Introduction to SAS, Working with continuous variables

Steve Simon

Created 2021-05-30

Overview

- Using variable labels
- Printing a small piece of data
- Simple descriptive statistics
- Printing row with smallest/largest value
- Missing value logic
- Simple transformations
- Histograms
- Correlations
- Scatterplots

author: Steve Simon

date: created 2021-05-30

purpose: to produce slides for module01 videos

Here is an overview of the topics I want to cover in module02.

Review definitions

- Categorical
  - Small number of possible values
  - Each value associated with a category
- Continuous
  - Large number of possible values
  - Potentially any value in an interval

Before we start, let's review a couple of definitions.

A **categorical variable** is a variable that can only take on a small number of values. Each value is usually associated with a particular category.

Examples of categorical variables are

sex (Male or Female),

race (White, Black, Native American, etc.),

cancer stage (I, II, III, or IV),

birth delivery type (Vaginal, C-section).

A **continuous variable** is a variable that can take on a large number of possible values, potentially any value in some interval.

Examples of continuous variables are

Birth weight in grams,

gestational age,

fasting LDL level.

There are some variables that are on the boundary between categorical and continuous, but it is not worth quibbling about here.

The point to remember is that the types of graphs that you use and the types of statistics that you compute are dependent on many things, but first and foremost on whether the variables are categorical, continuous, or a mixture.

Today, you will see examples involving mostly continuous variables.

Semicolons are important

- Ends every SAS statement
- Easy to forget
- Use this to your advantage
  - Several short lines
  - Indent continuations

Before I go too far, let me mention and important thing. Every SAS statement ends in a semicolon. This is important. You will forget a semicolon and it will lead to a cryptic error message. So here's a quick hint. If you get an error message on a certain line of code, look to see if you forgot a semicolon on the previous line. It happens to me all the time and I've been using SAS for decades.

Example of stretching statement across multiple lines.

One long line

```
statement option1 option2 option3 option4;
```

versus several short lines.

```
statement
  option1
  option2
  option3
  option4;
```

The use of semicolons is nice, in a way, because it allows you to stretch a complicated SAS statement across two or more rows of your program. This can often make your program more readable. It is hard to read a long line of code. Your eye has to scan left to right and you can sometimes lose track of which line you are on. Most newpapers place their articles in narrow columns because it makes them easier to read.

There is no official rule of thumb on this, but I do try to keep my lines below 50 characters. I also try to indent substatements with a data step or procedure. I use blank lines between data steps and procedures.

Don’t obsess about this now, but you’ll see a fairly consistent coding style that I use for my SAS code. You don’t have to follow my format, of course, which might be a bit too extreme for your tastes. Just experiment with things a bit until you can settle on a layout that you are comfortable with.

SAS code, documentation header

```
* m02-5507-simon-continuous-variables.sas
* author: Steve Simon
* date: created 2021-05-30
* purpose: to work with continuous variables
* license: public domain;
```

Here is the documentation header for a program that will show how to work with a dataset with mostly continuous variables.

SAS code: filename, libname, ods statements

```
%let p=q:/introduction-to-sas;

filename fat
  "&p/data/fat.txt";

libname intro
  "&p/data";

ods pdf file=
  "&p/results/m02-5507-simon.pdf";
```

The filename statement tells you where the raw data is stored.

The libname statement tells you where SAS will store any permanent datsets.

The ods statement tells you that SAS is going to store the results with a particular filename and in pdf format.

SAS code, input statement (1/2)

```
* Part01. Read in the data;

data intro.fat;
  infile fat;
  input
    case
    fat_brozek
    fat_siri
    dens
    age
    wt
    ht
    bmi
...
```

The input statement is very long.

SAS code, input statement (2/2)

```
...
    ffw
    neck
    chest
    abdomen
    hip
    thigh
    knee
    ankle
    biceps
    forearm
    wrist;
```

Here's the rest of it.

Rules for variable names

- Can use mix of
  - letters (A-Z, a-z),
  - numbers (0-9)
  - underscore (_)
  - no blanks
  - no symbols
- Can't start with a number
  - "a1" but not "1a"
- Capitalization not important
  - BMI, Bmi, bmi are same
- Up to 32 characters in length

There are important rules for variable names in SAS. You can use a mix of letters, numbers, and the underscore. You can't use blanks or any special symbols like the dollar sign ($) or the dash/minus sign.

You can't start with a number. So "a1" is okay, but "1a" is not.

Capitalization is not important in SAS. So you can call your variable "BMI" with all caps, or "Bmi" with mixed capitals or "bmi" with all lower case. SAS treats all of these the same.

Your variable name has to be 32 characters or less in length.

