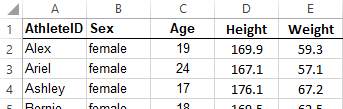
**Subject Characteristics with SAS Studio**

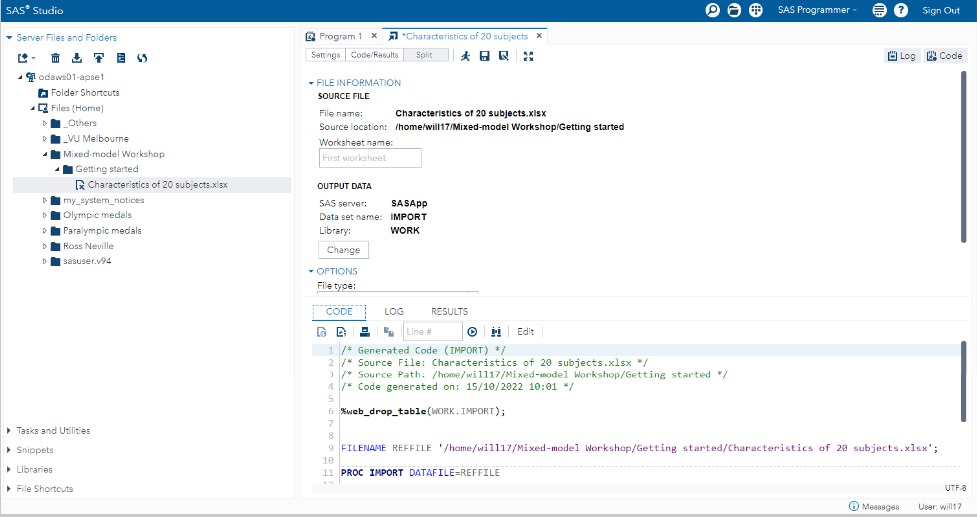
This document explains how to get some simple statistics for subject characteristics, how to get an effect statistic (a correlation coefficient), and how to do some scatterplots. If you are reading this, you must have already downloaded the compressed file **Mixed-model Workshop.zip**, you have obtained a SAS profile, you have worked through **How to access SAS Studio.docx** to create a folder **Getting started** within a folder **Mixed-model Workshop** in SAS Studio, and you have uploaded **Characteristics of 20 subjects.xlsx**.

1. Find the spreadsheet **Characteristics of 20 subjects.xlsx** on your computer and open it with Excel. Here are the first few rows of the table:



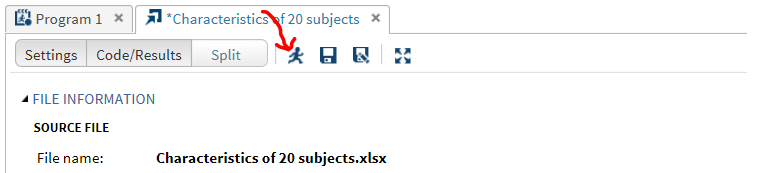
The units for Age, Height and Weight are obviously years, cm and kg. Click on the **Generate data** tab at the bottom, to see how I generated the data as samples from normally distributed variables with chosen means and standard deviations. Click in the cells to see how it's done. **Age** and **Height** are simple enough, but **Weight** is tricky: you have to assume a correlation between Height and Weight (here 0.70 for females and 0.70 males). Change the population values to see what happens. The data shown in Sheet1 are just one of the infinite number of samples.

1. Now open SAS Studio via this link <https://welcome.oda.sas.com/login>, if it's not already open. Find and double-click Characteristics of 20 subjects.xlsx in the navigation frame. You get this:

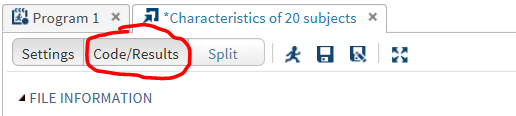


SAS has generated a special data tab Characteristics of 20 subjects consisting of a Split window showing Settings (upper window) and Code/Results (lower window). The code hasn't been executed yet, because SAS is waiting to see which spreadsheet you want. It is defaulting to First worksheet (underneath Worksheet name:), which is the data we want to import.

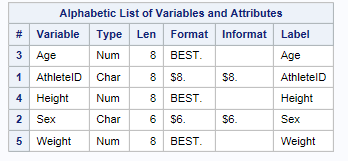
1. Click the Run icon:

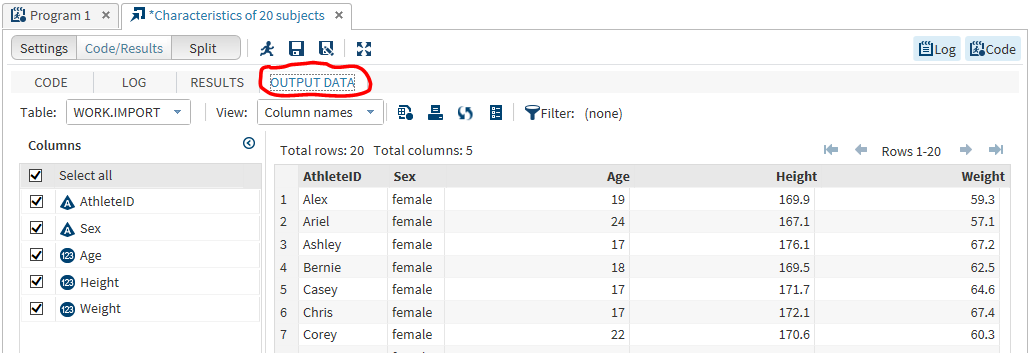


1. To see better what's happened, click on the Code/Results window, which will close the Settings and fill the right-hand window with the choice of three windows you need to know about…

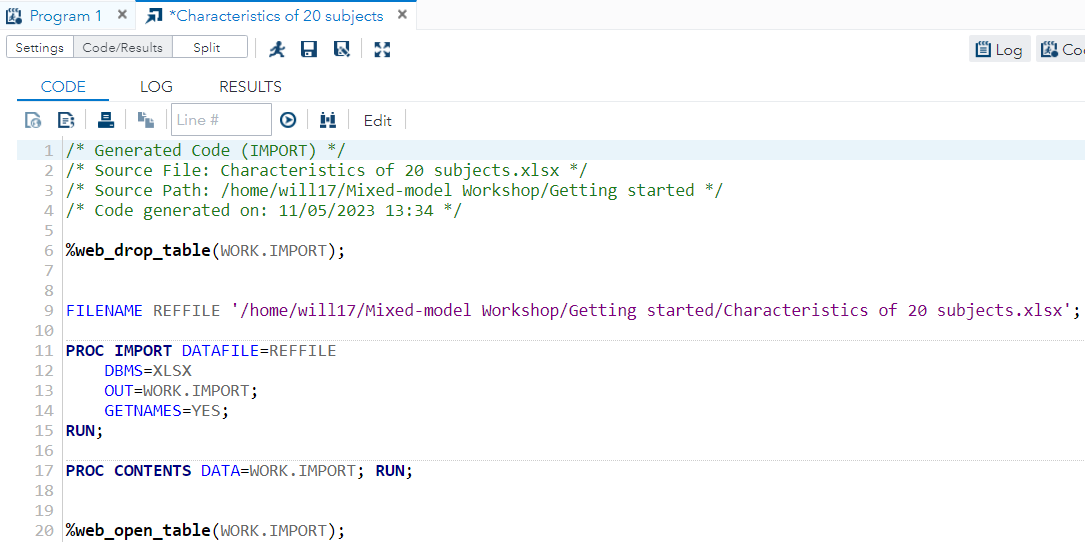


…starting with the RESULTS window. This is the output produced by the CONTENTS procedure (Proc Contents). Scroll through it to see all this boring information. The only useful output is this:

  
It's a good check on your data, but to see the data themselves click on the OUTPUT DATA tab:



Now click on the CODE tab and you will see the code for the procedure that imported the data (Proc Import) and showed the contents of the data (Proc Contents):



The code at the top in pale green and demarcated by /\* and \*/ is "comment" text that doesn't get executed when you run the code. SAS ignores anything between /\* and \*/.

