Introduction to SAS, Importing a wide range of data formats

Overview

- Space delimited files
- Comma delimited files
- Tab delimited files
- Tilde delimited files
- Fixed width files
- String data
- First line names

author: Steve Simon date: Created 2021-05-30 purpose: to produce slides for module02 videos license: public domain

Here is an overview of what I want to cover in module03.

Importing choices (1 of 2)

- A wide range of formats
  - Space delimited
  - Comma separated values
  - Tab delimited
  - Fixed format
- Strings in your input

I want to show a few simple examples of importing data. There are several common formats and SAS can handle all of these easily.

Importing choices (2 of 2)

- proc import
  - First row includes variable names
  - Binary data files
- Manual reformatting
  - Global search and replace
  - Not usually a good idea
- Skipping rows
- Converting strings to numbers

Some data sets use the first row to represent variable names while others don't.

If you have to, you can manually reformat the data. Use the global search and replace function in your text editor program. I generally try to avoid this. If your data set changes, you have to redo the reformatting step, which is tedious and error prone. You'd be better off learning a few tricks to get SAS to read a nonstandard data set.

Sometimes you have to skip a few rows. Sometimes you have to convert strings to numbers.

Space delimited, example

```
4 8 40
8 16 80
12 24 120
16 32 160
24 48 240
```

You've already imported a file similar to this. It is a space delimited file with one blank between each number. This is the simplest case, and requires very little code.

Space delimited, SAS code (1/3)

```
* m03-5507-simon-import-space-delimited.sas
* author: Steve Simon
* creation date: 2019-07-01
* purpose: to import data with spaces as
delimiters
* license: public domain;
```

Here is the SAS code to read and print this file, starting with the documentation header.

Space delimited, SAS code (2/3)

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

ods pdf file=
    "&path/results/m03-5507-simon-import-space-
delimited.pdf";

libname perm
    "&path/data";

filename raw_data
    "&path/data/space-delimited.txt";
```

Here are the lines to tell SAS where everything belongs

Space delimited, SAS code (3/3)

```
data perm.space_delimited;
  infile raw_data;
  input x y z;
run;

proc print
    data=perm.space_delimited(obs=2);
  title1 "First two rows of data";
run;

ods pdf close;
```

Here is the rest of the program.

Break #1

- What have you learned
  - Space delimited files
- What's next
  - Comma delimited files

Comma delimited, example

```
4,8,40
8,16,80
12,24,120
16,32,160
24,48,240
```

Many text files use commas to separate individual numbers. These often use a special file extension, .csv. These files are especially popular with Excel users, as they are about as easy to import and export as any other type of simple text files.

Here's a simple example of a comma delimited file.

Comma delimited, SAS code (1/3)

```
* m03-5507-simon-import-comma-delimited.sas
* author: Steve Simon
* creation date: 2019-07-01
* purpose: to import comma delimited files
* license: public domain;
```

Here is the SAS code to read and print this file, starting with the documentation header.

Comma delimited, SAS code (2/3)

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

ods pdf file=
    "&path/results/m03-5507-simon-import-comma-
delimited.pdf";

libname perm
    "&path/data";

filename raw_data
    "&path/data/comma-delimited.csv";
```

Here are the lines to tell SAS where everything belongs

Comma delimited, SAS code (3/3)

```
data perm.comma_delimited;
  infile raw_data delimiter=",";
  input x y z;
run;

proc print
    data=perm.comma_delimited(obs=2);
  title1 "First two rows of data";
run;
ods pdf close;
```

Here is the rest of the program.

Here is the SAS code to read and print this file. Nothing has changed other than the name of the file and the delimiter option on the infile statement.

Break #2

- What have you learned
  - Comma delimited files
- What's next
  - Tab delimited files

Tab delimited, example

```
4   8   40
8   16  80
12  24  120
16  32  160
24  48  240
```

I've already told you that tabs are evil, but if you encounter a tab delimited file, don't panic. You can handle it.

Here's a simple example of a tab delimited file. Note that the numbers are left justfied, which is a hint that there are tabs lurking in the file. The tabs here stop at columns, 5, 9, and 13, which is a bit weird, but may just represent how my system treats tabs. The classic thing to look for in a tab delimited file, besides the right justification, is the semi-aligned, but not perfectly aligned numbers. This doesn't happen with this data because all of the numbers are three digits or less, but it can happen when the numbers take up a bit more room and the numbers are uneven in size.