I'm using the variable names provided but if you create your own names, use brief (but descriptive) name for EVERY variable in your data set. There's no precise rule, but names should be around 8 characters long. Longer variable names make your typing tedious and much shorter variable names makes your code terse and cryptic.

I'm a bit more terse with these variable names than I normally would be just to reduce the amount of typing you have to do.

You should avoid special symbols in your variable names especially symbols that are likely

to cause confusion (the dash symbol, for example, which is also the symbol for subtraction). You should also avoid blanks. These rules are pretty much universal across most statistical software packages.

There are times when you'd like to have a blank in your variable name and you can use two special symbols in SAS (and most other statistical pacakges).

Here's how you can simulate blanks without actually using blanks.

the underscore symbol (above the minus key on most keyboards) and

the dot (period). (Note that SAS does not allow you to use the period this way.)

These symbols create some artificial spacing that mimics the blanks. Another approach is "CamelCase" which is the mixture of upper and lower case within a variable name with each uppercase designating the beginning of a new "word".

Recommendations for variable names

- Avoid generic names (x1, var01, etc.)
- Keep it short
  - Use commonly known abbreviations…
  - …but nothing cryptic
- Use all lower case (age, not AGE or Age)
- Separate words with underscores
  - fat_brozek, not fatbrozek
- Alternative: CamelCase
  - FatBrozek
- Caution: Writer's Exchange website
  - www.writersexchange.com

Your variable names should by descriptive. Avoid generic names like x1, var01, and so forth.

Keep things short. You can use commonly known abbreviations, such as "wt" for "weight". But avoid any cryptic abbreviations.

I like to keep everything in lower case. It is more readable than all upper case, and easier to remember than a mixture of upper and lower case. Some people prefer an initiial upper case, and there's nothing wrong with that. It is important, however, to be consistent.

You can use two or three short words in a variable name, but be sure to separate them using underscores. So the variable for percentage body fat as measured by Brozek's equation is "fat" and "brozek" separated by an underscore. Some people prefer CamelCase, where each word starts with an initial capital letter: capital F fat, capital B brozek.

The one thing you do want to avoid is just running two or three words together and all lower case. There's a story about a group that started up in the era before the web called Writer's Exchange. As you can guess this was a resource for new authors. When the web came out, they decided to put their resources up on a website, www.writersexchange.com. That seemed logical enough, but then someone notices that you could read the website as "www dot writer sex change dot com". Not exactly the image they wanted.

Break #1

- What have you learned so far
  - Rules for variable names
- What's next
  - Using variable labels
  - Printing a small piece of data

SAS variable labels

- Longer description of a variable
  - Can include blanks, special symbols
  - Internal documentation
  - Labels substituted on some (but not all) output
- Required in this class (see grading rubric)
- Recommendations for variable labels
  - Judicious use of upper and lower case
  - Spell out abbreviations
  - Specify units of measurement
  - Any other important details

SAS offers an opportunity for you to add documentation to your program about individual variables. These are called variable labels. They have almost no restrictions. You can use blanks, or special symbols like a dollar sign or a dash. The documentation that variable labels provide is mostly internal, but these labels are substituted in a few places like some graphs.

I strongly recommend use of variable labels and will require them for any homework you submit in this class. See the grading rubric for details.

What makes a good variable label? First take advantage of a mixture of upper and lower case to make your labels more readable. Spell out any abbreviations, even obvious abbreviations. If your variable has a measurement unit, specify that unit in your variable label. If there are other important details, put these in the variable label as well.

SAS code, labels (1/2)

```
* Part02. Add variable labels;

  label
    case="Case number"
    fat_brozek="Percentage body fat using
Brozek's equation, 457/Density - 414.2"
    fat_siri="Percent body fat using Siri's
equation, 495/Density - 450"
    dens="Density"
    age="Age (yrs)"
    wt="Weight (lbs)"
    ht="Height (inches)"
...
```

Every variable in a SAS program should have a label. This label will make some (but not all) of the SAS output more readable. it is also part of the internal documentation of your program. Note that some of these labels do not fit well in this Powerpoint slide, but that's okay.

SAS code, labels (2/2)

```
...
    neck="Neck circumference (cm)"
    chest="Chest circumference (cm)"
    abdomen="Abdomen circumference (cm) at the
umbilicus and level with the iliac crest"
    hip="Hip circumference (cm)"
    thigh="Thigh circumference (cm)"
    knee="Knee circumference (cm)"
    ankle="Ankle circumference (cm)"
    biceps="Extended biceps circumference (cm)"
    forearm="Forearm circumference (cm)"
    wrist="Wrist circumference (cm) distal to the
styloid processes"
  ;
run;
```

SAS code, proc print

```
* Part03. Print a small piece of the data;

proc print
    data=intro.fat(obs=10);
  var case fat_brozek fat_siri dens age;
  title1 "The first ten rows and five columns";
  title2 "of the fat data set";
run;
```

It's always a good idea to print out a small piece of your data to make sure everything is okay.

