

# The Use and Abuse of the Program Data Vector

Jim Johnson  
Principal Scientific Programmer



- A Few Comments
- Program Data Vector
  - What it is
  - What it contains
  - How it works
  - Tools and examples
- Use and Abuse
  - Tools and examples



- All examples use SAS version 8.2
- “A hundred different ways to do everything in SAS”
- Assume a RUN; statement in examples



## PROGRAM DATA VECTOR

- Storage place in memory for all variables encountered in the data step
- Variables may contain indicators flagging them to be kept, dropped, retained, or renamed



## PROGRAM DATA VECTOR

- user variables
- automatic variables
- temporary variables



## AUTOMATIC VARIABLES

- \_ERROR\_ : indicates if an error has occurred in the data step.
- \_N\_ : number of times the data step has executed.



## TEMPORARY VARIABLES

- *first.BY-variable* and *last.BY-variable*
  - always occur in pairs
  - one pair for every BY-variable in the BY statement
  - value: 1= true; 0=false



## TEMPORARY VARIABLES

- *IN=variable*
  - user specified variable name
  - indicates if a specific data set contributed to the current observation
  - value: 1=true; 0=false



## TEMPORARY VARIABLES

- *END=variable*
  - user specified variable name
  - indicates the end of the input data has been reached
  - value: 1=true; 0=false



## Automatic and temporary variables

- not written to output data set
- cannot be kept, dropped or renamed
- user defined variables can be assigned the value of automatic and temporary variables



## PROGRAM DATA VECTOR

- used to populate the output data set
- all variables in the output data set come from program data vector
- not all variables on program data vector are written to the output data set



## TOOLS TO MANIPULATE THE PROGRAM DATA VECTOR

- drop
- keep
- rename
- where



## SAS STATEMENTS versus SAS DATA SET OPTIONS



## SAS STATEMENTS

- appear in data step code
- actions are more global
- take affect at different times
- may have different effects



## SAS STATEMENTS

- drop age sex race;
- rename (first=alpha);
- where region = 'LRQ';



## SAS DATA SET OPTIONS

- apply to a specific data set
- used on
  - input data sets
  - output data sets



## SAS DATA SET OPTIONS

- specified in parentheses following data set name to which it applies
- equal sign (=) and option details follow option name



## SAS DATA SET OPTIONS

- data demog(drop=age sex race);
- set source(rename=(first=alpha));
- set body(where=(region = 'LRQ'));



## WHERE SPECIFIED

- Statements
  - anywhere in the SAS data step
- Data set options
  - only in the DATA statement or input statements such as SET, MERGE, or UPDATE



## HOW GLOBALLY THEY APPLY

- Statements
  - all data sets used in the data step
- Data set options
  - only to the data set to which they are attached



## WHEN THEY TAKE AFFECT

- DROP, KEEP, and RENAME statements
  - as the data is written to the output data set
- WHERE statement
  - as the data is read from the input data set



## WHEN THEY TAKE AFFECT

- Data set options
  - when specified on an input data set: as the data is read into the program data vector
  - when specified on an output data set: as the data is written from the program data vector



## DROP STATEMENT

- Indicates variables in the program data vector NOT to be written to the output data set
- Takes affect as the observation is written
- Can be specified anywhere in the data step



## DROP STATEMENT

- One DROP statement applies to all output data sets
- Multiple DROP statements can be used
- All DROP statements apply to all output data sets



## DROP STATEMENT

- Variables listed on DROP statement are available for use in the data step until the observation is output



## DROP= DATA SET OPTION

- When used on input data sets
  - specifies variables not to be read into the program data vector
- When used on output data sets
  - specifies variables not to be written from the program data vector



## DROP= DATA SET OPTION

- Must be specified for each data set to which it applies
- Multiple DROP= data set options may be specified for one data set



## KEEP STATEMENT

- Indicates variables in the program data vector that should be written to the output data set
- Takes affect as the observation is written
- Can be specified anywhere in the data step



## KEEP STATEMENT

- One KEEP statement applies to all output data sets
- Multiple KEEP statements can be used
- All KEEP statements apply to all output data sets



## KEEP STATEMENT

- Variables **NOT** listed on KEEP statement are available for use in the data step until the observation is output



