Chapter 1 Getting Your Data into SAS

1.1 The SAS Language

Many software applications are either menu driven or command driven (enter a command and you see the results), but SAS is neither. SAS uses statements to write a series of instructions called a **SAS program**. The primary purpose of each SAS program is to process and analyze data through a collection of statements.

It began in the late 60's and early 70's as a statistical package. SAS originally stood for Statistical Analysis System. However, unlike many competing statistical packages, SAS is also an extremely powerful, general-purpose programming language. SAS is a predominant software in the pharmaceutical industry and most Fortune 500 companies. In recent years, it has been enhanced to provide state-of-the-art data mining tools and programs for Web development and analysis.

The only way to learn a programming language is to practice (make errors, correct the errors, and make new errors). Once you understand and master all the "words" (SAS commands), you can write good "essays" (SAS program).

Note: SAS as a statistical analysis program, it requires us to identify appropriate statistical technique that should be carried out given a data set. For each of the statistical methods, there is a specific SAS command, and we provide a sequence of statements executed in order. Each statement gives instructions to SAS and must be placed correctly in the program.

We will use SAS Studio supported by the SAS University Edition (Mac OS X) for all the illustrations in this course. The full version – SAS Educational Analytics has more features than the University Edition. Only the interface differs a little. We will now open up a SAS session.

1.1.1 Interface of SAS/SAS Studio

The main window of SAS Studio consists of a navigation pane on the left and a work area on the right.

a. **Explorer/Navigation window** – find your program and datasets here.

The navigation pane provides access to your server files and folder shortcuts, your tasks and snippets, the libraries that you have access to, and your file shortcuts. The Server Files and Folders section is displayed by default.

The work area is used to display your data, code, tasks, logs, and results.

b. **Code editor window** – type your SAS program here.

SAS Studio includes a color-coded, syntax-checking editor for editing new or existing SAS programs. You can also edit SOURCE entries in SAS catalogs. The editor includes a wide

variety of features such as autocompletion, automatic formatting, and pop-up syntax help. With the code editor, you can write, run, and save SAS programs. You can also modify and save the code that is automatically generated when you run a task. The menu bar in the code editor window. Hover over and read each one.

c. **Log window** – contains messages by SAS keeping notes for all analysis and showing errors and warning.
d. **Results window (Results)** – display the output tables and figures

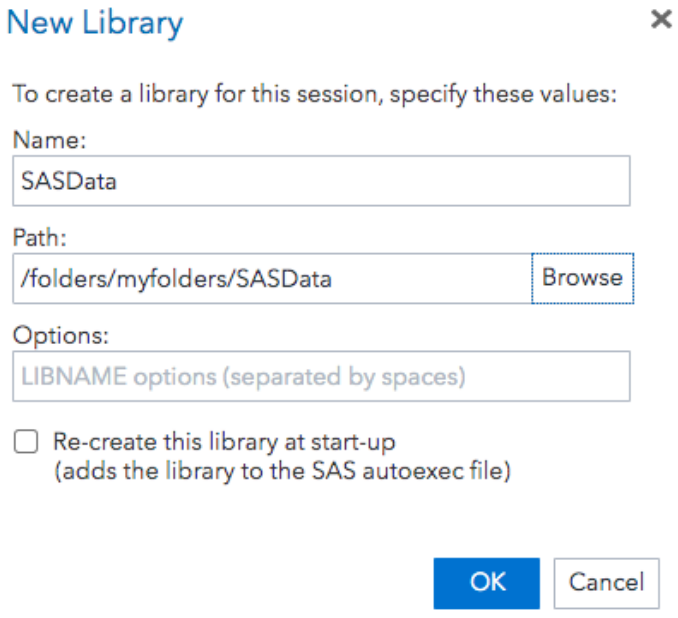
If there are no errors, the results open automatically. If there are errors, the Log tab opens by default.

