

PSTAT 130



SAS BASE PROGRAMMING

- Lecture 11 -

Objectives



- **OUTPUT Statement**
 - Create Multiple Records from a Single Observation
 - Write to Multiple SAS Data Sets
 - Create a Single Record from Multiple Observations
- **Create Running Totals**
 - SUM Statement
- **First./Last. variables**
 - With Single and Multiple BY variables
- **DROP= and KEEP=**
 - Limit the variables written to a SAS data set
- **FIRSTOBS= and OBS=**
 - Limit the number of observations read from a data set

Default Output



- By default, SAS writes a data record to the output data set when it reaches the RUN statement at the end of the DATA step.

2. Automatic
Return

```
data forecast;  
    set data2.growth;  
    <additional SAS statements>;  
run;
```

1. Automatic
Output

The OUTPUT Statement



- If you use an OUTPUT statement in the DATA step
 - SAS writes a record immediately
 - SAS DOES NOT write automatically at the end of the DATA step
- With an OUTPUT statement, you can
 - Create multiple records from one observation
 - Write to multiple output data sets
 - Combine information from multiple observations into a single record (when combined with RETAIN statement)

The OUTPUT Statement



- General form

```
OUTPUT <SAS-data-set-1.....SAS-data-set-n>;
```

Create Multiple Obs. Example



- Take each patient's current weight and forecast forward for the next five years
 - Assume 5% weight gain per year
 - Example: John Jones currently weighs 175 lbs. We project that he will weigh 5% more each year:

Year	Projected Weight
1	183.8
2	192.9
3	202.6
4	212.7
5	223.3

- Use the OUTPUT statement to write forecasted weight to the output data set for each of the next five years

Class Exercise 1



- Use the `admit` data set in the `data1` folder to create weight projections for the next 5 years.
 - Create a new data set called `work.weightproj`
 - Assume a 5% weight gain per year
 - Create the variable `year`
 - Calculate the variable `projweight` based on the previous weight and percent increase
 - Create a report from the `work.weightproj` data set
 - ✦ Suppress observation numbers
 - ✦ Display `name`, `year`, `projweight`
 - ✦ Display 'Patient Name', 'Year', and 'Projected Weight' as column headers

Desired Output



Patient Name	Year	Projected Weight
Murray, W	1	176.400
Murray, W	2	185.220
Murray, W	3	194.481
Murray, W	4	204.205
Murray, W	5	214.415
Almers, C	1	159.600
Almers, C	2	167.580
Almers, C	3	175.959
Almers, C	4	184.757
Almers, C	5	193.995
Bonaventure, T	1	129.150
Bonaventure, T	2	135.608
Bonaventure, T	3	142.388
Bonaventure, T	4	149.507
Bonaventure, T	5	156.983

Correct Log Output



NOTE: There were 21 observations read
from the data set DATA1.ADMIT.
NOTE: The data set WORK.WEIGHTPROJ has
105 observations and 12 variables.

Write to Multiple SAS Data Sets



- How can we use the `data1.admit` data set to create three separate output data sets?
 - Patients with Low activity level
 - Patients with Moderate activity level
 - Patients with High activity level

The DATA Statement



- General Form

```
DATA <data-set-name-1> ... <data-set-name-n>;
```

Name of the **FIRST**
data set being created
by this data step

Name of the **nth** data
set being created by
this data step

data1.admit



Name	Level	Sex	Age	Height	Weight
Murray, W	HIGH	M	27	72	168
Almers, C	HIGH	F	34	66	152
Bonaventure, T	LOW	F	31	61	123
Johnson, R	MOD	F	43	63	137
LaMance, K	LOW	M	51	71	158
Jones, M	HIGH	M	29	76	193
Reberson, P	MOD	F	32	67	151
King, E	MOD	M	35	70	173
Pitts, D	LOW	M	34	73	154
Eberhardt, S	LOW	F	49	64	172
Nunnelly, A	HIGH	F	44	66	140
Oberon, M	LOW	F	28	62	118
Peterson, V	MOD	M	30	69	147
Quigley, M	HIGH	F	40	69	163
Cameron, L	MOD	M	47	72	173
Underwood, K	LOW	M	60	71	191
Takahashi, Y	MOD	F	43	65	123

