Introductions – Overview of SAS

Welcome to our SAS tutorials. This first tutorial will provide a basic overview of the SAS environment and SAS programming. We don't want you to try to follow along with this tutorial (at least for your first viewing), instead we hope to give a quick overview of how SAS works and point out a few important ideas and potential issues. For subsequent viewings, you can follow along using any SAS dataset you have previously created.

We will be using SAS version 9.4 for these tutorials, however, version 9.3 is similar and 9.2 is similar enough but will require a little extra code which I can explain to individual students as needed. Higher versions hopefully will also be similar for at least a few versions.

We begin by opening SAS. It is always best to open SAS and then open any SAS files. I have a shortcut on my desktop and I have SAS pinned to my taskbar. You can also go to the start menu and all programs and find SAS in the menu. Choose SAS 9.4 (English) (or similar for different versions of SAS).

Now that we have opened SAS, the first thing I usually do is maximize the program. Notice there are three windows by default. The explorer window on the left. I don't use this area too often but I never close it either.

The log file at the top. This is a very important window as here you will find information about any errors in your SAS programs. You should always check the LOG file even if you feel there are no errors. When you first open SAS you should see a message similar to this with all blue text. Any other color might require attention.

If SAS is misbehaving, check the initial LOG file to make sure there are no system errors or problems. Closing and restarting can also often solve strange problems with SAS. Make sure you never have two open sessions of SAS as only the first will work.

Rarely, if SAS crashes dramatically, you may need to restart your computer and then restart SAS or close SAS manually with the Task Manager in order to start fresh. This sometimes happens to me when I have been doing a lot in SAS for hours without starting a new session. It rarely happens to students in my courses but might be more likely the further you go in your SAS programming.

The editor window is where we write our SAS programs which we will begin doing shortly.

First, I want to introduce you to the concept of a SAS library. On this computer, I have a folder on my desktop called "My SAS Library". Here's what it looks like. It is a folder I created which now contains numerous SAS data sets created when I imported them into SAS and told SAS to store them in this library. We will go through that process as a "work-along" tutorial later for now I just wanted you to understand the idea.

In SAS, I can see all of SAS libraries by double clicking on the libraries in the explorer window. This shows a number of libraries – some of which are automatically available, so see which ones you have when you first install SAS. I have created the AMY library, the TEMP library, and the SASPH library. For this tutorial, I will be using a dataset stored in the Amy library.

You can see the datasets in a library by double clicking on the library name. You can open data files but normally I do not do this as it can cause problems when working with the data – if you do open datasets, be sure to close them before you go back to what you are doing in SAS. Let's use that method to look at the dataset called FGHM113.

I prefer to use SAS programming to view my data so let's close this window and get back to what we are doing. If you browse inside the explorer window, to get back out you need to land on the window again and click on the UP FOLDER icon. I will click twice to get back to the view when we opened SAS. We will continue in the next tutorial.

Introductions – A First SAS Program

Welcome. This tutorial will provide a basic overview of SAS programming and add to our understanding of working in SAS. We don't want you to try to follow along with this tutorial (at least for your first viewing), instead watch and get an idea about what you will soon be doing in SAS. For subsequent viewings, you can follow along using any SAS dataset you have created.

We begin by opening SAS. It is always best to open SAS and then open any SAS files. I have a shortcut on my desktop and I have SAS pinned to my taskbar. You can also go to the start menu and all programs and find SAS in the menu. Choose SAS 9.4 (English) (or similar for different versions of SAS). I usually maximize SAS.

In SAS there are PROCEDURE steps and DATA steps. We will start with running a procedure. I will land in the editor window and type PROC PRINT DATA=AMY. FGHM113; RUN;.

For the dataset: AMY is the name of a SAS library which is linked to a folder on my computer and the period separates the library name from the dataset name. The dataset we are using is FGHM113, which is a small subset of the Framingham data.

Notice that now the untitled document name has a * at the end. This indicates the file has unsaved changes. Before I run the code and see the results, I am going to save this SAS file, also more usually called a SAS program. Possibly a SAS program file is most accurate. Any usual method of saving is fine, using the file menu or the disk icon or keyboard shortcuts.

Generally I do not save SAS programs in any of my SAS libraries because I don't like for my library folders to get too cluttered. You can do whatever you wish as far as where to store your SAS program files. For these tutorials, I am storing them in a folder called SAS Programs in SAS Tutorials. I will call this file Topic0B and SAS will add the .sas extension to the file.

SAS commands are not case sensitive but values of variables as well as anything else that appears in quotation marks will be case sensitive. I will tend to capitalize the SAS code in these tutorials but this is not necessary.

The enhanced editor, which I am using here, color codes keywords. Eventually this will help you spot problems in your SAS code. For now just try to notice the patterns in the code you are learning so you can recognize when things aren't the color you expect – this usually indicates a problem!!

Here the procedure begins with PROC and then the particular procedure we wish to use, PRINT. Then we specify the data to be used with DATA = AMY. FGHM113. The AMY specifies the library name, the period specifies the split between the library name and the dataset name of FGHM113.

To run an entire SAS program you can land anywhere on the editor as long as you are not selecting any particular text and the run the program using either the running man icon or the keyboard shortcut of F3 or by right-clicking and choosing submit all from the menu.

To run a portion of a SAS program you can select the section desired and then submit only those results. The methods are the same except you use submit selection in the choice from the right-click menu.

In this case we see a printout of our dataset.

Let's look at our log file. Notice it shows the code submitted and notes regarding how many observations were used.

Let’s look at a common error, a missing semicolon. Each SAS command ends in a semicolon. If we remove the one after the name of the dataset first you may notice that the word RUN is no longer in bold as it was. This can help you find missing semicolons. If we resubmit, what happens? In our log file, we see a lot of red text – this is usually not good!! SAS has a difficult time finding missing semicolons and tends to give an error message which just sounds like SAS is confused. In this case it is smart enough to suggest a semicolon as the first choice but sometimes it’s not so good at catching them.

Notice that in the editor window, it says PROC PRINT RUNNING – this is the sign of a failed procedure sometimes – but not always. You can easily clear the log file by landing in the window and clicking the blank paper icon. Let’s put the semicolon back and submit again.

Another possible problem is if you mistakenly select code. For example, here I will select on the first line of code, without also submitting the RUN statement. Notice that SAS does nothing. The log file has the line of code but no error. This is a sign of this kind of issue. Notice that it still says PROC PRINT RUNNING in the editor title bar. We could simply submit the missing components and it would work but here I want to show you how to use the BREAK feature in SAS.

Click on the circle with the exclamation mark in the menu. Choose cancel submitted statements and click OK. It gives you one last chance to change your mind. Here we select Y to cancel submitted statements and click OK. Notice that now it does not say RUNNING in the editor title bar and in the LOG it cancels the print procedure without producing any output.

Another potential error is putting components in the wrong place. Be sure to follow the examples provided and think about if there is a reason that something might need to be in the particular order we illustrate – for example you have to create a variable before you can use it or label it, etc. We will try to illustrate this after we have learned some procedures where this is a common problem.

Comments can be left in your SAS code by adding /* to the beginning of the comment and */ to the end. I will often provide comments in SAS code to help explain or remind about SAS code components.

In my SAS settings I request both the Results Viewer (which is HTML) and the plain text OUTPUT listing. In either case we see a basic printout of our entire dataset. We will be returning to PROC PRINT and other procedures in the future tutorials.

To choose the type of output we want to display, go to TOOLS – OPTIONS – PREFERENCES. Then go to the RESULTS tab and determine if you wish to add the output listing in text format or not. Usually the default settings look the same as they do here except the first box is not checked.

The TOOLS menu changes based upon which SAS window you have selected. Let’s land on the editor window. Notice that I have line numbers to the left of my code. I find this helpful but it does not seem to be the default setting. To add these line numbers, be sure you are in the editor window and go to TOOLS – OPTIONS – ENHANCED EDITOR – under the GENERAL tab, select show line numbers.

Before leaving, notice that the left menu is now showing the results tab. At the bottom left you can select between the results and explorer tabs as needed. And on the bottom under the editor and output, you can move between the editor, log, and output windows.

When we close SAS it will ask if we are sure we want to end the session, click OK and then do you wish to save the changes to the SAS program, here I will click YES.

For now, that's the end of this introduction to SAS programming. We will get into the specifics as needed throughout the semester.

Introductions – Dataset used for Tutorials

Welcome to this discussion of the dataset used for these tutorials. You can read the details in the Pulse Dataset Information posted with the tutorials which I have opened here.

This data represents a somewhat controlled experiment conducted by a statistics professor in Australia in his classes over numerous years where he attempted to randomize students to either sit or run. Pulse measurements were taken before and after along with other variables which we will see shortly.

I removed three students from the source dataset so that the original dataset I am posting for these tutorials consists of 107 students.

I mentioned the attempt to randomize. The second paragraph of the information discusses the methods used to try to obtain compliance. Initially he used a coin but who is to say that students did what their coin demanded? Then he assigned them based upon the forms. We will investigate this later in the tutorials.

The variables in the original dataset are

- **HEIGHT** in centimeters,
- **WEIGHT** in kilograms,
- **AGE** in years,
- **GENDER**
    - coded as 1 = Male and 2 = Female,
- **SMOKES** and **ALCOHOL** - are you a regular smoker or drinker
    - coded as 1 = Yes and 2 = No,
- Frequency of **EXERCISE** is
    - coded as 1 = High, 2 = Moderate, 3 = Low,
- **TRT** - whether the student ran or sat between pulse measurements
    - coded as 1 = Ran and 2 = Sat
- **PULSE1** and **PULSE2** in beats per minute
- **YEAR** - year of the course (93, 95, 96, 97, 98)

There are numerous questions we might like to answer some of which are mentioned in the document. We will discuss them as they are addressed in later tutorials.

On page 2 we also have some information about the new variables we will create later in the tutorials. In particular we will create BMI from the height and weight variables and categorize students into BMI standard groups.

Finally, we have provided the dataset in numerous formats. The original data will be provided in EXCEL in two formats, as well as comma separated (CSV), tab delimited text (.TXT), SAS dataset (.SAV), and SAS dataset (.SAS7BDAT).

Note: SAS can use some of these file types. In particular we will look at CSV files and you will work extensively with SAS datasets.

That's it for the data information for now. Thanks for watching!

Introductions – Tips for Watching Tutorials

Welcome - this video will discuss some tips for watching these tutorials. The videos are stored on YouTube and can be viewed either on the tutorial page or via YouTube.

To illustrate we use this page with videos for another course using the software package R.

You can see the video embedded in the page and you can watch it here. You can click on the YouTube link at the bottom of the video to view in YouTube. This doesn't currently offer any major differences but sometimes the website viewer lags behind YouTube in making changes.

With the software tutorials, you may need to see the relatively small text in the video clearly. To change the settings, use the gear icon at the bottom. Setting to the highest possible setting will make the video as clear as possible.

Viewing the video in full-screen will also likely be necessary to view the details.

You can also speed-up or slow-down the video speed using the gear icon.

You can turn on the closed captions in the settings accessed by the gear icon or by clicking on the cc icon.

The transcripts are also available as separate documents.

The best way to learn from these videos is to follow along yourself. To do this it is often helpful to have the videos in one monitor, screen, or window and the software in another.

Alternatively you might watch the videos on one device and work in the software on another; pausing the tutorials between tasks as you work in the software.