Make good use of the SAS Studio users' guide when you have questions:
https://documentation.sas.com/api/docsets/webeditorug/3.8/content/webeditorug.pdf?locale=en

1.1.2 SAS Library

Before we start writing SAS program, let's create a SAS library. In the navigation panel, click on Libraries. It shows four buttons – new library, delete selected libraries and tables, properties, and refresh libraries section. Click on the first button to create a new library.

Then you will see the following window pop-up. Let's name the folder as SASData. Then click Browse to give a path. We will store our generated SAS data sets in this folder.

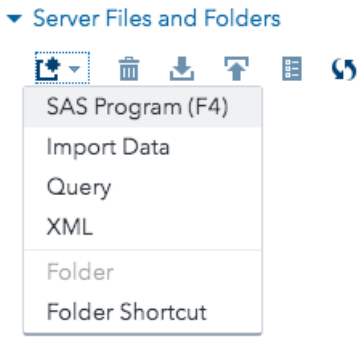


Next, click on My Libraries. In addition to the SAS folder that we just created (SASData), you should also see SASHELP, SASUSER, WEBWORK, and WORK folders. SASHELP shows the datasets that are provided by SAS. The SASUSER library is read only, as in any SAS server environment. You cannot save content to this library. Webwork is the default output library in interactive mode. The WORK folder contains all temporary datasets when a specific file directory is not used.

1.1.3 A Sample SAS Program

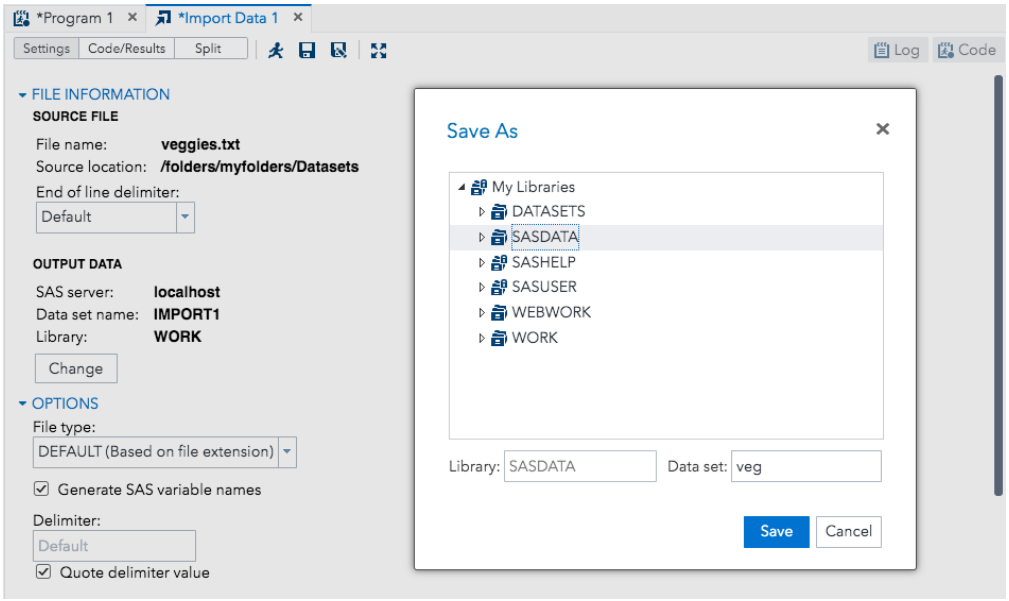
We will use *veggies.txt* in this example.

To create a new program, click the first button under Server Files and Folders (in the navigation panel on the left) or click F4.



a. Using Import Data feature in SAS Studio

Next to the SAS program, we can see **Import Data**. This is a nice feature of SAS studio. Click on Import Data, drag (from My Folders on the left panel) and drop your data file, or click select file. Then click on settings, Change the OUTPUT DATA (Library: SASDATA; Data set: veg).