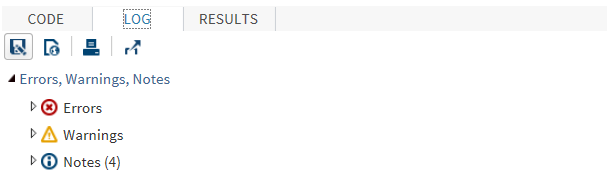
The %web\_drop\_table (WORK.IMPORT) and %web\_open… are called *macros*, which are blocks of code that SAS has already written and which are hidden. You can write a program as a macro, when you have lots of variables to analyze in the same way. Forget about macros for now. By the way, SAS uses *table* to refer to a dataset.

The **FILENAME** statement is the way SAS generated a name for the file to use in **PROC IMPORT**. The name is REFFILE, but it can be anything starting with a letter.

**PROC CONTENTS** includes a reference to the dataset, which SAS has called WORK.IMPORT. You can refer to anything with WORK in front of the period just as the "anything" itself. So DATA=IMPORT would work here.

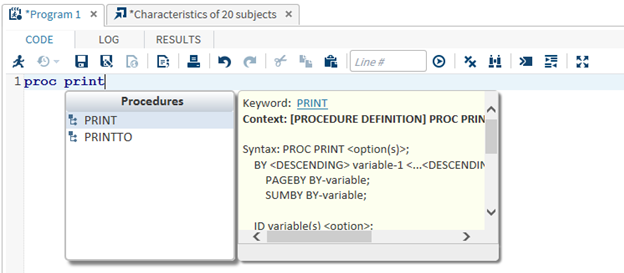
**RUN;** occurs often. All it means is execute the preceding code. The only time you really need it is at the very end of a program, because otherwise the last step in the program won't run.

1. Now click on the LOG tab. You will check this window often to make sure the code has run properly. The Errors, Warnings and Notes at the top are filters to allow you to view just those things:

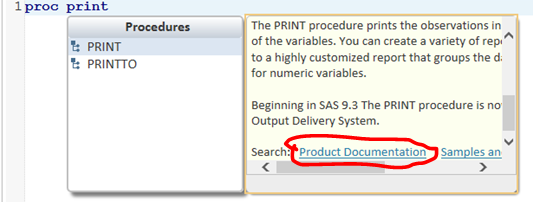


If there are no errors (no number in brackets), the program has run correctly, but scroll through the window anyway. You can see that it reproduces the code and includes any errors, warnings and notes that SAS deems appropriate.

1. Now let's write and run the simplest of all possible programs in SAS. Click on the Program 1 tab and type **proc print**. You will notice a prompting window opens and changes as you type the words. Here's what it looks like by the time you get to the end of **print**:

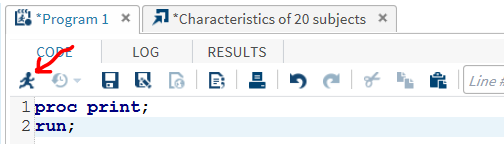


You can click on the PRINT link or scroll down and click on the Product Documentation link to get help on the procedure:

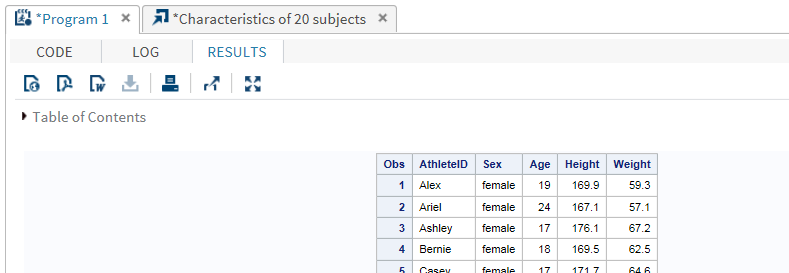


These links open up pages with lots more links, and it's not obvious which link is the best. That's why I have provided you with **Useful help links for SAS.htm**.

1. Now put a semi-colon after the **proc print**, hit the Enter key, type **run;** (do not forget the semi-colon!), then click the Run button:

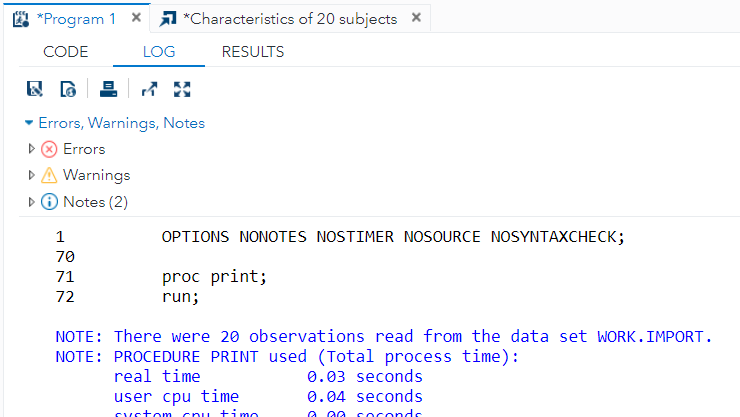


1. SAS runs the program and opens the RESULTS window, thus:



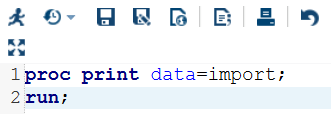
Scroll down to see all the observations.

1. Now click on the LOG tab to see what happened:



Notice that SAS automatically selected the IMPORT dataset. You didn't have to specify it in the Proc Print, because IMPORT was the last dataset created. But let's specify it anyway, for practice.

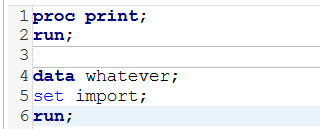
1. Click on the CODE tab and add data=import, then run it:



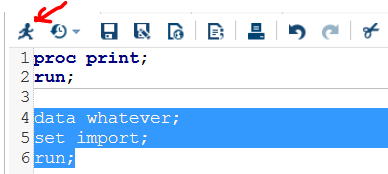
You get exactly the same RESULTS and LOG. Notice that SAS deleted the previous results and the previous log: they are not duplicated. You can't change this feature, which is probably a good thing, because it keeps everything tidy.

1. Let's make another incredibly simple program, this time involving a "data step". SAS consists almost entirely of *data* steps and *proc* steps. You create and modify datasets in data steps; you analyze datasets to get results or output with procs, although most procs can also produce data sets.

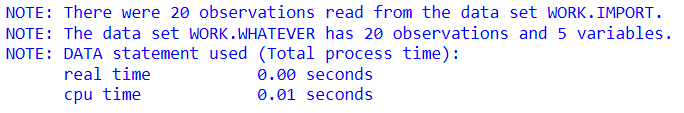
Click on the CODE tab, click the cursor at the end of Line 2, enter a blank line and then type these extra three lines:



1. To submit only this new code, highlight it and click the **Run** icon:

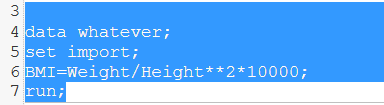


Now the LOG window opens, because there are no results and the RESULTS window has been cleared:

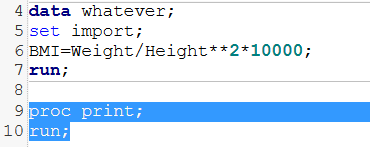


You have succeeded in creating another dataset, whatever, which is just a copy of import.

