a log-linear model for the prediction of both the random intercept and residual variances. Note that variances were estimated rather than standard deviations (as used in chapter 6) given the scale of the outcome.

- **Chapter 9**
  - The data to be read in were initially in multivariate (wide) format, in which one row contains all observations for a person. The initial data step restructured the data into stacked (long) format of one row per occasion per person, as is needed for **PROC MIXED**.
  - **PROC MIXED** was used to estimate the multivariate longitudinal models both time-varying variables (the risky behavior outcome and the monitoring predictor) were included in the likelihood, such that their random intercept, random linear age slope, and residual variances were partitioned by the model. However, this was only possible for the undirected effects version (in which the relationships between monitoring and risky behavior were specified using covariances rather than directed effects). Although it may be possible to estimate the directed effects version using **PROC NLMIXED**, I have only been able to estimate directed effects at level 2, not at level 1 (see my last attempt in example 5f in PSYC 945 from Spring 2014).
  - The slopes-as-outcomes models were also estimated using **PROC MIXED**. Note that these models were included primarily for pedagogical purposes and are NOT recommended (see chapter 9).
    - First, for the random effects version, the predicted random effects per person were requested via the **SOLUTION** option on the **RANDOM** statement and then saved to a SAS data file via an **ODS OUTPUT** for **SolutionR**. The fixed effects were then added back in to create the actual intercepts and slopes given by the level-2 equations (i.e., as needed given that random effects are deviations from their fixed effects).
    - Second, for the fixed effect version, the general fixed intercept was removed from the model via the **NOINT** option on the **MODEL** statement. **PersonID** was treated as a categorical predictor, such that the main effect of **PersonID** and its interaction with **agec18** then created the predicted intercept and age slope for each person, which were saved to a SAS data file via an **ODS OUTPUT** for **SolutionF**.

- **Chapter 10a**
  - The data to be read in were already in stacked format, so no further data transformation was necessary, except for the creation of the specific centered predictors needed.
  - The method of fake people was again used as a means to create predicted outcomes for use in plotting interactions. Note that missing data were deliberately left for cases with impossible values (i.e., that would create dead people).
  - Because time was unbalanced, the **REPEATED** statement was not used given that it was never anything other than the default **TYPE=VC** (the other types are for balanced data).

- **Chapter 10b**
  - The data to be read in were already in stacked format, so no further data transformation was necessary, except for the creation of the specific centered predictors needed.
  - The method of fake people was again used as a means to create predicted outcomes for use in plotting.
  - The three-level models required two **RANDOM** statements (to structure the **G** matrix for levels 2 and 3), which can be distinguished by the ID variables given in their **SUBJECT=** options. In this example, **SUBJECT=PersonID** indicated level 3 (for unique persons), whereas **SUBJECT=PersonID*burst** indicated level 2 (for the unique combination of person and burst). An equivalent way of denoting the nesting for level 2 would be **SUBJECT=burst(PersonID)** for the nesting of bursts within persons.

- Although it could have been used given balanced sessions at level 1, the `REPEATED` statement was not used for the univariate models given that it was never anything other than the default `TYPE=VC` in this example.
- For the multivariate model, the `REPEATED` statement was used to structure the **R** matrix for repeated `DV` outcomes within a session (within a burst within a person) via `SUBJECT=PersonID*burst*session`.

- **Chapter 11a**
  - The data to be read in were in multivariate format with respect to waves within students, but were stacked with respect to students within classes. A significant amount of data manipulation was required to create all the summary variables needed with which to build the model predictors, resulting in several intermediate data files. These were then used to summarize data at the different levels of analysis and for some specific models as well.
  - The three-level models again required two `RANDOM` statements (to structure the **G** matrix for levels 2 and 3), which can be distinguished by the ID variables given in their `SUBJECT=` options. In this example, `SUBJECT=ClassID` indicated level 3 (for unique classes), whereas `SUBJECT=ClassID*StudentID` indicated level 2 (for the unique combination of class and student). An equivalent way of denoting the nesting for level 2 is `SUBJECT=StudentID(ClassID)` for the nesting of students within classes.
  - The saturated means models included **G** and **R** matrices for classes and students that were unstructured across occasions, such that the `RANDOM` statement did not include an intercept, and instead included the categorical predictor of `wave` (as did the `REPEATED` statement).
  - All level-1 and level-2 predictors were treated as observed variables that were not included in the likelihood.

- **Chapter 11b**
  - The data to be read in were in multivariate format with respect to waves within students, but were stacked with respect to students within classes. A significant amount of data manipulation was required to create all the summary variables needed with which to build the model predictors, resulting in several intermediate data files. These were then used to summarize data at the different levels of analysis.
  - The specification of fixed effects of year-specific classes relied on a similar logic as in the multivariate models. Because the class effects are nested within years, the differences between classes in a given year were created through what look like interaction terms between year-specific indicators and class ID categorical predictors (e.g., `ClassID_year0*aclass0`), but in which the indicator variables function as a "switch" to only allow the class differences for the relevant cases. The year-specific effects of class grade were specified this way as well.
  - The models for crossed random effects of year-specific classes included one `RANDOM` statement per year (whose `SUBJECT=` is the class ID for that year), but none of them included a general random intercept. Instead, each included a random effect for its year-specific indicator variable, which created a random intercept that was switched on or off for each case as needed (see chapter 11 for more explanation). Because these custom random intercepts were included in separate `RANDOM` statements, no covariances were estimated among them.
  - Note that in the models with crossed random effects, the number of subjects reported in the Dimensions output table is 1, which indicates the number of unique repetitions of the **V** matrix (i.e., there is one observation for each unique pairing of student and class in those models).

- **Chapter 12**
  - The data to be read in were already in stacked format, so no further data transformation was necessary, except for the creation of the specific centered predictors needed.
  - The method of fake people was again used as a means to create predicted outcomes for use in plotting.

- The models for crossed random effects for subjects and items included one **RANDOM** statement for each level-2 part of the **G** matrix, which can be distinguished via the ID variables in their **SUBJECT=** option. A general random intercept was included for both subjects and items, as well as random effects for item predictors across subjects. However, the use of two separate **RANDOM** statements means that any random effects for subjects will not have covariances with any random effects for items, but that random effects from the same dimension will have covariances given the **TYPE=UN** option on each **RANDOM** statement.
- Note that in the models with crossed random effects, the number of subjects reported in the Dimensions output table is 1, which indicates the number of unique repetitions of the **V** matrix (i.e., there is one observation for each unique pairing of subject and item in those models).