

# SAS® Summer Training Institute

## Working with Data in SAS

Jason Wachsmuth

Research Data Analyst

Public Policy Center

[jason-wachsmuth@uiowa.edu](mailto:jason-wachsmuth@uiowa.edu)

While exploring whether vehicle sportiness has an impact on owner satisfaction, the goal of this session is to teach you multiple ways to:

- Read data into SAS
- Work with data
- Analyze data



# 3 methods to access data today include using the:

- DATALINES statement for inline data
- INFILE statement for external data
- SET statement for SAS data sets



## The DATALINES statement:

- Informs SAS that data lines follow
- Must be used with the INPUT statement

## The INPUT statement:

- Defines the arrangement of values
- Assigns values to variable names
- Defines the variable as character or numeric
- Syntax: INPUT variable <\$>

# Example:

```
data dogs;  
input breed $ class $ hi_weight;  
datalines;  
Pug Toy 18  
Beagle Hound 24  
Boxer Working 71  
;
```



# Results:

| VIEWTABLE: Work.Dogs |        |         |           |
|----------------------|--------|---------|-----------|
|                      | breed  | class   | hi_weight |
| 1                    | Pug    | Toy     | 18        |
| 2                    | Beagle | Hound   | 24        |
| 3                    | Boxer  | Working | 71        |

Exercise: Write code to read inline data. Copy the text from file x1.txt to use as input. Name the variables Make, Model, Type, Horsepower, and Weight. Call the output data set x1\_01.

```
data x1_01;  
  input make $ model $ type $ horsepower weight;  
  datalines;  
Acura MDX SUV 265 4451  
Buick Rainier SUV 275 4600  
Hummer H2 SUV 316 6400  
Infiniti FX35 Wagon 280 4056  
Infiniti FX45 Wagon 315 4309  
MINI Cooper Sedan 115 2524  
Pontiac Aztekt SUV 185 3779  
Pontiac Montana Sedan 185 3803  
Pontiac Vibe Wagon 130 2701  
Saturn VUE SUV 143 3381  
Scion xB Wagon 108 2425  
Subaru Baja Truck 165 3485  
Subaru Outback Wagon 165 3430  
Toyota RAV4 SUV 161 3119  
Toyota Tacoma Truck 142 2750  
Volvo V40 Wagon 170 2822  
Volvo XC70 Wagon 208 3823  
;
```

# The INFILE statement:

- Identifies the external file to read
- Must be used with the INPUT statement

```
data dogs_01;  
  infile "c:\data\in\dogs.txt";  
  input breed $ class $ hi_weight;  
run;
```





Exercise: Write code to read file x1.txt. Name the variables Make, Model, Type, Horsepower, and Weight. Call the output data set x1\_02.



```
data x1_02;  
  infile "c:\data\in\x1.txt";  
  input make $ model $ type $ horsepower weight;  
run;
```

A variety of options are available to use if data does not conform to the defaults. When:

- Character values are longer than 8 bytes then specify an informat with the colon modifier e.g., `:$30.`

- The delimiter is not a single blank

```
delimiter = ',' /*or ';' , '09'x, etc*/
```

- There are missing values

```
dsd
```

- There is a header row

```
firstobs=2
```

Exercise: Write code using modified list input and the appropriate infile options to read file x2.txt. Use the values found in the header row as variable names. Name the output x2.