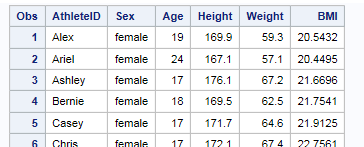
1. Finally, let's do another data step with a bit more sophistication. Let's create a variable BMI (body mass index), which is given by Weight/Height2. In SAS you write BMI=Weight/Height\*\*2\*10000;. You need the \*10000, because our variable Height is in centimeters, but for BMI it needs to be in meters. Copy or type this equation as an extra line in the data step, before the run, then run it.



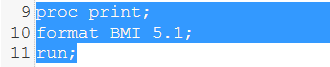
Oops, no output! We obviously need a proc print; run; to see if the equation is right:



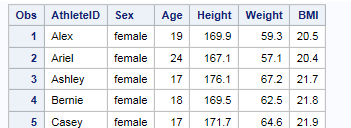
Run it, and you get this:



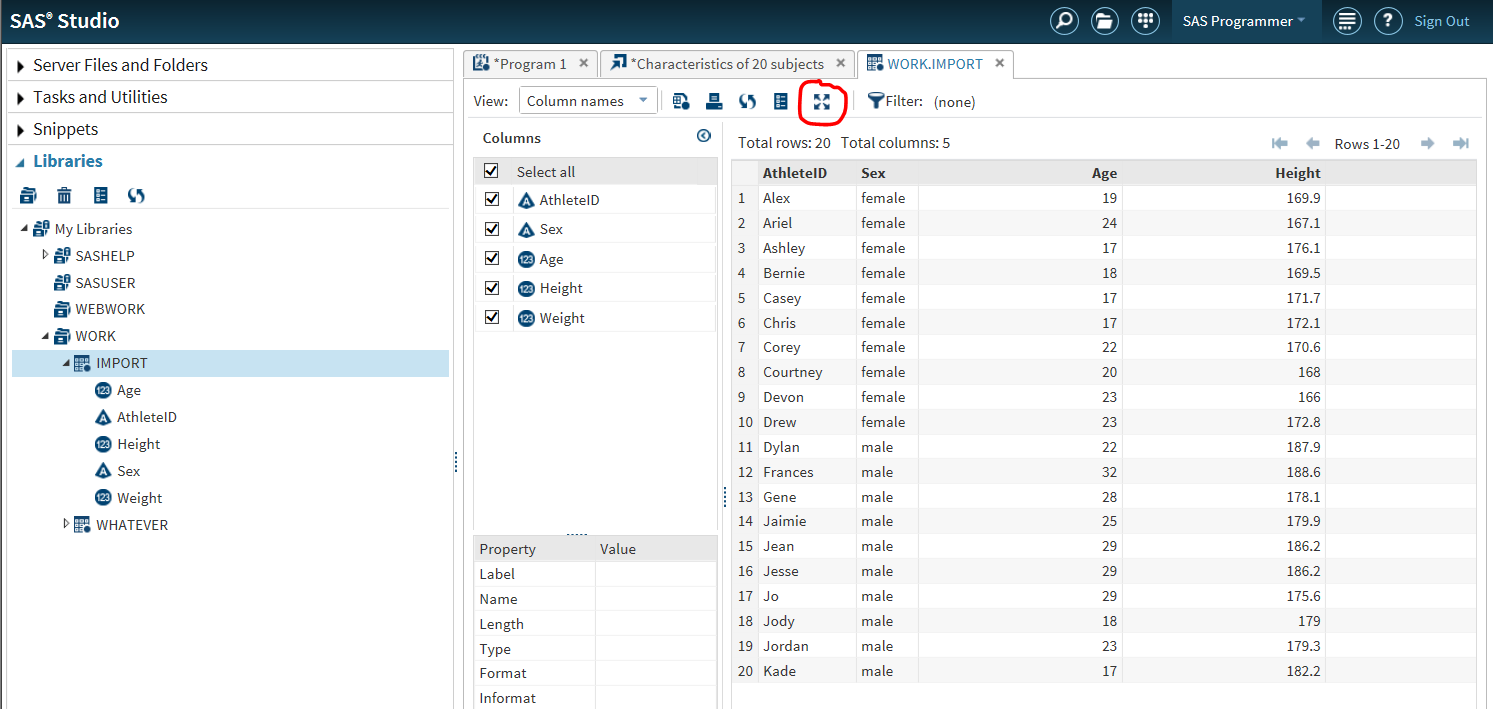
The numbers look right, but I'm not happy about all those decimal places for BMI. We control that with a format statement, here format BMI 5.1;. You can add it to the data step or the proc step. Let's put it in the proc step and run it:



You get this:

Cool!

1. There are zillions of other things you can do in data steps, but that will do for now. Before we move on to data analysis, there's one other helpful feature of SAS studio and one more kind of statement you need to know about…
2. In the navigation frame on the left, click to open Libraries/My Libraries/Work. You will see the two datasets you have created. Single-click on one of them, and you can see the variables. Double-click and you see another kind of data window (e.g., for IMPORT):

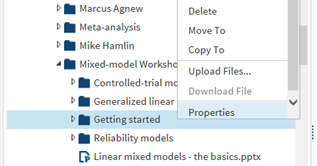
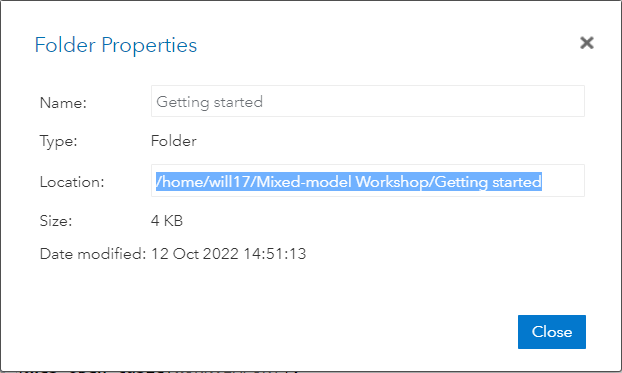


Click on the maximize icon (shown circled) to get a better view of the data. Click on it again to return to the previous view.

By the way, SAS uses the word *library* to refer to datasets or data files. A printing or listing of the library is called a *table*. Confusing!

All the datasets in the WORK folder disappear when you **Sign Out**. To make a permanent dataset, you have to use a LIBNAME statement and an associated data step. Here's the LIBNAME I am going to use:  
libname ss '/home/will17/Mixed-model Workshop/Getting started/';

The LIBNAME points to where I want to store a permanent copy of a dataset. I got that by copying part of the FILENAME statement in LOG window of the Characteristics of 20 subjects tab. Another way to get it is to right-click on the folder where you want to store the dataset, select Properties, triple-click in the Location window to select it, and copy to the clipboard:

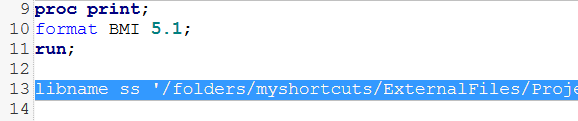
 

This is what's on the clipboard:  
/home/will17/Mixed-model Workshop/Getting started

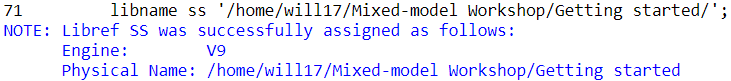
So I have to add libname ss, put quotes around the directory information, and add a semi-colon at the end. Add it to Program 1 as a new line.

**BEWARE:** SAS does not recognize "smart" quotes. Never use “ ” and ‘ ’; always use " " and ' '. Untick the option for smart quotes in Word via File/Options/Proofing/AutoCorrect options/Autoformat As You Type.