## KEEP= DATA SET OPTION

- When used on input data sets
  - specifies variables to be read into the program data vector
- When used on output data sets
  - specifies variables to be written from the program data vector



## KEEP= DATA SET OPTION

- Must be specified for each data set to which it applies
- Multiple KEEP= data set options may be specified for one data set



## KEEP STATEMENT used with KEEP= DATA SET OPTION

- If the KEEP= data set option is used on input
  - KEEP= data set option functions on input
  - KEEP statement functions on output



## KEEP STATEMENT used with KEEP= DATA SET OPTION

- If the KEEP= data set option is used on output
  - KEEP statement takes affect first
  - KEEP= data set option may cause data to be lost



```
data one;  
  a=1; b=2; c=3; output;  
run;
```

```
data two(keep=a c);  
  set one;  
  keep c;  
run;
```

WARNING: The variable a in the DROP, KEEP, or RENAME list has never been referenced.

NOTE: The data set WORK.TWO has 1 observations and 1 variables.



When both DROP and KEEP  
statements are used,

OR

When both DROP= and KEEP= data  
set options are used,

- The order does not matter
- DROP executes first



```

data one;
  a=1; b=2; output;
run;

data two;
  set one(drop=a keep=a);
run;

```

NOTE: The data set WORK.TWO has 1 observations and **0 variables**.



```

data two;
  set one;
  drop a;
  keep a;
run;

```

WARNING: The variable a in the DROP, KEEP, or RENAME list has never been referenced.  
NOTE: The data set WORK.TWO has 1 observations and **0 variables**.



```

data one;
  a=1; b=2; output;
  a=2; b=2; output;
run;

```

```

data two;
  set one(drop=a);
  if a=1;
run;

```

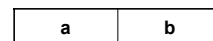
NOTE: Variable a is uninitialized.  
NOTE: The data set WORK.TWO has **0 observations** and **2 variables**.



```

data one;
  a=1; b=2; output;
  a=2; b=2; output;
run;

```



```

data two;
  set one(drop=a);
  if a=1;
run;

```

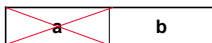
NOTE: Variable a is uninitialized.  
NOTE: The data set WORK.TWO has 0 observations and 2 variables.



```

data one;
  a=1; b=2; output;
  a=2; b=2; output;
run;

```



```

data two;
  set one(drop=a);
  if a=1;
run;

```

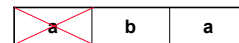
NOTE: Variable a is uninitialized.  
NOTE: The data set WORK.TWO has 0 observations and 2 variables.



```

data one;
  a=1; b=2; output;
  a=2; b=2; output;
run;

```



```

data two;
  set one(drop=a);
  if a=1;
run;

```

**NOTE: Variable a is uninitialized.**  
NOTE: The data set WORK.TWO has 0 observations and 2 variables.



## WHERE STATEMENT

- Indicates a subset of observations
- Takes affect as the observation is read from the input data set
- If an observation does not meet the WHERE condition, it is never read into the program data vector



## WHERE STATEMENT

- Can be specified anywhere in the data step
- One WHERE statement applies to all input data sets
- When multiple WHERE statements are specified, the last one will be used.



```
data one;  
  a=1; b=2; output;  
  a=2; b=2; output;
```

```
data two;  
  set one;  
  where b=2;  
  where a=1;  
run;
```

NOTE: Where clause has been replaced.  
NOTE: The data set WORK.TWO has 1 observations and 2 variables.



## WHERE= DATA SET OPTION

- When used on input data sets
  - subsets data as it is read into the program data vector
- When used on output data sets
  - subsets data as it is written from the program data vector



## WHERE= DATA SET OPTION

- Must be specified for each data set to which it applies
- Only one WHERE= data set option can be used with any one data set



## WHERE STATEMENT used with WHERE DATA SET OPTION

- For input data sets
  - WHERE= data set option will be used
  - WHERE statement will be ignored
  - Data sets not using the WHERE= data set option will be subject to the WHERE statement



```
data two;
  set one(where=(a=1));
  where b=2;
run;
```

WARNING: The WHERE statement cannot be applied to a data set in the last SET/MERGE/UPDATE/MODIFY statement. Either the data sets listed failed with open errors or they already specify a WHERE data set option.