Class Exercise 2



- Use `data1.admit`
 - Create the data sets: `work.low`, `work.mod`, `work.high`
 - Output each record to the appropriate data set
 - Create a report for `work.low` displaying Name, Level, Sex, Height, and Weight. Suppress the observation numbers.
- Desired output: (`work.low`)

Name	Level	Sex	Age	Height	Weight
Bonaventure, T	LOW	F	31	61	123
LaMance, K	LOW	M	51	71	158
Pitts, D	LOW	M	34	73	154
Eberhardt, S	LOW	F	49	64	172
Oberon, M	LOW	F	28	62	118
Underwood, K	LOW	M	60	71	191
Ivan, H	LOW	F	22	63	139

RETAIN Statement



- By default, SAS sets the values of all variables to be missing at the start of each iteration of the DATA step.
- Use the RETAIN statement to keep the value of a calculated variable available for use in the next iteration.
- General Form:

```
RETAIN variable-1 < . . . variable-n>;
```

Example



- Patients sign up for a six-month weight loss program. They are weighed at the start of the program, and then at their Month 3 and Month 6 visits. The data set contains the weight and the date for each of the visits.
- We are interested in calculating how much weight each patient has lost since the last visit.

Class Exercise 3



- Create a data set called `work.visits` with the following data:

name	visit	weight
john	01/01/08	210
john	04/04/08	199
john	07/03/08	183
mary	02/14/08	175
mary	05/16/08	167
mary	08/04/08	153
dave	09/23/08	223
dave	12/28/08	215
dave	03/24/09	206

Class Exercise 3 - continued



- Create a data set called `work.weightloss` that reads in the `visits` data set
 - Calculate the `weightchange` using the last weight and current weight
- Desired output (v. 1):

Patient Name	Visit Date	Visit Weight	Change in Weight	Last Weight
john	01/01/08	210	.	210
john	04/04/08	199	11	199
john	07/03/08	183	16	183
mary	02/14/08	175	8	175
mary	05/16/08	167	8	167
mary	08/04/08	153	14	153
dave	09/23/08	223	-70	223
dave	12/28/08	215	8	215
dave	03/24/09	206	9	206

First.BY-Variable & Last.BY-Variable



- When a SORTED data set is read into DATA step using the SET and BY statements
 - Two variables are created that can be used to identify the First record in the BY group and the Last record in the BY group.
 - These variables are only available for the duration of the DATA step.
- General Form

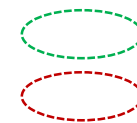
<i>First.BY-variable</i> <i>Last.BY-variable</i>

Class Exercise 4



- Now modify your data step for `weightloss` such that
 - It creates a new `baseweight` for each person
 - It calculates the difference between `baseweight` and `weight`
- Desired Output (v. 2):

Subject Name	Visit Date	Visit Weight	Baseline Weight	Change in Weight
dave	09/23/08	223	223	0
dave	12/28/08	215	223	8
dave	03/24/09	206	223	17
john	01/01/08	210	210	0
john	04/04/08	199	210	11
john	07/03/08	183	210	27
mary	02/14/08	175	175	0
mary	05/16/08	167	175	8
mary	08/04/08	153	175	22



First.Name

Last.Name

One Record from Multiple Obs.