1. Make sure you have the quote marks and semi-colon in the right place before you highlight it and Run it:

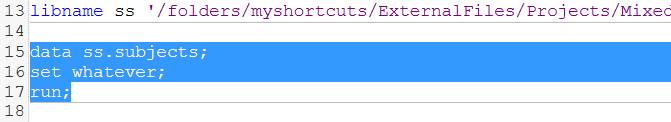


You should see something like this in the LOG:

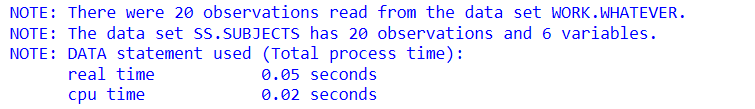


If it hasn't worked, correct the error and try again. If you missed out a quote mark, you get all sorts of weird statements in the LOG. Get it right!

1. Now let's make a permanent copy of the dataset. Add these lines of code and run them:

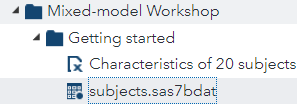


The LOG tells you it worked:



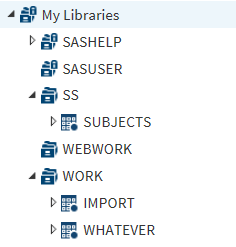
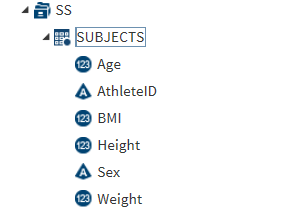
The ss can be anything at all starting with a letter. I always choose ss, because in my mind it stands for sas (data) set. SAS uses the ss to send a permanent data set called subjects to the directory shown.

Check out the navigation frame. The dataset is stored where I said to put it:

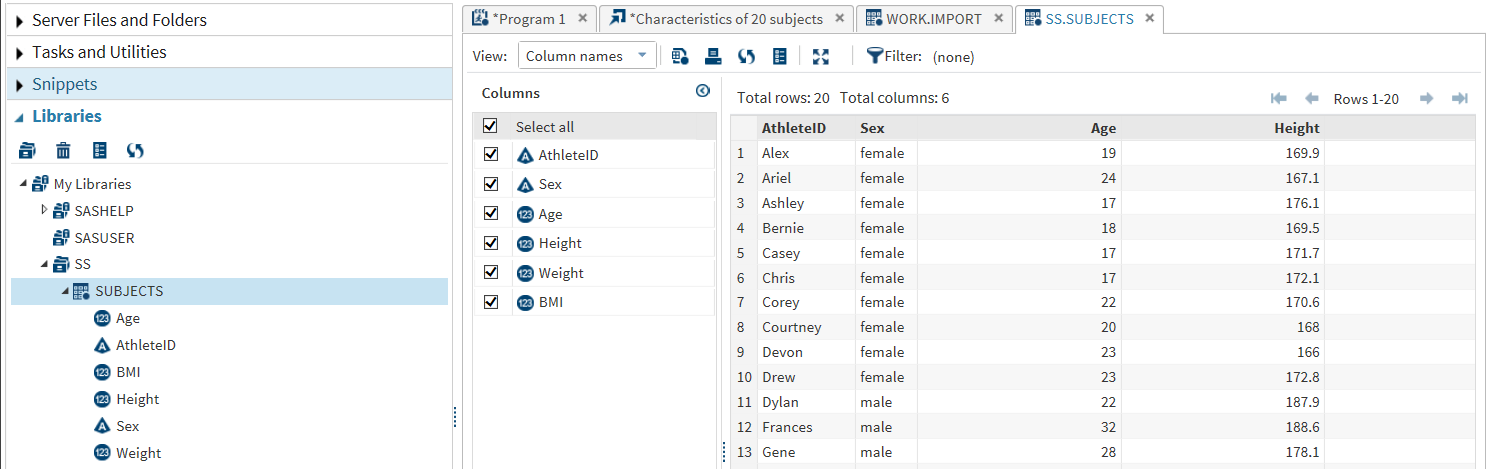


Notice it has a weird file extension, sas7bdat. This stands for *SAS Version 7 data*. Version 7 was a beta version of SAS in operation when SAS last changed the format of its datasets more than 10 years ago. Weird! You can double-click this one to view the data, but a single click doesn't show the variables.

Open the **Libraries** and you will a new entry SS in My Libraries, with SUBJECTS inside it. You can open it to see the variables…

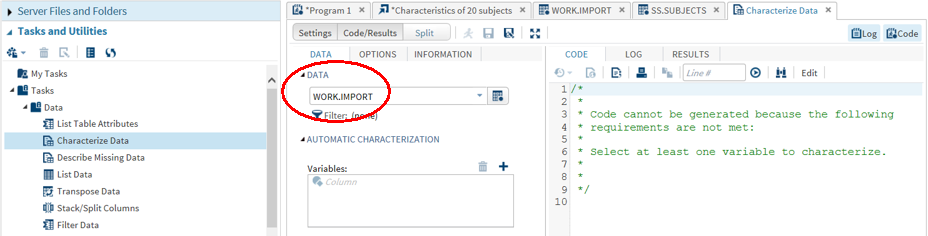
…and if you double-click it, you get a data window showing the dataset:



The dataset SUBJECTS is now available as ss.subjects in data and proc steps for the life of this SAS session. Once you sign off, you will need a libname statement in a new SAS session to access the dataset in the **Getting started** folder. You can use something other than ss in the libname statement, for example getstart, but then you will have to refer to the dataset in your program as getstart.subjects.

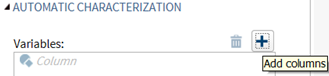
That's enough of an introduction to SAS. Now let's start doing analyses.

1. If you had taken a break at this point (as I had), SAS might have timed out, and you will have lost Program 1 and the tabs for WORK.IMPORT and SS.SUBJECTS. No problem, just re-open SAS Studio and run **Characteristics of 20 subjects.xlsx** (open it from the **Getting started** folder, if it's not open) to create the IMPORT dataset.
2. Let's generate some code using "point and click". In the **Navigation** frame, open Tasks and Utilities/Data, then double-click Characterize Data:

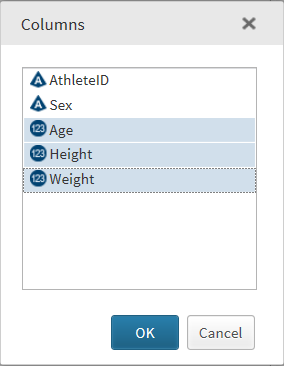


If WORK.IMPORT is not already in the DATA window (circled), find it by clicking on the icon at the right of that window, navigate to My Libraries/WORK, select IMPORT and OK.

1. Under AUTOMATIC CHARACTERIZATION, click the **+**...



…and select (Shift-click or Ctrl-click) the numeric variables Age, Height and Weight…

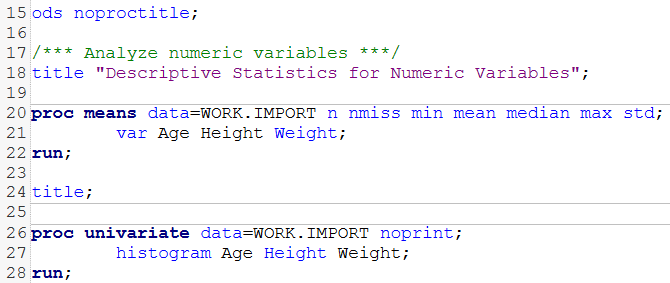


By the way, SAS uses the terms *columns* and *variables* interchangeably. Now click OK.

