

```
332
333
334 *****w*****CHALLENGE 12 - PUT YOUI
335
336 VARIABLE LABELS BP_Class 'Blood P
337
338 VALUE LABELS BP_Class
339 -2 'Low blood pressure'
340 -1 'Isolated Diastolic Hypotension'
341 0 'Healthy Range'
342 1 'Prehypertension'
343 2 'Stage 1 Hypertension'
344 3 'Pre-ISH'
345 4 'Isolated Systolic Hypertension'
346 5 'Stage 2 Hypertension'
347 6 'Hypertensive Crisis'.
348
349 FORMATS BP_Class(F1.0).
350
351 VARIABLE LEVEL BP_Class(ORDINAL)
352
353 *****AAA*****CHALLENGE 12 - PUT YOUI
354
355
356
```

ABSTRACT

This lesson covers data transformation and setting up your dataset ahead of analysis. It focuses on the development of good labeling and organizational skills.

Ayanda M. Masilela
CSSCR Workshops (2019)

INTRO TO SYNTAX IN SPSS PART 1

Data Transformations

TO USE THIS DOCUMENT

The document is arranged in sections which can be viewed in the Table of Contents. It contains syntax for the reader to copy and paste directly into the SPSS syntax file. The reader may also repeat the text by hand to emulate writing script.

You will be prompted on which syntax clusters you will need to copy or type into your syntax file with text that reads “*** COPY INTO SYNTAX FILE” after the “Sample Syntax” title.

Sample Syntax 7 *** COPY INTO SYNTAX FILE
FREQUENCIES chol /histogram normal.

Do not copy syntax into the file if not prompted.

There will be ample space between challenge sections for you to type and copy your own syntax.

EN...	17	
EN...	18	
EN...	19	
EN...	20	
EN...	21	
EN...	22	*****w*****CHALLENGE 1 - PUT YOUR SYNTAX HERE*****w*****
J 1**...	23	
RMATL...	24	
EN...	25	
EN...	26	
EN...	27	*****a*****CHALLENGE 1 - PUT YOUR SYNTAX HERE*****a*****
EN...	28	
EN...	29	
EN...	30	
EN...	31	
EN...	32	
EN...	33	
EN...	34	
EN...	35	
EN...	36	
EN...	37	
EN...	38	
EN...	39	
EN...	40	*****w*****CHALLENGE 2 - PUT YOUR SYNTAX HERE*****w*****
EN...	41	
EN...	42	
EN...	43	
EN...	44	
EN...	45	*****a*****CHALLENGE 2 - PUT YOUR SYNTAX HERE*****a*****
EN...	46	
EN...	47	
EN...	48	

Challenge sections do not immediately have copyable syntax in this document. Rather, a separate file for challenge syntax has been included in the file archive. Refer to it if you are unable to complete the challenge questions on your own. Your syntax may not match the challenge syntax, as you may have used a different method to accomplish a goal. This is fine. What's most important is that you find a method that best suits how you work within the software.

YOU WILL NEED

- SPSS 19 – this tutorial has not been tested on later generations of SPSS

BEFORE GOING FURTHER....

Do not save the .sav file!!! That will make changes permanent. Especially as you are practicing you may make mistakes. If you save those mistakes, they can't be undone.

If you make a mistake, you can simply reload the original file and repeat all of the successful syntax you had performed up to that point.

That being said... save your syntax after every successful cluster of code!

TABLE OF CONTENTS

1.....	GETTING STARTED
1...	Setting up file locations
1...	Working Across the Three SPSS Windows
2...	Why do we use syntax?
3...	Workflow
4...	Syntax Structure
7...	Working with the [Paste] Button in the GUI
8.....	PRELIMINARY TRANSFORMATIONS
8...	Labeling Variables and Values, and Observing Outcomes with Descriptive Functions
11...	Frequencies and Descriptives: Basic Organization and Analytical Tricks
13...	Crosstabs: An Overview of Categorical Relationships
14.....	BUILDING CHARTS AND THE VALUE OF [PASTE]
12...	Pasting Syntax and Making a Scatterplot
15...	A Quick Regression Analysis
17.....	EXTEA SECTION: THE EVOLUTION OF CHART SYNTAX
17...	Building Charts and Graphs with Legacy Dialogs
17...	Saving Chart Templates
18...	Comparing Chart Builder to Legacy Dialogs Graphs
20.....	DATA TRANSFORMATIONS: RECODING VARIABLES, ASSIGNING NEW VALUES, AND VARIABLE COMPUTATION
20...	Recoding Strings to Numerical Representations
23...	Recoding Numerical Representations into Other Numerical Representations
24...	Autorecode
25...	Recoding with IF
28...	Recoding Ranges of Values
29...	Computing New Variables
32...	Recoding/Computing Ranges of Values with IF
34...	DO IF
36...	Decision-Making and Representation: When Core Assumptions Leave Cases Out and How to Fix It.
43...	Challenge Grand Finale

GETTING STARTED

Go to the following links and read through the contents.
Keep these tabs open as a reference.

- <http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/Cdiabetes.html>
- <http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/diabetes.html>

Setting up file locations

- Go to the following website and download the files from the SPSS Syntax section:
<http://students.washington.edu/ayandm/tutfiles/SPSSstutfiles.html>
1. Download the first file archive from the SPSS Syntax section, Diabetes.zip
 - a. Right-click the Diabetes.zip file and “Unzip” it into your downloads folder, or a preferred working location

Be aware that as you set up your first line of code to retrieve the file, you will need to set up the appropriate pathname.

Working Across the Three SPSS Windows

1. Launch an instance of SPSS 19
 - a. At the top-left, click File → Open → Syntax
 - i. Find your syntax file, and click [Open]
 1. Diabetes_Student.sps is for you to type and/or paste syntax into as you follow along
 - a. Open this one for good practice
 2. Diabetes_Workbook.sps is the master file containing all of the correct syntax
 - ii. Feel free to begin with a completely blank syntax file if you prefer
 1. File → New → Syntax

This will launch a Syntax window and a blank data widow. At the top of the document, we'll open with the "Get FILE" function. It allows you to quickly launch an SPSS dataset (.sav file). It also lets you refresh a dataset quickly if you conduct a transformation that damages the dataset. Note that not all transformations can be undone!

- b. Look directly above at the item that says "Active" [DataSet 0]. Note how that tag matches that of the blank dataset.
- c. You'll need to set up the correct file path.

Sample Syntax 1 *** COPY INTO SYNTAX FILE
--

<code>get FILE=" C:\Users\[YOUR NET ID]\Downloads\Diabetes\diabetes.sav".</code>
--

- d. Highlight the entirety of the "get file" function in Line 1, and then press the green "play" button above, or type "Ctrl+R".

This will populate the data window, and will also bring up your output window. This results of this action aren't exactly exciting, but with the output window you can monitor the successes and failures of your code. You should now have three windows to monitor: Data, Output, and Syntax. As you develop your skills, you will be spending the bulk of your time in the Syntax and Output windows.

You can also track the success of your changes using the Data window. The Variable tab in the Data window is especially helpful as you begin calculating new variables, updating labels, and managing missing values. As you carry out these operations, changes will appear in the structure of your dataset and will be viewable in the Data and Variable views.

Why do we use syntax?

1. *It preserves the structure of the original dataset* – in some situations you don't want to overwrite the original. In fact, back up the original if you can.
2. *Sharing your process and memories* – Syntax keeps a paper trail of your logic in terms of how you have chosen to arrange your dataset, construct graphs, and carry out analyses
3. *It is repeatable* – You can execute all of these same commands on a different dataset with all of the same variables, or quickly swap out variables when applying similar methodologies to different datasets
 - a. Imagine receiving a batch of 300 new cases from a study that you have been managing for two years. You don't have to reinvent the wheel with organization or analytical techniques!
4. *It is secure* – Rather than sending whole datasets across the web, you can share your syntax with work partners working on the same or similar datasets hosted on their own servers. No need to move data more times than is necessary!

5. *It is light-weight and easy to share* – The file size is small and transfers through email or FTP swiftly. This is great for collaborative work on datasets that are too large or too confidential to transfer via the internet.