- Can use the RETAIN and OUTPUT statements, and First./Last. variables, to combine information from multiple data records, and write only one “summary” record

Class Exercise 5



- Modify your data step for `weightloss` such that it only displays the total weight change for each person
- Desired Output (v. 3):

Subject Name	Visit Date	Visit Weight	Baseline Weight	Change in Weight
dave	03/24/09	206	223	17
john	07/03/08	183	210	27
mary	08/04/08	153	175	22

Create a Running Total



- There are a few methods for creating a running total.
- Create a running total (`SaleRT`) for the variable `SaleAmt` located in `data2.daysales`
 - First, create a data set named `SaleTotal1`
 - ✦ Use the addition (+) operator
 - Next, create a data set named `SaleTotal2`
 - ✦ Use the `sum()` function

Running Total Example



```
data SaleTotal1;  
  set data2.daysales;  
  if _n_=1 then SaleRT = 0;  
  SaleRT = SaleRT + SaleAmt;  
  retain;  
run;
```

Successfully
creates a
running total



```
data SaleTotal2;  
  set data2.daysales;  
  SaleRT = sum(SaleRT, SaleAmt);  
  retain;  
run;
```

Successfully
creates a
running total



The SUM Statement



- The SUM statement (General Form)

$$\textit{Variable} + \textit{Expression};$$

- Creates the variable on the left side of the plus sign, if it does not already exist
- Initializes the variable to zero before the first iteration of the DATA step
- Adds the value of the *expression to the variable at execution*
- Automatically retains the variable
- Ignores missing values

Running Total Example



```
data SaleTotal13;  
  set data2.daysales;  
  SaleRT+SaleAmt;  
run;
```

Accumulating
Variable

SaleDate	SaleAmt	SaleRT
01APR2001	498.49	498.49
02APR2001	946.50	1444.99
03APR2001	994.97	2439.96
04APR2001	564.59	3004.55
05APR2001	783.01	3787.56
06APR2001	228.82	4016.38
07APR2001	930.57	4946.95
08APR2001	211.47	5158.42
09APR2001	156.23	5314.65

Accumulate a Total for BY Groups



- We can accumulate a total for each group of values.
Note: The input data set must be pre-sorted on the BY variable(s).
1. Set the Accumulator Variable to zero at the start of each BY group.
 2. Increment the accumulating variable with a sum statement.
 3. Output only the last observation of each BY group.

Accumulated Totals Example



- Use the `regsals` data set in the `data2` folder
 - Create a data set named `divsalaries` that only contains the variables `div` and `divsal`
 - Calculate `divsal` to be the accumulated total for each division

Accumulated Total Example



```
proc sort data=data2.regsals out=divsort;  
  by Div;  
run;  
data divsalaries;  
  set divsort;  
  by Div;  
  if First.Div then DivSal=0;  
  DivSal+Salary;  
  if Last.Div;  
  keep Div DivSal;  
run;
```



Obs	Div	DivSal
1	APTOS	310000
2	FINACE	163000
3	FLTOPS	318000
4	HUMRES	281000
5	SALES	373000

More First./Last. Variables

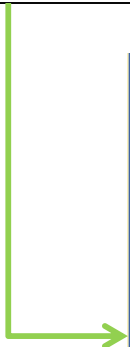


- What if we are now interested in grouping by region as well as division?
- We can create multiple First./Last. variables by specifying multiple BY-variables in our DATA step.

More First./Last. Variables



```
proc sort data=data2.regsals out=regsort;  
  by Region Div;  
run;  
data regtotal;  
  set regsort;  
  by Region Div;  
run;
```