The data option tells SAS what data set you want to print. If you omit this, SAS will print the most recently created data set.

The obs=10 option limits the number of rows printed to the first 10. For large data sets, you should always take advantage of this option.

The var statement limits the variables that you print to those that you specify. Again, this is important for large data sets.

Please do not ever print more than ten rows or more than five variables, if you can help it. Excessively lengthy outputs will lose you a few points (see the grading rubric).

The title statement tells SAS to provide a descriptive title at the top of the page of output.

The run statement says you're done with the procedure.

SAS output, proc print

15:02 Monday, May 31, 2021 1

The first ten rows and five columns of the fat data set

Obs	case	fat_brozek	fat_siri	dens	age
1	1	12.6	12.3	1.0708	23
2	2	6.9	6.1	1.0853	22
3	3	24.6	25.3	1.0414	22
4	4	10.9	10.4	1.0751	26
5	5	27.8	28.7	1.0340	24
6	6	20.6	20.9	1.0502	24
7	7	19.0	19.2	1.0549	26
8	8	12.8	12.4	1.0704	25
9	9	5.1	4.1	1.0900	25
10	10	12.0	11.7	1.0722	23

SAS output

This is what your output looks like.

Break #2

- What have you learned so far
  - Using variable labels
  - Printing a small piece of data
- What's next
  - Simple statistics
  - Printing row with smallest/largest value

SAS code, proc means

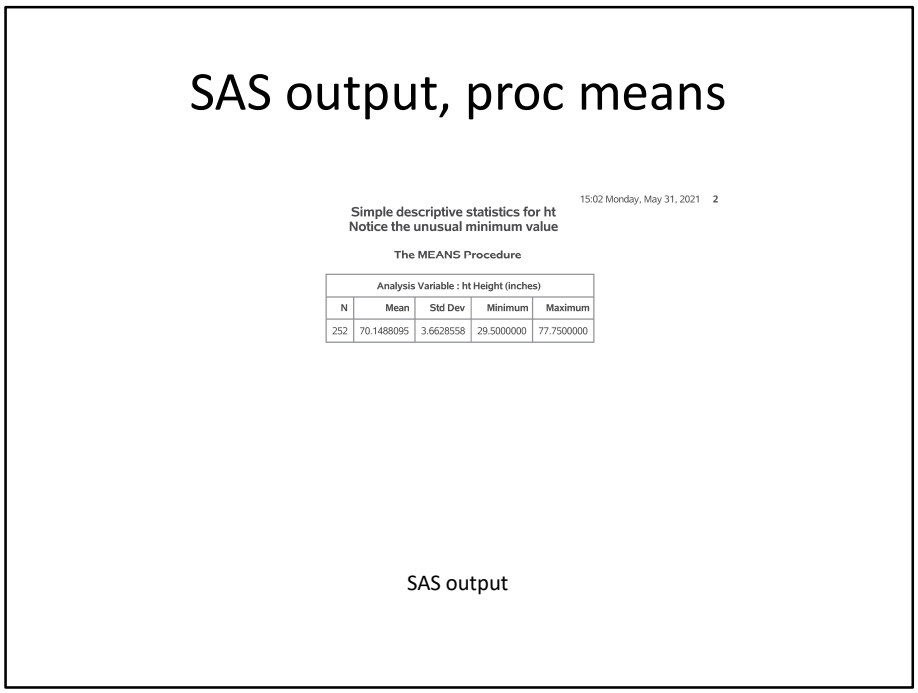
```
* Part04. Calculate simple statistics for ht;

proc means
    n mean std min max
    data=intro.fat;
  var ht;
  title1 "Simple descriptive statistics for ht";
  title2 "Notice the unusual minimum value";
run;
```

The means procedure will produce descriptive statistics for your data. By default, it will produce the count of non-missing values, the mean, the standard deviation, and the minimum and maximum values, but I am listing them explicitly here, just for show.

The data option tells SAS which data set you want descriptive statistics on, and the var statement tells SAS which variable(s) you want descriptive statistics on.

SAS output, proc means



15:02 Monday, May 31, 2021 2

Simple descriptive statistics for ht
Notice the unusual minimum value

The MEANS Procedure

Analysis Variable : ht Height (inches)				
N	Mean	Std Dev	Minimum	Maximum
252	70.1488095	3.6628558	29.5000000	77.7500000

SAS output

This is what your output looks like.