Workflow

1. Label the variables and values that you know
2. Conduct Data Transformations
3. Label and organize these newly calculated values
4. Conduct Analysis + Build Charts

Label the Features That You Know

Each dataset will come with variables and values for those variables. Sometimes the variable names and value names are more or less intuitive, but in most cases it is best to ascribe labels to make your outputs and charts easy to read. Do this early!

Conduct Data Transformations

This is where much of your organizational time will go. Data transformations, such as recoding or computing new variables, will help you to correctly format and label your dataset. Recoding is especially important in this case. Analytical functions are often not available for use on “string” datatypes. Thus, recoding strings to a numerical representation is essential for carrying out these more advanced operations.

Additionally, sometimes your data doesn’t arrive formatted or tabulated in a way that enables you to ask the right questions. It can take a little tinkering to get things in order.

Label and Organize These Newly Calculated Values

In the process of recoding, computing, etc., you will create new variables that will need to be tagged in an intuitive way. Fortunately, by the time you reach this step, you will have already gotten some practice from step 1. Be sure to make these changes immediately! There is a lot of work to be done, and backtracking to make corrections can be challenging and furthermore result in disorganized, unintuitive code.

Conduct Analyses + Build Charts

After working through the organizational process, it’s finally time to run analyses and build tables. These two functions have been paired because sometimes you can take advantage of

charts for exploratory analyses before initiating your deep dive. You can also build charts once you've conducted your analysis, and your results expose results that you find to be compelling.

Syntax Structure

Syntax has a few basic demands in order to function. Most importantly, every cluster of code must be ended with a period (.).

Sample Syntax 2
<code>get FILE="U:\Public Health SPSS\Diabetes\diabetes.sav".</code>

In some cases, you need to carry out multiple operations under one command. Everything that appears between the text “VARIABLE LABELS” and the “.” Are clustered into one operation.

Sample Syntax 3	
VARIABLE LABELS	Opens the command to being labeling variables
id 'Subject ID'	Assigns a label to the corresponding variable name
chol 'Total Cholesterol'	*
stab.glu 'Stabilized Glucose'	*
hdl 'High Density Lipoprotein'	*
glyhb 'Glycosylated Hemoglobin'.	* Note that the period in the last line closes the VARIABLE LABELS command.

Sample Syntax 3 (Alternative)	
VARIABLE LABELS id 'Subject ID' chol 'Total Cholesterol' stab.glu 'Stabilized Glucose' hdl 'High Density Lipoprotein' glyhb 'Glycosylated Hemoglobin'.	

Sample Syntax 3 and 3 (Alternative) carry out the same function, but the formatting is somewhat different. 3 separates each labeling operation line by line. Sometimes it is easiest to structure it this way for clarity and ease of editing, but either method is fine. 3 (Alternative) is another common way of arranging syntax. You may receive or use tutorial syntax in either format, but be aware that they carry out the same functions.

Another nuance to be aware of is how SPSS manages strings. In Sample Syntax 3, you will notice that the strings in the operation are surrounded with single quotes. Whether, such as in the case above, the string is being applied to a label or if a value is a string, it MUST be surrounded in single quotes, otherwise SPSS will not let the operation run.

When dealing with numerical representations of variables, you do not have to surround those entries in quotes.

Sample Syntax 4	
IF (frame_code = 2) FC2=3.	Opens an IF command converting a numerical value to another numerical value
IF (frame_code = 3) FC2=2.	Note that each IF command is immediately closed with a period.
IF (frame_code = 4) FC2=1.	*
IF (frame_code = 1) FC2=999.	*
MISSING VALUES FC2 (999).	Assigns the “missing” characteristic to the value 999

You can also add comments by surrounding text with an asterisk (*). With comments, you can add notes documenting what your syntax does. Comments will not execute as functions. Rather, if they are highlighted and run, the text will simply be printed in the output window. You will know that a line has been successfully commented if it turns gray.

Sample Syntax 5
This is a coment.

Note that you cannot add functional code in the line immediately following a comment. You will need to have at least one line break after the comment to begin using active code.

Sample Syntax 6
<p>X - Incorrect</p> <p>231: *This is a comment*</p> <p>232: IF (frame_code = 2) FC2=3.</p>
<p>✓ - Correct</p> <p>231: *This is a comment*</p> <p>232:</p> <p>233: IF (frame_code = 2) FC2=3.</p>

In addition to the main command, SPSS also reads subcommands. They appear below the main function and are always led with a slash (/) and appear between the main command and the (period) that closes the overall argument. These subcommands enable you to add to or dictate the parameters of an output.

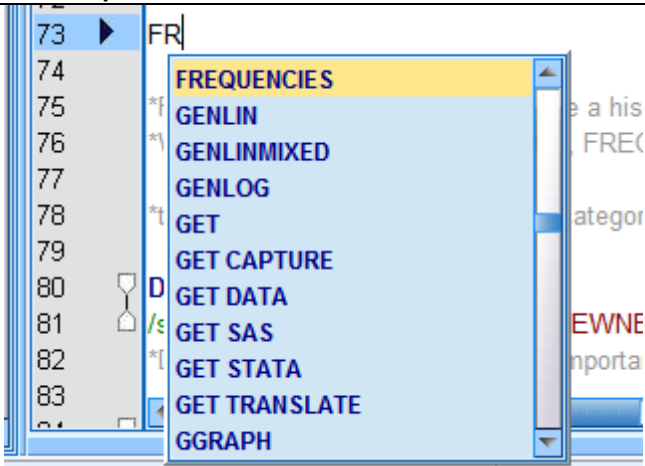
Sample Syntax 7	
FREQUENCIES chol	Main command
/histogram normal.	Subcommand with a functional request for additional parameters

Sample Syntax 7 *** COPY INTO SYNTAX FILE
FREQUENCIES chol
/histogram normal.

In this case, we are running a frequency on the variable “chol”. With the subcommand, we are asking SPSS to output this frequency analysis with a histogram. The parameter within the subcommand, “normal”, tells SPSS to include a normal curve on the requested histogram.

Colorful Syntax

As you can see from Sample Syntax 7, the text is quite colorful. This is all significant. When a function has been successfully enclosed, the initial command will be **bold and blue**. Subcommands are /green, and are led with a slash (/). The parameter of the green subcommand is **dark reddish**. Variable names and values maintain plain black text. If you’ve typed in your syntax correctly, it should adopt all of these colors. If something is amiss, such as a missing period, the main command will usually turn **bold and bright red**.

Quick Tip 1	
	<p>As you begin typing commands or subcommands into the syntax editor, a drop list of functions will populate. Take advantage of this to prevent spelling errors. Additionally, when you begin working with subcommands, only parameters that are functional within the subcommand will populate the drop menu.</p> <p>Enjoy!</p>

Working with the [Paste] Button in the GUI

Working in the Graphical User Interface (GUI) does not make you a charlatan. However, it is still important to preserve your syntax, and therefore your process, at all times. The [Paste] button makes this possible.

1. Working with [Paste] – look to the navigation bar at the top-left
 - a. Analyze → Descriptive Statistics → Descriptives
 - i. Add the variable “ratio” to the Variable(s) dialog box
 - ii. Click [Options]
 1. Fill the check boxes for Skewness and Kurtosis
 2. Click [Continue]
 - iii. In the main window, click [Paste]

Look to the bottom of your syntax file. It should have pasted syntax for the Descriptives function you just activated in the GUI.

Sample Syntax 8 (Matches the [Paste] output)

```
DESCRIPTIVES VARIABLES=chol
  /STATISTICS=MEAN STDDEV MIN MAX KURTOSIS SKEWNESS.
```

You will not know every single syntax line and arrangement for every single function, and that is okay! This takes time to learn, and sometimes getting a little help from the GUI to catalog your progress is worthwhile. Especially on more complex tasks such as creating graphs, it can become extremely difficult to remember and cleanly execute every last syntax detail.

Sample Syntax 8 (Short Hand)

```
DESCRIPTIVES chol
  /STATISTICS MEAN STDDEV MIN MAX KURTOSIS SKEWNESS.
```

- Here, you can see a little bit of shorthand.
- This rendition doesn't have the “Variables=” text. Either is fine, but typically when typing out syntax, the short hand form will most often be used.