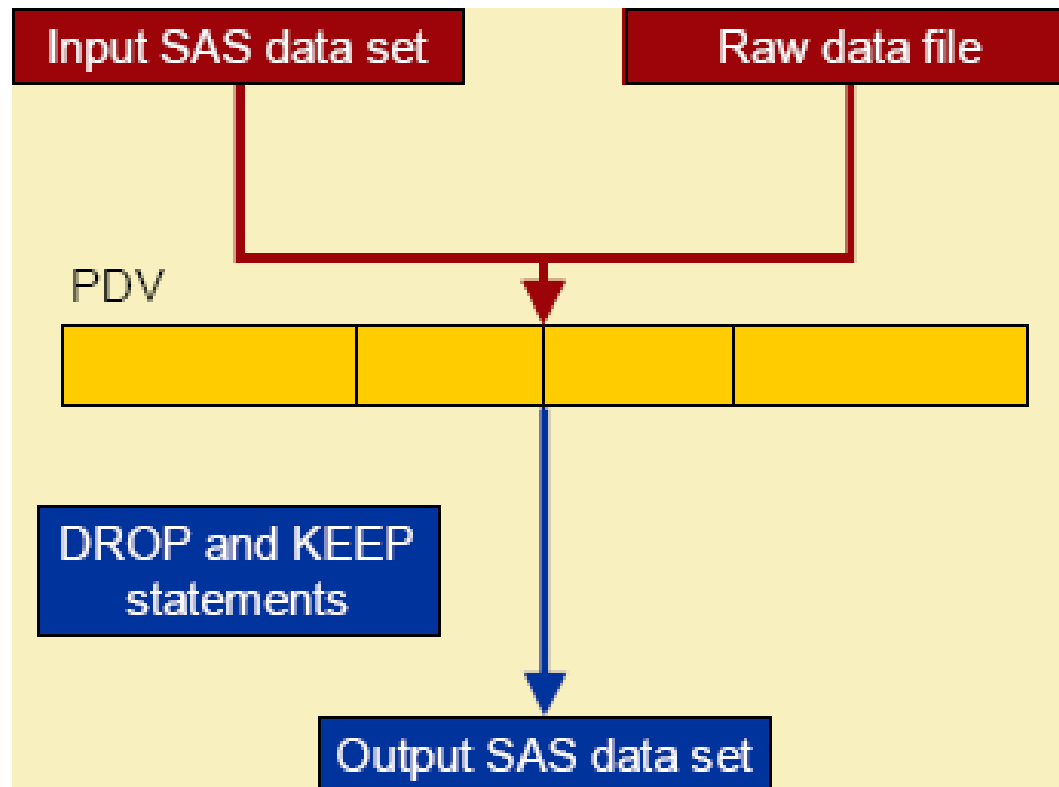
Region	Div	First. Region	Last. Region	First. Div	Last.Div
C	APTOPS	1	0	1	0
C	APTOPS	0	1	0	1
E	APTOPS	1	0	1	0
E	APTOPS	0	0	0	0
E	APTOPS	0	0	0	1
E	FINACE	0	0	1	0

Control Variable Output



- In the DATA step:
 - By default, SAS writes all the variables from the input data set(s) to every output data set.
 - The DROP and KEEP statements can be used to control which variables are written to every output data sets.

The DROP and KEEP Statements



DROP= & KEEP= Data Set Options



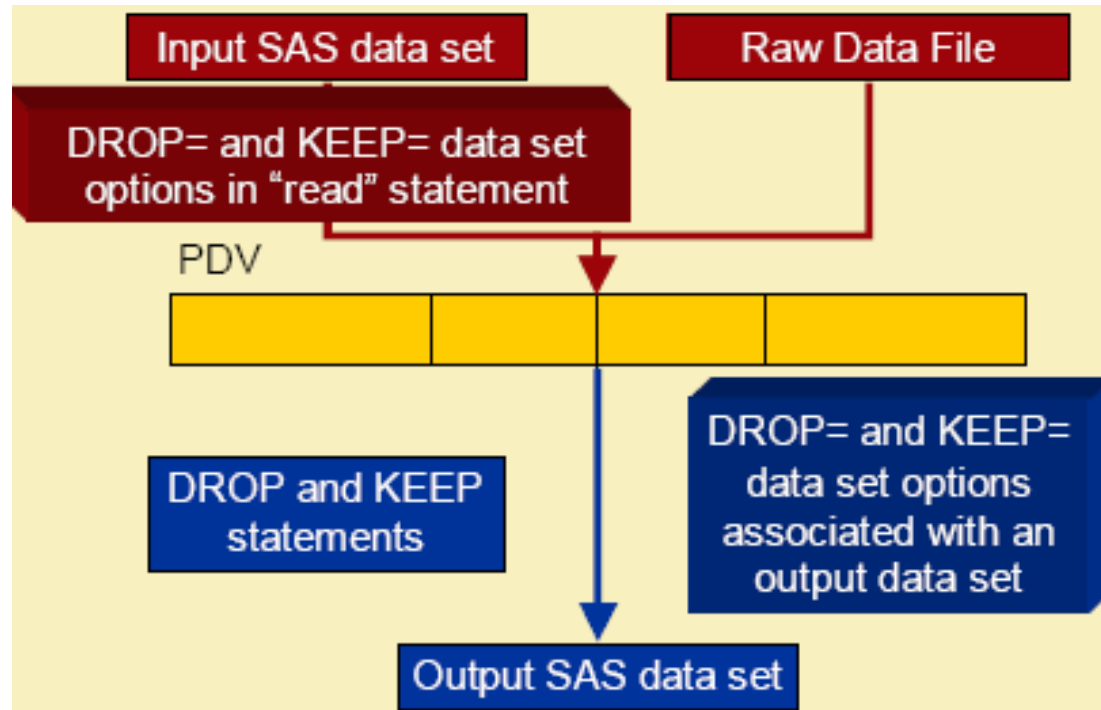
- **DROP=**

```
SAS-data-set (DROP=variable-1  
                variable-2  
                ...  
                variable-n)
```

