# SAS Tutorial: The SAS Data Step

This tutorial is meant to serve as a short introduction to the SAS data step. SAS operates in two primary operations – the SAS data step and the SAS procedure (or PROC).

Within this tutorial I will reference page numbers from the 5th edition of *The Little SAS Book* (LSB). You should read the noted pages so that you can obtain a better understanding of SAS. Note that not all of the discussion in LSB is directly relevant to our computing arrangement of using SAS OnDemand through JMP. **In particular students will never be able to use an INFILE statement to read in data. Students can only use data that the instructor has saved on the server, data read in using a CARDS or DATALINES statement, or data pushed up to the server through JMP.**

You can begin learning about the SAS data step by reading pp. 1-9 in LSB. You should also read all referenced pages in this tutorial. If you happen to have a different edition of LSB, then simply use the index. Each of these SAS topics can easily be found in the index, as can most other SAS topics since they are organized by keywords.

## Part 1: Creating SAS Data Sets Using the DATALINES Statement

We will begin by creating two data sets to use as examples by using the DATALINES statement. Note that in these two data steps we are performing a ***fixed width text read*** within a DATALINES statement. This type of data input requires that the spacing be exact as SAS will read the first three characters (and a space counts as a character) on the line and assign them to dimkey, the next four characters on the line and assign them to x, and the next four characters on the line and assign them to y. If you do not have the spacing exactly as it is presented in this example, then you will get an error message from SAS.

```
*************************************************************************;
* Creating SAS data sets using a DATALINES statement - see LSB p. 36. ;
*************************************************************************;
data temp1;
input dimkey $3. x 4.0  y 4.2 ;
* Note that the $ indicates that dimkey is a character variable and not
a numeric variable like x and y.  $, 8.0, and 8.2 are called "informats".
See pp. 46-47 in LSB for a list of SAS informats.;
* Note that the first entry must be all of the way to the left.;
* The code must be typed exactly like this - left justified with 1 space
between clumns;
* Note that DATALINES does not act consistently across platforms - PC versus
Linux;
datalines;
01 100 12.2
02 300 7.45
03 200 10.0
04 500 5.67
05 300 4.55
;
run;
```

```
data temp2;
input dimkey $3. z 4.0 first_name $9. last_name $10.;
* Note that for SAS to read this correctly, you will need to space this data
* out like a fixed width text file;
*23456789012345678901234567890;
datalines;
01 100 steve    miller
02 300 Steve    Utrup
04 500 JacK     wilsoN
05 300 AbRAham  LINcoln
06 100 JackSON  SmiTH
07 200 EarL     Campbell
08 400 WiLLiam  Right
;
run;
```

## Part 2: Print Your Data for Viewing

In JMP you will not be able to view the SAS data sets in your working directory. If you want to view them, then you will need to use the SAS procedure PROC PRINT.

```
*******************************************************************************;
* Print the data sets out for visual inspection;
* See LSB pp. 108-111;
*******************************************************************************;
title 'Data = temp1';
proc print data=temp1; run; quit;

title 'Data = temp2';
proc print data=temp2; run; quit;
```

| Obs | dimkey | x | y |
|---|---|---|---|
| 1 | 01 | 100 | 12.20 |
| 2 | 02 | 300 | 7.45 |
| 3 | 03 | 200 | 10.00 |
| 4 | 04 | 500 | 5.67 |
| 5 | 05 | 300 | 4.55 |

| Obs | dimkey | z | first_name | last_name |
|---|---|---|---|---|
| 1 | 01 | 100 | steve | miller |
| 2 | 02 | 300 | Steve | Utrup |
| 3 | 04 | 500 | JacK | wilsoN |
| 4 | 05 | 300 | AbRAham | LINcoln |
| 5 | 06 | 100 | JackSON | SmiTH |

| Obs | dimkey | z | first_name | last_name |
|---|---|---|---|---|
| 6 | 07 | 200 | EarL | Campbell |
| 7 | 08 | 400 | WiLLiam | Right |

## Part 3: Print a PROC CONTENTS

The standard method of listing out the column (or variable) names on a SAS data set and their data type is by using PROC CONTENTS.

```
*************************************************************************;
* Use PROC CONTENTS to list the metadata for the data sets;
* See LSB pp. 70-71;
*************************************************************************;
title ; * Reset the title statement to blank;
proc contents data=temp1; run; quit;
proc contents data=temp2; run; quit;
```