PRELIMINARY TRANSFORMATIONS

Labeling Variables and Values, and Observing Outcomes with Descriptive Functions

Copy and paste the following lines of code.

Sample Syntax 8 *** COPY INTO SYNTAX FILE
DESCRIPTIVES ratio.

We have an output for a variable called “ratio.” The authors have already carried out one calculation for the "ratio" variable. It used chol/hdl to calculate the HDL ratio. HDL ratio is used as a predictor for heart disease.

- The ideal ratio is 3.5
- Ideally, non-HDL will be lower than 130.
- Healthy HDL ranges are 35 to 65 mg/dL for men
- Healthy HDL ranges are 35 to 80 mg/dL for women.

But before we get ahead of ourselves, let's practice applying intuitive labels to our variables.

Sample Syntax 9 *** COPY INTO SYNTAX FILE
VARIABLE LABELS ratio 'Cholesterol/HDL Ratio'.
DESCRIPTIVES ratio.

Check the output. The label should now be more intuitive. We can do this for all of our variables. We'll use the author's labeling system to generate our own for the dataset. Fortunately, with syntax we don't have to do this one-by-one like we would in the GUI.

Sample Syntax 10 *** COPY INTO SYNTAX FILE
VARIABLE LABELS id 'Subject ID' chol 'Total Cholesterol' stab.glu 'Stabilized Glucose' hdl 'High Density Lipoprotein' glyhb 'Glycosylated Hemoglobin'.

Quit Tip 2: When running multiple lines of code separated by a period, you must highlight every line and then hit RUN.

```

12
13 FREQUENCIES frame
14 /histogram normal.
15
16 RECODE frame ('small'=1)('medium'=2)('large'=3)('=999) INTO frame_code.
17 MISSING VALUES frame_code(999).
18 VARIABLE LEVEL frame_code(Ordinal).
19 EXECUTE.
20
21 VALUE LABELS frame_code 1 'Small' 2 'Medium' 3 'Large' 999 'Missing'.
22

```

“Why can't we just change the variable names?”

SPSS is very picky about the character length, use of spaces, and special characters for variables. We instead use labels to apply more intuitive listings for our entries. When we create outputs, SPSS will then substitute those new labels for a more readable output. These labels also automatically apply themselves to charts.

Note that when you are carrying out syntax, you will use the variable name, NOT the value label. Variable names are short and sweet for this reason.

Sample Syntax 10 (Alternative)

```
VAR LAB id 'Subject ID' chol 'Total Cholesterol' stab.glu 'Standardized Glucose' hdl 'High Density Lipoprotein' glyhb 'Glycosylated Hemoglobin'.
```

Here, you can see another sample of shorthand in the usage of “VAR LAB”.

CHALLENGE 1

Task 1: Apply Variable Labels to the Remaining Variables

Hint: Use the author’s notes as your guide for developing label names

You will make corrections to the following variables.

```

bp.1s
bp.1d
bp.2s
bp.2d
time.ppn

```

Type your syntax between the “CHALLENGE 1 - PUT YOUR SYNTAX HERE” lines in the Syntax file.

Take a moment to save your syntax file!

Frequencies and Descriptives: Basic Organization and Analytical Tricks

Frequencies are a great way to get an overview of your dataset. The results can be similar to that of the DESCRIPTIVES function, but with a few nuances.

Sample Syntax 11 *** COPY INTO SYNTAX FILE

```
FREQUENCIES chol
/histogram normal.
```

FREQUENCY allows you to produce a histogram, and performs most of the same analytical tasks as DESCRIPTIVES upon request. Descriptives does not produce a histogram.

*When working with categorical data, FREQUENCY is best. When working with continuous data, FREQUENCY or DESCRIPTIVES can serve your needs.

CHALLENGE 2

Task 1: Try running a FREQUENCY on a categorical variable

Hint: Examine the “Data” view to identify string variables

If you tried to run FREQUENCIES with a /HISTOGRAM on a categorical variable of string values, you will have received an error message

Warnings

```
frame is a string so a histogram cannot be produced.
```

The /HISTOGRAM function can only tabulate categorical variables if they have a numerical representation. Not to worry, for now. We will address this issue later as we work on recoding data.

DESCRIPTIVES performs another important function - that of calculating Z-Scores. We will explore that function later

Sample Syntax 12 *** COPY INTO SYNTAX FILE

```
DESCRIPTIVES chol
/STATISTICS mean MIN max stddev SKEWNESS KURTOSIS.
```


You can also do a basic ANOVA exploration with the MEANS function by applying the “/STATISTICS” subcommand. This is unique because it gives you access to the foundations of an analytical technique while working with strings. It is however limited in that this method does not allow for Post-Hoc testing.

Sample Syntax 13 *** COPY INTO SYNTAX FILE

<pre>MEANS chol by location /STATISTICS.</pre>
--

Let’s quickly review the unique outputs for “Frequencies” versus “Descriptives”

Sample Syntax 14 *** COPY INTO SYNTAX FILE

<pre>FREQUENCIES weight /HISTOGRAM normal.</pre>
--

Sample Syntax 15 *** COPY INTO SYNTAX FILE

<pre>DESCRIPTIVES weight /STATISTICS mean stddev skewness kurtosis /SAVE.</pre>

- FREQUENCIES is unique in that it allows you to develop a chart on-the-fly using the Histogram function.
 - Unfortunately, DESCRPTIVES does not allow this function.
- DESCRPTIVES does however offer the rapid calculation of Z-Scores using the “/SAVE” subcommand.
 - This is a great method for outlier detection.

Not every command has a /SAVE subcommand, and the /SAVE subcommand does something different for each main command, but this particular application is worth remembering!

Take a look at your Data table and sort the dataset DESCENDING and examine the outputs.

- To do this, right-click the column header, and select “Sort Descending”
- Next, sort it ASCENDING.

Do you see any values greater than 3.29 or less than -3.29? If so, those are outliers within the dataset. In brief, 99.9% of data will fall within 3.29 standard deviations of the mean. Anything beyond that is an outlier.

Crosstabs: An Overview of Categorical Relationships

The CROSSTABS command is a great way to investigate proportional relationships between categorical characteristics. Just like how FREQUENCIES and DESCRIPTIVES have special hidden tools, CROSSTABS also has its own collection of tools. With it, you can command a Chi-Square analysis, and also generate charts.

Sample Syntax 16 *** COPY INTO SYNTAX FILE

```
CROSSTABS gender by location
/STATISTICS=CHISQ
/BARCHART.
```

- Your output should convey a rather lengthy table of proportional relationships, as well as a Chi-Square output.
- Additionally, we commanded CROSSTABS to produce a bar chart in quite the same way we asked FREQUENCIES to produce a histogram in the previous section.

CHALLENGE 3

Task 1: Run a FREQUENCY on the following variable: glyhb

Task 2: Run DESCRPTIVES on the following variables: bp.1s and bp.1d

Task 3: Run MEANS on glyhb by frame

Task 4: Run CROSSTABS on gender by frame

Hint: Examine the “Data” view to identify string variables. Take advantage of copy-paste, swapping out the appropriate variables when necessary.

BUILDING CHARTS AND THE VALUE OF [PASTE]

Pasting Syntax and Making a Scatterplot

If SPSS syntax is new to you, memorizing syntax may seem pretty daunting. Starting with the basics is key, and eventually you will work up to being able to set up sophisticated models and data transformations. There are however some situations that are too textually dense for even the most skilled coder to emulate. Sometimes it is simply impractical to use syntax to carry out certain functions. We'll walk through this problem using the Chart Builder, then practice using the [Paste] button.

1. Click Graphs → Chart Builder
 - a. On the bottom-left, click "Scatter/Dot" from the list
 - b. Click and drag the first display box into the large window above
 - i. Click the variable "age" on the left list, and drag it to the X-Axis
 - ii. Click the variable "glyhb" and drag it to the Y-Axis
 - c. Click the Titles/Footnotes tab below the big window
 - i. Fill the check box for "Title 1"
 1. 8 In the "Content" box to the right, type *Glycosylated Hemoglobin by Age*, and click "Apply"
 - d. Click [Paste] in the main window

Your chart editor window will disappear, and an impressive cluster of code will generate at the bottom of your document. You will not be expected to memorize all of this code. For most purposes, it seems impractical to type in this much text from memory, so instead take advantage of the [Paste] button to preserve your syntax.