- **KEEP=**

```
SAS-data-set (KEEP=variable-1  
              variable-2  
              ...  
              variable-n)
```

Controlling Variable Input and Output



Control Variable Input and Output



```
data army(keep=Code Airport);  
    set data2.military(drop=City State Country);  
    if Type eq 'Army' then output;  
run;
```

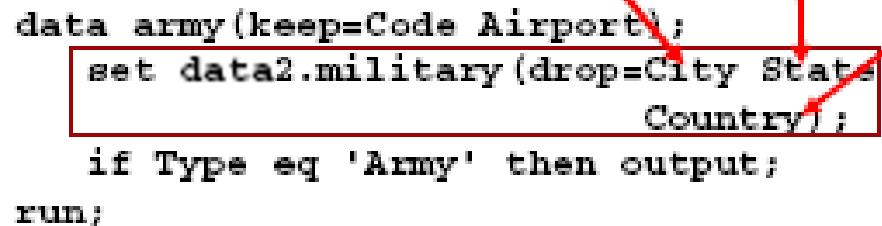
Control Variable Input



Variables

Type	Code	City	State	Country	Airport
------	------	------	-------	---------	---------

```
data army(keep=Code Airport);  
set data2.military(drop=City State  
Country);  
if Type eq 'Army' then output;  
run;
```

A diagram illustrating the SAS code. A red box highlights the 'drop=City State Country' part of the 'set data2.military' statement. Red arrows point from the 'City', 'State', and 'Country' headers in the 'Variables' table to the corresponding variables in the red box. Another red arrow points from the 'Code' header to the 'keep=Code' part of the 'data army' statement. A third red arrow points from the 'Airport' header to the 'keep=Code Airport' part of the 'data army' statement.

PDV

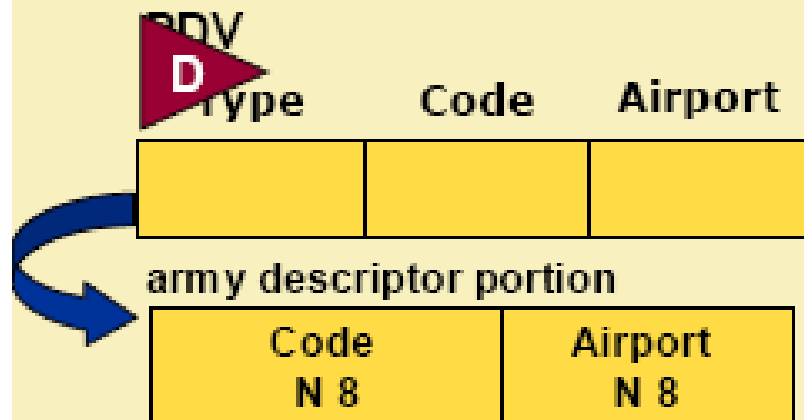
Type	Code	Airport

Control Variable Output



Type	Code	City	State	Country	Airport
------	------	------	-------	---------	---------

```
data army(keep=Code Airport);  
  set data2.military(drop=City State  
                     Country);  
  if Type eq 'Army' then output;  
run;
```