Notice the unusual minimum value. While this is not totally outside the realm of possibility, you should always ask when you see something unusual like this.

First, let's look at this value in the context of the other values in this row of data.

SAS code, print the smallest value

```
* Part05. Look at smallest value;

proc sort
    data=intro.fat;
  by ht;
run;

proc print
    data=intro.fat(obs=1);
  title1 "The row with the smallest ht";
  title2 "Note the inconsistency with wt";
run;
```

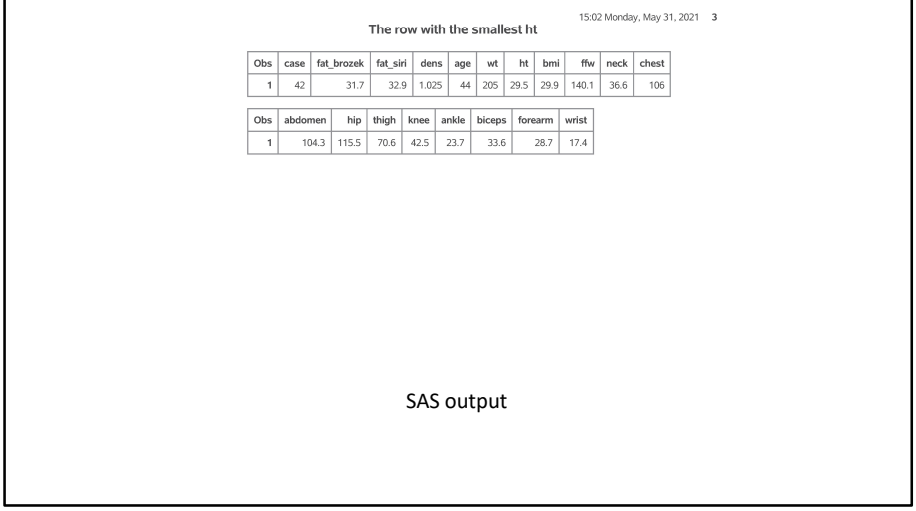
First, let's look at this value in the context of the other values in this row of data.

You do this by sorting the data so that the shortest subject becomes the first row of the data and the tallest subject becomes the last. Then print just the very first row of your data.

Warning: be careful about sorting your data if you can't get the data easily back to the original order. It might be okay, but there are times when you'd like your data all the way back and that means data in the original order. This data set has a case variable that you can resort by in order to get back ot the original order.

If you don't have a case variable, store the sorted data in a separate location: something along the lines of proc sort data=x out=y.

SAS output, print the smallest value



15:02 Monday, May 31, 2021 3

The row with the smallest ht

Obs	case	fat_brozek	fat_siri	dens	age	wt	ht	bmi	ffw	neck	chest
1	42	31.7	32.9	1.025	44	205	29.5	29.9	140.1	36.6	106

Obs	abdomen	hip	thigh	knee	ankle	biceps	forearm	wrist
1	104.3	115.5	70.6	42.5	23.7	33.6	28.7	17.4

SAS output

This is what your output looks like.

There is no possible way that a height of 29.5 inches could be paired with a weight of 205 pounds.

With this outlier on the low end, you might consider doing nothing other than noting the unusual value.

Alternately, you could delete the entire row associated with this value. Finally, you might consider converting the small ht value to a missing value code.

There is no one method that is preferred in this setting. If you encounter an unusual value, you should discuss it with your research team, investigate the original data sources, if possible, and review any procedures for handling unusual data values that might be specified in your research protocol.

Your data set may arrive with missing values in it already. Data might be designated as missing for a variety of reasons (lab result lost, value below the limit of detection, patient refused to answer this question) and how you handle missing values is way beyond the scope of this class. Just remember to tread cautiously around missing values as they are a

minefield.

Notice that I store the revised data sets with the row removed and with the 29.5 replaced by a missing value in different data frames. This is good programming practice. If you ever have to make a destructive change to your data set (a change that wipes out one or more values or a change that is difficult to undo), it is good form to store the new results in a fresh spot. That way, if you get cold feet, you can easily backtrack.

We'll use the data set with the 29.5 changed to a missing value for all of the remaining analyses of this data set.

Logic statements involving missing value codes are tricky. SAS stores missing value codes as the most extreme legal negative number. So if you want, for example, to exclude negative values, make sure that you account for missing values as well.

```
(ht < 0) & (ht ~= .)
```

SAS code, printing the largest value

```
* Part06. Look at the largest value;

proc sort
    data=intro.fat;
  by descending ht;
run;

proc print
    data=intro.fat(obs=1);
  title1 "The row with the largest ht";
  title2 "This seems quite normal to me";
run;
```

Just for the sake of completeness, let's look at the row of data with the largest height value. Add the keyword desc to sort the data in reverse order.