1. You'll now see these variables in the Variables box…



and you'll see this code in the CODE window (below a whole lot of comment text). You might have to resize some windows (by dragging the middle of the window margin) to see right to the end of the line of the **proc means** code:



ods stands for output delivery system, and noproctitle suppresses an unnecessary title in the output.

**proc means** invokes the means procedure with the current dataset. The n nmiss min etc. are keywords indicating all the simple statistics that will be produced; n stands for number of observations, and nmiss stands for number of missing values.

var Age Height Weight indicates that these variables will be analyzed.

**proc univariate** invokes the univariate procedure, which is like proc means, except that it can provide a lot more statistics. Here those statistics are suppressed by the noprint option.

histogram indicates that a frequency histogram will be produced for the variables.

The apparently inconsistent colors of variables is due to the fact that some variable names are also keyword options (those in blue). Don't worry about it.

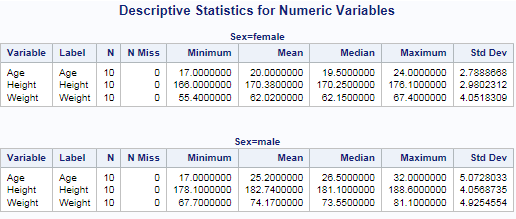
1. But wait! Before we run the program, we should get separate characteristics for the females and males. So click on ROLES, click the **+**, and select Sex:

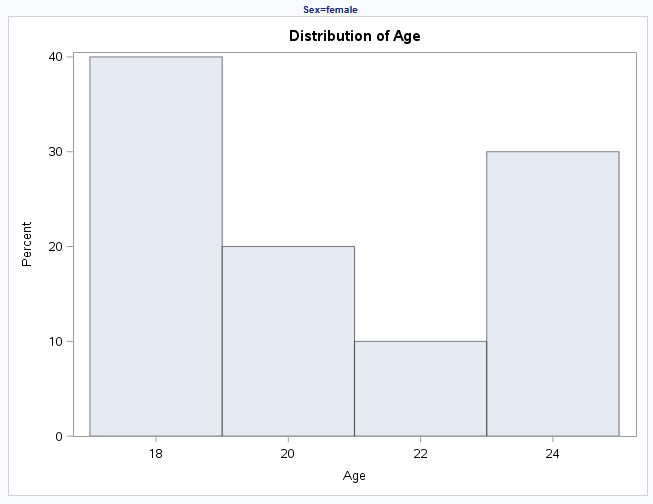
 

1. Now click the Run icon:

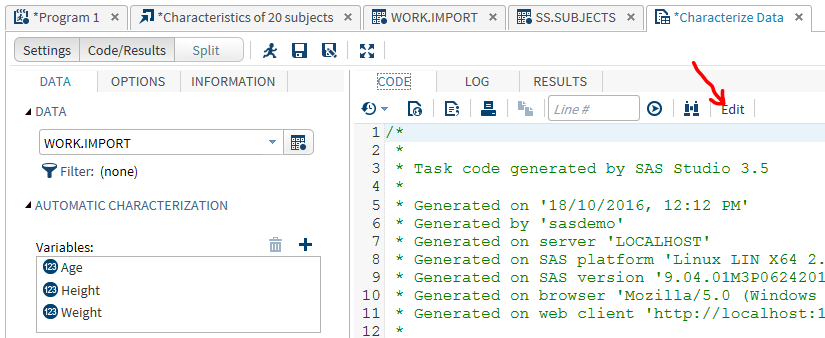


1. Maximize the RESULTS window, or drag the margins, or change the magnification so you get a full view of the output, all of which should be reasonably obvious:

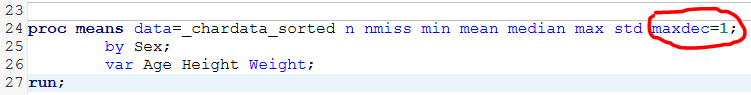


 etc., etc.

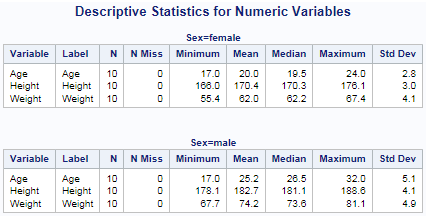
1. I hate all those decimal places! To change that, you have to modify the code. Click on the CODE tab, then click on Edit (see arrow):



A new program window will open, with all the same code. Add maxdec=1 at the end of the **proc** **means** statement (circled)…



…then Run the whole program and get this:



Much better!

If you have a close look at the program, you will notice that SAS used the **proc** **sort** procedure to make sure the dataset was sorted alphabetically by sex. SAS got the sort procedure to output a new dataset **\_chardata\_sorted** (the name is informative but irrelevant) before running the proc means and the proc univariate with the new dataset "by sex". Finally SAS used proc delete to delete the new dataset. It's not clear why they did all that, because both procedures allow you to get separate output for the females and males using the CLASS statement, without any sorting. Let's do that now with proc means. It's a simple way to check for obvious outliers in a new dataset. Copy and paste this bit of code into the bottom of Program 1, and click the Run icon:

proc means maxdec=1 data=import;

class sex;

run;

Here's some similarly simple code for proc univariate:

proc univariate plot data=import;

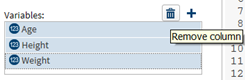
class sex;

run;

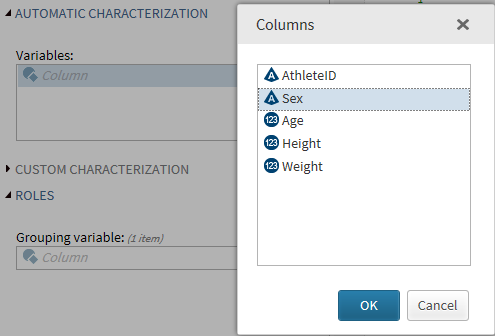
Proc univariate now provides output for all the numeric variables in the dataset. You could limit the output to one or more variables with the var statement, of course.

The univariate statistics include the highest and lowest values and a frequency histogram on its side. I don't take much notice of the box plot or the probability plot (which addresses the question of how non-normal the data are).

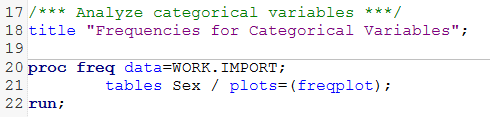
1. You should also check out the categorical (nominal, classification or non-numeric) variables in any new dataset. Click on the Characterize Data tab, select the three variables and click the Trash icon, and do the same for **Sex** in the Grouping variable window:

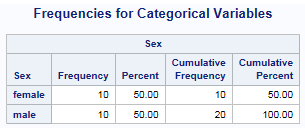
Now go back to the variables window, click the **+**, select Sex, OK:



You should see this in the CODE window. Choosing nominal variables has invoked **proc freq**:



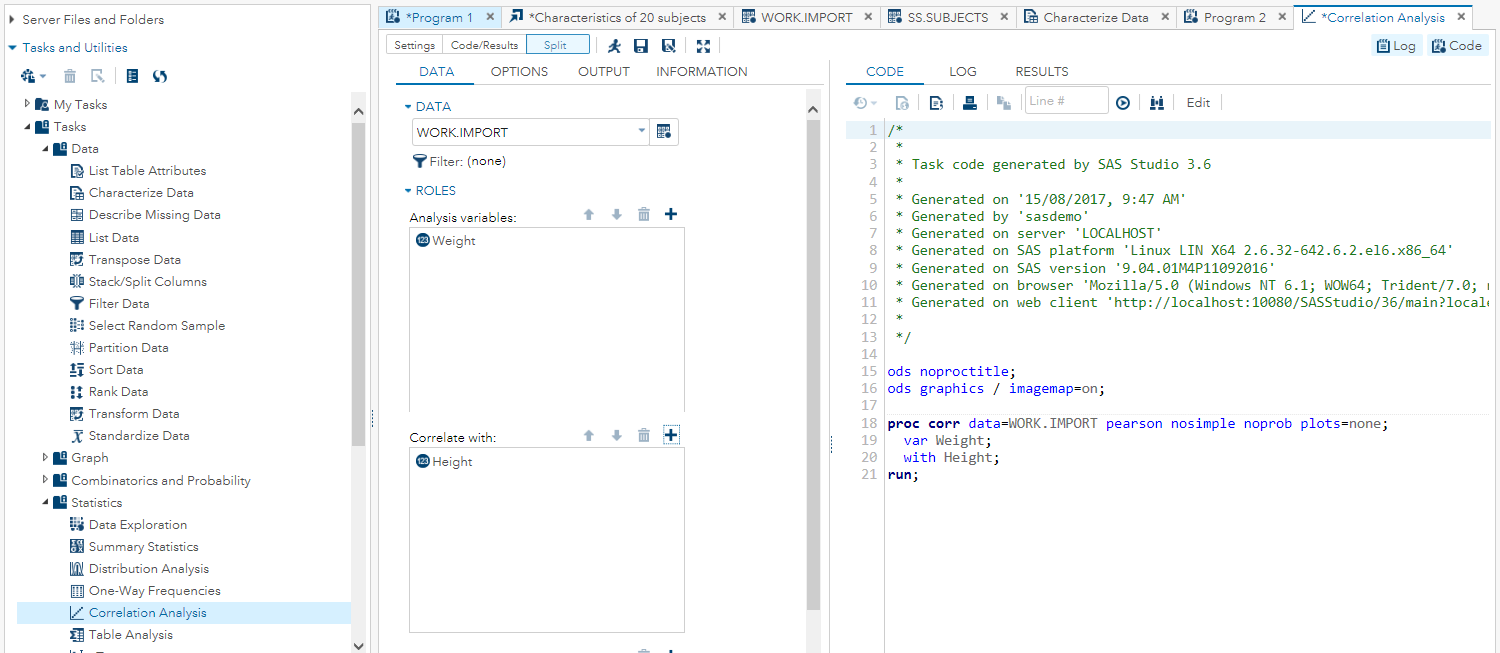
1. Now click the Run icon and inspect the output for this table and an equally boring histogram:



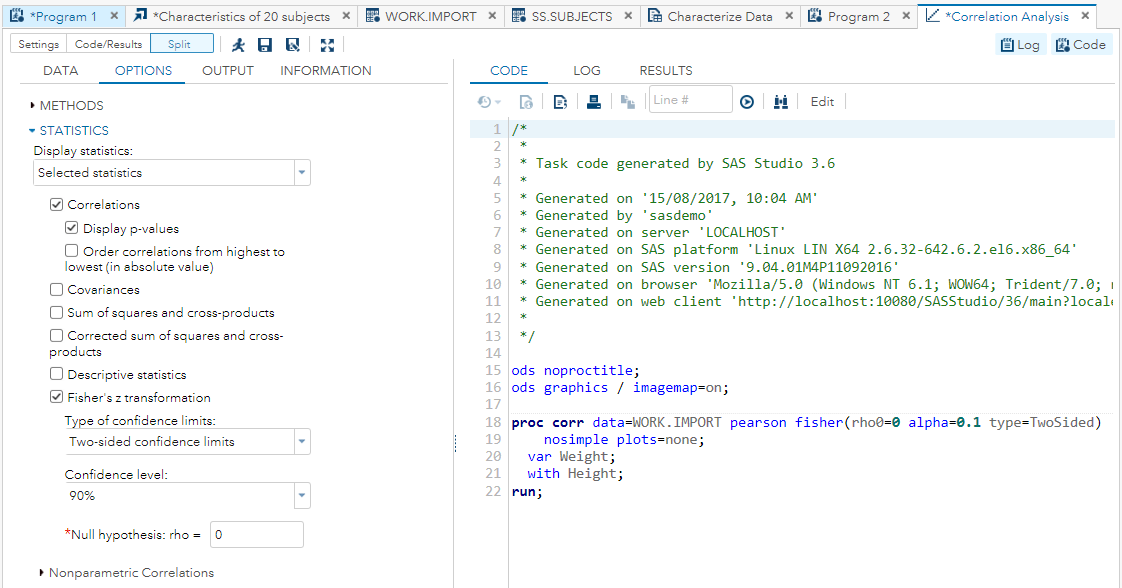
1. If you run this simple code…  
   proc freq data=import;run;  
   …you will get frequencies of every value of the numeric variables, as well as all the nominal variables Try it. Then type or copy this into the bottom of Program 1 and Run it:  
   proc freq data=import;  
   tables \_character\_;  
   run;

The variable \_character\_ refers to all nominal variables. The output is very boring, but when you have a complex dataset, proc freq is a good way to check for misspellings and to remind you of the structure of the data. There is also \_numeric\_, but I seldom use it.

1. Now let's generate some effect statistics and inferential statistics: correlations, and their confidence limits and p values. Scroll down the Tasks and Utilities/Tasks window to Statistics, click to open it, double-click Correlation Analyses, select IMPORT as the data, **Weight** as the Analysis variable, and Correlate with: **Height**:

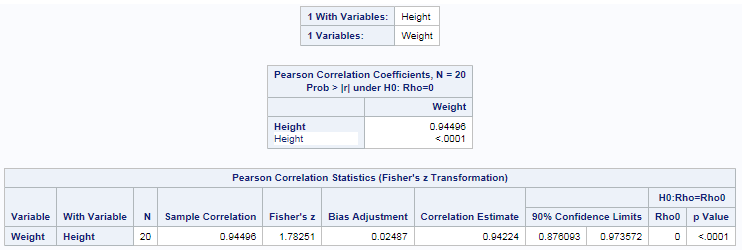


1. Notice that we have invoked proc corr. Let's change the options by clicking on the OPTIONS tab, opening STATISTICS and selecting the following:



Now click Run icon: 

…and you will get these RESULTS:



The first panel shows the correlation. It has too many decimal places. You should show correlations with only two: 0.94. Underneath it is the p value, which SAS has called **Prob> |r| under H0: Rho=0**. This is the way SAS presents a so-called test of the null hypothesis. **Rho** is the Greek letter ρ, which is used to represent the population correlation. The null hypothesis is the assumption that ρ=0. So, if ρ=0, the p value is a measure of the probability that you would observe anything greater than the sample value (>0.94 and <-0.94). If p is <0.05, the effect is significant at the 5% level, so you can conclude that ρ>0. The p value is so small that SAS shows <0.0001; otherwise it would be an exact value.

Here are some explanations of the second panel...

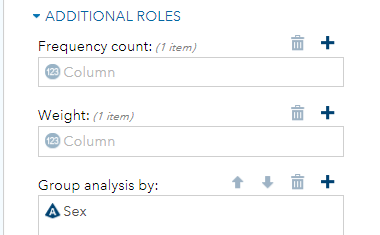
**Fisher's z** is the Fisher transformation of the correlation coefficient: z = 0.5ln[(1+r)/(1-r)]. You don't have to know this formula, and you don't need the Fisher z. SAS derives the confidence limits for the Fisher z via its sampling standard error of 1/√(N-3) and the assumption of normality, then back-transforms them to the usual correlation coefficient.