## WHERE STATEMENT used with WHERE DATA SET OPTION

- For output data sets
  - WHERE statement will be used as the data is READ into the program data vector
  - WHERE= data set option will be used as the data is WRITTEN from the program data vector



```
data one;
  a=1; b=2; output;
  a=2; b=2; output;
  a=1; b=3; output;

data two(where=(a=1));
  set one;
  where b=2;
run;
```

NOTE: The data set WORK.TWO has 1 observations and 2 variables.



## WHERE versus SUBSETTING IF



- WHERE statement is executed as observations are read. Observations not meeting the condition are never read into the program data vector
- Subsetting IF is executed **after** the observation reaches the program data vector



- WHERE used with BY
  - WHERE is executed first
- Subsetting IF used with BY
  - BY statement is executed first



- WHERE with MERGE
  - WHERE is executed first
- Subsetting IF with MERGE
  - MERGE is executed first

## RENAME STATEMENT

- Specifies variables to be renamed and their new names
- Takes affect as the observation is written to the output data set
- Can be specified anywhere in the data step

## RENAME STATEMENT

- One RENAME statement applies to all output data sets
- Multiple RENAME statements can be used
- All RENAME statements will apply to all output data sets

## RENAME STATEMENT

- The program data vector contains the variables with their OLD names, therefore, programs should continue to use the OLD variable names for processing in the current data step

## RENAME= DATA SET OPTION

- When used on input data sets
  - renames variables as data is read into the program data vector
  - program data vector contains variables by the NEW names. Programs should use NEW variable names.

## RENAME= DATA SET OPTION

- When used on output data sets
  - renames variables as data is written from the program data vector
  - program data vector contains variables by the OLD names. Programs should use OLD variable names.

## RENAME= DATA SET OPTION

- Must be specified for each data set to which it applies
- Multiple RENAME= data set options can be used with any one data set

## HIERARCHY

1. DROP
2. KEEP
3. RENAME

(subject to timing described previously)

```
data one;
  a=1; b=3; c=3; output;
  a=2; b=1; c=3; output;
  a=2; b=2; c=3; output;
  a=3; b=2; c=3; output;
  a=3; b=3; c=3; output;
run;
```

```
data two(keep=new_code rank a b
          rename=(rank=index));
  set one(rename=(c=code1)) end=last;
  by b;
  drop b;
  rank=b;
  where a=2;
  rename code1=new_code;
  keep code1 rank a b;
run;
```

```
data two(keep=new_code rank a b
          rename=(rank=index));
  set one(rename=(c=code1)) end=last;
  by b;
  drop b;
  rank=b;
  where a=2;
  rename code1=new_code;
  keep code1 rank a b;
run;
```

last	FIRST.b	LAST.b	_ERROR_	_N_
------	---------	--------	---------	-----

```
data two(keep=new_code rank a b
          rename=(rank=index));
  set one(rename=(c=code1)) end=last;
  by b;
  drop b;
  rank=b;
  where a=2;
  rename code1=new_code;
  keep code1 rank a b;
run;
```

a	b	code1
---	---	-------

```

data two(keep=new_code rank a b
         rename=(rank=index));
set one(rename=(c=code1)) end=last;
by b;
drop b;
rank=b;
where a=2;
rename code1=new_code;
keep code1 rank a b;
run;

```

a	b	code1
---	---	-------



```

data two(keep=new_code rank a b
         rename=(rank=index));
set one(rename=(c=code1)) end=last;
by b;
drop b;
rank=b;
where a=2;
rename code1=new_code;
keep code1 rank a b;
run;

```

a	b	code1	rank
---	---	-------	------



```

data two(keep=new_code rank a b
         rename=(rank=index));
set one(rename=(c=code1)) end=last;
by b;
drop b;
rank=b;
where a=2;
rename code1=new_code;
keep code1 rank a b;
run;

```

a	<del>b</del>	code1	rank
---	--------------	-------	------



```

data two(keep=new_code rank a b
         rename=(rank=index));
set one(rename=(c=code1)) end=last;
by b;
drop b;
rank=b;
where a=2;
rename code1=new_code;
keep code1 rank a b;
run;