This is valuable for visualizations because, as noted before, you can standardize the procedures for generating charts across multiple datasets.

2. Highlight the chart syntax at the bottom of the file and run it.

The visualization has generated a scatter plot that plausibly indicates a relationship between age and rising Glycosylated Hemoglobin levels. Note that one measure of Type II Diabetes risk is a Glycosylated Hemoglobin level of 7 or higher.

3. Double-click the visualization window to activate the Chart Editor

- a. Click Options → Add Reference Line to Y-Axis
 - i. Observe the appearance of the dialogue icon. This will add a reference line to the middle of your display
 - ii. Click the reference line, and a Properties window will open
 1. Set the position to 7
 - a. Click [Apply]
 - iii. Add another reference line
 1. Set the position to 5.6
 - a. Click [Apply]
 - iv. If you prefer, you can also add a fit line
 1. Click Elements → Fit Line at Total
 - a. Look to the top-right. That is your R-Square value
 2. Click the new fit line and go to the Properties window
 - a. Click the “Lines” tab, and pick a unique color to make it stand out from the reference lines
 - i. Click [Apply]
- b. Click back to your Outputs window to refresh the chart, then close the chart editor window

A Quick Regression Analysis

The visualization conveys that there may be a positive relationship between increasing age and rising Glycosylated Hemoglobin levels, but we’ll need to run a regression analysis to examine this conjecture further.

The chart builder gave us a quick R-Square, but we need to investigate further. Run the CORRELATIONS command to see if there is a statistical significance to this finding.

Sample Syntax 17 *** COPY THE LEFT BOX INTO SYNTAX FILE	
CORRELATIONS glyhb age.	Note that for this command, “by” is not required in the syntax to relate to variables.

According to this test, a relationship is plausible given the significance value... let's continue with a regression model.

Sample Syntax 18	
REGRESSION	Opens the regression analysis command
/DEPENDENT glyhb	Subcommand that assigns the dependent variable
/METHOD=ENTER age	Subcommand that assigns the independent variable. There are a multitude of methods, so it is important to assign ENTER to control the usage of AGE
/DESCRIPTIVES MEAN STDDEV CORR SIG N.	Subcommand requesting descriptive statistics. You can assign whatever you like!

Sample Syntax 18 *** COPY INTO SYNTAX FILE
REGRESSION /DEPENDENT glyhb /METHOD=ENTER age /DESCRIPTIVES MEAN STDDEV CORR SIG N.

Based on this result, it seems there is a positive correlation, but as we learned previously our R-Square value is quite low. Age simply can't be the only factor, but there is a correspondence here nonetheless.

EXTEA SECTION: THE EVOLUTION OF CHART SYNTAX

You may skip this if you like. Syntax is included in the master file.

Building Charts and Graphs with Legacy Dialogs

Chart syntax wasn't always so complicated. Let's take a step back in time with the "Legacy Dialogs" function. SPSS has seen many iterations, and the authors of the program have done a good job of preserving many of those functions. Here we will examine the evolution of chart builder syntax. As both options are available, you are free to use whichever method you prefer. Be advised that self-help materials online will utilize both current dialogues and legacy dialogues.

4. Go Graphs → Legacy Dialogues → Scatterplot
 - a. Click "Define"
 - i. Assign the variables to the appropriate axes
 - b. Add a title
 - c. Click [Paste]

Sample Syntax 19

GRAPH

```
/SCATTERPLOT(BIVAR)=age WITH glyhb  
/TITLE='Glycosylated Hemoglobin by Age'.
```

5. Now double-click the new graph to open the editor.
 - a. Click Elements → Fit Line at Total. That's your fit line!
6. Go to your Output window and refresh. Stretch the chart to the right a little bit as well.
 - a. Look to the top right. That's your R-Square!

Saving Chart Templates

This is a rare moment when we were able to make a change in the GUI, but there was no offering for pasting syntax. In this case, it is worthwhile to save a chart template. This option IS

NOT accessible through the standard chart builder. Use your judgment as to the best method for managing chart construction across multiple datasets or worksites.

7. Double-click the scatterplot you made using Legacy Dialogs
 - a. At the top-left, click File → Save Chart Template
 - b. Fill the checkbox for “All Settings”
 - c. Click [Continue]
 - i. Save the file like any other, if you wish.

This file can then be used to apply the same structure and appearance to future scatterplots. Again, this only works with Legacy Dialogs, but is a worthwhile trick to remember.

Comparing Chart Builder to Legacy Dialogs Graphs

There’s quite a contrast in terms of textual representation here.

LEGACY DIALOGS
Sample Syntax 19
GRAPH /SCATTERPLOT(BIVAR)=age WITH glyhb /TITLE='Glycosylated Hemoglobin by Age'

CHART BUILDER
Sample Syntax 20
GGRAPH /GRAPHDATASET NAME="graphdataset" VARIABLES=age glyhb MISSING=LISTWISE REPORTMISSING=NO /GRAPHSPEC SOURCE=INLINE. BEGIN GPL SOURCE: s=userSource(id("graphdataset")) DATA: age=col(source(s), name("age")) DATA: glyhb=col(source(s), name("glyhb")) GUIDE: axis(dim(1), label("age")) GUIDE: axis(dim(2), label("Glycosylated Hemoglobin")) GUIDE: text.title(label("GH by Age")) ELEMENT: point(position(age*glyhb)) END GPL.

Take a moment to read through the *Sample Syntax 20*.

You may be able to interpret some of the commands. However, the syntax is extensive and requires a great deal of precision. Take advantage of the [Paste] button to manage these more complex functions. Building to the level that you are able to recall these structures takes time, and SPSS has built-in mechanisms to make sure that you are building and conveying your process clearly. Take advantage of those tools.

DATA TRANSFORMATIONS: RECODING VARIABLES, ASSIGNING NEW VALUES, AND COMPUTATION

Recoding Strings to Numerical Representations

Sample Syntax 20 ***	
FREQUENCIES frame	Outputs a frequency analysis
/format notable	Stops the output from producing tables. This saves space and is useful when working with continuous data
/histogram normal.	Produces a histogram with a normal curve

Sample Syntax 20 *** COPY INTO SYNTAX FILE	
FREQUENCIES frame	
/histogram normal.	
*We'll leave out the /format subcommand this time around	

We got a lot of the outcomes that we wanted, but due to "frame" being a string, SPSS was unable to generate a histogram. Furthermore, there were some blank values.

- We'll need to recode frame to a numerical representation.
- We'll also need to assign a value for missing.

Sample Syntax 21 (STRING to NUMERICAL)	
RECODE frame ('small'=1)('medium'=2)('large'=3)(''=999) INTO frame_code.	Opens the recode command and assigns new numerical values to the original strings. Note that the strings are surrounded in single quotes The Parameter INTO dictates the creation of a whole new variable called frame_code. See below on notes about (''=999)
MISSING VALUES frame_code(999).	Assigns the value of 999 to missing values
VARIABLE LEVEL frame_code(Ordinal).	SPSS differentiates nominal, ordinal, and scale variables as “Levels”. The level is set to “Ordinal” since frame size does indicate an increase of some kind
EXECUTE.	Forces the recoded values to populate in the newly generated variable column

Sample Syntax 21 (STRING to NUMERICAL) *** COPY INTO SYNTAX FILE	
RECODE frame ('small'=1)('medium'=2)('large'=3)(''=999) INTO frame_code. MISSING VALUES frame_code(999). VARIABLE LEVEL frame_code(Ordinal)). EXECUTE.	

This was a short line of code, but a lot has happened.

- First, observe that ‘small’, ‘medium’, and ‘large’ are all surrounded in quotes, indicating that they are strings.
- Next, observe the (''=999). SPSS reads blanks as valid entries, so it is necessary to assign a value to the entry in order to then isolate it. SPSS needs explicit instruction.
 - '' with nothing between the quotes represents a blank string, therefore we are telling SPSS that *blank string=999*.
- The MISSING VALUES function tells SPSS how to treat all entries listed as 999. You can assign any value you like, but make sure that your “missing value” tag is intuitive and obvious.
- VARIABLE LEVEL sets the organizational interpretation of the data.
 - Nominal: Codes are simply a label used to differentiate one case or set of cases from another, but otherwise have no other intrinsic quantitative value
 - Ordinal: Codes represent a ranking of some kind from greatest to least, etc.
 - Scale: Indicates continuous significance. There is the possibility of increase or decrease in amounts (distances, etc.) and furthermore these amounts can be broken down into subunits.
- EXECUTE forces data transformations to populate newly generated columns.