Remember to check the OPTIONS – Generate SAS variable names. If the data file contains the variable names on the first line, check this box (leave it as default); otherwise, please de-select this box.

Then click on Run (click on little running man on the top panel). There are multiple ways to check if the dataset is loaded properly.

- See Log window: is there any error or warning?
- See Results window: there should be tables showing the attributes of the data file.
- Open the SAS dataset directly: My libraries → SASDATA → double click on VEG.

b. Use INFILE command to import a text file

A traditional way to import a txt datafile is to use INFILE command. In the Code Editor, we can start writing statements.

```
*Read in the dataset from a txt file and convert it to a SAS dataset*;
data veggies;
    infile "/folders/myfolders/Datasets/veggies.txt";
    input name $ code $ days number price;
    costperseed = price/number;
run;
```

Read data using a DATA step

- Reading dataset or changing variables in a dataset – we use DATA step.
- The first line (data veggies) defines the name of the SAS dataset.
- **INFILE** specifies the path where the file is located. If the data lines start from the second line (first line is the variable name), we add **FIRSTOBS=2** right after the path quote telling SAS to start reading from the 2nd line.
- **INPUT** specifies the variables. By default, SAS treats all variables as numeric. If reading character/string variables, put a $ after the variable name.
- **RUN** is required for all procedures/steps – telling SAS to execute the commands.

Change/create variable in a DATA step

- In the example, we create a new variable (cost per seed) as price divided by number.
- Various operations can be specified in a DATA step. For instance,
    - DROP statement: remove any variables from a data set;
    - KEEP statement: instead of dropping, it tells SAS which variables to keep;
    - See SAS Help document for a list of operators. Here is a snapshot.

Arithmetic Operators

Symbol	Definition	Example	Result
**	exponentiation	a**3	raise A to the third power
*	multiplication [1]	2*y	multiply 2 by the value of Y
/	division	var/5	divide the value of VAR by 5
+	addition	num+3	add 3 to the value of NUM
-	subtraction	sale-discount	subtract the value of DISCOUNT from the value of SALE

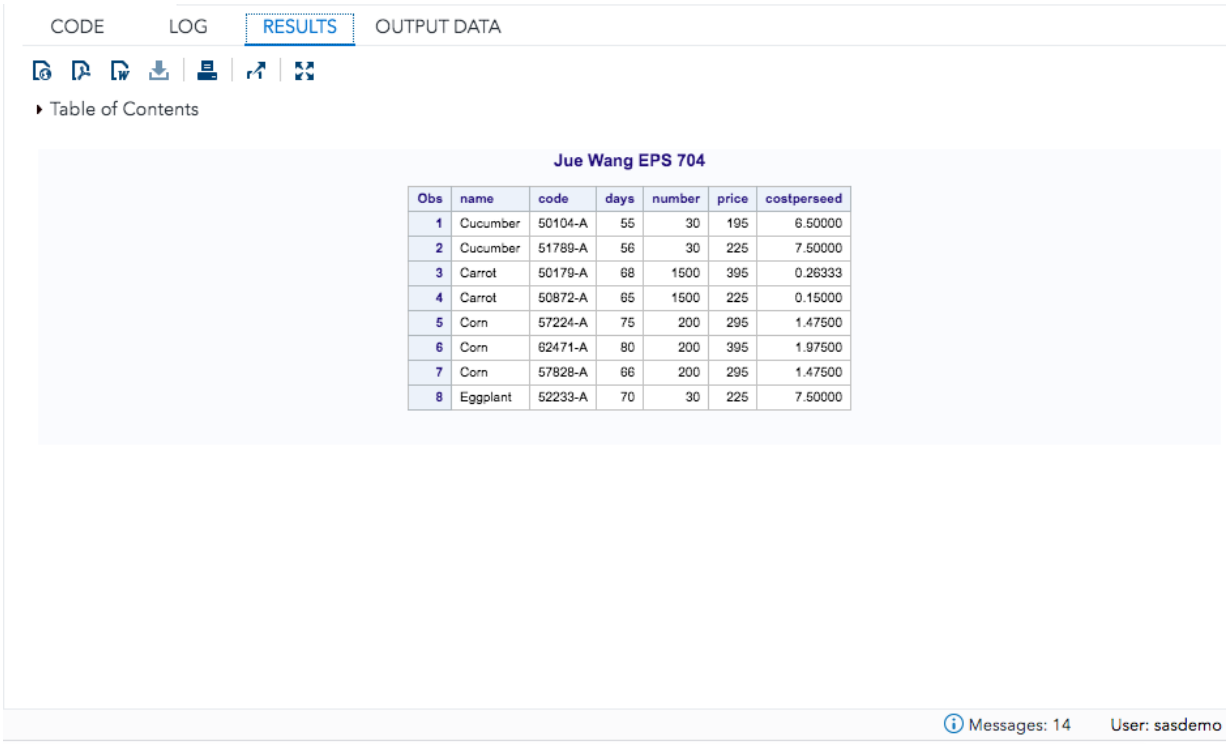
1 The asterisk (*) is always necessary to indicate multiplication; 2Y and 2(Y) are not valid expressions.