## Part 4: Manipulating SAS Data Sets

Here are some examples of some common manipulations that you will perform on SAS data sets and some examples of SAS character functions since our example data sets have string variables. Examples of SAS numeric functions can be found on pp. 80-81 of LSB.

```
*************************************************************************;
* Manipulate SAS data sets;
*************************************************************************;
data temp1;
      set temp1;
      w = 2*y + 1;

      * See LSB pp. 84-85;
      if (x < 150) then segment=1;
      else if (x < 250) then segment=2;
      else segment=3;
run;

data temp2;
      set temp2;
      * See pp. 78-79 for "Selected SAS Character Functions";
      proper_first_name = propcase(first_name);
      upper_last_name   = upcase(last_name);
      first_initial = substr(upcase(first_name),1,1);
      last_initial  = substr(upcase(last_name),1,1);
      initials = compress(first_initial||last_initial);
run;
```

```
title 'Data = temp1';
proc print data=temp1; run; quit;

title 'Data = temp2';
proc print data=temp2(obs=15); run; quit;
```

| Obs | dimkey | x | y | w | segment |
|---|---|---|---|---|---|
| 1 | 01 | 100 | 12.20 | 25.40 | 1 |
| 2 | 02 | 300 | 7.45 | 15.90 | 3 |
| 3 | 03 | 200 | 10.00 | 21.00 | 2 |
| 4 | 04 | 500 | 5.67 | 12.34 | 3 |
| 5 | 05 | 300 | 4.55 | 10.10 | 3 |

| Obs | dimkey | z | first_name | last_name | proper_first_name | upper_last_name | first_initial | last_initial | initials |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 01 | 100 | steve | miller | Steve | MILLER | S | M | SM |
| 2 | 02 | 300 | Steve | Utrup | Steve | UTRUP | S | U | SU |
| 3 | 04 | 500 | JacK | wilsoN | Jack | WILSON | J | W | JW |
| 4 | 05 | 300 | AbRAham | LINcoln | Abraham | LINCOLN | A | L | AL |
| 5 | 06 | 100 | JackSON | SmiTH | Jackson | SMITH | J | S | JS |
| 6 | 07 | 200 | EarL | Campbell | Earl | CAMPBELL | E | C | EC |
| 7 | 08 | 400 | WiLLiam | Right | William | RIGHT | W | R | WR |

## Part 5: Subsetting SAS Data Sets

Frequently you will not want all of the observations out of a SAS data set. Instead, you will only want a few of the observations based on a set of selection criteria. In this case you will want to subset your data. Follow this example and then print out the resulting data sets.

```
*************************************************************************;
* Subsetting Your Data;
* See LSB pp. 86-87, 328-330;
*************************************************************************;
data s1;
      set temp1;
      if (segment=1);
run;

data s2;
      set temp1;
      if (segment ne 1) then delete;
run;
```

```
* This will work, but it is not really proper;
data s3;
      set temp1;
      where (segment=1);
run;

* A where clause really belongs in the set statement;
data s4;
      set temp1 (where=(segment=1));
run;

* Verify that all of the data sets are the same;
title ; * Reset the title statement to blank;
proc print data=s1; run; quit;
proc print data=s2; run; quit;
proc print data=s3; run; quit;
proc print data=s4; run; quit;
```

**Part 6: Combining SAS Data Sets**

There are multiple ways to combine SAS data sets. In the most general sense you are either *stacking* the data sets or *merging* the data sets. The techniques that we will demonstrate here are traditional SAS techniques. If you happen to already know some SQL, then you could also perform these operations using traditional SQL language in a PROC SQL statement. Run the example code, print out the output, and compare the results. Merging data sets is a basic but important SAS operation that every SAS user needs to understand.

```
*************************************************************************;
* Stacking two data sets together;
* See LSB pp. 180-181;
*************************************************************************;
data stacked_data;
      set temp1 temp2;
run;

title 'Data = stacked_data';
proc print data=stacked_data; run; quit;


*************************************************************************;
* Creating an ordered stack of two data sets;
* See LSB pp. 182-183;
*************************************************************************;
* First sort the data by the dimkey;
proc sort data=temp1; by dimkey; run; quit;
proc sort data=temp2; by dimkey; run; quit;

* Now order the stack by including a BY statement;
data ordered_stack;
      set temp1 temp2;
      by dimkey;
run;
```

```
title 'Data = ordered_stack';
proc print data=ordered_stack; run; quit;

*************************************************************************;
* Combine the columns into one data step by merging the two data sets
* together.  See LSB pp. 184-187.;
*************************************************************************;
* Note that our data are already sorted.  Data must be sorted in order to
  be merged;

data merged_data;
      merge temp1 temp2;
      by dimkey;
run;

title 'Data = merged_data';
proc print data=merged_data; run; quit;


*************************************************************************;
* LEFT JOIN, RIGHT JOIN, and INNER JOIN using an IN statement;
* See LSB 200-201;
*************************************************************************;
* Note that our data are already sorted.  Data must be sorted in order to
  be merged;

* LEFT JOIN;
title 'LEFT JOIN OUTPUT';
data left_join;
      merge temp1 (in=a) temp2 (in=b);
      by dimkey;
      if (a=1);
run;

proc print data=left_join; run; quit;


* RIGHT JOIN;
title 'RIGHT JOIN OUTPUT';
data right_join;
      merge temp1 (in=a) temp2 (in=b);
      by dimkey;
      if (b=1);
run;

proc print data=right_join; run; quit;

* INNER JOIN;
title 'INNER JOIN OUTPUT';
data inner_join;
      merge temp1 (in=a) temp2 (in=b);
      by dimkey;
      if (a=1) and (b=1);
run;

proc print data=inner_join; run; quit;
```