SAS output, printing the largest value

The row with the largest ht 15:02 Monday, May 31, 2021 4

Obs	case	fat_brozek	fat_siri	dens	age	wt	ht	bmi	ffw	neck	chest
1	96	17.3	17.4	1.0991	53	224.5	77.75	26.1	185.7	41.1	113.2

Obs	abdomen	hip	thigh	knee	ankle	biceps	forearm	wrist
1	99.2	107.5	61.7	42.3	23.2	32.9	30.8	20.4

SAS output

This is what your output looks like.

These values seem reasonable to me.

Break #3

- What have you learned so far
  - Simple statistics
  - Printing row with smallest/largest value
- What's next
  - Handling outliers
  - Missing values

Remove outlier

```
* Part07. Removing the entire row;

data intro.fat1;
  set intro.fat;
  if ht > 29.5;
run;
```

This code removes the entire row of data. Notice that I store the modified data under a new name. That way, if I regret tossing the entire row out, I can easily revert to the original data.

Change to missing value

```
* Part08. Converting the outlier to a missing
value;

data intro.fat2;
  set intro.fat;
  if ht=29.5 then ht=.;
run;
```

This code converts the height to a missing value, but keeps the original data.

There is no one method that is preferred in this setting. If you encounter an unusual value, you should discuss it with your research team, investigate the original data sources, if possible, and review any procedures for handling unusual data values that might be specified in your research protocol.

Your data set may arrive with missing values in it already. Data might be designated as missing for a variety of reasons (lab result lost, value below the limit of detection, patient refused to answer this question) and how you handle missing values is way beyond the scope of this class. Just remember to tread cautiously around missing values as they are a minefield.

Notice that I store the revised data sets with the row removed and with the 29.5 replaced by a missing value in different data frames. This is good programming practice. If you ever have to make a destructive change to your data set (a change that wipes out one or more values or a change that is difficult to undo), it is good form to store the new results in a fresh spot. That way, if you get cold feet, you can easily backtrack.

We'll use the data set with the 29.5 changed to a missing value for all of the remaining

analyses of this data set.

Logic statements and missing values

```
* Part09. Faulty approach for filtering out
negative values;

proc print
    data=intro.fat2;
  where ht < 0;
  title1 "ht < 0 will include ht = .";
run;
```

Here's an important thing to remember about missing values. SAS stores missing value codes as the most extreme legal negative number. This can sometimes lead to surprising and misleading results.

Every procedure in SAS has its own default approach to missing values and often provides you with one or more alternatives. You have to review this carefully for each and every statistical procedure that you run. If you do data manipulations involving missing values, you have to make sure that the result correctly reflects what you want.

SAS output, proc print with faulty logic

ht < 0 will include ht = . 15:02 Monday, May 31, 2021 5

Obs	case	fat_brozek	fat_siri	dens	age	wt	ht	bmi	ffw	neck	chest
252	42	31.7	32.9	1.025	44	205	.	29.9	140.1	36.6	106

Obs	abdomen	hip	thigh	knee	ankle	biceps	forearm	wrist
252	104.3	115.5	70.6	42.5	23.7	33.6	28.7	17.4

SAS output

This is what your output looks like.

In order to prevent this from happening, you need to check for missingness before applying any other logic statement.

The proper way to search for negative ht values

```
where (ht < 0) & (ht ~= .)
```

You may hate having to do this and wish that SAS would have handled things differently. Different packages, like R, have a three valued logic system where every logic statement (well, almost every logic statement) involving missing values codes to MISSING rather than to TRUE or FALSE. This sometimes works better, but sometimes the SAS approach works better.

SAS code, nmiss option

```
* Part10. Counting missing values;

proc means
    n nmiss mean std min max
    data=intro.fat2;
  var ht;
  title "Using the nmiss statistic";
run;
```

If you are concerned at all about missing values (and you should be), ask for the number of missing values in proc means using nmiss.

SAS output, proc means with nmiss option

Using the nmiss statistic 15:02 Monday, May 31, 2021 6

The MEANS Procedure

Analysis Variable : ht Height (inches)					
N	N Miss	Mean	Std Dev	Minimum	Maximum
251	1	70.3107570	2.6142960	64.0000000	77.7500000

SAS output

This is what your output looks like. Note that your data set has 251 observations and 1 missing value.

Break #4

- What have you learned
  - Missing values
- What's next
  - Simple transformations
  - Histograms

SAS code, simple transformations

```
* Part11. Simple transformations;

data converted_units;
  set intro.fat2;
  ht_cm = ht * 2.54;
  wt_kg = wt / 2.2;
run;

proc print
    data=converted_units(obs=10);
  var ht ht_cm wt wt_kg;
  title1 "Original and converted units";
run;
```

You can do simple transformations like unit conversions in SAS. Create a new dataset with the data statement. Use the set command to tell SAS that you plan to use and modify an existing dataset.