Right now, a SAS dataset is generated and placed in the WORK library (as a temporary file). To permanently create and store a SAS data file, we add an affiliation to the data name.

```
*Read in the dataset from a txt file and convert it to a SAS dataset (Library SASDATA)*;
data sasdata.veggies;
    infile "/folders/myfolders/Datasets/veggies.txt";
    input name $ code $ days number price;
    costperseed = price/number;
run;

proc print data=sasdata.veggies;
run;
```

c. Print data using a procedure called PROC PRINT. **PROC** stands for procedure. The output table appears in the Results window. Highlight/Select the codes and hit run (running man). Remember always check the data either in the library, OUTPUT DATA window (in SAS studio) or using PROC PRINT to display in an output table (RESULTS window) to make sure they were read in correctly!



Obs	name	code	days	number	price	costperseed
1	Cucumber	50104-A	55	30	195	6.50000
2	Cucumber	51789-A	56	30	225	7.50000
3	Carrot	50179-A	68	1500	395	0.26333
4	Carrot	50872-A	65	1500	225	0.15000
5	Corn	57224-A	75	200	295	1.47500
6	Corn	62471-A	80	200	395	1.97500
7	Corn	57828-A	66	200	295	1.47500
8	Eggplant	52233-A	70	30	225	7.50000

Notes:

- **Semicolons conclude every complete statement in SAS!**
- The colors are helpful for detecting coding errors.
    - Comments begin with a * and are color coded in green.
    - DATA or PROC calls (define keywords) are in dark blue
    - Statement in the quotes are purple.
    - Numeric constants are teal.

- Quick ways in SAS studio to import data, create new program, open a SAS dataset, or search any file in SAS library. See the three buttons on the very top panel:

1.2 More Ways to Create SAS Datasets

a. Use DATALINES to enter data responses

Instead of importing a data set, we can always input data responses directly in the Code Editor. **DATALINES** statement in a DATA step – tells SAS to expect data input

Example code:

```
*Input data directly using DATALINES and store it in the Library SASDATA*;
data sasdata.veggies2;
    input name$ code$ days number price;
    costperseed = price/number;
    datalines;
    Cucumber 50104-A 55 30   195
    Cucumber 51789-A 56 30   225
    Carrot   50179-A 68 1500 395
    Carrot   50872-A 65 1500 225
    Corn     57224-A 75 200  295
    Corn     62471-A 80 200  395
    Corn     57828-A 66 200  295
    Eggplant 52233-A 70 30   225
    ;
run;
```

Here, the sasdata.veggies2 command stores the veggies2 dataset into the sasdata folder. You can see the only difference is to use DATALINES instead of INFILE command in a DATA step. Please note that the other commands should be inserted before DATALINES.