```

a	<del>b</del>	code1	rank
---	--------------	-------	------



```

data two(keep=new_code rank a b
         rename=(rank=index));
set one(rename=(c=code1)) end=last;
by b;
drop b;
rank=b;
where a=2;
rename code1=new_code;
keep code1 rank a b;
run;

```

a	<del>b</del>	new_code	rank
---	--------------	----------	------



```

data two(keep=new_code rank a b
         rename=(rank=index));
set one(rename=(c=code1)) end=last;
by b;
drop b;
rank=b;
where a=2;
rename code1=new_code;
keep code1 rank a b;
run;

```

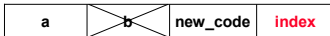
a	<del>b</del>	new_code	rank
---	--------------	----------	------



```

data two(keep=new_code rank a b
rename=(rank=index));
set one(rename=(c=code1)) end=last;
by b;
drop b;
rank=b;
where a=2;
rename code1=new_code;
keep code1 rank a b;
run;

```



a	new_code	index
2	3	1
2	3	2



## RETAIN STATEMENT

- Specifies variables not to be set to missing at the beginning of each iteration of the data step
- Only variables NOT in the input data sets can be retained
- Retaining a variable from an input data set has no effect



## RETAIN STATEMENT

- Variables created with a sum statement are automatically retained
- Subsequent data sets in the SET statement will reset all non-retained variables to missing
- Retained variables with no initial value, and no value from processing, will not be written to the output data set



```

data one;
a=1; b=1; c=0; output;
a=1; b=2; c=0; output;

```

```

data two;
a=2; d=3; output;
a=2; d=4; output;

```

```

data three;
set one two;
retain b 0 e;
run;

```



```

data one;
a=1; b=1; c=0; output;
a=1; b=2; c=0; output;

```

```

data two;
a=2; d=3; output;
a=2; d=4; output;

```

```

data three;
set one two;
retain b 0 e;
run;

```

a	b	c	d
1	1	0	.
1	2	0	.
2	.	.	3
2	.	.	4



```
data one;
  a=1; b=1; c=0; output;
  a=1; b=2; c=0; output;
```

```
data two;
  a=2; d=3; output;
  a=2; d=4; output;
```

```
data three;
  set one two;
  retain b 0 e;
run;
```

a	b	c	d
1	1	0	.
1	2	0	.
2	.	.	3
2	.	.	4

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

## HOW TO USE, ABUSE, MANIPULATE, AND EXPLOIT THE PROGRAM DATA VECTOR

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

## CHANGE LENGTH OF AN EXISTING VARIABLE

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data one;
  var='ABC';
```

```
data two;
  set one;
  attrib var length=$5;
```

WARNING: Length of character variable has already been set. Use the LENGTH statement as the very first statement in the DATA STEP to declare the length of a character variable.

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data one;
  var='ABC';
```

```
data two;
  attrib var length=$5;
  set one;
```

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

## **MISSING AS A** DATA STEP KEYWORD

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data one;
  date = '04may2003'd; output;
  date = .;          output;
  date = '06may2003'd; output;
```

```
data two;
  set one;
  if date = missing;
```

NOTE: Variable missing is uninitialized.  
NOTE: The data set WORK.TWO has 1 observations  
and 2 variables.



```
data one;
  date = '04may2003'd; output;
  date = .;          output;
  date = '06may2003'd; output;
```

```
data two;
  set one;
  if date = missing;
```

NOTE: Variable missing is uninitialized.  
NOTE: The data set WORK.TWO has 1 observations  
and 2 variables.

date 8.
---------



```
data one;
  date = '04may2003'd; output;
  date = .;          output;
  date = '06may2003'd; output;
```

```
data two;
  set one;
  if date = missing;
```

NOTE: Variable missing is uninitialized.  
NOTE: The data set WORK.TWO has 1 observations  
and 2 variables.

date 8.
---------



```
data one;
  date = '04may2003'd; output;
  date = .;          output;
  date = '06may2003'd; output;
```

```
data two;
  set one;
  if date = missing;
```

NOTE: Variable missing is uninitialized.  
NOTE: The data set WORK.TWO has 1 observations  
and 2 variables.

date 8.	missing 8.
---------	------------



```
data one;
  text = 'ABC'; output;
  text = ""; output;
  text = 'DEF'; output;
```