Taking notes is also a good idea. You will be more likely to remember how to complete a task from your own notes. You will be asked to complete many of these tasks numerous times during the semester as we build our knowledge of statistical analysis.

That's all the basic advice I have about using these tutorials. Thanks for watching!

Topic 1A – Linking Folder with a SAS Library

Welcome - this video will go through how to link a folder on your computer with a library in SAS. This allows easy access to and storage of SAS data sets. We will always use data stored in a library in our tutorials and highly urge you to use a library in your own work.

To begin you need to create a folder on the computer where you are running SAS (or in some location which SAS can access). In my case, I used the folder for my courses and created a sub-folder called My SAS Data. You can name the folder anything you wish but if you change the name, move the folder, or in any way change the path of this new folder, the link created will be broken and your SAS library will no longer work and you will have to recreate the link to the folder.

There are other ways to access data in SAS and you are welcome to investigate these on your own and use any method that works for you in this course.

Once you have created the folder and are happy with its name and location, open SAS.

Double click on Libraries and right-click in any white space and choose new (alternatively you can land in the explorer window and go to file and new). This opens a dialog box.

In the name, choose the name of the library in SAS, this does not need to match your folder name and should be as short as you can make it. I will use BIO as the library name.

Click the box "Enable at startup" – this tells SAS to make this connection each time you start SAS, otherwise the connection will disappear when you close this SAS session.

Click on Browse and find your folder. Go INSIDE the folder and click OK. Then click OK again.

Now we see a library called Bio in the libraries. If we open the folder, there is no data so far.

To move up in the directory structure, make sure you are inside the explorer window and use the up folder icon in the tools below the menus.

Now you have a SAS library connected to a folder on your computer. Be sure to remember where the folder is and again … if you change anything in the path of that folder, don't be surprised that you can no longer access your data! If that happens, just return to this tutorial and repeat the process to link the folder to a library in SAS.

One thing before we quit. If you need to delete a library for any reason (including that it has become broken), simply land on the library and either go to edit – delete or right-click and delete to remove it from your list.

That is everything you need to know about linking a folder to a SAS library for now. We will look at importing and working with data in future tutorials. Thanks for watching!

Topic 1B – Importing Data from CSV File

(In order to follow along you need the original dataset in CSV format – pulse.csv.)

Welcome - this video will go through importing data from a comma separated or CSV file and saving as a SAS dataset in a SAS library. SAS seems to have issue importing EXCEL files in recent versions so I simply suggest using CSV files as an alternative. In practice, if you have an EXCEL file, simply save it as a CSV file and then import the CSV file into SAS.

To begin we will open SAS. You should already have a SAS library linked with a folder on your computer.

From SAS we go to FILE – IMPORT DATA. Here we need to select Comma Separated Values (*.csv) from the dropdown list under Standard Data Source. Then click NEXT.

Browse for the data file in CSV format. Click on the file and click OPEN. Then click NEXT.

We want to store our dataset in our library. In this case I will use the BIO library created in an earlier tutorial. We select BIO from the library list.

In the member box, choose a name for your dataset. I will call it PULSECSV. You can click FINISH from here. If you click NEXT it gives you an additional option to save the SAS code needed for this import. I will click FINISH.

In the log file, we see that the dataset was successfully created and that it has 107 observations and 11 variables. Remember to always check your log file!!

We can print this dataset to verify. PROC PRINT DATA=BIO.PULSECSV; RUN; We can select the code and click the running man or use F3 to submit.

We can see the dataset, and it looks as expected. The variables are as we discussed in the introduction to the dataset.

We have imported the data from a CSV file, saved it as a SAS dataset for future use in our SAS library, and now we will save our SAS code for future use. Land on the editor window and go to FILE – SAVE. That's all for this video. Thanks for watching!

Topic 2A – Permanently Labeling Variables

(In order to follow along you need to have created the SAS dataset PULSECSV from the Topic 1B tutorial.)

Welcome - this video will go through how to permanently add more meaningful labels to variables in a dataset. Often we want the variables names in a SAS dataset to be as short as possible as we have to type them a lot. So we would like to add a more descriptive label which is used in the final results.

To illustrate, I have already created a short SAS program which I will now open. I will go to FILE. If you have recently used a SAS file, you will find it under Recent Files. Here I will open Topic2A.sas.

First I am going to run some code to create a histogram for PULSE1 which is the variable containing the students' resting pulse rates. We will come back to this code later, for now I only want to focus on the output.

When we look at the histogram created we see the x-axis label simply has the variable name PULSE1. Ideally this should be more descriptive, especially the further away from you the audience becomes.

Before we begin, let's print our data and run another procedure on our data, PROC CONTENTS.

In our PROC PRINT, we will add an option, after the dataset name, we add (OBS = a number). So our first line becomes PROC PRINT DATA=BIO.PULSECSV (OBS=5); This will print only the first 5 observations. We just want to get a look, we don't need to see the whole dataset. In this case the option is in parentheses since it is an option to the dataset name itself.

Then we add our second line, RUN; and then submit this set of code. The dataset looks as expected.

Now let's look at a new procedure, PROC CONTENTS. This provides additional information about the dataset. I use it at the beginning of most every analysis to remind myself of the variable names in the dataset.

We begin with PROC CONTENTS DATA = BIO.PULSECSV then I usually add the VARNUM option to the PROC CONTENTS statement – this is not in parentheses since it is an option for the entire PROC statement not the dataset itself. This option will provide the list of variables in the order they appear in the dataset instead of the default which is alphabetical. Then we add the semicolon and then RUN; and submit the code.

PROC CONTENTS provides three tables. The first provides information about the file including its name, number of observations, number of variables, etc. The 2nd table has the complete file path of the dataset – this can be useful if you have forgotten where you put your folder for your library!

The most useful part is the table with the list of variables. You can copy variable names from this list if it is helpful.

Now let's work on labeling the variables in this data set. We will create a new dataset through this process.

To do this we use the other kind of STEP in SAS, a DATA step. DATA steps are used to process data manipulations such as creating new variables, removing observations, labeling variables, and other data only processing.

PROCEDURE steps, on the other hand, generally provide some data analysis but rarely are used to modify data.

We start with DATA and then the name of our NEW SAS data set to be created. I will call the new dataset PULSE_STEP1 but I want to store it in my BIO library. So we have DATA BIO.PULSE_STEP1;.

The next command is the SET statement which provides the dataset you wish to pull from which will be modified. It is ALWAYS a good idea to name the new dataset something different as it is possible to have an error which overwrites your original data with nothing! This is never good! So here we have SET BIO.PULSECSV; since we wish to pull in our data from our PULSECSV dataset imported earlier.

Now we begin our labeling. You can have many label statements with one for each variable or one label statement with multiple variables. In SAS, the command does not end until the semicolon regardless of the how many lines of code you take for that command. Here we will use one label statement and label all of the variables in the dataset.

We begin with the LABEL command and then the first variable name, which is HEIGHT. Then after the variable name we put and = and then in quotations, the label for the variable, in this case "Height (cm)."

We repeat this for all of the variables in the dataset. Notice that I will not put the translations of the codes in the label for the variable, only the variable label itself. So for Gender, I will actually simply leave it as Gender and for SMOKES, I will just use "Regular Smoker?", etc. After the last variable, we add the semicolon that ends the command. Then a RUN; to end the DATA step.

Now we submit our DATA step and look at our log file to make sure it was successful. There are no errors and everything seems good here.

Now let's repeat the PROC PRINT and PROC CONTENTS on the new dataset.

Notice that the PROC PRINT results stay the same. In PROC CONTENTS, we now see the labels in a column in the variable list.

Now if we recreate the histogram on the new dataset, we do not change the variable names, they stay the same, we still ask for PULSE1. But now we see the full description as the label in the histogram's x-axis.

We have provided permanent labels for our variables using a DATA step and viewed the results in PROC CONTENTS and a histogram. That's all for this tutorial!

Topic 2B – Translating Categorical Variables

Welcome - this video will go through how to create translations for categorical codes using PROC FORMAT followed by linking the translations to specific variables using a FORMAT statement in a DATA step. This will be required for any dataset where the raw data for categorical variables are numbers. This will result in output which shows the actual description instead of the coded raw values.

In order to follow along you need to have the SAS dataset PULSECSV from the Topic 1B tutorial. We will go back through the process of labeling the variables here and add the new skill of translating the coded categorical variables. I will go to the recent files and open the Topic2B code.

In SAS there are two steps. First we must define the translations called user-defined formats in SAS. This does not immediately connect the translations with any variables until we add that connection. For example, if you have 15 questions which are 1 = Yes and 2 = No, you only need one translation which specifies 1 = Yes and 2 = No and then you can link that translation to all 15 Yes/No variables at once.

I suggest the format names be DIFFERENT from your variable names so that you can easily keep track of which is which.

In this dataset we will create translations which will be used for the following variables:

- GENDER where 1 = Male and 2 =Female
- SMOKES and ALCOHOL where 1 = Yes and 2 = No
- EXERCISE where 1 = High, 2 = Moderate, and 3 = Low
- TRT where 1 = Ran and 2 = Sat

We have two Yes/No variables with the same exact code. The rest of the translations will only be used by one variable.

To begin let's run PROC PRINT and PROC contents on the data. We will use the dataset PULSECSV and create a new dataset, PULSE_STEP2 when we are finished.

When we print the data, we see that the variables mentioned do have numbers as their raw data.

In PROC CONTENTS, we will see a difference after we have formatted the variables. For now, all of the formats are the same, this is the default format for values which are stored as numbers imported through CSV files.

Now let's look at the frequency distributions for our categorical variables using the raw data. Notice that there are no variable labels and the coded values are used in the output frequency tables.

Now let's look at the code for PROC FORMAT – remember – this does NOT make any connections to any variables in any dataset yet.

The procedure begins with PROC FORMAT;. There is no data assigned with PROC FORMAT as this procedure does not analyze data. It only defines the translations, called formats in SAS, which we will need later.

Each VALUE statement provides the definition for one translation. We need four here. One to be used for our variable GENDER named GDR, one named YN for the variables SMOKES and ALCOHOL which are both the same code, one for EXERCISE named EXER, and one for TRT called TREAT. Each value statement ends in a semicolon. The spacing is not important, I simply line mine up this way to make it easy to check my code. We end with a RUN statement. Again, I highly suggest that the format names and the variable names NOT be the same. This causes beginning SAS students confusion.

We can submit this code now or wait until we write the code to connect these to our variables, either way, we must submit the PROC FORMAT code before we use any of the formats in a DATA step.

Let's submit it now and look at the log file. For each format created, you should see a note that the format has been output. These formats are only stored for the given session so when we work with this dataset in the future, we will need to begin our SAS file with the PROC FORMAT code. We will illustrate this in the next tutorial after we have gone through the basic process here.

Now we will create a new dataset in a DATA step which will label the variables and connect the categorical variables with the appropriate formats.

We begin with DATA and the name of the new dataset to be created BIO.PULSE_STEP2. Our SET statement pulls in data from the BIO.PULSECSV dataset. Then we have the same label statement we used in the Topic 2A tutorial on that topic. We are repeating it here since in practice we would usually prefer to complete all of these tasks with one DATA step, not two.

Then we have a FORMAT statement which tells SAS which formats to associate with which variables. The variable name (or list of variable names) comes first followed by the format name which is ended with a period. The period tells SAS this is a format name instead of a variable name. All format names always end in a period when used in a FORMAT statement.