We can also add labels using LABEL command in DATA step. For example,

Variable	Label
Name	Vegetable name
Code	Product code
Days	Days to germination
Number	Number of seeds
Price	Retail price

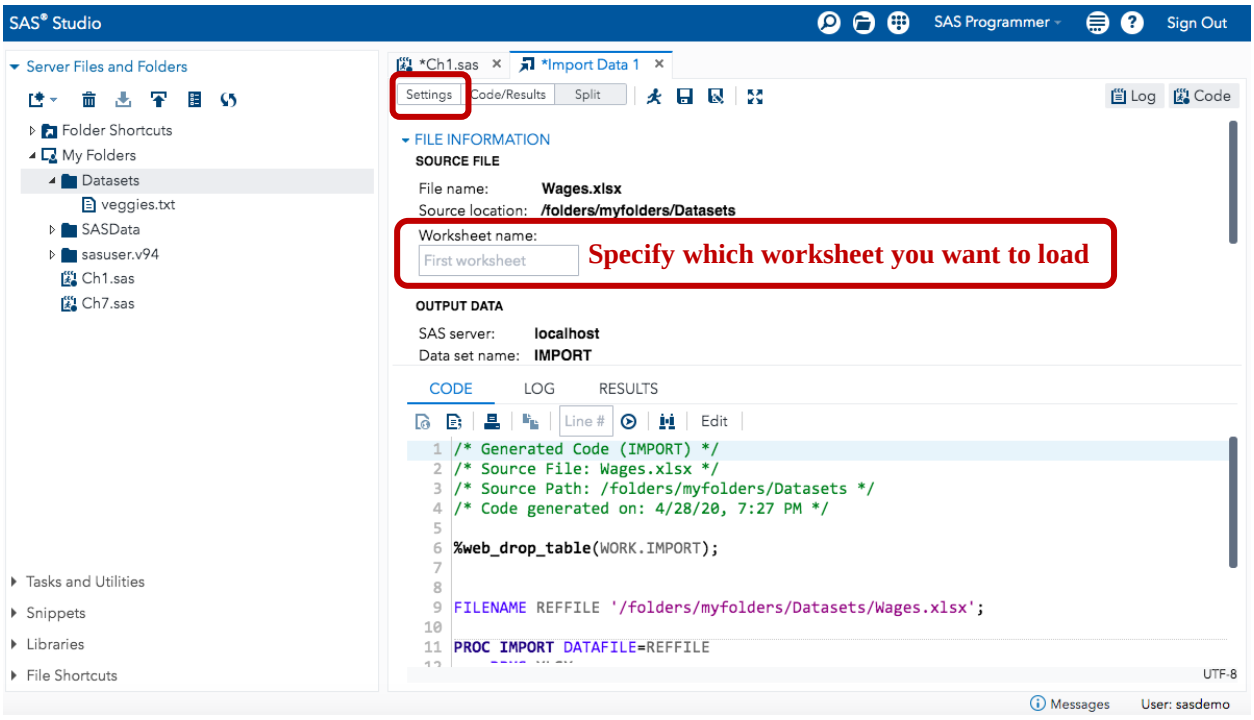
```
*Add LABEL and using SET *;
data sasdata.veggies3;
    set sasdata.veggies2;
    label name = "Vegetable name"
          code = "Product code"
          days = "Days to germination"
          number = "Number of seeds"
          price = "Retail price"
          costperseed = "Cost of seed";
run;
```

- SET statement: Instead of entering the data lines again, we can use the **SET** statement to retrieve the previously created SAS data set (veggies 2) and create a new SAS data set (veggies 3) with the labels. The SET statement is being used a lot in a DATA step.
- LABEL statement: It basically tells SAS to give specified labels to corresponding variables. Pay close attention to the position of semicolon.

b. Use Import Data to read Excel Spreadsheet

SAS can read data from almost any source. Most commonly used are text files and Excel spreadsheets. We just showed data import from a .txt file. Now let's see how to import an Excel spreadsheet. The easiest way is to use Import Data function in SAS Studio. Let's use dataset *Wages.xlsx*.