- Especially when conducting data transformations, SPSS does not automatically populate newly generated columns with the data you have tabulated.
- In previous steps, we bypassed this by using FREQUENCIES and DESCRIPTIVES to check outputs, which forces the recodes to populate
- EXECUTE resolves this issue, and also does not produce an output table, saving space and time.

This is an important step, but in the process we lost the value labels that tell us intuitively what the numerical recodes mean.

Sample Syntax 22 *** COPY INTO SYNTAX FILE

VALUE LABELS frame_code 1 'Small' 2 'Medium' 3 'Large' 999 'Missing'.
--

Take a look at the Values column for frame_code in the Variable View window. It's okay, but the decimals are a little bit messy. They may even deceive a reader into thinking that this is continuous data. Let's remove the two decimal points so that only the "1", "2", and "3" remain.

Before progressing, observe that the "width" (number slots to the left of the decimal) of frame_code is also 8.

Sample Syntax 23 *** COPY LEFT BOX INTO SYNTAX FILE
--

FORMATS frame_code(F1.0).	
----------------------------------	--

	For many functions, it is important to make sure that there isn't a space between the variable name and the transformation assignment.
--	--

Now check back to the Variable View. Decimals will now be 0, and the width will be 1. This will look much cleaner for your output tables.

Recoding Numerical Representations into Other Numerical Representations

The syntax for converting numerical representations to different numbers is nearly identical. Just be aware that you do not have to use single quotes, as these are not strings.

We'll generate a new variable called `frame_code2`.

Sample Syntax 24 (Numerical to Numerical) *** COPY INTO SYNTAX FILE

```
RECODE frame_code (1=3) (2=2)(3=1)(ELSE=COPY) INTO frame_code2.
MISSING VALUES frame_code2(999).
VARIABLE LEVEL frame_code2(Ordinal).
EXECUTE.
```

There wasn't too much change here, just that the values for 1 and 3 were flipped. If you don't want to change values, you have two options as demonstrated in the `RECODE` line

- You can command that the initial value should equal the converted values (2=2) in this case
- Or you can use the (ELSE=COPY) function
 - Anything that has not been explicitly assigned a new code is simply copied over from the original column
 - In this case, the missing value marker of 999 was copied.
- We then had to repeat the missing value assignment and variable level
- Execute forces the changes

CHALLENGE 4

Task 1: Assign value labels to `frame_code2`

Task 2: Format the new numerical codes such that there are no zeros after the decimal

Hint: The code in *Sample Syntax 24 (Numerical to Numerical)* simply flipped "large" to be 1 and small to be 3.

Since this recode is not essential to the completion of this project, we can delete it before moving on. *This cannot be undone, so use caution when removing variables that are original to the dataset.*

Sample Syntax 25 *** COPY INTO SYNTAX FILE

```
DELETE VARIABLES frame_code2.
```

Autorecode

AUTORECODE is a great option if you have a large batch of data to recode and the order of the data isn't significant. This is also a backdoor way of managing missing string values, but we'll save that for an appendix entry!

This method also has one major advantage...

- Unlike the standard recodes that we did above, AUTORECODE preserves and converts the values of the recoded column into the value labels of the new numerical column.
- If, say, you had entries for all countries in the world from Afghanistan through Zimbabwe. That's over 200! It would be a nightmare to label them by hand.
- AUTORECODE manages the labeling process for you

Sample Syntax 26	
SORT CASES location.	Sorting is essential for automated transformations. This eases SPSS's burden in tabulating which values require a specific input
AUTORECODE location	Opens the autorecode command and applies it to location
/INTO loc_code.	Into establishes the creation of a new variable column
/GROUP	Group ensures that code assignments fall uniformly on the corresponding entries
/PRINT.	Print produces readable text in the output file displaying the outcome of the autorecode

Sample Syntax 26 *** COPY INTO SYNTAX FILE
SORT CASES location. AUTORECODE location /INTO loc_code /GROUP /PRINT.

- Autorecode requires that cases be sorted by the variable upon which the code will be conducted
- Note the change in Syntax versus the RECODE function
 - **/INTO** is now a subcommand
- **/GROUP** instructs SPSS to maintain the same code across all cases with the same characteristics
 - Chicago will always be assigned a 23

- Denver will always be assigned a 55
- The /PRINT command displays the final assignments for the recode. Now check the outcome with a FREQUENCY.

Sample Syntax 27 *** COPY INTO SYNTAX FILE

FREQUENCIES location / histogram normal .
--

Much better! Now we have a histogram and all of the correct labels.

A final note: if there are any missing values, SPSS will assign them with autorecode. You can update that value to any representation that you want, or keep it as-is. Fortunately, with this variable we did not have that problem.

Recoding with IF

As you move independently through your work, you will learn that often the term “Recode” is used even though the syntax for RECODE is not present at all. There are many other ways to generate new variables and representations, and among the most popular is the IF function.

It is possible to use the IF function to carry out the recode of frame to frame_code.

ORIGINAL EXAMPLE

Sample Syntax 21 (STRING to NUMERICAL)

RECODE frame ('small'=1)('medium'=2)('large'=3)(''=999) INTO frame_code. MISSING VALUES frame_code(999). VARIABLE LEVEL frame_code (Ordinal). EXECUTE.
--

We'll generate a new variable called `frame_code3`.

ALTERNATIVE EXAMPLE USING IF	
Sample Syntax 28	
IF (frame='small') frame_code3=1.	IF commands function line-by-line, ending with a period each time
IF (frame='medium') frame_code3=2.	In this case, you are sending multiple commands to populate the same variable column one factor at a time
IF (frame='large') frame_code3=3.	Make sure that when you are running clusters of IF codes, you highlight every line of code and execute them all at once. <i>You don't want to miss something!</i>
IF (frame='') frame_code3=999.	Assigns a missing numerical value to a string
MISSING VALUES frame_code(999).	Assigns the missing characteristic to 999
VARIABLE LEVEL frame_code (Ordinal).	Sets the variable level
EXECUTE.	Forces the population of the new column

Enter the alternative syntax to generate a new variables called `frame_code3`.

ALTERNATIVE EXAMPLE USING IF *** COPY INTO SYNTAX FILE	
Sample Syntax 28	
IF (frame='small') frame_code3=1.	
IF (frame='medium') frame_code3=2.	
IF (frame='large') frame_code3=3.	
IF (frame='') frame_code3=999.	
MISSING VALUES frame_code(999).	
VARIABLE LEVEL frame_code (Ordinal).	
EXECUTE.	

Let's look at some differences...

TABLE 1		
RECODE	VS.	IF
Recode handles the conversions all at once		IF splits the process into single commands that all end in a period
Recode needs the processing instruction INTO		IF utilizes "="
Runs a single lengthy batch of code with one period		Runs several short sections of code, all of which end with a period and therefore must be explicitly run independently

A pseudocode verbalization of the IF structure reads like an "IF-THEN" statement.

"IF frame is equal to small THEN frame_code3 is equal to 1."

Whereas the RECODE pseudocode would read something like...

"RECODE values within the variable frame: small equals 1, medium equals 2, large equals 3, blanks equal 999, INTO frame_code3."

It's a bit harder to think about in such lengthy terms. Perhaps this is why IF is so popular. Regardless, decide what works best for you, as both methods are perfectly viable options.

The IF function is among the most versatile in SPSS. As you master your craft, you will be able to deploy it in a variety of ways. This also means that its documentation can be a bit muddy and difficult to self-teach. Nonetheless, as you become faster at interpreting syntax, the myriad applications of IF will fall into place.

CHALLENGE 5

Task 1: delete the variable frame_code3.

Recoding Ranges of Values

Sometimes it is necessary to recode based on a range of values, not simply a categorical selection. Fortunately, RECODE has this kind of flexibility.