Now we submit our data step. Looking at the log file, we can see that there are no errors and the notes indicate the DATA step was a success.

Looking a the results of PROC PRINT, we now see that the values of the categorical variables are translated. The raw data are still numbers but SAS will show us the description instead in most output.

Looking at the results of PROC CONTENTS we see our format names in the format column in the variable list for the categorical variables.

Finally, when we create the frequency distributions, we see that we have the variable labels at the top of the table and the translations for the values of the variables instead of the codes. This is our goal with this process. We do not want to change the raw data but we want to see the description instead of the code in our analyses.

In our next tutorial we will look at how to work with this dataset in a new session. Please be sure to watch that tutorial carefully as once we have associated the formats permanently, it can cause confusion for beginning students when they come back to work with that data.

That's all for this tutorial!

Topic 2C – Using a Formatted SAS Dataset

Welcome - this video will go through how to use a dataset which has permanently assigned user-defined formats which provide translations for coded categorical variables.

In order to follow along you need to have the SAS dataset PULSE_STEP2 from the Topic 2B tutorial.

Normally, you will already have such a dataset stored in your library from when you created the dataset to begin with. In this tutorial, I will start from one step back. Suppose someone gives you a dataset and the formats needed and you need to access it in SAS.

Before recording this tutorial I removed the STEP2 dataset from my folder on my computer. You can see here that there is no file called PULSE_STEP2 in this directory.

If we go to SAS and look in this library, again you will see that there is no STEP2 file.

Now I will take the file from my desktop and put it back in my folder.

When we return to SAS we will need to browse out of the library and back in to refresh or go to VIEW and REFRESH. Now you can see that the STEP2 file is there.

In order to access a SAS dataset, all you need to do is put the .SAS7BDAT file into the folder on your computer which is associated with your SAS library, it will then be immediately available.

However, there is one issue with this dataset. If we try to print the dataset using PROC PRINT we get the following errors – these errors may be common for you for a while until you get used to this process. The errors clearly say that certain formats cannot be found or could not be loaded.

The issue here is that we did not yet submit the PROC FORMAT code to define these translations in this session. Formats are only held for the current session (unless you learn more complex ways of handling them which is possible but beyond what we need for this course).

Let's submit the PROC FORMAT code and then print the dataset again. Now everything works just as expected.

We can again look at the frequency tables to check that all coded categorical variables are correctly translated.

It will be very important to keep the PROC FORMAT code used for datasets in this course and to submit that code at the beginning of each SAS session where you wish to analyze that data.

Now our data is ready for most of the analyses we will learn this semester. That's all for this tutorial!

Topic 2D – Computing New Variables

Welcome - This video will discuss how to compute new variables from other variables in a dataset.

In order to follow along you will need the SAS dataset PULSE_STEP2 (after labeling all variables and translating the original coded categorical variables).

New variables are created in a DATA step. In practice it is best to avoid having more DATA steps than needed. Usually, you can continue to modify your original data step later if you need to make changes.

If you do work in stages, you must be careful to make sure you start from and save to correct datasets as common errors include starting from the wrong dataset or overwriting your original data with data that contains some kind of problem which would require reimporting the original data and starting over. We work in stages in these tutorials due to the fact that they only cover one topic.

Our SAS code begins with the formats needed for this dataset. We will submit this code to begin.

Now we start our DATA step. The first line gives the new dataset name so we will make this step 3 with DATA BIO.PULSE_STEP3. Then in our SET statement we add the dataset we want to use as our starting data. We will use the step 2 dataset with SET BIO.PULSE_STEP2.

We will begin with a common transformation which is often used in practice, the natural logarithm. This transformation is sometimes used to create a variable which is more symmetric than the original variable. We will use the original variable WEIGHT.

In SAS (and often in the field of statistics), the natural logarithm function is denoted simply as LOG instead of LN. This is a difference in notation between mathematicians and biologists (who would likely use LN) and statisticians (who tend to use LOG instead). In statistics we rarely use LOG base 10 and so there is no confusion (for statisticians). For students, it is often confusing!!

Given that, creating this new variable is very easy. We first decide on a name for our new variable. I will call it LOGWT. If you prefer, you can still name it LNWT or anything that reminds you what this variable represents.

Then we write our statement to create this new variable in the form, NEWVAR = EQUATION; Here we have LOGWT = LOG(WEIGHT); Weight was the original variable we wish to transform and LOG is the function we are using. That's it. Once we submit this data step, we will have a new variable called LOGWT. We will wait and look at the results after we create our next variable.

Now we will create a variable for body mass index which we will call BMI from the height and weight variables in our dataset. The formula for BMI is the weight in kilograms divided by the square of the height in meters. Our weight is in kilograms but our height is in centimeters so we will need to divide it by 100 to get the height in meters.

This equation is a little more complicated but the process is the same. We start by giving our new variable name BMI, followed by the equals sign, and then our equation WEIGHT/(HEIGHT/100*HEIGHT/100). You must be careful about order of operations and grouping in these calculations. In SAS you use ** to provide powers. Here instead of squaring the denominator, I simply multiplied HEIGHT/100 by itself.

We are going to give these new variables permanent labels. They are numeric so they don't need any new translations.

We add a LABEL statement and add LOGWT = "Natural Log of Weight" and BMI = "Body Mass Index" then add our semicolon to end that LABEL statement.

Now we add our RUN statement and then we can submit our code and check our log file. The original dataset had 11 variables, now we see that this was successful and that the new dataset has 13 variables.

Running PROC CONTENTS (on our STEP3 data) with the VARNUM option, we see in the variable list we have our two new variables with their assigned labels. There is no format for these variables since they were not imported using the wizard and we have not assigned any as no format is needed here.

Running PROC PRINT for the first 10 observations, the values seem reasonable for our new variables.

To see the effect of the transformation of WEIGHT, we can compare the histograms and boxplots for WEIGHT and LOGWT. Here we have copied the four graphs created into a document. We can easily see that where WEIGHT was skewed right, the new variable LOGWT is much more symmetric.

Transformations always have advantages and disadvantages so be sure to read more about any transformation you plan in practice in order to fully appreciate the ramifications.

Body Mass Index is a common measure so it is good to know that it is easy to calculate in SAS given the height and weight. We will use it in a future tutorial to learn how to categorize a quantitative variable.

We can compute many variables this way. Another possibility would be to construct standardized versions of a variable by taking (VARIABLE – mean)/std.

We will save this SAS code as Topic2D.sas. That’s all for this tutorial.

Topic 2E – Categorize a Quantitative Variable

Welcome - This video will discuss how to categorize a quantitative variable. This is a skill often used in practice when there are commonly defined groups based upon a measured variable.

For example we might want to provide information about body mass index categorized into underweight, normal, overweight, or obese or classify individuals having high or low values for a particular variable such as high systolic or diastolic blood pressure. In addition there are often other reasons to categorize variables varying from the intended audience to the need to handle non-linear predictors in multiple regression analysis.

Here we will categorize the body mass index variable (created in Topic 2D) into a binary version which looks at overweight (BMI ≥ 25) vs. not overweight (BMI < 25) and a multilevel version which categorizes into the standard categories.

We don't have any missing data here but in practice if you do, you will need to make sure that they are handled correctly. In SAS missing data are ordered as "negative infinity" so that if you try to select all values less than a number, the missing values will be included. If you always specify two endpoints for each category then you can completely avoid this issue and we will do that here.

In order to follow along you will need the SAS dataset PULSE_STEP3 (after creating two new variables).

Before we start creating the variables, let's run PROC MEANS on our STEP3 data to determine the minimum and maximum for the variable we will be categorizing, BMI.

We can see that for BMI, the minimum is around 16.5 and the maximum 32.1. We will go slightly outside this range in creating our groups as needed. Also notice there are 107 observations.

We use a DATA step to create new variables of any kind. For this type of categorization, we use if-then statements. There are other options such as IF-ELSE statements but we will only learn this one way to categorize quantitative variables.

In practice it is best to avoid having more DATA steps than needed. Usually, you can continue to modify your original data step later if you need to make changes.

If you do work in stages, you must be careful to make sure you start from and save to correct datasets as common errors include starting from the wrong dataset or overwriting your original data with data that contains some kind of problem which would require reimporting the original data and starting over. We work in stages in these tutorials due to the fact that they only cover one topic.

Our SAS code has the formats already in use for this dataset. Before we submit this code, since we will need two new formats, we will go ahead and add them here.

We will be creating two categorized versions one with two levels and one with four so I am naming the formats BMI_TWO and BMI_FOUR. SAS does not allow us to end a format name with a number so I used the word instead.

The two-level variable has two groups: < 25 and ≥ 25. In practice we usually use words to describe groups but since the project asks you to use numeric ranges, we will do that here. Since SAS doesn't easily allow us to put the greater than or equal symbol, I will use 25+ to indicate that the 25 is in the upper group.

For the four-level variable, we can use an algebra style notation to indicate whether endpoints are included or not. You can put anything that explains exactly what happens on the endpoints as long as it is short enough to be reasonable for translations. Here we use < 18.5 for the first level, algebra notation of [18.5, 25) and [25, 30) for the middle levels and 30+ for the last level.

Now we submit this code and check the log file to make sure all of our new code worked. Be sure to check each format in the log file as it will run what it can but skip formats with errors. If you don't look back to the top, you may miss an earlier error.

Now we start our DATA step. The first line gives the new dataset name so we will make this step 4 with DATA BIO.PULSE_STEP4. Then in our SET statement we add the dataset we want to use as our starting data. We will use the step 3 dataset with SET BIO.PULSE_STEP3.

We start by creating a binary variable which looks at overweight (or worse) vs. less than overweight. We need to decide on a name for this variable before we begin, we will use BinaryBMI.

The cutoff for overweight is a body mass index of 25. Technically 25 is considered overweight so we want it counted in the higher category.

We add the following statement: IF 15 <= BMI < 25 THEN BinaryBMI = 1;

This statement tells SAS to create a new variable BinaryBMI with a value of 1 if the original BMI variable falls between 15 and 25, including only the lower endpoint. Notice that this says mathematically that BMI must be greater than or equal to 15 but less than 25.

Now we add the second statement: IF 25 <= BMI <= 33 THEN BinaryBMI = 2;

This defines the new variable BinaryBMI to be 2 when BMI is 25 or larger. We chose the 33 and 15 based upon the min and max calculated earlier – we chose a reasonable integer outside of the range of the data. Any values outside the range of the data would work.

Now we add the code for a multi-level version of BMI which I decided to name BMIGroups. Here we have four IF-THEN statements, one for each level of our new categorized version. The cutoffs used are given in the dataset information document.

Now, let's label and format these new variables. I will label BinaryBMI as "Binary Body Mass Index" and BMIGroups as "Body Mass Index Categories" in a LABEL statement.

Then we connect the two new variables to the formats we created earlier using a FORMAT statement.

Now, we can run this DATA step and look at the log file. The dataset used had 13 variables, we added two new variables, and the log file says our resulting dataset STEP4 has 15 variables.

Running PROC print, we can see we have two new variables BinaryBMI and BMIGroups. Since we already formatted the variables, we see these formats in our PROC PRINT results but in fact there are coded values in the background that now we do not see. If you want to see that step, you can run the DATA step before adding the FORMAT statement and see the difference.