Import Data → Drag-n-drop or select your Excel data file → Specify worksheet name → Click on Settings.



- Change the OUTPUT DATA (Library: SASDATA; Data set: wages). We can leave the OPTIONS as default. Then click on Run.
- Next, look at Codes/Results → Check OUTPUT data to see if they were loaded correctly.
- Meanwhile, check CODE editor. It generated the SAS codes using PROC IMPORT. You can save it for a future use.

Note: The Import Data function is recommended as it is easy to use. It also works well with comma or tab delimited data files.

1.3 Two Commonly Used Procedures – PROC PRINT and PROC SORT

a. Statements in PROC PRINT

- Simply specify the dataset to be printed.
- In PROC PRINT, we can specify the display format in the output table. For example,

```
proc print data=sasdata.veggies3;
     format costperseed dollar.2;
run;
```

The FORMAT statement requests the dollar format to print costperseed variable, and the 2 following the period (.2) specifies 2 decimal places after the decimal point (or 2 digits). Here is what looks like in the output table.

Obs	name	code	days	number	price	costperseed
1	Cucumbe	50104-A	55	30	195	$6.50
2	Cucumber	51789-A	56	30	225	$7.50
3	Carrot	50179-A	68	1500	395	$0.26
4	Carrot	50872-A	65	1500	225	$0.15
5	Corn	57224-A	75	200	295	$1.48
6	Corn	62471-A	80	200	395	$1.98
7	Corn	57828-A	66	200	295	$1.48
8	Eggplant	52233-A	70	30	225	$7.50

Another common use is the display format for dates: **mmddyy** would give you 05/18/2020.

- Options in the PROC line: The NOOBS stops the observation numbers being printed.
- VAR statement specifies the variables that we want to print. If VAR statement is missing, all variables will be printed.
- WHERE statement allows us to print observation that meet a particular criterion.

SAS codes (Code editor):

```
proc print data=sasdata.veggies3 noobs;
     where costperseed > 1;
     format costperseed dollar.2;
     var name code costperseed;
run;
```

SAS output (Results window):

name	code	costperseed
Cucumbe	50104-A	$6.50
Cucumber	51789-A	$7.50
Corn	57224-A	$1.48
Corn	62471-A	$1.98
Corn	57828-A	$1.48
Eggplant	52233-A	$7.50

b. PROC SORT – sorts the data set by a variable

- BY statement specifies which variables to sort by. By default, it sorts in an ascending order. To sort in a descending order, use keyword DESCENDING.

Example I: sort data by cost of seed in a descending order.

```
*Uses of PROC SORT-- sort by one variable*;
proc sort data=sasdata.veggies3;
  by descending costperseed;
run;
```

Example II: sort data by cost of seed within each number of seed. Both are in descending order. In addition, we name the sorted dataset as veggies4 which is a new dataset in SASdata library.

```
*Uses of PROC SORT-- sort by one variable within another variable*;
proc sort data=sasdata.veggies3 out=sasdata.veggies4;
  by descending number descending costperseed;
run;
```

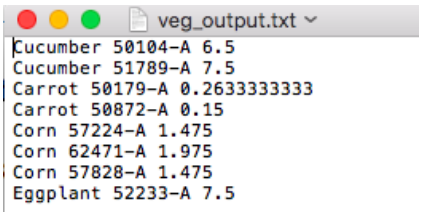
Check OUTPUT DATA to see the differences.

1.4 Export SAS Datasets – save a dataset outside of SAS

a. Save a dataset as a plain text file using **DATA** step.