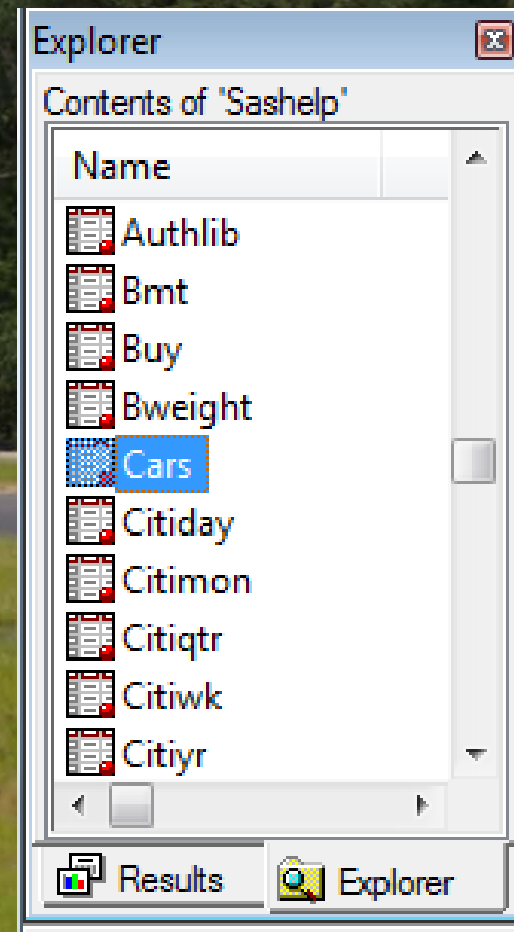
```
data x2;  
  infile "c:\data\in\x2.txt" delimiter=',' dsd firstobs=3;  
  input make :$30. model :$30. type $ horsepower weight;  
run;
```

The SET statement reads observations from a data set.

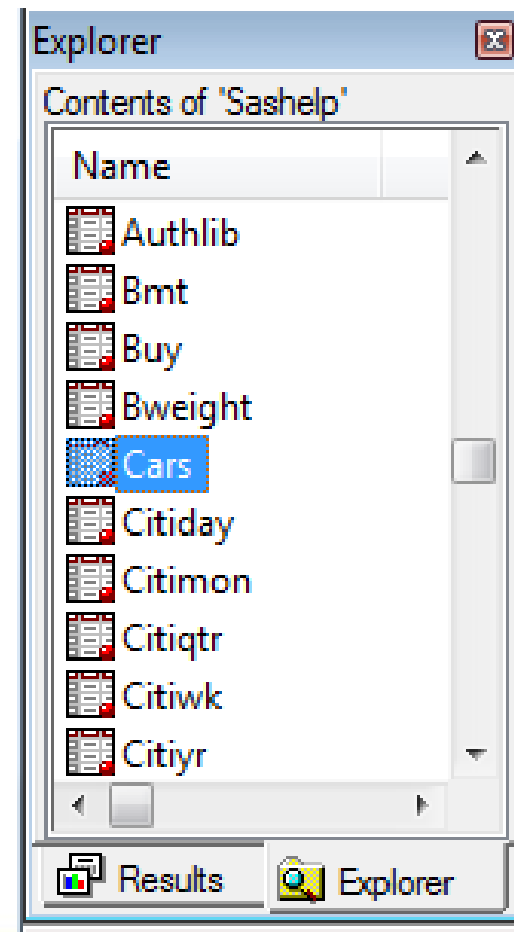
```
data dogs_02;  
set dogs_01;  
run;
```



Exercise: Write code to read the permanent data set highlighted in this Explorer window. Name the output cars\_01.



```
data cars_01;  
set sashelp.cars;  
run;
```





# Data can be modified using:

- Assignment
- IF-THEN/ELSE
- DO
- MERGE
- PROC SORT
- Functions
  - Numeric
  - Character
  - Date

Assignment statements are useful for creating new variables.

```
year = 2015;  
president = "Obama";  
weight_lb = weight_kg*2.2;
```

An IF-THEN statement executes a SAS statement for observations that meet specific conditions.

```
if year = 2015 then delete;  
if year = 2015;  
if year = 2015 then president = "Obama";  
if president = "Obama" then president = "Barack Obama";
```

# Use the optional ELSE statement for efficiency.

```
data cats_02;  
  set cats_01;  
  if type = "Rescue" then id = 1;  
    else if type = "Russian Blue" then id = 2;  
    else if type = "Maine Coon" then id = 3;  
    else id = .;  
run;
```