Control Which Obs. are Read



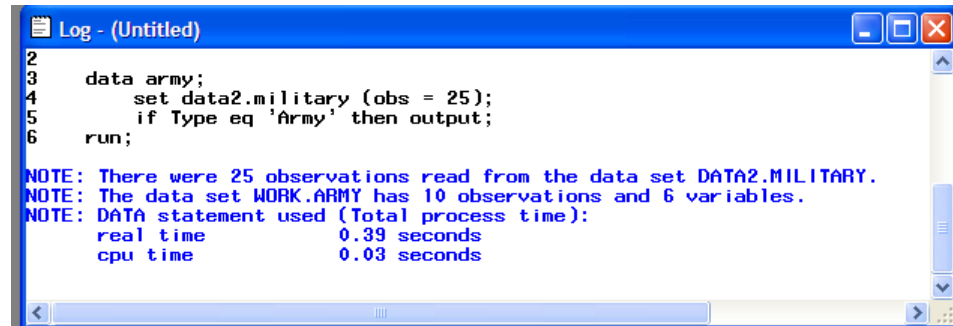
- The OBS= data set option specifies the last observation to be read from the input data set
- General Form:

```
SAS-data-set (OBS=n)
```

- Example:

```
data army;  
    set data2.military(obs=25);  
    if Type eq 'Army' then output;  
run;
```

Control Which Obs. are Read



The screenshot shows a SAS Log window titled "Log - (Untitled)". It contains the following SAS code and output:

```
2  
3   data army;  
4       set data2.military (obs = 25);  
5       if Type eq 'Army' then output;  
6   run;
```

NOTE: There were 25 observations read from the data set DATA2.MILITARY.
NOTE: The data set WORK.ARMY has 10 observations and 6 variables.
NOTE: DATA statement used (Total process time):
 real time 0.39 seconds
 cpu time 0.03 seconds

Control Which Obs. are Read



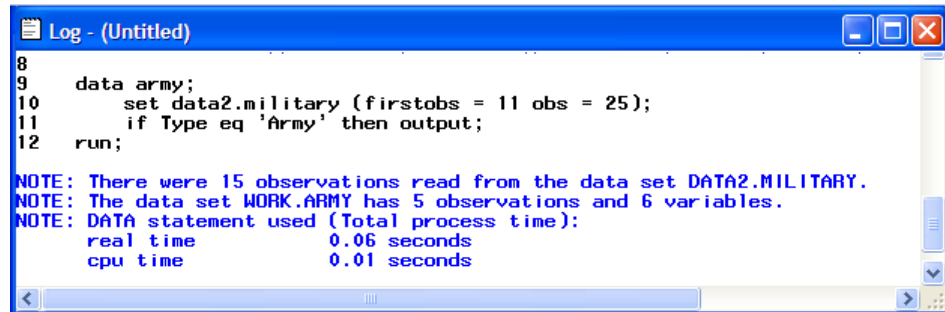
- The FIRSTOBS= data set option allows you to skip records at the top of the input file.
- General Form:

```
SAS-data-set (FIRSTOBS=n)
```

- Example (starts reading at 11th record):

```
data military;  
    set data2.military(firstobs=11 obs=25) ;  
    if Type eq 'Army' then output;  
run;
```


Control Which Obs. are Read

A screenshot of a SAS Log window titled "Log - (Untitled)". The window has a blue title bar and standard Windows window controls. The log text is as follows:

```
8  
9   data army;  
10      set data2.military (firstobs = 11 obs = 25);  
11      if Type eq 'Army' then output;  
12  run;  
  
NOTE: There were 15 observations read from the data set DATA2.MILITARY.  
NOTE: The data set WORK.ARMY has 5 observations and 6 variables.  
NOTE: DATA statement used (Total process time):  
      real time           0.06 seconds  
      cpu time            0.01 seconds
```