We can also run PROC FREQ now to check our work. Our total for each variable is 107 which is correct. Also due to the way we created these variables, we know that the first two levels of the BMIGroups should end up in the first level of BinaryBMI and our total of 15+73 does indeed equal 88. The two upper levels in BMIGroups should end up in the second level of BinaryBMI and again our total of 17+2 equals 19.

We have successfully created a binary and a multilevel version for BMI in our data. They are correctly labeled and translated and ready for further analysis. That's all for this tutorial!

Topic 3A – Copying Results into Word

Welcome - this video will go through how to copy results from SAS into WORD. We will use the results from Topic 2A which contain some tables and graphs.

To begin, with SAS open, I reopen that SAS file, Topic2A.sas. I will run the last three procedures. PROC PRINT, PROC CONTENTS, and PROC SGPLOT.

To follow along you will need the Topic2A.sas program as well as have the PULSECSV dataset in your SAS library. Note that if your library name is different than mine (BIO), you will need to change the library name to yours in all SAS code before submitting.

This will create the HTML output for these procedures. You can save the HTML output, but often you will only want portions of the output and so my suggestion is to copy and paste. We will look at working with HTML output in another tutorial. We don’t grade you on your formatting skills so just do your best. We are also happy to help if you have issues.

For most students it is easy enough to copy from the SAS Results Viewer and then paste into a word document. I have a blank Word document open. We start with the histogram. Land on the histogram and right-click and select copy. Then in an open word document, use CTRL-V or right-click and paste.

You can play around with which options look best for certain object types. Graphs will generally copy well using any method. These are usually very large so you can resize them by dragging a corner or the bottom or side. Be sure that text is readable in any submissions for this course. I will add a few extra lines after the graph to prepare for our next paste.

Graphs are usually very easy. Tables cause students a little more frustration but again, we aren’t going to grade you on formatting. As long as the information is there you should be fine.

To select a table simply drag your mouse over all text in that table. Copy, then paste into WORD. This doesn’t usually keep the formatting of the colors and borders. If you wish you can use word to edit the table, adding borders, changing the font, font-size, colors, etc. as you desire. Word now has many auto formats which might work well.

You can also take a screen shot using the PrtScn key or any other screen capture method and pull the table into word that way. Here I have taken a screenshot of that same table. I will paste into word and crop the image. Then I resized the image and formatted the document until I was happy. Centering and spacing the results.

Don’t forget to save your WORD file! Here I will save the file as Topic3A.docx.

We have illustrated how to copy graphs and tables from the Results Viewer in SAS into a WORD document. This will be an important skill for presenting your data analyses in this course and in any future data analysis endeavors.

One additional comment, if you wish to create a PDF file, you can simply save your word document as a PDF file using that file menu option. That’s all for this tutorial!

Topic 3B – ODS RTF

This tutorial covers how to export SAS output into a RTF or rich text format document. This can help when you need to export many results but can also be more difficult or time consuming than simply copying and pasting if you have to remove a lot of unneeded items.

We will use the results from Topic 2A which contain some tables and graphs. We have already edited this file as Topic 3B.sas, adding the needed new code.

ODS stands for output delivery system. By default you are using ODS HTML output. Here we will specifically ask for RTF output which can easily be edited in Word and saved as a Word document or PDF file.

We will create two sets of output. For the first, we will use the default options. Before our first graph, we add ODS RTF FILE = "H:_SAS\Topic3BPart1.RTF"; This asks for RTF output to be stored in a file in my H drive in a folder called _SAS and named Topic3BPart1.RTF. You will need to provide a file path for your file on your system instead of H:_SAS.

After the PROC CONTENTS but before our DATA step, I add ODS RTF CLOSE; which tells SAS to stop exporting to the RTF file and save the file.

Note: You can use PDF instead of RTF here but the PDF files are not easy to edit and you will usually need to edit your output, at least for this course. Search the internet to learn more if you are interested.

Let's submit this portion of code from the beginning to the end of our RTF request. If the document doesn't open automatically (you may even get an error), don't worry, unless you have an error in your actual code, the file will be saved where you requested so go there to see the results. Here the document opens automatically in Word. You will notice that by default, SAS generally puts one graph or procedure per page – unless a procedure needs more pages. Some procedures also have additional automatic breaks in the output.

You will also notice that it puts the titles in the header area. I find this to be more difficult to edit than an alternative option I will show you shortly. Otherwise, the results look very nice and are easily edited. Do save this file as a Word document for class instead of leaving it as a RTF file. You can do that by going to FILE – SAVE AS and choosing a Word file type (.docx or .doc).

I submitted the data step and now let's look at a few nice options for RTF. To get rid of some of the header information we can add an option of BODYTITLE and to have everything flow without any page breaks we can add STARTPAGE = NEVER. There are many other options; you are welcome to search to learn more about ODS RTF.

Again we add the closing statement ODS RTF CLOSE and then submit this section of code. Now you notice that the only thing in the header is the date/time information and that it doesn't have any page breaks until they are needed in the document.

If you want to remove the date/time information, you can add (and submit) the following statement before the ODS RTF command (or at the very beginning of your SAS code): OPTIONS NONUMBER NODATE;

Normally you would simply add the initial statement at the beginning of your code and the close at the end. But be careful not to include items like a full print of your data. Try to make sure that only the results you want are included in the file to minimize the amount of editing needed.

That's it for this tutorial, hopefully you will find this a helpful skill for working with output in SAS!

Topic 4 – Frequency Distributions

Welcome - this video covers creating frequency distributions for categorical variables. In order to follow along you will need the SAS dataset PULSE_STEP1 (after labeling the variables). I have opened SAS and opened the SAS program, Topic4.sas

We will start with printing the dataset, followed by looking at the variable list from PROC CONTENTS.

Notice that the raw data are numbers for our categorical variables. We cover how to translate these codes in our output in other tutorials. We will discuss the difference after we have produced the results for this data.

Again, I often use the results of PROC CONTENTS to remind myself of the exact variable names for variables in my SAS dataset.

The procedure we will use to obtain frequency distributions is PROC FREQ. We select the dataset with DATA = BIO.PULSE_STEP1;

The command to obtain frequency distributions is TABLES, we will be requesting the table for the variable GENDER. End the line with a semicolon and add our RUN statement.

Select the code and submit to see the results. Notice that the values of the variable are given as 1 and 2. This will not be allowed for submissions on assignments in the course but for our first assignment the raw data are NOT coded so the method in this tutorial will work to produce output which is descriptive enough.

Now, we will analyze all of our categorical variables at one time so I simply add GENDER, SMOKES, ALCOHOL, EXERCISE, and TRT to the TABLES statement.

Here we will add an option to obtain bar charts for our variables. The / is how SAS specifies an option to a statement that is not the main PROC statement. The option we add is PLOTS = FREQPLOT.

Submit this code to see the results. We obtain a frequency table and a plot for each variable. Notice the variable labels added in an earlier tutorial are appearing in both the tables and the graphs.

We can request bar charts directly using PROC SGPLOT and the statement VBAR (for vertical bars) or HBAR (for horizontal bars). You can only produce bar charts one at a time using SGPLOT but the SGPLOT procedure may offer more control over how the plot is displayed if you investigate.

I don’t generally alter SAS graphs. If I need a graph to look differently, I tend to create it in another software package but SAS does have the ability to do just about anything if you have a high enough programming skill level in SAS.

That's all for creating frequency distributions for categorical variables. Thanks for watching!

Topic 5A – Numeric Measures using PROC MEANS

Welcome – this tutorial will cover calculating numeric measures for one quantitative variable using PROC MEANS. In order to follow along you will need the SAS dataset PULSE_STEP1 (after labeling the variables). I have opened SAS and opened the SAS program, Topic5A.sas

We will start with printing the dataset, followed by looking at the variable list from PROC CONTENTS.

Notice that the raw data are numbers for our categorical variables. We cover how to translate these codes in our output in other tutorials. However, that won't matter for this tutorial as we will be focusing on our quantitative variables: HEIGHT, WEIGHT, AGE, PULSE1, and PULSE2.

We begin with the default output for PROC MEANS. We choose our data and use a VAR statement to select only one variable to analyze. If you do not specify a variable in the VAR statement, or possibly if you don't even put a VAR statement, SAS will analyze all of the variables in the dataset which it THINKS are numbers. This is usually not a good approach. It is better to request the variables of interest.

Let's select and submit this code. By default we only get these 5 values. The sample size N, the Mean, Standard Deviation, Minimum, and Maximum. Notice also that we tend to get a ridiculous number of decimal places.

We can request additional values by adding keywords to the PROC MEANS statement. Here we add N MEAN STD MIN Q1 MEDIAN Q3 MAX to the first line of PROC MEANS, after the data set name but before the semicolon. We will also request rounding with MAXDEC=3.

We will also add all of our variables to the list. Notice that the variable name WEIGHT changes colors, this is because WEIGHT is sometimes an option to variables in SAS and is unfortunate in this case. This doesn't cause SAS any problems but sometimes these kinds of oddities cause new student (or old students) of SAS to be concerned.

Let's submit this code. Notice that it puts all of the variables in one table. This actually doesn't copy very well, especially with the labels in the 2nd column. You may find you prefer the final results if you still request results for each variable individually. Unfortunately that would require running 5 separate instances of PROC MEANS.

That's all for this video on calculating numeric measures for one quantitative variable using PROC MEANS. This is very useful, with one tool we have found all of the numeric measures we need. Thanks for watching!

Topic 5B – Creating Histograms and Boxplots

Welcome – this video covers creating histograms and boxplots using PROC SGPLOT. In order to follow along you will need the SAS dataset PULSE_STEP1 (after labeling the variables). I have opened SAS and opened the SAS program, Topic5B.sas. We will start with printing the dataset, followed by looking at the variable list from PROC CONTENTS.

Notice that the raw data are numbers for our categorical variables. We cover how to translate these codes in our output in other tutorials. However, that won't matter for this tutorial as we will be focusing on our quantitative variables: HEIGHT, WEIGHT, AGE, PULSE1, and PULSE2.

To create a histogram we use PROC SGPLOT and a HISTOGRAM statement which contains the variable of interest. For each variable, we will need a new PROC SGPLOT procedure. We can't add multiple variables to the HISTOGRAM statement (or other similar one-variable plot requests in SGPLOT).

The default histogram code is simple. The variable of interest goes in the histogram statement. We aren't using any options. We begin our procedure as expected and end it as expected with a RUN statement. We get a nice default histogram for height where we can see the variable label is being used on the x-axis instead of the variable name.

Now let's add two density curve overlays. These are estimates of what the population might look like under certain assumptions. We still begin with a histogram, then we can request additional plots on the same variable with this same SGPLOT procedure.

The first DENSITY command with TYPE = NORMAL will add a normal curve with a mean and a standard deviation equal to those values from our sample. In other words, we are assuming the population is a normal distribution with mean and standard deviation the same as that observed in our sample. This tends to be too strong of an assumption in practice except for a small subset of variables.

The second DENSITY command with TYPE = KERNEL will add a "best guess" curve that makes no assumptions about what the population curve might look like – this tends to produce a graph that looks very similar to the histogram. The closer this looks to the normal curve, the more reasonable it is that the sample taken was drawn from a normally distributed population. In the case of height, the variable seems fairly normally distributed with only minor differences between the red KERNEL curve and the blue NORMAL curve.

Now we can repeat this for the rest of our variables. For the remaining variables, WEIGHT is mildly skewed right (but clearly so), AGE is highly skewed right, PULSE1 is basically symmetric with one or two very large outliers – possibly we will remove this observation later but it does require some investigation. For PULSE2, the graph is bimodal (and skewed right) with the left cluster being narrow and the right cluster more spread out. Since about half of the students ran and half sat, this isn't surprising.