For this example, we'll work on recoding ranges of total cholesterol scores into risk categories. We'll also tidy up the variable and value labels.

Sample Syntax 29	
RECODE chol (LOWEST THRU 200 = 0)(200 THRU 220 = 1)(220 THRU 240=2)(241 THRU HIGHEST = 3)(ELSE = 999) INTO total_chol.	THRU functions much like a simplified logical argument. LOWEST assigns everything up to and including 200 the specified value. HIGHEST assigns everything from 241 and greater the specified value ELSE tells SPSS how to handle anything that hasn't received an explicit transformation assignment. In this situation, the one patient that did not have a lipid profile receives a value of 999.
MISSING VALUES total_chol(999).	Assigns 999 as the missing indicator
VARIABLE LEVEL total_chol(Scale).	Changes variable type to Scale
FORMATS total_chol(F1.0).	Removes decimals and reduces the number of spaces to the left of the decimal to 1
EXECUTE.	Forces the transformation

Sample Syntax 29 *** COPY INTO SYNTAX FILE	
RECODE chol (LOWEST THRU 200 = 0)(200 THRU 220 = 1)(220 THRU 240=2)(241 THRU HIGHEST = 3)(ELSE = 999) INTO total_chol. MISSING VALUES total_chol(999). VARIABLE LEVEL total_chol(Scale). FORMATS total_chol(f1.0). EXECUTE.	
VARIABLE LABELS total_chol 'Total Cholesterol Score Rank'.	
VALUE LABELS total_chol 0 "Optimal" 1 "Borderline High" 2 "High" 3 "Very High".	

An important nuance to remember here is that this method works best for integers. It can be done with decimal value cutoffs, but exercise additional caution if deploying THRU in those cases.

Computing New Variables

Sometimes a simple recode won't generate the analytical angles that we want, so we'll have to compute a new variable. COMPUTE is a powerful function. So powerful that it would be difficult to cover every function in this short overview, so we'll stick with the basics of...

- Applying an operator and a constant to a variable
 - [inches]/12=[feet], for example
- Using two (though potentially more) variables to calculate a new variable

We'll start with a simple unit conversion of height from inches into feet.

Sample Syntax 29 *** COPY INTO SYNTAX FILE

COMPUTE Height_ft=(height/12).

- Here we used a constant as the basis of the computation

Sample Syntax 30 *** COPY INTO SYNTAX FILE

IF missing(height) Height_ft=999.
--

MISSING VALUES Height_ft(999).

EXECUTE.

CHALLENGE 6

Task 1: Sort the dataset by Height_ft.

Next, we'll calculate Body Mass Index with the intention of using it for its original purpose: assessing the health of large groups of people.

CHALLENGE 7	
Task 1: Calculate a new variable called BMI using the BMI formula	
BMI Formula: $703 \times ([\text{weight in pounds}]/[\text{height in inches}^2])$	
Hint: This task simply combines using a constant as well as two variables already present in the dataset	
IMPORTANT: SPSS uses the expression ** (double asterisk) for exponents in equations.	
After you have generated the formula, paste the code in the left box into the syntax file to manage missing variables.	
IF missing(height) OR missing(weight) BMI=999. MISSING VALUES BMI(999). EXECUTE.	Note the inclusion of the logical function “or” in the generation of a missing value placeholder. If either height or weight is missing, the new entry will receive a 999.

We’ll shelf this computation for now and revisit it in the analysis portion.

Now we’ll look into Waist-to-Hip ratio as a marker for health risk. As noted previously, this is a better individual indicator of heart disease risk, Type II Diabetes risk, and fertility issues in some women.

Sample Syntax 31 *** COPY INTO SYNTAX FILE
COMPUTE WH_Ratio=(waist/hip). VARIABLE LABELS WH_Ratio 'Waist-Hip Ratio'.

We didn’t run an EXECUTE this time. Click over to your Data view and look at the newly computed column.

It’s blank! As noted at the beginning of this section, many data transformations do not automatically populate new columns. We either have to tell SPSS to do this with an EXECUTE, or run a computation to force a completion.

Run DESCRIPTIVES to examine the calculation. It looks like there were two missing cases, which is good for a dataset of this size, but we’ll have to manage missing values.

CHALLENGE 8
Task 1: Sort cases by WH_Ratio.

Now click to the data view. We can see the two missing entries. We can correct them with an “IF”.

Sample Syntax 32 * COPY INTO SYNTAX FILE****IF** missing(waist) **OR** missing(hip) WH_Ratio=999.**MISSING VALUES** WH_Ratio(999).**EXECUTE.**

- Observe the logical operator “or”
 - We use this to designate any situation where one or the other variable is missing data
 - That is, if a patient had measured hip but not waist, that case would be tallied as missing

In the first recode section, we had a somewhat different experience with handling missing values. Previously, we had to manage the complication of a blank string being counted as a valid entry. This time, we had to manage a blank numerical entry. SPSS treats these two situations differently. A blank numerical entry is automatically treated as missing, whereas SPSS treats a blank string entry as a valid title. Thus, we needed to use a different but easier method for managing missing numerical data.

Recoding/Computing Ranges of Values with IF

COMPUTE is a versatile tool, but even it has its limits. Again, we can take advantage of IF to compute variables while using logical operators, such as >, <, =, AND, OR... etc.

Go to the following link if you don't already have it open in your browser
<http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/diabetes.html>.

The authors have indicated several risk factors for cardiovascular disease and Type II Diabetes. I think that calculating the diabetes indicator will be the simplest way to start.

- The diagnosis value is for Type II Diabetes is 7
- Prediabetes diagnosis values are between 5.6 and less than 7
- A Glycosylated Hemoglobin level of less than 5.6 is considered healthy

Sample Syntax 33 *** COPY INTO SYNTAX FILE

```
IF (glyhb<5.6) T2D=0.
IF (glyhb>=5.6 AND glyhb<7) T2D=1.
IF (glyhb>=7)T2D=2.
VARIABLE LABELS T2D 'Type II Diabetes'.
VALUE LABELS T2D 0 'Not Diabetic' 1 'Pre-Diabetic' 2 'Diabetic'.
IF missing(T2D) T2D=999.
MISSING VALUES T2D(999).
EXECUTE.
```

- As noted above, each IF line represents one function.
- Be aware of AND and OR logicals
 - You can get some wild results if you lose the wrong operator!

We can dial up the level of selection using multiple variables. We'll revisit the waist-to-hip ratio topic.

First we'll recode risk levels for men

- Low is 0.95 or lower
- moderate is 0.96-1.0
- high is 1.0 or higher

Sample Syntax 34 * COPY INTO SYNTAX FILE**

IF (gender='male' **AND** WH_Ratio<=0.95) WH_Risk=0.

IF (gender='male' **AND** (WH_Ratio>0.95 and WH_Ratio<1)) WH_Risk=1.

IF (gender='male' **AND** (WH_Ratio>=1)) WH_Risk=2.

Run a frequency to force the computation and check your work.

CHALLENGE 9

Task 1: Calculate risk categories using the women's criteria

Task 2: Manage missing values

Task 4: Assign a data type (level)

Task 4: Remove decimals from the code values

Task 5: Check your work with a frequency

Hint: Copy-paste is your friend! Make sure to swap out the correct variables and constant values.

DO IF

DO IF is a powerful conditional tool which can be a little confusing at first but once you have mastered it, it is immensely powerful. It can save time and reduce the amount of text needed to execute large blocks of commands, giving your code a cleaner, less error-prone structure. Similarly to IF, it fits nicely in conjunction with other functions. For this example, we'll run the WH_Ratio process again and derive the same outcome with a different method.