The DO statement specifies a group of statements to be executed as a unit.

```
data favs_02;  
set favs_01;  
  if make = "Lotus" then do;  
    model = "Exige S Coupe";  
    type = "Sports";  
    lbs2hp = 7.5;  
end;  
else if make = "Ford" and model = "" then do;  
  model = "Mustang Boss 302";  
  type = "Sports";  
  lbs2hp = 8.1;  
end;  
run;
```

# Exercise:

- Using the cars\_01 data set, reassign Model's value 'Aztekt' to 'Aztek'.
- Next, calculate the average MPG.
- Then, together we will calculate vehicle sportiness and output cars\_02.

```
/*  
- MPG_avg = (MPG_City + MPG_Highway)/2;  
- if LBS per Horsepower is lt 11, or  
- at least 100 Horsepower per L  
***/
```

```
data cars_02;  
  set cars_01;  
  if model = "Aztekt" then model = "Aztek";  
  lbs2hp = weight/horsepower;  
  if lbs2hp lt 11 then sporty = 'Y';  
  else if horsepower ge enginesize*100 then sporty = 'y';  
  else sporty = 'n';  
run;
```

The MERGE statement joins observations from two or more data sets into a single observation. Use a BY statement after a MERGE statement to match variables by specific values.

```
data dogs_03;  
merge dogs_02 owners;  
by id;  
run;
```





However, all data sets must be sorted by the variables used in the BY statement in separate steps before merging the data. This is done with the [SORT](#) procedure.

```
proc sort data = dogs_02;  
by id;  
run;
```

```
proc sort data = owners;  
by id;  
run;
```

# Example:

```
proc sort data = dogs_02;
```

```
by id;
```

```
run;
```

```
proc sort data = owners;
```

```
by id;
```

```
run;
```

```
data owners_dogs_01;
```

```
merge dogs_02 owners;
```

```
by id;
```

```
run;
```



Exercise: Merge the cars\_02 and satisfaction data sets by Make, Model, and Drivetrain. Name the output cars\_03.

```
libname in "c:\uisug\aug15\data\in";
```

```
proc sort data = cars_02;
```

```
by make model drivetrain;
```

```
run;
```

```
proc sort data = in.satisfaction;
```

```
by make model drivetrain;
```

```
run;
```

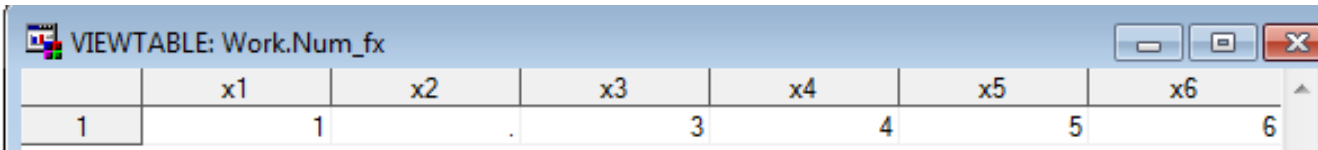
```
data cars_03;
```

```
merge cars_02 in.satisfaction;
```

```
by make model drivetrain;
```

```
run;
```

A function performs a computation or a manipulation on arguments and returns a value.



|   | x1 | x2 | x3 | x4 | x5 | x6 |
|---|----|----|----|----|----|----|
| 1 | 1  | .  | 3  | 4  | 5  | 6  |

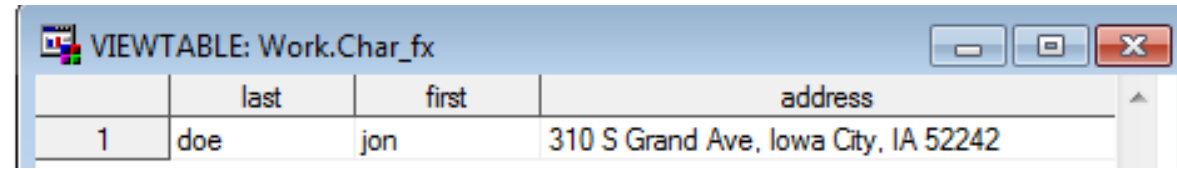
Statement

```
num = n(1, ., 3, 4, 5, 6);  
num = nmiss(of x1-x6);  
min = min(of x1-x6);  
max = max(11, of x1-x6, 7);  
tot = sum(of x1-x6);  
/*tot = x1+x2+x3+x4+x5+x6;  
avg = mean(of x1-x6);  
avg = int(mean(of x1-x6));  
avg = round(mean(of x1-x6));
```