Now we look at boxplots. We will still use PROC SGPLOT but the command is now either VBOX (my personal preference) or HBOX. The first creates vertical boxplots and the second horizontal. You can choose either for any boxplots requested. Here we have the code for both VBOX and HBOX for HEIGHT and then we have the code for the remaining variables using VBOX only. I don't usually use any boxplot options for modifying a single boxplot but later we will cover making side-by-side boxplots for comparing groups in Case CQ.

That's it for creating histograms and boxplots for one quantitative variable at a time using PROC SGPLOT. Thanks for watching!

Topic 5C – Creating QQ-Plots

Welcome – this video will illustrate creating QQ-Plots and other plots using PROC UNIVARIATE. This procedure can also provide a very detailed numerical summary of a quantitative variable but here we wish to see only its graphical output.

In order to follow along you will need the SAS dataset PULSE_STEP1 (after labeling the variables). I have opened SAS and opened the SAS program, Topic5C.sas. We will start with printing the dataset, followed by looking at the variable list from PROC CONTENTS.

Notice that the raw data are numbers for our categorical variables. We cover how to translate these codes in our output in other tutorials. However, that won't matter for this tutorial as we will be focusing on our quantitative variables: HEIGHT, WEIGHT, AGE, PULSE1, and PULSE2.

To create Quantile-Quantile plots, commonly abbreviated QQ-Plots, we use PROC UNIVARIATE as PROC SGPLOT does not offer this plot option. Note that the QQ does not stand for the variable types as in our CASE QQ where we are looking at two quantitative variables, the Q's in QQ-plot stand for quantile which is the method of comparison of our one quantitative variable to the theoretical distribution of interest – in our case, the normal distribution. Although they will look like scatterplots, there is only one variable from our sample represented in the graph.

We begin with PROC UNIVARIATE and select our data set as usual. We add the NOPRINT option since we only want to see graphs in our output. If you wish to see the full default output, remove that option. If you wish to see everything UNIVARIATE can do replace it with the ALL option. We end our first line with a semicolon.

Then we have a VAR statement to specify the variables we want to analyze. Similar to PROC MEANS if we do not put a VAR statement, we will obtain results for all of the variables in our dataset. We definitely don't want these kind of results for our categorical variables! In this case we will begin with one variable HEIGHT.

Next we illustrate three graphs designed to compare to a normal distribution. We always request the first, the QQ-plot but the others are probability plots and probability-probability plots. The more all three of these follow a straight line the more normal. The last will always match at beginning and end so the deviations to consider are those in the middle of the plot. A perfectly normal distribution would follow the line perfectly but that is not expected in real data unless the sample size is extremely large and the population is normally distributed.

Then we request the histograms as well. First with the default settings, then with the normal curve and kernel overlays. The idea is the same as SGPLOT but the code is different. The NOPRINT option here will suppress a table of information that displays about the normal estimation process. If you wish to see that output, remove the NOPRINT option.

Here are the plots from that code.

We can add all of the variables of interest to the VAR statement. Here I requested only the QQPLOT and one HISTOGRAM for each of the variables. We have the same results mentioned for the histograms in the previous tutorial.

In the case of height, the variable seems fairly normally distributed with only minor differences between the red KERNEL curve and the blue NORMAL curve. The only difference in this histogram and the one from SGPLOT is some minor formatting differences including the added text which relates to the estimation of the overlay curves.

For the QQ plot, notice the dots are generally randomly bouncing around the line and staying very close to it, except the first and last observations. This confirms that this variable is close to normally distributed based upon this sample.

For the remaining variables, WEIGHT is mildly skewed right (but clearly so) – in the QQ-plot we see this as an upward curve in SAS output, AGE is highly skewed right – again an upward curve in our QQ-plot.

PULSE1 is basically symmetric with one very large outlier – the QQ-plot clearly shows this one observation as unusual and the rest of the data are reasonably normal if that observation was ignored, potentially it is an error from self-collected pulse readings.

For PULSE2, the histogram is bimodal (and skewed right) with the left cluster being narrow and the right cluster more spread out. Since about half of the students ran and half sat, this isn't surprising. The QQ-plot shows an S-shaped pattern which overall has an upward curve. There are numerous "S-Shaped" patterns so be sure to look at both the histogram and QQ-plot to determine the whole picture.

Really for each quantitative variable you would like to use all three plots, a histogram, a boxplot, and a QQ-plot to develop a complete picture of the distribution of the values of the variable in your sample.

That's it for creating QQ-plots (and histograms) using PROC UNIVARIATE. Thanks for watching!

Topic 6A –Two-Way (Contingency) Tables

Welcome – this video discusses exploratory data analysis for Case CC, two categorical variables – specifically, here we will be creating two-way tables, also called contingency tables, which provide a breakdown of the number of observations in each combination of the levels of the two categorical variables. Percentages will also be automatically calculated by the software.

In order to follow along you will need the SAS dataset PULSE_STEP2 (after labeling all variables and translating the original coded categorical variables).

Since we will be working with a dataset with permanently assigned formats, we will need to begin our SAS program with the PROC FORMAT code which defines those formats.

The PROC FREQ procedure and the TABLES statement are used here also but the request is different. In order to request two-way tables, we place a * between variables. Multiple requests can be made in one tables statement and variables can be grouped in parentheses. The variables to the left of the * are the rows and those to the right, the columns.

Here we request an individual two-way table which has ALCOHOL in the rows and EXERCISE in the columns. Then we requests four tables, one for each of the variables GENDER, SMOKES, ALCOHOL, and EXERCISE in the rows where the column variable will be TRT.

These tables report a number of values by default. You can see the order in the upper left corner. We get

- The FREQUENCY which is the overall count in that cell
- The PERCENT which is the overall percentage for that cell out of the total, this is usually not of interest in any way as we are interested in looking at the distribution of one variable across the levels of the other variable but are not interested in the percentage out of the overall total
- The ROW PCT which is the percentage for that cell out of the total for THIS ROW
- The COL PCT which is the percentage for that cell out of the total for THIS COLUMN

The row and column percentages are potentially useful but often only one of these will be useful. We break down the response variable within the levels of the explanatory variable but usually not the other way around. Normally I place my explanatory variable in the rows but this is a personal choice. Unless we specify otherwise in a specific assignment, the choice is yours regarding which variable is in the rows and which in the columns.

Notice that we do see the translations for the variable values as well as the variable labels in the output.

That’s all for this tutorial on two-way tables for exploratory data analysis in Case CC.

Topic 6B –Two-Way (Contingency) Tables

Welcome – this video discusses inferential methods for Case CC, two categorical variables – specifically, here we will be creating two-way tables as we have before and adding to this the calculation of the standard test, the chi-square test for independence and the non-parametric alternative, Fisher's exact test. We will also show how to add the expected cell counts used for the standard chi-square test to our output.

In order to follow along you will need the SAS dataset PULSE_STEP4. The code for our formats which translate our categorical variables is given first. We have two new formats for two variables we will create in the DATA step that follows.

Then we have the needed data manipulation code. For the remaining analyses, I wanted to remove the observation with the unusually large resting pulse. I don't remember exactly the value but it was clearly larger than 140 when I looked at a boxplot of PULSE1.

We begin with the DATA statement and name our new dataset BIO.PULSE_STEP5. Then we pull our data from BIO.PULSE_STEP4 in the SET statement. This was the dataset after categorizing body mass index.

To remove observations we can use a DELETE command at the end of an IF-THEN statement. All values satisfying the IF condition will be removed from the dataset. Here we use IF PULSE1 > 140 THEN DELETE. This removes all observations with a PULSE1 value larger than 140. There should only be one of these.

Then we categorize the quantitative variable weight into a binary variable with the next two IF-THEN statements and a 5-level categorical variable with the next 5 IF-THEN statements. Finally we label and format the two new variables BINARYWT and WTGroups.

Now we submit the PROC FORMAT and DATA step. Notice that in the log file, we have one less observation in the STEP5 data than the STEP4 data as it gives the notes: There were 107 observations read from the data set BIO.PULSE_STEP4. The data set BIO.PULSE_STEP5 has 106 observations and 17 variables.

To begin we will look at the treatment variable – whether the student ran or sat – as the single row variable – and pair it with other categorical variables in this version of the dataset – gender, regular smoker?, regular drinker?, and frequency of exercise.

This is the same code we used for Case CC in exploratory data analysis. We simply add a few options. We use PROC FREQ on the STEP5 data. We use a tables statement. The * indicates we want a two-way table. The form is ROW*COLUMN. Here we put TRT first, which will be the rows, and then the * followed by, in parentheses, the list of variables of interest to pair with TRT. In this case we have GENDER, SMOKES, ALCOHOL, and EXERCISE.

Now we add the forward slash followed by our new options, CHISQ FISHER EXPECTED. CHISQ gives the results for the chi-square test, FISHER gives the results for Fisher's exact test (this can be computationally intensive for large datasets and/or multi-level variables), EXPECTED adds the expected cell counts to the two-way table results reported.

Submitting this code, we get the two-way table for the first pair – treatment by gender. In each cell, we get the frequency (count), expected count, percent, then row percentages, and finally column percentages. We are most interested in the row percentages here since we want to compare the distribution of a column variable between the two treatments (RAN or SAT).

The distribution of gender is similar for both those who sat and those who ran. Under the table we get the results of the tests. Before looking at the results, it is a good habit to check for warnings about the expected counts.

In this case, none of the cells have expected counts less than 5 and SAS gives no warning. Thus we can feel comfortable applying the Chi-square test for these variables.

For 2x2 tables, we want the Continuity Corrected Chi-squared test which has a test statistic of 0.2105 and a p-value of 0.6464 in the Prob column. For 2x2 tables, you may get the results of Fisher's exact test whether you request it or not. Here the 2-sided p-value for Fisher's exact test is 0.5568. None of the other results are needed here but you are welcome to research their uses on your own.

So there is not enough evidence that there is an association between gender and the treatment variable. This is a good thing since we would like our treatment groups to be similar with respect to other variables.

For treatment vs. regular smoker, we do have 1 cell with an expected count less than 5. This expected count was 4.57 which is not too bad but Fisher's exact test may be more appropriate. Notice, as is often the case, we do get the same overall conclusion from both tests with the Continuity corrected p-value for the Chi-square test of 0.4908 and for Fisher's exact test of 0.3554. Again there is not enough evidence of an association between smoking status and treatment.

For treatment vs. regular drinker, there are no concerns with using the Chi-square test as none of the cells have expected counts less than 5, the smallest expected count is 16.6. The Continuity corrected p-value for the Chi-square test is 0.3923 and for Fisher's exact test is 0.3160. So again there is not enough evidence of an association between regular drinker and the treatment variable.

For treatment vs. frequency of exercise, there are no concerns with using the Chi-square test since the minimum expected count is 5.81. For any size larger than 2x2, there is no continuity corrected p-value and we use the standard Pearson Chi-square (this is the same person that developed Pearson's correlation coefficient but they are different methods for each specific situation).

The two-sided p-value for the Chi-square test is 0.7703. Since we requested Fisher's exact test we obtain it here with a p-value of 0.7906. So again there is not enough evidence of an association between frequency of exercise and the treatment.