Sample Syntax 35	
DO IF gender = 'male'.	Opens the DO IF Argument. This applies the conditional that the upcoming operations only apply to cases where gender is 'male'
IF (WH_Ratio<=0.95) WH_Risk2=0.	IF-based computation
IF (WH_Ratio>0.95 AND WH_Ratio<1) WH_Risk2=1.	*
IF (WH_Ratio>=1) WH_Risk2=2.	*
ELSE IF gender = 'female'.	Sets the alternative condition that gender = 'female'. Subsequent calculations will apply only to cases where gender is 'female'
IF (WH_Ratio<=0.80) WH_Risk2=0.	IF-based computation
IF (WH_Ratio>0.80 AND WH_Ratio<0.86) WH_Risk2=1.	*
IF (WH_Ratio>=0.86) WH_Risk2=2.	*
COMPUTE WH_Risk2 = missing(WH_Ratio).	Adopts the value for "missing" from WH_Ratio
END IF.	Closes the DO IF Argument. DO IF will remain red until END IF is inserted.
IF missing(WH_Risk2) WH_Risk2=999.	Populates a numerical representation for a missing value
MISSING VALUES WH_Risk2(999).	Assigns the missing value of 999 to WH_Risk2
EXECUTE.	Forces the computation of the new variable column

Sample Syntax 35 * COPY INTO SYNTAX FILE**

```

DO IF gender='male'.
IF (WH_Ratio<=0.95) WH_Risk2=0.
IF (WH_Ratio>0.95 AND WH_Ratio<1) WH_Risk2=1.
IF (WH_Ratio>=1) WH_Risk2=2.
ELSE IF gender='female'.
IF (WH_Ratio<=0.80) WH_Risk2=0.
IF (WH_Ratio>0.80 AND WH_Ratio<0.86) WH_Risk2=1.
IF (WH_Ratio>=0.86) WH_Risk2=2.
END IF.
IF missing(WH_Risk2) WH_Risk2=999.
MISSING VALUES WH_Risk2(999).
EXECUTE.

```

There is some nuance to DO IF. It is structured in an unusual way compared to all of the other Command-subcommand setups that we had explored previously. DO IF has two essential lines and two optional lines.

Essential	DO IF.	Main command that opens the DO IF function <i>and</i> sets forth the first conditional statement that isolates the cases that will receive a computation
Optional	ELSE IF.	Secondary command that sets up another parameter to isolate cases that will receive a computation that is different from the first <ul style="list-style-type: none"> You can use ELSE IF as many times as needed
Optional	ELSE.	Tertiary command that instructs SPSS on how to handle all other values not already addressed in DO IF or an ELSE IF line
Essential	END IF.	Closes the DO IF command.

For the sake of simplicity, this DO IF series was conducted on a single variable only. This is however not a requirement. As you become more skilled in being able to set up conditionals across multiple variables it can aid you in the management of messy datasets and many other applications. For now stick with the basics.

Delete the variable WH_Risk2, as we have already generated an identical variable in WH_Risk.

Sample Syntax 36 * Copy Into Syntax File**

```

DELETE VARIABLES WH_Risk2.

```


Decision-Making and Representation: When Core Assumptions Leave Cases Out and How to Fix It.

Next, we'll look at blood pressure readings. Doctors tend to focus primarily on Systolic readings, and less so Diastolic readings. As such, most common blood pressure tables will only convey a limited scope of possible Hypertensive conditions, and often don't include a discussion of Hypotensive conditions.

In this section, we'll work within those initial assumptions that render invisible certain medical conditions. We'll then manage deciphering better ways to represent a patient's condition when their readings don't fit neatly into categorical boxes.

It can be tempting to set up a series of logical conditionals to classify blood pressure-related risks combining both systolic and diastolic pressure readings. This however can cause some problems.

Let's observe with an intentional failure.

- Normal: Less than 120/80 mm Hg;
- Stage 1: Systolic between 130-139 or diastolic between 80-89;
- Stage 2: Systolic at least 140 or diastolic at least 90 mm Hg;
- Hypertensive crisis: Systolic over 180 and/or diastolic over 120

Sample Syntax 37 *** COPY INTO SYNTAX FILE

DESCRIPTIVES bp.1s bp.1d.

IF ((bp.1s <= 120) and (bp.1d <= 80)) BP_Risk=0.

IF ((bp.1s > 140 and bp.1s <= 180) or (bp.1d >= 90 and bp.1d <= 120)) BP_Risk=1.

IF (bp.1s > 180 or bp.1d > 120) or (bp.1s > 180 and bp.1d > 120) BP_Risk=2.

IF (bp.1s > 140 and bp.1d <90) BP_Risk=2.

VARIABLE LEVEL BP_Risk(Ordinal).

FREQUENCIES BP_Risk.

Observe the output window. Weird... the variables in question are very complete, with only 5 cases truly missing readings. But with these logical conditionals, a lot has been left out.

- The logical conditionals are based on the four standard blood pressure ranges, which only gauge one type of hypertension.
 - That is, the hypertensive condition in which both systolic and diastolic hypertension are elevated.

- The selection criteria leaves out cases with Isolated Systolic Hypertension (ISH), a type of hypertension that is more common in older adults.
- Additionally, these conditionals leave out people who have a lower but healthy diastolic pressure with an only slightly elevated systolic pressure.
 - This is akin to a *Pre-ISH* condition
 - Systolic readings are higher and worthy of concern, but Diastolic readings are in the low-healthy range
 - A doctor would certainly be intervening on these readings
- We should also account for people with low blood pressure.

If over a quarter of cases are left out, the measurement tool isn't working well. We'll need to find another way to represent the real experiences of the patients who were surveyed.

Instead, another approach would be to recode ranges of healthy systolic and diastolic readings individually, and then match those codes to develop unique identifiers as risk categories.

Use all the tools in your toolbox to stay organized. We'll build a table in the word processor to make sure we are organized. After that, we'll write out the script to recode based on this criteria.

Keep BP_Risk for now. It will be important later!

First, clarify your selection criteria for the final question.

TABLE 3		
Class	Systolic	Diastolic
Low blood pressure	=<90	=<60
Healthy Range	90> and =<120	60> and =<80
Prehypertension	>120 and =<129	60> and =<80
Pre-ISH	=>130 and =<140	=>52 and <80
Isolated Systolic Hypertension (ISH)	>140	<90
Stage 1 Hypertension	>130 and =< 139	>80 and =<90
Stage 2 Hypertension	>140	>90
Hypertensive Crisis	>180	>120

To save time, I deciphered the especially challenging blood pressure ranges and marked it "Pre-ISH". The outstanding classifications for Pre-ISH and ISH have been highlighted for clarity.

Now assign codes to each range on an individualized Systolic and Diastolic basis.

- Note that the specific criteria for Isolated Systolic Hypertension and Monitor Blood Pressure has been removed for the moment. That classification will become relevant again when we generate new codes based on the ranges below.
- Not that Diastolic only has five categories. This is because prehypertension still dictates that Diastolic pressure be less than or equal to 80 – a healthy level.
 - Again as noted above, Systolic readings tend to be the focus of medical inquiry, and shift most drastically

TABLE 4		
Systolic	Sys_Code	Label
<=90	-1	Low
90> and <=120	0	Healthy
>120 and <=129	1	Prehypertension
>=130 and <= 139	2	Stage 1
>140 <= 179	3	Stage 2
>180	4	Crisis

TABLE 5		
Diastolic	Dia_Code	Label
<=60	-1	Low
60> and <=80	0	Healthy
>80 and <90	1	Stage 1
>90 and <=120	2	Stage 2
>120	3	Crisis

The logical expressions use the values of higher classes as cutoffs.]

CHALLENGE 10
Task 1: Use your skills to generate new codes for Systolic blood pressure. Use the codes in the table above
Task 2: Use your skills to generate new codes for Diastolic blood pressure. Use the codes in the table above
Task 3: After finishing each task, execute the code and run a frequency to check your work.
Hint: You will use the variable bp.1s for Systolic, and bp.1d for Diastolic. Use the headers in the table above as the variable name for your newly calculated codes

Normally at this stage we would assign variable labels. You may do this, but since we have another transformation pending, you can rely on the tables on the previous page to guide your classification methods.

Sys_Code	Dia Code	Logical	BP_Class	Label
-1	-1	And	-1	Low blood pressure
0	0	And	0	Healthy Range
1	0	And	1	Prehypertension
2	1	Or	2	Stage 1 Hypertension
1 or 2	0	And	3	Pre-ISH
3	1	Or	4	Isolated Systolic Hypertension
3	2	Or	5	Stage 2 Hypertension
4	3	Or	6	Hypertensive Crisis

- Note that Pre-ISH has a wide range of Systolic readings and a healthy Diastolic Reading
- Note how Prehypertension also has a healthy Diastolic range
- Note how the Systolic reading for Isolated Systolic Hypertension actually exceeds that of Stage 1 hypertension
- In this case, the ordinal arrangement of risk considers both the Systolic and Diastolic readings

Now let's generate a script to create a new variable called BP_Class based on the table of classifications above.