**Bias** **Adjustment** and **Correlation** **Estimate** give the misleading impression that the sample correlation is biased high and needs to be adjusted down a bit. In fact, the sample correlation is biased *low*, and needs to be adjusted *up* a bit, by a factor of 1+(1-r2)/(2(N-3)), a formula devised by Olkin and Pratt in 1958. In other words, when you calculate a Pearson correlation coefficient, the answer you get tends to underestimate the true correlation. Apparently what SAS is showing here is an adjustment for the Fisher z that gives the confidence interval the correct coverage. Versions of SAS later than 9.4 may clarify this issue. It doesn't really matter here, because the adjusted value is practically identical to the sample value. The adjustment is a bit larger for smaller sample sizes and for correlations closer to 0.50.

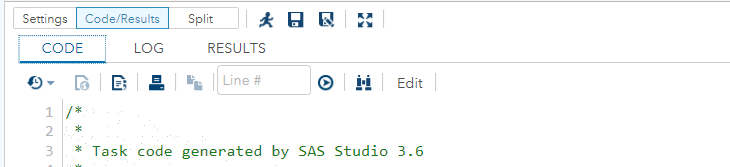
**90% Confidence Limits** represent the range of values of a statistic (here the correlation coefficient) within which the true (population) value is 90% likely to fall. So, the true correlation is somewhere between 0.88 and 0.97. The true value is the value you would get if you had an extremely large sample size; with such a sample size, you would find that the lower and upper confidence limits are practically identical to the sample correlation.

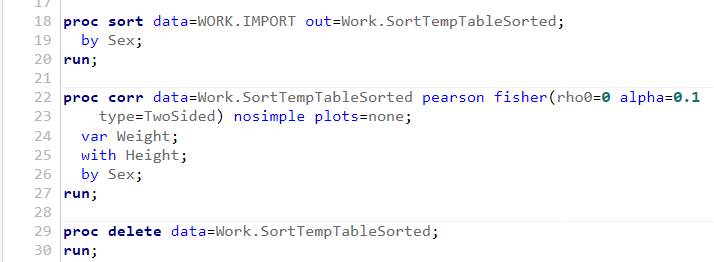
1. When you have distinct subgroups of subjects, as here with females and males, it is wrong to calculate one correlation for all the subjects. So let's calculate the correlations for the groups separately. Click on the Settings tab, then scroll down to ADDITIONAL ROLES, click on the **+** and add **Sex**…





…then click on Code/Results and CODE to view the program:



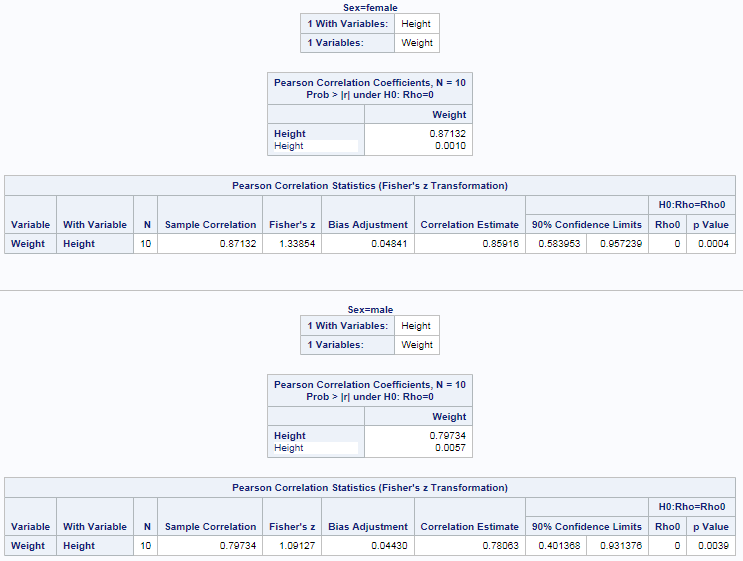


Once again, SAS is creating a new sorted dataset, using it with proc corr, then deleting the sorted dataset. See below for something simpler.

Now Run the program…



…and view the RESULTS:



If you look carefully at the confidence limits, you will see that they span the population values (0.70 for the females and 0.70 for the males) that were used to generate the sample in the spreadsheet that was used to generate the data. Or to put it another way, the confidence interval includes the true value. Well, that's nice, but it doesn't always happen. How often would you expect the 90% confidence interval NOT to include the population value? (This question is so easy that I refuse to give the answer!)

Notice that the correlations for each sex are lower than for the combined data. To understand why this happens, we need to see a scatterplot of the heights and weights. So let's do some graphs. First, though, here is the simpler code for proc corr. Type or copy and paste it into Program 1 and Run it:

proc corr fisher(alpha=0.1) data=import nosimple;

by Sex;

run;

We don't need a proc sort; by Sex; before the proc corr, because the dataset import is already sorted by Sex. This code gives a correlation matrix of all the numeric variables. If you omit nosimple, you will also get the basic simple statistics for the variables. If you want to limit the correlations to chosen variables, use these statements before the run;…

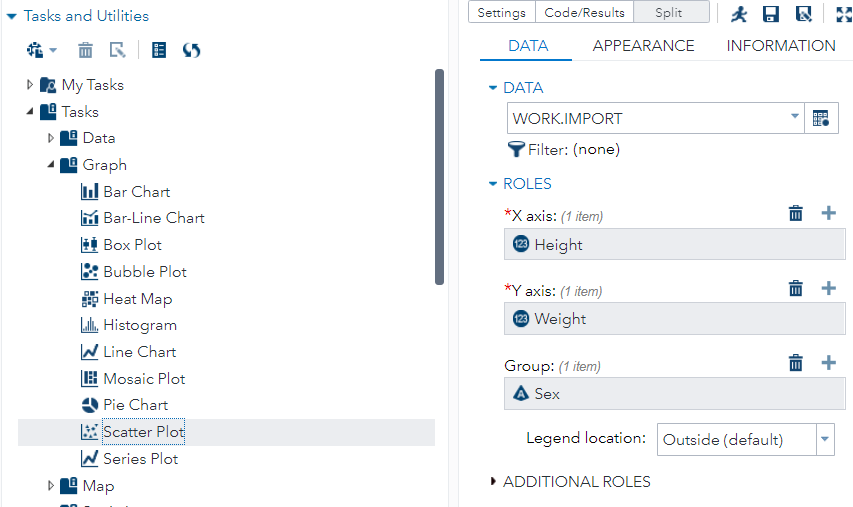
var Height Weight;

…or use these statements to further limit the output:

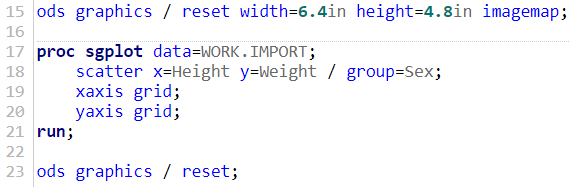
var Height;

with Weight;

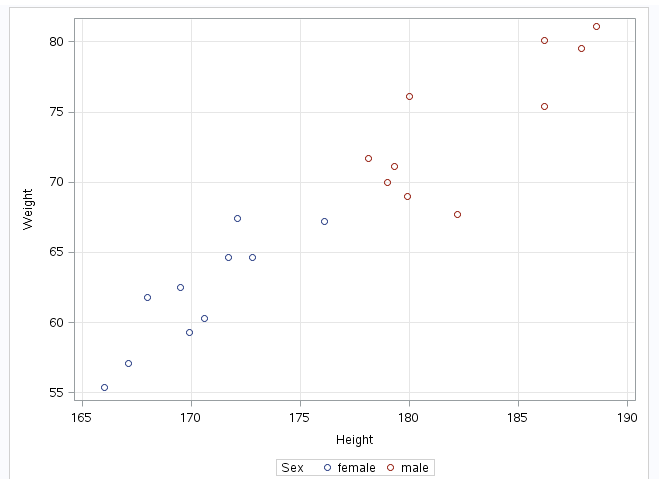
1. Now the graphs… In Tasks and Utilities, find Graph and double-click Scatter Plot. Then choose IMPORT for the data, **Height** for the X variable, **Weight** for the Y variable, and **Sex** for the Group variable:



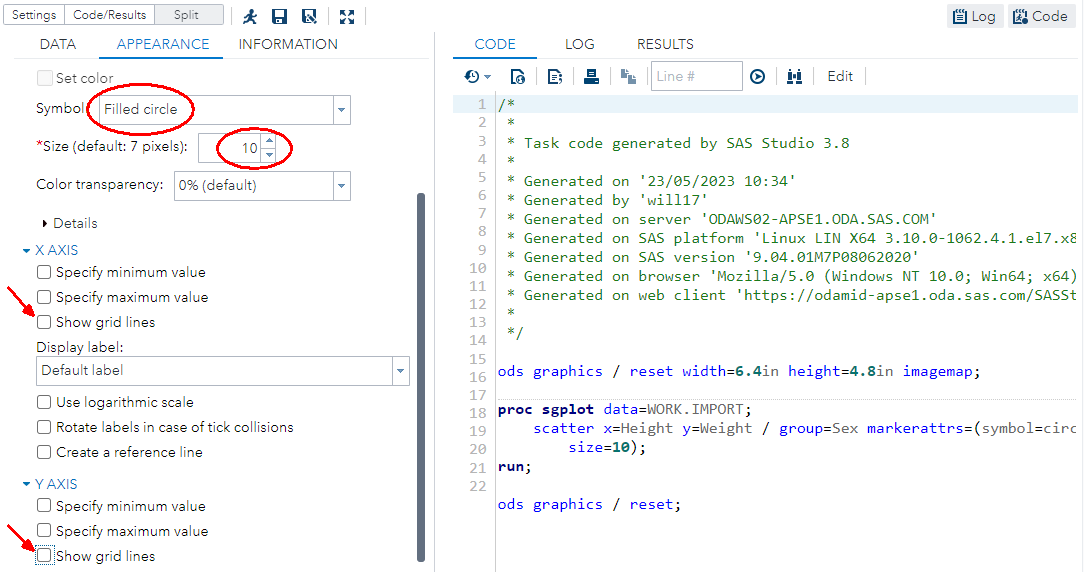
Here's the resulting code:



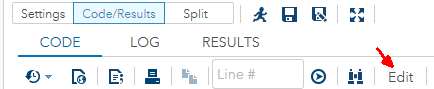
You can change the units for width and height to cm in APPEARANCE/GRAPH SIZE, but who cares? Run it and get this graph:



1. SAS is a bit naughty with its defaults for symbols: they should be filled and the colors should be more clearly different. Unfortunately, SAS is not friendly with customization of graphics. Click on the OPTIONS tab and you can fill the symbols, make them bigger, and remove the grid lines:

  
But to make filled symbols outlined with black, and to change the colors, you have to modify the code, as follows…

Click on the Code/Results tab and click on Edit:



In the new Program 3 window, add the highlighted text (or replace the program by copying and pasting this whole block of code (the additions are highlighted):

ods graphics / reset width=6.4in height=4.8in imagemap attrpriority=none;

proc sgplot data=WORK.IMPORT;

styleattrs

datacolors=(red blue)

datacontrastcolors=(black);

scatter x=Height y=Weight / group=Sex markerattrs=(symbol=circlefilled

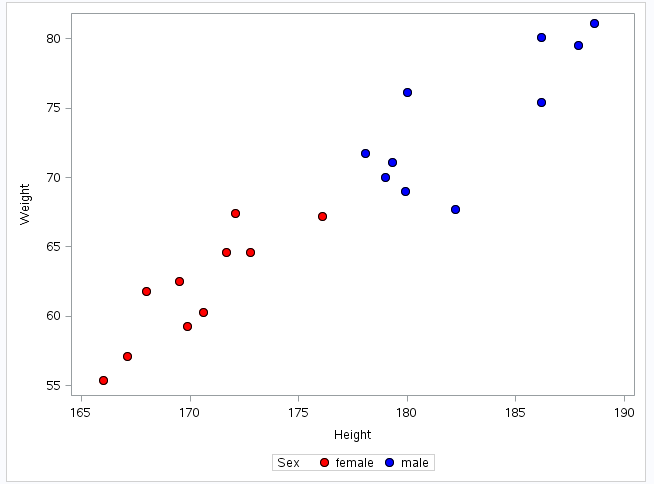
size=10) filledoutlinedmarkers;

run;

ods graphics / reset;

run;

Now Run this edited program and you should get this:



Much prettier, but it took me about 30 minutes of searching around the SAS site and trial-and-error coding to get this graph. What we need is an *interactive* graphing interface, where we can click and drag some items, and right-click on others to select options, just as we do in Excel. Frankly, Excel is a more practical graphing package than SAS Studio.

You can see a range of colors by starting to type a color (e.g., pink or light) after the first bracket in datacolors=(. Then choose the color you want from the palette that pops up.

1. I hope you can now see how the correlation for all the data is greater than for either group. With all the data, the scatter is overall closer to a line compared with the scatter for each group on its own. There are many studies in the journals where authors have put females and males together to get a higher correlation.
2. Here's some really simple easy code to get a rough graph like the above when you are exploring data:

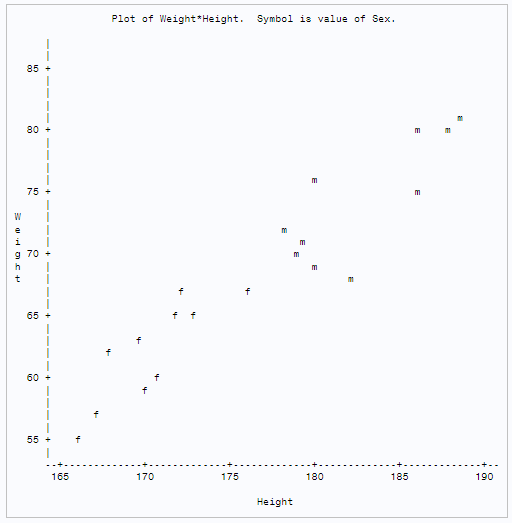
options ls=80 ps=45;

proc plot data=import;

plot Weight\*Height=Sex;

run;

The ls and ps stand for linesize and pagesize, which you can write out in full, if you like. The 80 and 45 refer to the width and height of the plot in keystrokes. This code goes all the way back to the 1960s, when SAS output was produced on an electric version of a typewriter. Here's what you get:



I love it, because it is so easy, but you can't publish a graph with this klunky format. To get a publishable graph, you have to spend literally hours with SAS. I conclude that the graphics in SAS are too unfriendly for occasional publication. Instead, do your graphs with Excel or a graphing package like GraphPad Prism or SigmaPlot. Either way, you still have to clean up the graphs in PowerPoint or other vector-graph editor. I say more about this when we do graphs in the reliability models.

1. To finish, you could add comments to Program 1 to remind you of what the block of code represent, then save it as a **.sas** program file by clicking on either the **Save** or **Save as** icon…



…and save it to the **Getting started** folder with a name like **Simple programs**. But I've done it for you! You can view the file by opening it with Word on your laptop (it's a simple text file), but you won't see the nice colors. So upload that file to the **Getting started** folder and take a look.

1. Close all the windows without saving, then Sign Out.
2. When you're ready to start mixed modeling, view the slideshow **Linear mixed models - the basics.pptx** in the folder **Mixed-model Workshop**. Then open the folder **Reliability models** and within it, open the file **SAS reliability mixed models.docx**.