These are all good news for our experiment but let's try to find some related variables. Let's look at GENDER vs our two categorical versions of weight. Let's submit this code.

For gender vs. the binary weight variable, we selected a fairly large value as the cutoff and so we don't end up with any females in the highest category. This isn't necessarily a problem as it is the expected counts that are important. However, here we do have 2 out of the 4 cells with expected counts less than 5, with a minimum of 4.16. This isn't too bad but Fisher's exact test would be more reliable.

We may prefer to use Fisher's exact test as we just mentioned and its p-value is 0.0034. Thus there is enough evidence of an association between gender and weight (not particularly surprising). If it had been reasonable to use the chi-square test, since this is a 2x2 table the appropriate chi-square test would be the continuity corrected with a p-value of 0.0105.

For gender vs. the multi-level weight variable, there is no concern for using the Chi-square test with a minimum expected cell count of 7.4. The p-values for everything are <0.0001 but we are interested in the Chi-square p-value in the first row and the Fisher's P-value in the 2nd row of the last table. The Table Probability in the Fisher's output is NOT a p-value but part of how the test is conducted.

So in this way of investigating we also find clear evidence of an association between gender and weight.

Learning to read the output in these tables is an important learning objective so please let us know if you have questions. That's all for this tutorial.

Topic 7A – Numeric Summaries by Groups

Welcome – this tutorial will cover exploratory data analysis in Case CQ (or QC) where we have one quantitative variable and one categorical variable. Here we will cover calculating numeric measures for the quantitative variable WITHIN EACH LEVEL of the categorical variable using PROC MEANS and a CLASS statement.

In order to follow along you will need the SAS dataset PULSE_STEP2 (after labeling all variables and translating the original coded categorical variables).

Here we will look at the quantitative variable PULSE1 which is the resting pulse rate by each of the categorical variables. This will require a different PROC MEANS procedure for each request.

Since we will be working with a dataset with permanently assigned formats, we will need to begin our SAS program with the PROC FORMAT code which defines those formats.

PROC MEANS is also used to create numerical summaries by groups. Only one grouping variable can be defined but multiple quantitative variables can be requested in the VAR statement although we won't need to do so here in this tutorial.

The code for PROC MEANS is identical as that used for one quantitative variable except after the main PROC MEANS statement we add a CLASS statement with the categorical variable we want to use to define groups for this analysis. We have one request for each of the categorical variables, GENDER, SMOKES, ALCOHOL, EXERCISE, and TRT. In each case we are only looking at PULSE1 but you could add other quantitative variables to the VAR statement if desired.

Here we obtain the requested numerical summaries for resting pulse rate within the levels of each of our categorical variables. This output, combined with side-by-side boxplots will provide a good comparison of the distribution of the quantitative variable within the levels of the categorical variable.

Finding numerical summaries is very easy using PROC MEANS. It can be useful for one quantitative variable or for situations such as this involving one quantitative variable and one categorical variable.

That's all for this tutorial on numerical summaries for a quantitative variable by groups defined by a categorical variable.

Topic 7B – Side-By-Side Boxplots

Welcome – this video covers creating side-by-side boxplots for Case CQ (or QC) where we have one quantitative variable and one categorical variable.

In order to follow along you will need the SAS dataset PULSE_STEP2 (after labeling all variables and translating the original coded categorical variables).

Here we will look at the quantitative variable PULSE1 which is the resting pulse rate by each of the categorical variables. This will require a different PROC SGPLOT procedure for each request.

Since we will be working with a dataset with permanently assigned formats, we will need to begin our SAS program with the PROC FORMAT code which defines those formats.

PROC SGPLOT is also used to create side-by-side boxplots by groups. We can use HBOX or VBOX.

The code for PROC SGPLOT is identical as that used for one quantitative variable except after the variable name in the VBOX (or HBOX) statement, we add a forward slash to indicate we want to request an option and add the CATEGORY option with the categorical variable we want to use to define groups for this analysis.

We have one request for each of the categorical variables, GENDER, SMOKES, ALCOHOL, EXERCISE, and TRT. In each case we are only looking at PULSE1. To analyze other quantitative variables would also require additional SGPLOT procedures as we can only look at one quantitative and one categorical variable at a time with this procedure.

Looking quickly through the boxplots we see that females have a slightly higher resting pulse rate and there is one male with an extreme outlier – could be an error.

Comparing smokers and non-smokers is difficult due to the small sample size for smokers in the data – overall the comparison is less overall variation but this could be a result of the lack of data.

There is more variation among regular drinkers than non-drinkers but the center is about the same.

For exercise, it does seem there is a weak trend that the more you exercise, the lower your resting pulse rate – as we might expect.

And finally for treatment – the groups have a similar distribution of resting pulse rates – we will adjust for initial pulse rate in our analysis but it is still good that our groups were reasonably similar for our Ran and Sat groups to begin with.

The outlier does concern me – so likely I will investigate and possibly remove it in the next set of data manipulations where we will create some new variables.

That’s all for this tutorial on side-by-side boxplots in Case CQ (or QC).

Topic 7C – Two Independent Samples T-test

Welcome – this video covers the two-sample t-test for independent samples. This test will compare the mean of a quantitative variable for the two groups defined by a binary categorical variable which falls into Case CQ.

In order to follow along you will need the SAS dataset PULSE_STEP5. Here we will start by looking at how resting pulse rates compared between the two treatments. Hopefully we have relatively comparable groups from the beginning.

First we submit the needed formats. The procedure used for this test is PROC TTEST. This procedure can also be used for one-sample t-tests and paired t-tests although the code used varies.

We begin with PROC TTEST on our STEP5 data. The rest of the code is the same as for PROC MEANS in Case CQ. We have a CLASS statement for our binary categorical explanatory variable and a VAR statement for our quantitative response variable. Finally we have our RUN statement. Let's submit this code.

For the two-sample t-test, our first decision is whether or not we feel comfortable assuming equal variances. The test for equal variances has a null hypothesis that the variances are equal and an alternative that they are not equal. The p-value needed is in the "Equality of Variances" table under the "Pr > F" column. Here the p-value for the test of equal variances is 0.4709 and so we fail to reject the null hypothesis that the variances are equal.

Although we cannot prove the null hypothesis is true, we do not have enough evidence that they are different. This seems reasonable given that the standard deviation of the resting pulse rates for those who Ran is 11.3840 and for those who Sat is 10.3087. Although there may be a difference it is not statistically significant and any difference that may exist would likely be too small to have a big impact on the results of the t-test.

Thus we will continue with the t-test where equal variances are assumed. This is the first row in the output. The t-statistic is -0.74 and the two-sided p-value is 0.4610 under the "Pr > |t|" column in the "Pooled" row. Thus there is not enough evidence that the mean resting pulse rate differs between the two treatment groups.

We can see that the confidence interval (in the "Pooled" row) ranges from -5.7789 to 2.6381 which includes zero and thus we cannot show a statistically significant difference. This is good in that we would like our two groups to be similar with respect to resting pulse rates.

Notice that we get numeric and graphical summaries of interest for each treatment group: we get the mean, standard deviation, standard error, min and max, confidence intervals for the mean and standard deviation. Then at the end we get histograms with normal and kernel density curves with boxplots underneath for each treatment group in a panel followed by QQ-plots for each treatment group in another panel.

We can also calculate summaries using PROC MEANS and construct side-by-side boxplots using PROC SGPLOT. The only new code we have below is that we add the CLM option to the PROC MEANS keyword list to obtain confidence intervals for the population mean to our requested summaries. Submitting this code we can see the confidence intervals are added to our proc means results. And looking at the boxplots confirms the lack of a statistically significant difference between the mean resting pulse between those who RAN and those who SAT.

Now we will look at an example where we will be able to find a significant difference between means. For the categorical explanatory variable, we choose gender and for the quantitative response variable we choose weight. Let's submit this PROC TTEST code.

Again, our first decision is whether or not we feel comfortable assuming equal variances. Here the p-value for the test of equality of variances is 0.0003. So there is evidence that the variances of the two groups are different.

From the summary table this seems reasonable as the standard deviation for males is 14.3 and that for females is 8.5.

Thus we need to use the t-test which does not assume equal variances which is the 2nd row in the table. The test-statistic is 7.92 and the p-value is <0.0001 under the "Pr > |t|" column in the "Satterthwaite" row. Thus there is a highly statistically significant difference in the mean weight between males and females.

The confidence interval (in the "Satterthwaite" row) ranges from 13.3408 to 22.2765. Since these are calculated as Group 1 (Males) minus Group 2 (Females) and this value is always positive, we know the mean for males is larger than that for females and we can interpret this interval by saying we are 95% confident that the mean weight among males is 13.3 to 22.3 kg greater than that for females.

Again, we can review the other numeric and graphical summaries to better see what our comparison might actually mean in practice. We can calculate our own numeric summaries and construct graphs. Here we again use PROC MEANS and create side-by-side boxplots using PROC SGPLOT.

Notice here I have split the PROC MEANS requests into two groups, the sample size, mean, standard deviation, and confidence interval in the first request and the five-number summary in the second. This can help keep the tables short and here the grouping makes sense.

We can clearly see the difference in the distribution of weight for females and males from the boxplots. From the PROC MEANS results, we can see that for this sample the difference in the means is 74.9-57.1 = 17.8 and for the medians is 73-56 = 17.

That's all for this tutorial on two-sample t-tests in Case CQ.

Topic 7D – One-Way ANOVA (Analysis of Variance)

Welcome – this video covers the one-way ANOVA. This test compares the means of a quantitative variable for the groups defined by a categorical variable. The groups must be independent samples.

For example, we cannot use this to look at multiple measurements taken on the same individual over time as the same person would be reflected in each group and the measurements in one group and those in another would be dependent, not independent.

Although the one-way ANOVA test can be used for two-groups, in that case we will always use the two-independent samples t-test in this course. In order to follow along you will need the SAS dataset PULSE_STEP5. Here we will start by looking at the relationship between height and the multi-level weight groups we created in an earlier tutorial. First we submit the needed formats.

This is a new procedure. PROC GLM. The code is similar to PROC REG with the possibility of adding a CLASS statement (PROC REG doesn't let us add classification variables in this way). We start with PROC GLM on our data. Then we have a CLASS statement which contains the multi-level categorical explanatory variable of interest. Here we are using WTGroups, our categorical 5-level weight variable.

Then we have a MODEL statement. This is the same as for PROC REG except now we can have a categorical variable on the right of the equals sign. The quantitative response variable HEIGHT goes on the left of the equals sign and our categorical explanatory WTGroups on the right. That is all you will need for this course but I will show you how to request post-hoc tests such as Tukey and Bonferroni multiple comparisons.

To do this we use an LSMEANS statement which contains the categorical variable we wish to use for our comparisons, in this case WTGroups. We have added this statement twice, once to request Tukey's comparisons and then to request Bonferroni's. Then we have a RUN statement and a QUIT statement. Like PROC REG, PROC GLM needs a QUIT statement to completely end the procedure. Let's submit this code.

For this course, we are primarily interested in the main ANOVA table. Here we have a F-value of 34.18 and a p-value of < 0.0001 in the "Pr > F" column. Thus there is enough evidence that the mean height is different among some of the weight categories. We also get a graph of side-by-side boxplots appropriate for this comparison.

The Tukey and Bonferroni output provides the mean height for each group followed by a pairwise table with the p-values for all possible combinations of weight groups. There are some groups which are not statistically significantly different but many of them are. The last graph displays this information but I find it difficult to read and prefer the pairwise table in combination with the values of the means in practice.