The conversions done here will turn height and weight into centimeters and kilograms, respectively.

SAS output, simple transformations

15:02 Monday, May 31, 2021 7

Original and converted units

Obs	ht	ht_cm	wt	wt_kg
1	77.75	197.485	224.50	102.045
2	77.50	196.850	188.15	85.523
3	76.00	193.040	216.00	98.182
4	76.00	193.040	244.25	111.023
5	75.50	191.770	194.00	88.182
6	75.25	191.135	171.50	77.955
7	75.00	190.500	212.75	96.705
8	74.75	189.865	210.25	95.568
9	74.75	189.865	224.75	102.159
10	74.50	189.230	186.25	84.659

SAS output

This is your output with measurements both in the original units and metric. Notice that I did not print any more than 10 rows of data.

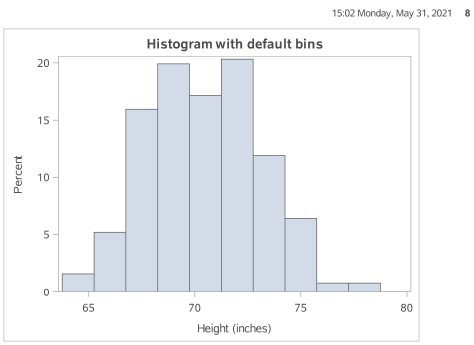
SAS code, Histogram

```
* Part12. Display a histogram;

proc sgplot
    data=intro.fat2;
  histogram ht;
  title "Histogram with default bins";
run;
```

Here's the code to create a histogram with the default option. Generally, it is wise to modify the defaults for any graphic image.

SAS output, Histogram



SAS output

This is your output with measurements both in the original units and metric. Notice that I did not print any more than 10 rows of data.

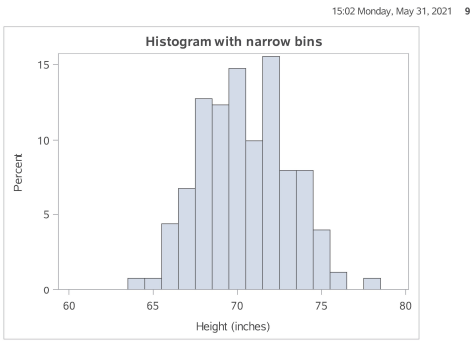
SAS code, Histogram with many bars

```
* Part13. Revised histogram with narrow bins;

proc sgplot
    data=intro.fat2;
  histogram ht / binstart=60 binwidth=1;
  title "Histogram with narrow bins";
run;
```

Here's the code to create a histogram with many bars. The first bar is centered at 60, and each bin has a width of 1 inch (plus or minus 0.5 inches)

SAS output, Histogram with many bars



SAS output

This is the revised histogram. You can also go in the opposite direction.

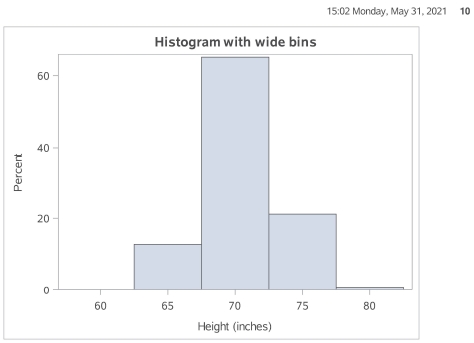
SAS code, Histogram with fewer bars

```
* Part14. Revised histogram with wide bins;

proc sgplot
    data=intro.fat2;
  histogram ht / binstart=60 binwidth=5;
  title "Histogram with wide bins";
run;
```

Here's the code to create a histogram with few bars. The first bar is again centered at 60, but now each bin has a width of 5 inches (plus or minus 2.5 inches).

SAS output, Histogram with many bars



SAS output

This is the revised histogram. There is no “correct” version of the histogram. Try several widths and see which one gives the clearest picture of your data.

Break #5

- What have you learned
  - Simple transformations
  - Histograms
- What's next
  - Correlations
  - Scatterplots

Correlations

- Informal interpretation
  - between +0.7 and +1.0: strong positive association
  - between +0.3 and +0.7: weak positive association
  - between -0.3 and +0.3: little or no association
  - between -0.3 and -0.7: weak positive association
  - between -0.7 and -1.0: strong negative association

The correlation coefficient is a single number between -1 and +1 that quantifies the strength and direction of a relationship between two continuous variables. As a rough rule of thumb, a correlation larger than +0.7 indicates a strong positive association and a correlation smaller than -0.7 indicates a strong negative association. A correlation between +0.3 and +0.7 (-0.3 and -0.7) indicates a weak positive (negative) association. A correlation between -0.3 and +0.3 indicates little or no association.