Sample Syntax 38 *** Copy into Syntax File
IF ((Sys_Code = -1) AND (Dia_Code = -1)) BP_Class = -1.
IF ((Sys_Code = 0) AND (Dia_Code = 0)) BP_Class = 0.
IF ((Sys_Code = 1) AND (Dia_Code = 0)) BP_Class = 1.
IF ((Sys_Code = 2) OR (Dia_Code = 1)) BP_Class = 2.
IF ((Sys_Code = 1) OR (Sys_Code = 2)) AND (Dia_Code = 0) BP_Class = 3.
IF ((Sys_Code = 3) OR (Dia_Code = 1)) BP_Class = 4.
IF ((Sys_Code = 3) OR (Dia_Code = 2)) BP_Class = 5.
IF ((Sys_Code = 4) OR (Dia_Code = 3)) BP_Class = 6.
EXECUTE.
FREQUENCIES BP_Class.

The outcome accounts for 387 cases, which is a good result. We know from previous FREQUENCIES and DESCRIPTIVES that 5 cases do not have data. We do however need to account for the missing cases.

A good way to do this is by assigning a missing value, splitting the file, and running DESCRIPTIVES on the variables with missing data.

Challenge 11

Task 1: Assign 999 as a missing value to BP_Class.

Next, we'll split the file and run DESCRIPTIVES. SPLIT FILE will subdivide the dataset based on a categorical variable, in this case BP_Class.

Sample Syntax 39	
SORT CASES BP_Class.	Always sort the dataset by the split variable
SPLIT FILE BP_Class.	Subdivides the dataset by category
DESCRIPTIVES bp.1s bp.1d.	You can run DESCRIPTIVES on multiple variables at the same time

Sample Syntax 39 *** COPY INTO SYNTAX FILE

SORT CASES BP_Class.

SPLIT FILE BP_Class.

DESCRIPTIVES bp.1s bp.1d.

Take a look at the output for 999. Observe the minimum and maximum for “First Systolic Blood Pressure” and “First Diastolic Blood Pressure”. It appears that there are folks who have a healthy range for Systolic blood pressure, but a low value for Diastolic blood pressure. This condition is called [Isolated Diastolic Hypoptension](#). It primarily appears in older people, especially those who are on medications that lower blood pressure.

The heart beats in two phases, systole and diastole.

- Systole: the heart squeezes and pushes blood through the circulatory system
- Diastole: the heart relaxes, vacuuming blood into itself in preparation for pushing Systole once again

Put colloquially, hypertension is when the heart is heart “squeezing too hard” and “not relaxing enough.” In the case of Isolated Diastolic Hypoptension, the heart “relaxes too much.” The result is the less powerful “vacuuming in” of blood, causing lightheadedness or lethargy. The resulting dizziness can increase fall risk, which is especially hazardous for older people.

This is a medical condition that has received more attention over the past decade, and we ought to make sure that people with this condition are clearly represented in the dataset.

We'll need to make a few more adjustments to the classification table

- Add a line item for Isolated Diastolic Hypotension
- Generate a logical expression for selection criteria
- Update the code values to reflect the inclusion of this new classification

TABLE 4		
Systolic	Sys_Code	Label
<=90	-1	Low
90> and <=120	0	Healthy
>120 and <=129	1	Prehypertension
>=130 and <= 139	2	Stage 1
>140 <= 179	3	Stage 2
>180	4	Crisis

TABLE 5		
Diastolic	Dia_Code	Label
<=60	-1	Low
60> and <=80	0	Healthy
>80 and <90	1	Stage 1
>90 and <=120	2	Stage 2
>120	3	Crisis

TABLE 7				
Sys_Code	Dia Code	Logical	BP_Class	Label
-1	-1	And	-2	Low blood pressure
0	-1	And	-1	Isolated Diastolic Hypotension***
0	0	And	0	Healthy Range
1	0	And	1	Prehypertension
2	1	Or	2	Stage 1 Hypertension
1 or 2	0	And	3	Pre-ISH
3	1	Or	4	Isolated Systolic Hypertension
3	2	Or	5	Stage 2 Hypertension
4	3	Or	6	Hypertensive Crisis
*** There was one case (id=20261) in which the patient had a perhypertensive systolic score and a low blood pressure diastolic score. This was classed as an ISH case. The alteration is reflected in the syntax on the next page. To reiterate, people rarely fit neatly into boxes, especially when looking at two interrelated variables simultaneously.				

The affected values have been highlighted. Given this change, we'll need to add a line of logic into the original cluster of script to account for the new classification.

But first, turn off SPLIT FILE.

Sample Syntax 40 * Copy into Syntax File**

SPLIT FILE off.

Sample Syntax 41 * Copy into Syntax File**

IF ((Sys_Code = -1) **AND** (Dia_Code = -1)) BP_Class = -2.
IF ((Sys_Code = 0 or (Sys_Code = 1)) **AND** (Dia_Code = -1)) BP_Class = -1.
IF ((Sys_Code = 0) **AND** (Dia_Code = 0)) BP_Class = 0.
IF ((Sys_Code = 1) **AND** (Dia_Code = 0)) BP_Class = 1.
IF ((Sys_Code = 2) **OR** (Dia_Code = 1)) BP_Class = 2.
IF ((Sys_Code = 1) **OR** (Sys_Code = 2)) **AND** (Dia_Code = 0) BP_Class = 3.
IF ((Sys_Code = 3) **OR** (Dia_Code = 1)) BP_Class = 4.
IF ((Sys_Code = 3) **OR** (Dia_Code = 2)) BP_Class = 5.
IF ((Sys_Code = 4) **OR** (Dia_Code = 3)) BP_Class = 6.
EXECUTE.
FREQUENCIES BP_Class.

And now for formatting and cleanup...

Challenge 12

Task 1: Assign variable labels – use what is intuitive for you!

Task 2: Assign value labels based on the TABLE 7 above

Task 3: Remove the decimals from the format of the numerical code

Task 4: Change the variable level to “ordinal”

Hint: Copy-pasting from a table can vastly speed up this process. Just be sure to enclose the labels in double-quotes.

Challenge Grand Finale

While breezing through these lessons, there were a few transformations that were skipped or never addressed. These final three transformations will finish the preparation steps necessary to continue on to the analysis section.

Challenge 13
Task 1: Create an ordinal coding system based on guidelines for healthy HDL levels
<ul style="list-style-type: none"> • Lower than 40 is considered to be risky • Between 40 and 60 is considered to be lower risk • Greater than 60 is optimal
<ul style="list-style-type: none"> • Women's HDL levels often clock higher than that of men's, with a healthy range as high as 80 in some cases

Challenge 14
Task 1: Compute a new variable for non-HDL cholesterol
Task 2: Create an ordinal coding system for non-HDL cholesterol
<ul style="list-style-type: none"> • Ideally, non-HDL will be lower than 130. • Between 130 and 159 is considered borderline • Between 160 and 189 is considered high • Over 190 is very high

Challenge 15
Task 1: Create an ordinal coding system on guidelines for a healthy Total Cholesterol ratio
<ul style="list-style-type: none"> • The optimal ratio is less than 3 for women and less than 3.5 for men • A moderate ratio is 3.0-4.4 for women and 3.5-5.0 for men • A high ratio is more than 4.4 for women and more than 5.0 for men

Before we depart, let's run FREQUENCIES on all of the newly created ordinal variables to check the work. If you've been running frequencies after each transformation, you may skip this step. Sometimes it can be challenging to backtrack through an Output file to observe outcomes.

Sample Syntax 42 * Copy into Syntax File**

```
FREQUENCIES TCR_Scale  
non_hdl_code  
Dia_Code  
BP_Class  
Sys_Code  
WH_Risk  
T2D.
```

Be sure to browse you Variable View to make sure that everything looks the way you want it to!



You have made it to the end of the tutorial!

Stay tuned for Intro to SPSS Syntax Part II on data analysis.