We can also add our own numeric and graphical summaries with PROC MEANS and PROC SGPLOT.

Now we will run through that one more time to compare the mean Age between the 5 weight groups. Submitting this code and looking at the ANOVA table, we see that the p-value is 0.3797 and thus there is not enough evidence that the mean age is different between any of the 5 weight groups. The boxplots seem to confirm that any differences seen could be due to chance. We can add the PROC MEANS and PROC SGPLOT results as well.

That's all for this tutorial on one-way ANOVA in Case CQ.

Topic 7E – Non-Parametric Tests for Case CQ

Welcome – this video covers the non-parametric alternatives to the two-independent samples t-test and one-way ANOVA. We will cover the alternative to the paired t-test in a different tutorial.

These tests compare the median of the quantitative variable instead of the mean. If the distributions in the groups are symmetric then this could also be considered a test for the means but not in the case of skewed distributions.

One assumption is that there is only a location shift between the groups. This can limit the usefulness of this test in practice and we should check to see if the distributions are similar using side-by-side boxplots or other graphical displays. We must also remember though that this assumption is about the population and so small deviations are not likely a problem but major differences would be.

We will look at Height vs. gender, Height vs weight categories, and Age vs. weight categories.

All of these have relatively similar distributions. The relatively minor differences seen in the shapes could be due to chance. Thus these tests are reasonable.

We begin with the Wilcoxon Rank-Sum test, also known as the Mann-Whitney U-test. This is for two groups and is analogous to the two independent samples t-test.

First we submit the needed formats. The NPAR1WAY procedure with identical code is used for both the Wilcoxon and Kruskal-Wallis test and it is similar to PROC TTEST or PROC MEANS code.

We begin with PROC NPAR1WAY on our STEP5 data. Then we provide our categorical explanatory variable in the CLASS statement. For a two-level variable, this will produce the Wilcoxon Rank-Sum test. Here we choose gender as our explanatory variable. Then we have a VAR statement with our quantitative response variable. Here we choose height. We have a RUN statement and a QUIT statement to finish this procedure.

Submitting this code, we see a table with some information about the scores used for this test within each gender. Then we have a table with the Wilcoxon Two-Sample Test results. We get information about the statistic and then two possible tests, one based upon a normal approximation and one based upon a t-based approximation.

The two-sided Pr is what we would want regardless of which of these we use. Both p-values are trying to approximate the same probability and either could be used. In this case both are <0.0001 and so there is enough evidence that the median height is different between males and females. We also get the Kruskal-Wallis test (which also has a p-value <0.0001). Just like the ANOVA, this would be valid for two groups but usually would be used when there are more than two-groups. The boxplots given in these results are based upon the SCORES not the original data so they aren't very useful to us.

Now we will look at two cases where the Kruskal-Wallis test is appropriate. This is analogous to the One-Way ANOVA. We will look both height and age vs. weight categories. We put WTGroups in the CLASS statement and both the quantitative variables of interest, height and age, in the VAR statement. This will provide two tests. One for Height vs. WTGroups and one for Age vs. WTGroups.

Submitting this code, not surprising, the test finds that some medians are statistically significantly different between the weight categories for height (p-value <0.0001) but not for age (p-value = 0.4395). This seems reasonable given the results obtained for ANOVA and our exploratory results.

That's all for this tutorial on the Wilcoxon Rank-Sum test and the Kruskal-Wallis test.

Topic 8A – Setting up Data for Paired T-test

Welcome – this video will modify the current dataset so that we can conduct a paired t-test on the actual differences for each of our two treatment groups. We want to conduct a test to determine if the mean pulse rate changes among those who ran and again, another test for those who sat.

In order to follow along you will need the SAS dataset PULSE_STEP5. We will calculate the differences between pulse2 (after the treatment) and pulse1 (before the treatment). We will expect an increase for those who ran and basically no difference for those who sat.

First we submit the needed formats. Then we have a DATA step defining a new dataset BIO.PULSE_STEP6 followed by a SET statement bringing in data from BIO.PULSE_STEP5.

To create a new variable we begin with the new variable name. Variable names can't have any spaces and there are some characters that are not allowed. Here we name the new variable DIFF_2v1 and follow this with an equals sign and then our equation PULSE2 – PULSE1 to calculate the difference of interest. Then we provide a variable label of DIFF (Pulse2 – Pulse1) in a LABEL statement. Finally we have our RUN statement.

We submit this code and check our log file. We don't see any errors. There were 17 variables and 106 observations in the STEP5 data and here we see that the new data has 106 observations and 18 variables.

We will check the data now using PROC PRINT. We will only look at 10 observations. Here we will add a VAR statement to PROC PRINT. This allows you to print only the variables you want to see and specify the order in which you want to see them. Very nice for checking only subsets of variables of a larger dataset.

For these first 10 observations, we can see the treatment, the original pulse measurements and their difference. Everything seems in order. In PROC CONTENTS we can see that our new variable is labeled and named as expected.

Finally we can look at the boxplots to get an idea how these differences are distributed for each treatment group.

We could conduct a two-sample t-test to compare these differences but here it is VERY clear without any statistical methods that there is a big difference in the change in pulse for these two groups. For those who SAT the values cluster near zero and have very little variation. For those who RAN, the change in pulse rates seem to have both the mean and median a little over 50 beats per minute and a very wide spread.

That's all for this tutorial.

Topic 8B – Exploratory Analysis of Difference

Welcome – this video will investigate the differences in pulse rates directly using exploratory methods.

In order to follow along you will need the SAS dataset PULSE_STEP6. First we submit the needed formats.

Now we will look at PROC MEANS on the difference by treatment. I split this into two sets for nice output. Running this code, we see that among those who ran, the mean change in pulse is 52.4 and the median is 54.5 with a standard deviation of 20.6. Among those who sat, the mean is -1.03 and the median is -1, with a standard deviation of 3.9.

Then we create QQ-plots for both treatments. Here we are using a new statement the WHERE statement, which allows us to choose only a portion of the observations of the dataset based upon the values of other variables in the dataset. Here we run PROC UNIVARIATE only for observations WHERE TRT=1. This selects only those who RAN since the value of TRT was 1 for that group. We ask for a QQPLOT with the diagonal line for the normal distribution.

We repeat this for those who SAT with the statement WHERE TRT = 2 and submit this code. Notice SAS doesn't label which group is used so we have to know or add our own titles so that we know which it is.

We might be mostly interested in these QQ-plots to see if normality is reasonable, especially for small sample sizes. We can see that the data do appear to be reasonably normally distributed since the observed values fall close to the line of values representing the expected values for a normal distribution.

We can create side-by-side boxplots as we did in the previous tutorial. But since we are interested in looking at paired t-tests within each of these groups, we might not really want to see them side-by-side, at least not as our only plot. We might want to get a better picture of each group individually, especially for those who SAT as the scale makes it difficult in the side-by-side boxplots.

To solve this we can again use the WHERE statement. We create a simple, single variable boxplot without the CATEGORY option and add a WHERE TRT = 1 for the first set of code and WHERE TRT = 2 for the second set. This will give a single boxplot for the two groups. These are not well-labeled by default, here I added titles to the plots which helps. If you don't clear your titles they are in effect until the next title statement comes so you can get badly labeled results if you don't immediately clear the titles.

In these boxplots, we can now see that the line at the median is a little lower than zero and that the distribution extends a little more in the negative direction than the positive.

We could create similar code for histograms or any other exploratory analysis of interest.

That's all for this tutorial.

Topic 8C – Paired T-test and Non-Parametric Alternatives

Welcome – this video will conduct the paired t-test. There are a few ways we can do this. Generally you only need to do one of these for any requested analyses in this course.

In order to follow along you will need the SAS datasets PULSE_STEP6. First we submit the needed formats. We will only look at results for those who RAN but a similar process could be used for those who SAT.

First we will use the original pulse measurements. Here we get to choose the order of the difference, we will choose the same order we used when we created the difference ourselves, PULSE2 – PULSE1. To do this we use PROC TTEST on our STEP6 data. We do NOT have a CLASS statement in a paired t-test as we did for the two-sample t-test because these are not independent samples here.

We are adding a WHERE statement to select only individuals who RAN which has TRT =1.

Then we have the PAIRED statement. This is only for paired t-tests and defines the differences with a * between the variables. We have PULSE2 * PULSE1 to calculate the differences as PULSE2 – PULSE1. Then we have our RUN statement. Submitting this code we first get two tables with numeric summaries of the differences.

Then we get the results of the paired t-test itself, we see the t-value is 16.88, this is the test statistic, and the p-value is <0.0001 indicating there is a statistically significant difference in the mean pulse rate before and after treatment among those who Ran.

The confidence interval (from the previous table) suggests with 95% confidence that, among those who ran, the mean pulse rate after treatment is between 46.1 and 58.7 beats faster than the mean pulse rate before treatment.

Notice that SAS automatically gives us a nice graphical display of the distribution of the differences with a histogram and a boxplot underneath. Then it gives some other graphs which you might find helpful for looking at paired differences. The paired profiles show the change for each individual with a line representing the mean change.

The agreement plot shows both measurements and a line to indicate where there would be no change. Since all values are higher than the line on the x-axis representing the second pulse measurement this shows all values increased from the first to the second pulse measurements. Finally we get a QQplot of the differences. The differences are reasonably normally distributed based upon this plot and the earlier histogram.

Alternatively we can use the differences directly. We can do this using either PROC TTEST or PROC UNIVARIATE, however PROC UNIVARIATE also is the method of obtaining the non-parametric tests, so this is more useful in practice.

To use PROC TTEST, we use PROC TTEST and add an option to that statement to specify the null hypothesis value, H0 = 0. We still have our WHERE TRT = 1 statement. Since we are using the differences directly, we use a VAR statement instead of the PAIRED statement. We end with a RUN statement and submit this code.

The results are the same where they are provided. The first three tables, histogram, and QQPLOT are exactly as before. We don't get the paired profiles or the agreement plot since SAS does not know the differences are anything other than a single variable.

To use PROC UNIVARIATE, we add an option to that statement which specifies the null hypothesis MU0=0 and an option to calculate confidence intervals CIBASIC. Again we have the WHERE statement to select only those who RAN and a VAR statement to specify the variable of interest. We request a QQPLOT with a normal line. Submitting this code will conduct the t-test, the sign test, and the signed-rank test.

We have all of the usual PROC UNIVARIATE output including the summary measures in the first two tables. We get the confidence intervals and tests in the next two tables. You can see the confidence interval for the mean is the same as we found using PROC TTEST and the t-value and p-value as well.

Notice that after the Student's t row, we have a row for the Sign test and then one for the Signed Rank test. All of which agree that the p-value is < 0.0001.

Then we get the rest of the UNIVARIATE output, including the QQPLOT we requested.

We can also request summaries ourselves using PROC MEANS and PROC SGPLOT. Here we request numeric summaries and a boxplot of the differences for this group.

That's all for this tutorial on paired t-tests in Case CQ.

Topic 9A – Basic Scatterplots

Welcome – this video covers creating scatterplots in Case QQ where we have two quantitative variables. We will use PROC SGPLOT and illustrate both the SCATTER statement and the LOESS statement.

In order to follow along you will need the SAS dataset PULSE_STEP2 (after labeling all variables and translating the original coded categorical variables).