Don't take these rules too literally. You're not trying to make definitive statements about your data set. You are just trying to get comfortable with some general patterns that occur in your data set. A complex and definitive statistical analysis will almost certainly not agree with at least some of the preliminary correlations noted here.

The corr procedure produces, by default, a square correlation matrix of all the numeric variables. The noprob and nosimple options cut down on the amount of information printed. The with statement produces a rectangular correlation matrix.

SAS code, Simple correlations

```
* Part15. Calculate correlations;

proc corr
    data=intro.fat2
    noprob nosimple;
  var fat_brozek fat_siri;
  with neck -- wrist;
  title "Correlation matrix";
run;
```

SAS output, Correlations (1/2)

Correlation matrix 15:02 Monday, May 31, 2021 **12**

The CORR Procedure

10 With Variables:	neck chest abdomen hip thigh knee ankle biceps forearm wrist
2 Variables:	fat_brozek fat_siri

Pearson Correlation Coefficients, N = 252

	fat_brozek	fat_siri
neck Neck circumference (cm)	0.49149	0.49059
chest Chest circumference (cm)	0.70289	0.70262
abdomen Abdomen circumference (cm) at the umbilicus and level with the iliac crest	0.81371	0.81343
hip Hip circumference (cm)	0.62570	0.62520
thigh Thigh circumference (cm)	0.56128	0.55961

SAS output

SAS output, Correlations (2/2)

Correlation matrix 15:02 Monday, May 31, 2021 13

The CORR Procedure

Pearson Correlation Coefficients, N = 252		
	fat_brozek	fat_siri
knee Knee circumference (cm)	0.50779	0.50867
ankle Ankle circumference (cm)	0.26678	0.26597
biceps Extended biceps circumference (cm)	0.49303	0.49327
forearm Forearm circumference (cm)	0.36328	0.36139
wrist Wrist circumference (cm) distal to the styloid processes	0.34757	0.34657

SAS output

Notice that I deliberately avoided printing out the entire corelation matrix, which would span several pages of output. This is something you yourself should strive for, especially when you are running something that you want to share with others. Don't overwhelm your reader with things that they may not need. If your readers do want more information, it is better for them to tell you rather than you giving them more in the first meeting that they need.

SAS code, New dataset with correlations

```
* Part16. Save the correlations in a separate
data file.;

proc corr
    data=intro.fat2
    noprint
    outp=correlations;
  var fat_brozek fat_siri;
  with neck -- wrist;
run;

proc print
    data=correlations;
  title "Correlation matrix output to a data
set";
run;
```

You can save the correlations in a separate data file.

SAS output, New dataset with correlations (1/2)

15:02 Monday, May 31, 2021 14

Correlation matrix output to a data set

Obs	_TYPE_	_NAME_	fat_brozek	fat_siri
1	MEAN		18.938	19.151
2	STD		7.751	8.369
3	N		252.000	252.000
4	CORR	neck	0.491	0.491
5	CORR	chest	0.703	0.703
6	CORR	abdomen	0.814	0.813
7	CORR	hip	0.626	0.625
8	CORR	thigh	0.561	0.560
9	CORR	knee	0.508	0.509
10	CORR	ankle	0.267	0.266
11	CORR	biceps	0.493	0.493

SAS output

The output is a bit unusual because SAS wants to include means and standard deviations in your output. You can and should remove this. It would be easy enough to do (use the where statement), but I wanted to show you the full data set.

SAS output, New dataset with correlations (2/2)

15:02 Monday, May 31, 2021 15

Correlation matrix output to a data set

Obs	_TYPE_	_NAME_	fat_brozek	fat_siri
12	CORR	forearm	0.363	0.361
13	CORR	wrist	0.348	0.347

SAS output

SAS code, Modified correlations (1/2)

```
* Part17. Modify these correlations.;

data correlations;
  set correlations;
  if _type_="CORR";
  drop type;
  fat_brozek=round(100*fat_brozek);
  fat_siri=round(100*fat_siri);
run;

proc sort
    data=correlations;
  by descending fat_brozek;
run;
```

Saving as a data file allows you to manipulate the individual correlations. Here we multiply the correlations by 100, round them, and sort them. This can often simplify the interpretation of large correlation matrices.

This code does the reordering and printing.