```
data two;
  set one;
  if text = missing;
```

NOTE: Variable missing is uninitialized.  
NOTE: The data set WORK.TWO has 1 observations  
and 2 variables.

text \$3	missing \$3
----------	-------------



RETAINED  
OR NOT



var1 var2

```
X .  
X .  
Z .  
Z .
```

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

var1 var2

```
X .  
X .  
Z .  
Z .
```

var1 var2

```
X 1  
X 1  
Z 2  
Z 2
```

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

var1 var2

```
X .  
X .  
Z .  
Z .
```

```
data two;  
set one;  
by var1;  
if first.var1 then var2 + 1;  
run;
```

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

var1 var2

```
X .  
X .  
Z .  
Z .
```

```
data two;  
set one;  
by var1;  
if first.var1 then var2 + 1;  
run;
```

var1 var2

```
X 1  
X .  
Z 1  
Z .
```

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

var1 var2

```
X .  
X .  
Z .  
Z .
```

```
data two;  
retain var2 0;  
set one;  
by var1;  
if first.var1 then var2 + 1;  
run;
```

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

var1 var2

```
X .  
X .  
Z .  
Z .
```

```
data two;  
retain var2 0;  
set one;  
by var1;  
if first.var1 then var2 + 1;  
run;
```

var1 var2

```
X 1  
X .  
Z 1  
Z .
```

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

var1 var2

```
X .  
X .  
Z .  
Z .
```

```
data two;  
set one(drop=var2);  
by var1;  
if first.var1 then var2 + 1;  
run;
```

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

var1 var2

```
X .  
X .  
Z .  
Z .
```

```
data two;  
set one(drop=var2);  
by var1;  
if first.var1 then var2 + 1;  
run;
```

var1 var2

```
X 1  
X 1  
Z 2  
Z 2
```

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

## OVERWRITTEN VARIABLE ATTRIBUTES

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data one;  
name='Mary';  
  
data two;  
label name='Subject Name';  
name = 'Scott';  
  
data three;  
set one two;  
put name=;  
run;
```

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data one;  
name='Mary';  
  
data two;  
label name='Subject Name';  
name = 'Scott';  
  
data three;                name=Mary  
set one two;              name=Scot  
put name=;  
run;
```

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data one;  
name='Mary';  
  
data two;  
label name='Subject Name';  
name = 'Scott';  
  
data three;                Subject  
set one two;              Obs Name  
                            1 Mary  
                            2 Scot  
  
proc print;  
run;
```

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data one;
  name='Mary';
```

name \$4

```
data two;
  label name='Subject Name';
  name = 'Scott';
```

name \$5 label='Subject Name'

```
data three;
  set one two;
  put name=;
run;
```

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data one;
  name='Mary';
```

name \$4

```
data two;
  label name='Subject Name';
  name = 'Scott';
```

name \$5 label='Subject Name'

```
data three;
  set one two;
  put name=;
run;
```

name \$4 label='Subject Name'

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

## CHANGE A VARIABLE'S ATTRIBUTES WITHOUT CHANGING ITS NAME IN ONE DATA STEP

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data demog;
  age = '25';
```

```
data demog(rename=(num_age=age)
  drop=age);
  set demog;
  num_age=input(age,8.);
```

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data demog;
  age = '25';
```

```
data demog(rename=(num_age=age)
  drop=age);
  set demog;
  num_age=input(age,8.);
```

age \$2

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data demog;
  age = '25';
```

```
data demog(rename=(num_age=age)
  drop=age);
  set demog;
  num_age=input(age,8.);
```

age \$2

num\_age 8.

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data demog;
  age = '25';

data demog(rename=(num_age=age)
  drop=age);
  set demog;
  num_age=input(age,8.);
```

<del>age \$2</del>	num_age 8.
--------------------	------------

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data demog;
  age = '25';

data demog(rename=(num_age=age)
  drop=age);
  set demog;
  num_age=input(age,8.);
```

<del>age \$2</del>	age 8.
--------------------	--------

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

## TEMPLATES

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data template;
  attrib pt length=8 label='subject number' format=z6.;
  attrib sex length=$6 label='subject gender';
  attrib race length=8 label='ethnic origin' format=race.;
run;
```

NOTE: variable pt is uninitialized.  
NOTE: variable sex is uninitialized.  
NOTE: variable race is uninitialized.  
NOTE: The data set WORK.TEMPLATE has  
1 observations and 3 variables.