Result

```
5  
1  
1  
11  
19  
.*/  
3.8  
3  
4
```

# Select character functions:



|   | last | first | address                              |
|---|------|-------|--------------------------------------|
| 1 | doe  | jon   | 310 S Grand Ave, Iowa City, IA 52242 |

## Statement

```
name = cat(last, first);  
name = cats(last, first);  
name = catx(' ', last, first);  
name = propcase(catx(' ', last, first));  
city = scan(address, 2, ' ');  
state = scan(scan(address, 3, ' '), 1, ' ');  
len = length(address);  
zip = substrn(address, len-5, len);
```

## Result

```
doe jon  
doejon  
doe, jon  
Doe, Jon  
Iowa City  
IA  
36  
52245
```

# Select date functions:

| Statement  | Result                |
|--|-----------------------|
| <code>sas_date = '31dec1999'd;</code>            | 14609                 |
| <code>sas_date = date(); *'18aug2015'd;</code>   | 20319                 |
| <code>year = year(date());</code>                | 2015                  |
| <code>year = substrn(year(date()), 3, 2);</code> | 15                    |
| <code>month = month(date());</code>              | 8                     |
| <code>week = week(date());</code>                | 33                    |
| <code>x = intnx('year', date(), 0);</code>       | 20089 /*Jan 1, 2015*/ |
| <code>y = intnx('year', date(), -1);</code>      | 19724 /*Jan 1, 2014*/ |
| <code>z = intnx('month', date(), 1);</code>      | 20332 /*Sep 1, 2015*/ |

## Exercise:

- Delete rows from the cars\_03 data set if there are any missing owner satisfaction ratings.
- Calculate the average owner satisfaction rating using the MEAN function; call the variable o\_avg.
- Create a variable named Vehicle that is the product of concatenating variables Make and Model and is space delimited.
- Name the output cars\_04.



```
data cars_04;  
set cars_03;  
if nmiss(of o1-o10) gt 0 then delete;  
o_avg = mean(of o1-o10); /*avg owner satisfaction*/  
vehicle = catx(' ', make, model);  
run;
```

Last, we will review the results using:

- The PRINT procedure
- The MEANS procedure

# Exercise: Review the top 30 vehicles with the highest owner satisfaction ratings.

- Sort the cars\_04 data set by descending ratings (so the highest rated vehicles appear on top), and output a data set named cars\_05.
- Output the first 30 observations using [PROC PRINT](#).

```
proc sort data = cars_04 out = cars_05;  
by descending o_avg;  
run;
```

```
proc print data = cars_05 (obs=30);  
var vehicle lbs2hp mpg_avg sporty o_avg type;  
run;
```