SAS code, Modified correlations (2/2)

```
* Part18. Print the modified correlations.;

proc print
    data=correlations;
  title "Rounded and re-ordered correlation
matrix";
run;
```

SAS output, Modified correlations

15:02 Monday, May 31, 2021 16

Rounded and re-ordered correlation matrix

Obs	_TYPE_	_NAME_	fat_brozek	fat_siri
1	CORR	abdomen	81	81
2	CORR	chest	70	70
3	CORR	hip	63	63
4	CORR	thigh	56	56
5	CORR	knee	51	51
6	CORR	neck	49	49
7	CORR	biceps	49	49
8	CORR	forearm	36	36
9	CORR	wrist	35	35
10	CORR	ankle	27	27

SAS output

This is the output. You can see that measurements at the extremities are poor predictors of body fat. Apparently, we grow fat from the middle outward.

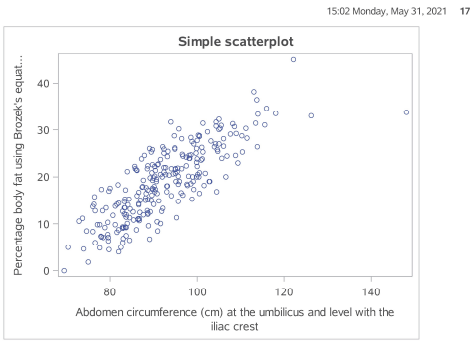
SAS code, Simple scatterplot

```
* Part19. Draw a scatterplot.;

proc sgplot
    data=intro.fat2;
  scatter x=abdomen y=fat_brozek;
  title "Simple scatterplot";
run;
```

A scatterplot is also useful for examining the relationship among variables. You can produce scatterplots several different ways, but the scatterplots produced by the sgplot procedure have the most flexibility.

SAS output, Simple scatterplot



SAS output

This plot shows a general upward trend.

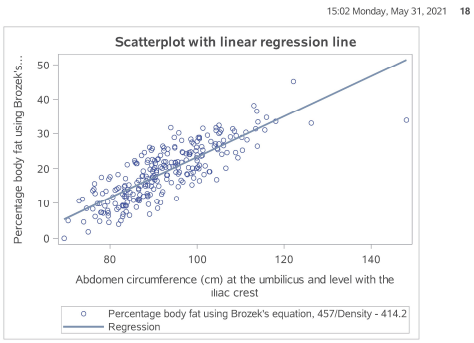
SAS code, Linear trend line

```
* Part20. Adding linear trend line.;

proc sgplot
    data=intro.fat2;
  scatter x=abdomen y=fat_brozek;
  reg x=abdomen y=fat_brozek;
  title "Scatterplot with linear regression
line";
run;
```

The reg statement adds a least squares trend line to your graph.

SAS output, Simple scatterplot with linear trend



SAS output

The trend line is very useful for large and noisy data sets. It also allows you to more quickly visualize extreme values. Notice, for example, that the person with the largest abdomen measure (the biggest gut, if I can be informal) is quite out of line with what you might expect the relationship to be.

SAS code, Smooth curve

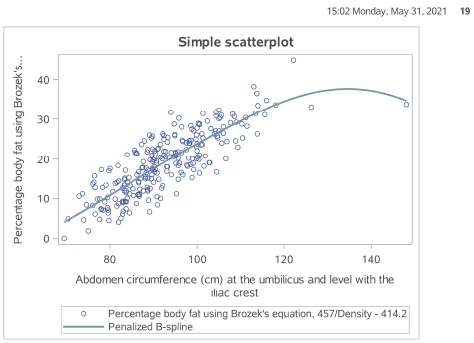
```
* Part21. Adding a smooth curve.;

proc sgplot
    data=intro.fat2;
  scatter x=abdomen y=fat_brozek;
  pbspline x=abdomen y=fat_brozek;
  title "Scatterplot with a smooth curve";
run;

ods pdf close;
```

The pbspline statement adds a smoothing spline to your graph. This allows you to informally investigate more complex relationships.

SAS output, Simple scatterplot with smooth curve



SAS output

The smoothing spline provides some evidence that the relationship is roughly linear a low and medium abdomen measurements, but tends to level off a bit at higher levels. Interpret this with caution, of course, because you have very little data at extrmemy high adbomen measures.

Don’t forget!

```
ods pdf close;
```

I always seem to forget this last statement and then I get upset with SAS for not providing the PDF output. But SAS can’t produce the PDF output until you tell it you are done. So don’t yell at your computer when it’s your own darn fault (just like Jimmy Buffet in the Margaritaville song).

Summary

– What have you learned
  - Using variable labels
  - Printing a small piece of data
  - Simple descriptive statistics
  - Printing row with smallest/largest value
  - Missing value logic
  - Simple transformations
  - Histograms
  - Correlations
  - Scatterplots