You can also detect tabs by opening the file in a text editor like notebook and playing around with adding extra spaces. A lack of action followed by a sudden hop is a pretty good indication that you are dealing with tabs.

Tab delimited, SAS code (1/3)

```
* m03-5507-simon-import-tab-delimited.sas
* author: Steve Simon
* creation date: 2019-07-01
* purpose: to import a comma delimited file into
SAS
* license: public domain;
```

Here is the SAS code to read and print this file. You cannot enter the tab character directly as a delimiter. Instead, you have to provide the hex code, 09. Note the use of quote marks and X.

This import worked fine, but beware. Some of these tab separated value files have a few stray blanks mixed in with the tabs, and this can cause havoc.

Tab delimited, SAS code (2/3)

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

ods pdf file=
    "&path/results/m03-5507-simon-import-tab-
delimited.pdf";

libname perm
    "&path/data";

filename raw_data
  "&path/data/tab-delimited.txt";
```

Here are the lines to tell SAS where everything belongs

Tab delimited, SAS code (3/3)

```
data perm.tab_delimited;
  infile raw_data delimiter="09"X;
  input x y z;
run;

proc print
    data=perm.tab_delimited(obs=2);
  title1 "First two rows of data";
run;

ods pdf close;
```

Here is the rest of the program.

Break #3

- What have you learned
  - Tab delimited files
- What's next
  - Tilde delimited files

Tilde delimited, example

```
4~8~40
8~16~80
12~24~120
16~32~160
24~48~240
```

You might even see a bizzare symbol as a delimiter. Something that almost never appears. The tilde(~) is a good example. We have not talked about string data yet, but if your data has lots of strings and it has a lot of unmatched quote marks, commas, and other weird things that might trip you up, the owner of the data might use a tilde because nothing else works as a delimiter.

Here's a simple example of a tilde delimited file.

Tilde delimited, SAS code (1/3)

```
No beginning found
```

Here is the SAS code to read and print this file. You cannot enter the tab character directly as a delimiter. Instead, you have to provide the hex code, 09. Note the use of quote marks and X.

This import worked fine, but beware. Some of these tab separated value files have a few stray blanks mixed in with the tabs, and this can cause havoc.

Tilde delimited, SAS code (2/3)

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

ods pdf file=
  "&path/results/import-tilde-delimited.pdf";

libname module01
  "&path/data";

filename raw_data
  "&path/data/tilde-delimited.txt";
```

Here are the lines to tell SAS where everything belongs

Tilde delimited, SAS code (3/3)

```
data module01.tilde_delimited;
  infile raw_data delimiter="~";
  input x y z;
run;

proc print
    data=module01.tilde_delimited(obs=2);
  title1 "First two rows of data";
run;

ods pdf close;
```

Here is the rest of the program.

Break #4

- What have you learned
  - Tilde delimited files
- What's next
  - Fixed width files

Fixed width, example

```
 4 8 40
 816 80
1224120
1632160
2448240
```

Sometimes you may run across a file with no delimiters at all. The numbers stand next to each other shoulder to shoulder. Here's an example. There's a space or two here or there because not every number is exactly the same number of digits. But there are plenty of numbers without any spaces behind them.

Why might you do this? If your file was very large, you may not want to afford the luxury of delimiters. The file is already straining to fit in and adding delimiters would just be too much.

These days, of course, storage is cheap, but you still might encounter a fixed width file now and then.

The only way an undelimited file would work is if there was a rigid structure where each number would reside in a specified column location. It's like a series of row houses and the address tells you which house is which.

The key that tells you what columns correspond to which numbers has to reside in a different file. And you have to translate this key to the proper SAS code.

In the fixed width file above, you would have to be told there are three numbers, the first in columns 1 and 2, the second in columns 3 and 4 and the last in columns 5 through 7.

Fixed width, SAS code (1/3)

```
* m03-5507-simon-import-fixed-width.sas
* author: Steve Simon
* creation date: 2019-07-01
* purpose: to import data in a fixed width format
* license: public domain;
```

Here is the SAS code to read and print this file. You cannot enter the tab character directly as a delimiter. Instead, you have to provide the hex code, 09. Note the use of quote marks and X.

This import worked fine, but beware. Some of these tab separated value files have a few stray blanks mixed in with the tabs, and this can cause havoc.