# Results show that although a sporty car is number 1, that is not the only factor that influences satisfaction.

| Obs | vehicle                                      | lbs2hp  | mpg_avg | sporty | o_avg | Type   |
|-----|--|---------|---------|--------|-------|--------|
| 1   | Mercedes-Benz SL55 AMG 2dr                   | 8.5903  | 17.5    | Y      | 9.8   | Sports |
| 2   | Toyota Prius 4dr (gas/electric)              | 26.2727 | 55.0    | n      | 9.8   | Hybrid |
| 3   | Honda Insight 2dr (gas/electric)             | 25.3425 | 63.0    | n      | 9.7   | Hybrid |
| 4   | Toyota Echo 2dr manual                       | 18.8426 | 39.0    | n      | 9.7   | Sedan  |
| 5   | Volkswagen Jetta GLS TDI 4dr                 | 30.0300 | 42.0    | n      | 9.7   | Sedan  |
| 6   | Dodge Viper SRT-10 convertible 2dr           | 6.8200  | 16.0    | Y      | 9.6   | Sports |
| 7   | Honda Civic Hybrid 4dr manual (gas/electric) | 29.3763 | 48.5    | n      | 9.6   | Hybrid |
| 8   | Toyota Corolla S 4dr                         | 19.4154 | 36.0    | n      | 9.6   | Sedan  |
| 9   | Toyota Echo 2dr auto                         | 19.3056 | 36.0    | n      | 9.6   | Sedan  |
| 10  | Toyota Echo 4dr                              | 19.0278 | 39.0    | n      | 9.6   | Sedan  |
| 11  | Honda Civic HX 2dr                           | 21.3675 | 40.0    | n      | 9.5   | Sedan  |
| 12  | Toyota Corolla CE 4dr                        | 19.2462 | 36.0    | n      | 9.5   | Sedan  |
| 13  | Toyota Corolla LE 4dr                        | 19.4154 | 36.0    | n      | 9.5   | Sedan  |
| 14  | Audi RS 6 4dr                                | 8.9422  | 18.5    | Y      | 9.4   | Sports |
| 15  | Mercedes-Benz SL600 convertible 2dr          | 8.9838  | 16.0    | Y      | 9.4   | Sports |
| 16  | Porsche 911 GT2 2dr                          | 6.5639  | 20.5    | Y      | 9.2   | Sports |
| 17  | Audi S4 Quattro 4dr                          | 11.2500 | 17.0    | n      | 8.9   | Sedan  |
| 18  | Saturn Ion1 4dr                              | 19.2286 | 30.5    | n      | 8.9   | Sedan  |
| 19  | Hyundai Elantra GT 4dr hatch                 | 19.5507 | 30.0    | n      | 8.8   | Sedan  |
| 20  | Pontiac Vibe                                 | 20.7769 | 32.5    | n      | 8.8   | Wagon  |
| 21  | BMW 545iA 4dr                                | 11.7354 | 22.0    | n      | 8.7   | Sedan  |
| 22  | BMW M3 convertible 2dr                       | 11.3544 | 19.5    | y      | 8.7   | Sports |
| 23  | Ford Mustang GT Premium convertible 2dr      | 12.8731 | 21.0    | n      | 8.7   | Sports |
| 24  | Honda Accord LX 2dr                          | 18.7125 | 30.0    | n      | 8.7   | Sedan  |
| 25  | Mercedes-Benz C32 AMG 4dr                    | 10.1433 | 18.5    | Y      | 8.7   | Sedan  |
| 26  | Mercedes-Benz CL600 2dr                      | 9.0730  | 16.0    | Y      | 8.7   | Sedan  |
| 27  | Pontiac GTO 2dr                              | 10.9559 | 18.0    | Y      | 8.7   | Sports |
| 28  | Scion xB                                     | 22.4537 | 33.0    | n      | 8.7   | Wagon  |
| 29  | Audi S4 Avant Quattro                        | 11.5765 | 18.0    | n      | 8.6   | Wagon  |
| 30  | Chevrolet Corvette convertible 2dr           | 9.2800  | 21.5    | Y      | 8.6   | Sports |

Exercise: Calculate the average owner satisfaction by mpg\_avg using the MEANS procedure.

```
proc sort data = cars_05;  
by mpg_avg;  
run;
```

```
proc means data = cars_05;  
var o_avg;  
by mpg_avg;  
run;
```

Exercise: To more efficiently look for potential owner satisfaction patterns, use the MEANS procedure within a macro.

```
%macro means(var1, var2);  
proc sort data = cars_05; by &var2.; run;  
  
proc means data = cars_05;  
var &var1.;  
by &var2.;  
run;  
%mend means;  
  
%means(o_avg,mpg_avg)  
%means(o_avg,origin)  
%means(o_avg,make)  
%means(o_avg,type)
```

# Congratulations on Completing:

## Working with Data in SAS