Since we will be working with a dataset with permanently assigned formats, we will need to begin our SAS program with the PROC FORMAT code which defines those formats.

We begin with PROC SGPLOT using our STEP2 data. To create a simple scatterplot, we use the SCATTER statement. We choose a quantitative variable for the each axis. Let's predict Y = weight using X = height. We end with the RUN statement. Submitting this code, we see the resulting scatterplot. Notice that our X and Y axis labels use the variable labels we created in an earlier tutorial. Overall the trend seems reasonably linear with one point on the left which does not seem to fit the rest of the data.

Now let's look at adding a LOESS curve to the scatterplot. A LOESS curve is a smooth curve which approximates the running average in the data. The amount of smoothing determines how many points will be part of the average for a particular x-value. The goal is to see the overall pattern in the data, including any hidden non-linearity but without too much detail. We don't want to fit a model which tries to hit every point but shows the general trend.

The procedure is the same. The only change is that we replace the SCATTER statement with the LOESS statement. This statement will produce both the scatterplot points and the smooth curve in one statement.

The SMOOTH option must be positive. Values close to zero will have less smoothing. Let's look at an example of too little smoothing where we set the SMOOTH option to 0.1. This clearly shows too much detail.

Now if we change the SMOOTH option to 1.2, this looks almost like a straight line and is likely too much smoothing.

Looking at SMOOTH = 0.5 and 0.6, this would be about what we would find most helpful for this data. Overall the trend is slightly non-linear. It is increasing, but the increase seems steeper as height increases. This may be mostly driven by the farthest left point.

If we look back to the SMOOTH = 0.1 graph you can see that although the graph jumps a lot, the overall pattern is relatively linear through the center of the ups and downs except for that last point where the graph jumps up to meet it just as the curve is relatively linear through though points in our SMOOTH = 0.5 graph.

You can also remove the SMOOTH option and allow SAS to choose the "best" smoothing value for your data. It does usually do fairly well in these situations.

These LOESS lines can help investigate the trend in the average Y as X changes and can help use determine if linearity is a reasonable assumption in Case QQ.

That's all for this tutorial on basic scatterplots.

Topic 9B – Grouped Scatterplots

Welcome – this video covers creating grouped scatterplots. This is actually not part of any of the formal cases we cover in this course as it looks at two quantitative variables and one categorical variable simultaneously. However, these graphs are potentially useful in practice and we do briefly mention them in the course materials.

In order to follow along you will need the SAS dataset PULSE_STEP2 (after labeling all variables and translating the original coded categorical variables).

Since we will be working with a dataset with permanently assigned formats, we will need to begin our SAS program with the PROC FORMAT code which defines those formats.

We begin with PROC SGPLOT using our STEP2 data. To create a grouped scatterplot, we use the SCATTER statement. We choose a quantitative variable for the each axis. Let's predict Y = weight using X = height. Now we add the forward slash to indicate options and add GROUP = GENDER to show a different color point for males and females. We end with the RUN statement. Submitting this code, we see the resulting scatterplot. Notice that, not surprisingly, females are clustered in the lower range of heights and males in the upper range of heights.

We can also use the LOESS statement with the GROUP option to draw running-average trend lines for each group. The commands are exactly the same as before with the addition of the GROUP = GENDER option in the LOESS statement. Here we will look at SMOOTH = 0.5 and the default smoothing.

This shows that, as we would expect, the males and females do cluster together and gives the best guess at the trend based upon the data within each group. We can see that on average, the slope seems to be a little steeper for males than females, even considering the potential outlier in the female group.

Grouped scatterplots with LOESS trend-lines can definitely help us investigate more complex trends in our data.

That's all for tutorial on grouped scatterplots.

Topic 9C – Pearson's Correlation Coefficient

Welcome – this video covers calculating Pearson's correlation coefficient for Case QQ where we have two quantitative variables which are linearly related. Everything we need to learn about this procedure in this course will be covered in this tutorial.

In order to follow along you will need the SAS dataset PULSE_STEP2 (after labeling all variables and translating the original coded categorical variables).

Since we will be working with a dataset with permanently assigned formats, we will need to begin our SAS program with the PROC FORMAT code which defines those formats.

We should verify that the relationship is reasonably linear using a scatterplot before conducting this analysis. The correlation will be calculated for all pairs of variables which are placed in the variables list.

We looked at height and weight earlier and found that, although it is best to analyze within each gender, the data are only slightly non-linear when considered together. When we calculate the correlation coefficient here, it will give a measure of the strength and direction of the simple linear regression model through this data.

We begin with PROC CORR using our STEP2 data. Similar to PROC MEANS and UNIVARIATE, we have a VAR statement to specify which variables to analyze. These must be quantitative variables. Here we will only look at height and weight but you can put as many variables in this list as you wish. It will provide the correlation between all possible pairs of variables. The order of the variables is not important for PROC CORR.

In the output we get a list of the variables analyzed, followed by some simple summary measures of each variable individually.

Then we get the correlations themselves. Many students find this difficult to read at first. The correlation we need is 0.74138 in this case, the first number in the columns which intersect the variables HEIGHT and WEIGHT – between height and weight in row #1 – or between weight and height in row #2. It repeats for both combinations even though the values are the same – a little strange but standard in most statistical packages.

The second value is the significance, which we will learn later is the p-value to determine if this correlation is statistically significantly different from zero. Is there a statistically significant linear correlation between height and weight? We don't have to be statisticians to know that there is but we will discuss the details later in the course.

Finally, the cells which only have a 1 in them represent the correlation between each variable with itself. This indicates, as is clearly the case, that HEIGHT is 100% correlated with itself and the same for WEIGHT. Notice this value does not come with a p-value underneath it and is not of interest to us but most software packages do something similar for pairwise correlations between variables.

You might try running this on many quantitative variables to see what happens. We will show that result later!

That's all for this tutorial on calculating Pearson's correlation coefficient.

Topic 9D – Simple Linear Regression - EDA

Welcome – this video covers calculating the simple linear regression equation for Case QQ where we have two quantitative variables which are linearly related.

In order to follow along you will need the SAS dataset PULSE_STEP2 (after labeling all variables and translating the original coded categorical variables).

Since we will be working with a dataset with permanently assigned formats, we will need to begin our SAS program with the PROC FORMAT code which defines those formats.

We should verify that the relationship is reasonably linear using a scatterplot before conducting this analysis.

We looked at height and weight earlier and found that, although it is best to analyze within each gender, the combined data are only slightly non-linear when considered together. This will calculate the values needed for the simple linear regression equation for this data.

We begin with PROC REG using our STEP2 data. Then we add a MODEL statement of the form MODEL Y = X. So here we are using MODEL Y = WEIGHT and X = HEIGHT. We end with a RUN statement and a QUIT statement. The QUIT statement is not entirely necessary but PROC REG is a procedure that technically does not stop running until you submit a QUIT command.

When we return for the inferential statistics component, we will add a few additional options but for the most part we will already be seeing some of the results we will need later.

By default we obtain 4 tables. The only one we are concerned with at this point is the very last called PARAMETER ESTIMATES. We need to find the slope and intercept of the linear regression equation. In SAS these are in the PARAMETER ESTIMATE column.

The row labeled INTERCEPT is clearly the INTERCEPT, here it is negative 121.173.

The row labeled with the x-variable – in this case Height – is the SLOPE, here it is 1.085.

So the regression equation would be Y-hat = -121.173 + 1.085(HEIGHT).

The slope implies that on average, for each 1 cm increase in height the mean weight increases by 1.085 kg.

Later we will discuss the ANOVA table and the interpretation of R-square in the Model summary table as well as other output and at that time we will discuss what might be problematic for this data. But for now that's all we need for simple linear regression in the exploratory data analysis section.

Topic 9E – Simple Linear Regression - Inference

Welcome – this video covers calculating the simple linear regression equation for Case QQ where we have two quantitative variables which are linearly related as well as looks at the inferential components.

In order to follow along you will need the SAS dataset PULSE_STEP4. First we submit the needed formats.

We should verify that the relationship is reasonably linear using a scatterplot before conducting this analysis. We looked at height and weight earlier and found that, although it is best to analyze within each gender, the data are only slightly non-linear when considered together.

Now we will calculate the values needed for the simple linear regression analysis for this data.

The new code is minimal. We still use PROC REG. Here we are using the STEP4 data. We do add one option in the PROC REG statement of PLOTS=DIAGNOSTICS(UNPACK). This will pull the graphs from the 3 by 3 diagnostic panel out and provide them full size. This is often preferred when you need to look closely at the assumptions.

Then we have our MODEL statement as before with our Y variable of WEIGHT on the left of our equals sign and our X variable of HEIGHT on the right of the equals sign. We add the forward slash for options and the CLB option which calculates confidence intervals for the parameter estimates of the "Beta" values in the model. In this model, this will provide confidence intervals for the intercept (Beta_0) and slope (Beta_1).

Submitting this code, we get results similar to those you have seen before.

First we get a summary of the number of observations read and used. Any multi-variable analysis will usually remove all observations where any of the variables have any missing observations so these two numbers can be different in practice, with less clean data.

Then we get the analysis of variance or ANOVA table. Regression and analysis of variance are mathematically the same process but the one-way ANOVA is a special case as is simple linear regression. This provides a breakdown of the sums of squares used in the regression model calculations and an overall test of whether or not the model explains a significant amount of variation. Here the F-value of this test is 128.15 and the p-value is < 0.0001.

Then we get a table with some numeric summaries related to this regression model. Of interest to us is the R-square value of 0.5496. This can be interpreted by saying that this model using height explains 55% of the variation in weight.

The last table provides the parameter estimates which we have seen before. Here we have added the confidence intervals for the parameter estimates. If the intercept is meaningful then this may be of interest but in this case, the intercept represents the mean weight for someone with a height of zero which is not possible in practice and therefore not of interest.

The confidence interval for the slope is definitely of interest as this provides bounds for the increase or decrease in the mean response for each 1-unit increase of the explanatory variable.

The INTERCEPT is negative 121.173.

The row labeled with the x-variable label – in this case Height (cm) – is the SLOPE, here it is 1.08458.

So, rounding these values to 3 decimal places, the regression equation would be Y-hat = -121.173 + 1.085(HEIGHT).

The slope implies that on average, for each 1 cm increase in height the mean weight increases by 1.085 kg and the 95% confidence interval suggests this increase could be as low as 0.895 to as high as 1.275.

For the graphs, we get a histogram of the residuals with a normal curve overlay. We want the residuals to be normally distributed which by this graph they seem to be.

Then we get the scatterplot of the residuals on the y-axis vs. the predicted values on the x-axis. We are looking for a random scatter of relatively equal spread around the center regardless of the location on the x-axis. If we see any pattern it can indicate non-linearity or non-constant variance depending on the type of pattern. Both of these issues are violations of the assumptions of this method. The more harsh the violation, the more concern we would have.

In this case there is some evidence of non-constant variance in that for large values on the x-axis there seems to be more variation. Possibly we have two clusters illustrating the different genders. We could investigate further by analyzing the two genders separately.

The next graph is similar except the residuals have been standardized into t-statistics. Values outside the lines indicate unusually large or small values.

Another graph we are interested in for this course is the QQPLOT a few plots down which shows the data are reasonably normal with some issues on the tails. Those of you taking an entire course in regression will learn more about some of the other plots in such a course.

The fit-plot at the end is a nice representation of the data and the model.

That’s all for this tutorial on simple linear regression in Case QQ.