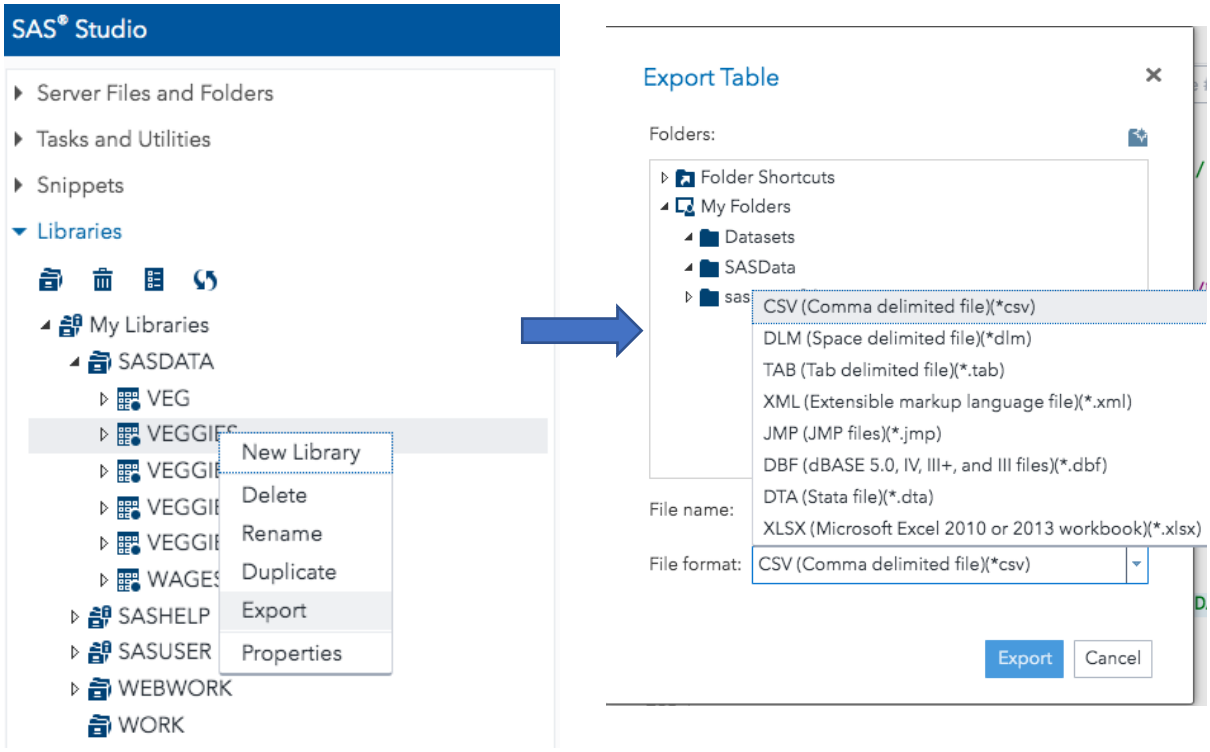
```
*Save the dataset as a plain text file using DATA step*;
data _null_;
   set sasdata.veggies;
   file "/folders/myfolders/Datasets/veg_output.txt";
   put name code costperseed;
run;
```

Saved .txt file:



b. Use EXPORT function in SAS Studio

In the navigation panel on the left, find a SAS dataset in the library. Right click on the dataset that you want to export. Then click on **Export**. An Export Table window pops up. Specify the folder to save the output data, provide a file name, and select the desired format for the output file. Next, click on Export to complete this procedure.



1.5 Final Comments

a. Data come in so many different formats. Getting data into SAS can be tricky at times. Learning different approaches for importing a dataset will guarantee a good start for running a SAS program.

b. **SAS names.** Simple naming rule: All SAS variable names and data set names can be no longer than 32 characters and must begin with a letter or the underscore (_) character. Dashes and spaces are not allowed. Here are a few invalid SAS names

Invalid SAS names	Reason
8_is_enough	Begins with a number
Price per pound	Contains blanks
Month-total	Contains invalid character (-)
Num%	Contains invalid character (%)