Fixed width, SAS code (2/3)

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

ods pdf file=
    "&path/results/m03-5507-simon-import-fixed-
width.pdf";

libname perm
    "&path/data";

filename raw_data
    "&path/data/string-data.txt";
```

Here are the lines to tell SAS where everything belongs

Fixed width, SAS code (3/3)

```
data module01.fixed_width;
  infile raw_data delimiter=",";
  input
    x 1-2
    y 3-4
    z 5-7;
run;

proc print
    data=module01.fixed_width(obs=2);
  title1 "First two rows of data";
run;

ods pdf close;
```

Here is the SAS code to read and print this file. Each variable is followed by a column range.

Break #5

- What have you learned
  - Fixed width files
- What's next
  - String data

String data, example

```
Alpha 4 8
Bravo 8 16
Charlie 12 24
Delta 16 32
Echo 24 48
```

Strings add a bit of complication. A string is a sequence of one or more letters, like a person's name, or a mixture of numbers and letters, like a person's address.

Here's a small text file with a string of four to six letters.

String data, SAS code (1/3)

```
* m03-5507-simon-import-string-data.sas
* author: Steve Simon
* creation date: 2019-07-02
* purpose: to import data that includes a string
* license: public domain;
```

Here is the documentation header.

String data, SAS code (2/3)

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

ods pdf file=
    "&path/results/m03-5507-simon-import-string-
data.pdf";

libname perm
    "&path/data";

filename raw_data
    "&path/data/string-data.txt";
```

Here are the lines to tell SAS where everything belongs

String data, SAS code (3/3)

```
data perm.string_data;
  infile raw_data delimiter=" ";
  input
    name $
    x
    y;
run;

proc print
    data=perm.string_data(obs=2);
  title1 "First two rows of data";
run;

ods pdf close;
```

Here is the SAS code to read and print this file. Designate a string variable by adding a dollar sign after the variable name.

Complications with string data

- Strings longer than eight characters
  - Informat statement
- Strings with delimiters
  - Use different delimiter
  - Enclose sting in quotes
- Strings with quotes
  - Use double quotes around string with single quote
    - "It's my bidthday!"
  - Use single quotes around string with double quote
    - 'Smile and say "Cheese!" when I take this picture'
  - Use escape codes

SAS will have problems with certain strings. I don't want to go into too much detail now, but here are some things to look out for.

SAS expects strings to be eight characters in length or less. If your string has more than eight character, you have to warn SAS with an informat statement.

If you have a space delimited file, but your strings have space in them, the simplest thing is to switch to a different delimiter, like a comma. What if your string has commas and spaces? Well you could try using a tilde, or some other obscure character. Another choice is to enclose your string in quotes.

What about strings with quotes in them. This can happen. I did a text analysis of the book A Chirstmas Carol by Charles Dickens and the text itself had lots of he said quotes and lots of apostrophes. There's always a work around. You can use double quotes to enclose a string with an apostrophe. Use single quotes to enclose a string with double quotes. There are also special escape codes.

Break #6

- What have you learned
  - String data
- What's next
  - First line names

First line names, example

```
name,x,y
Alpha,4,8
Bravo,8,16
Charlie,12,24
Delta,16,32
Echo,24,48
```

First line names, SAS code (1/3)

```
* m03-5507-simon-import-first-line-names.sas
* author: Steve Simon
* creation date: 2019-07-02
* purpose: to import data with variable names on
the first line
* license: public domain;
```

Here is the documentation header.

First line names, SAS code (2/3)

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

ods pdf file=
    "&path/results/m03-5507-simon-import-first-
line-names.pdf";

libname perm
    "&path/data";

filename raw_data
    "&path/data/first-line-names.csv";
```

Here are the lines to tell SAS where everything belongs

First line names, SAS code (3/3)

```
proc import
    datafile=raw_data dbms=dlm
    out=perm.first_line_names replace;
  delimiter=",";
  getnames=yes;
run;

proc print
    data=perm.first_line_names(obs=2);
  title1 "First two rows of data";
run;

ods pdf close;
```

proc import will read the file, and the getnames=yes subcommand will read the variable names from the first line of data.

Summary

– What have you learned
  - Space delimited files
  - Comma delimited files
  - Tab delimited files
  - Tilde delimited files
  - Fixed width files
  - String data
  - First line names