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data template;
  attrib pt length=8 label='subject number' format=z6.;
  attrib sex length=$6 label='subject gender';
  attrib race length=8 label='ethnic origin' format=race.;
run;
```

NOTE: variable pt is uninitialized.  
NOTE: variable sex is uninitialized.  
NOTE: variable race is uninitialized.  
NOTE: The data set WORK.TEMPLATE has  
**1 observations** and 3 variables.

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```
data template;
  attrib pt length=8 label='subject number' format=z6.;
  attrib sex length=$6 label='subject gender';
  attrib race length=8 label='ethnic origin' format=race.;
  if _n_=0;
run;
```

NOTE: variable pt is uninitialized.  
NOTE: variable sex is uninitialized.  
NOTE: variable race is uninitialized.  
NOTE: The data set WORK.TEMPLATE has  
**0 observations** and 3 variables.

**COVANCE**  
THE DEVELOPMENT SERVICES COMPANY

```

data template;
  attrib pt length=8 label='subject number' format=z6.;
  attrib sex length=$6 label='subject gender';
  attrib race length=8 label='ethnic origin' format=race.;
  if _n_=0;
run;

```

pt 8. label='subject number' format=z6.	sex \$6 label='subject gender'	race 8. label='ethnic origin' format=race.
---	-----------------------------------	--



```

data demog;
  pt =101;
  sex ='male';
  race =2;
  adddte ='15jan2002'd;
  dob ='22oct1985'd;
run;

```

pt 8.	sex \$4	race 8.	adddte 8.	dob 8.
-------	---------	---------	-----------	--------



```

data demog(drop=adddte dob);
  set template;
  demog;
run;

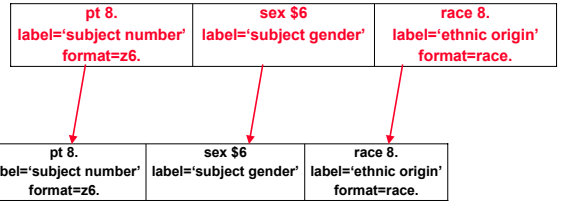
```



```

data demog(drop=adddte dob);
  set template;
  demog;
run;

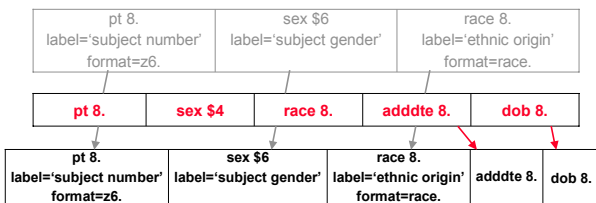
```



```

data demog(drop=adddte dob);
  set template;
  demog;
run;

```



```

data demog(drop=adddte dob);
  set template;
  demog;
run;

```

pt 8. label='subject number' format=z6.	sex \$6 label='subject gender'	race 8. label='ethnic origin' format=race.	<del>adddte 8.</del>	<del>dob 8.</del>
---	-----------------------------------	--	----------------------	-------------------



```

data client.demog;
  attrib pt length=8 label='subject number' format=z6.;
  attrib sex length=$6 label='subject gender';
  attrib race length=8 label='ethnic origin' format=race.;
  if _n_=0;
run;

```

NOTE: variable pt is uninitialized.  
NOTE: variable sex is uninitialized.  
NOTE: variable race is uninitialized.  
NOTE: The data set CLIENT.DEMOG has  
0 observations and 3 variables.



```

data demog;
  pt      =101;
  sex     ='male';
  race    =2;
  adddte  ='15jan2002'd;
  dob     ='22oct1985'd;
run;

```



```

proc append
  base = client.demog
  data = demog;
run;

```

NOTE: Appending WORK.DEMOG to CLIENT.DEMOG.  
WARNING: Variable adddte was not found on BASE file.  
WARNING: Variable dob was not found on BASE file.  
WARNING: Variable sex has different lengths on BASE and  
DATA files (BASE 6 DATA 4).  
ERROR: No appending done because of anomalies listed  
above.  
Use FORCE option to append these files.  
NOTE: 0 observations added.  
NOTE: The data set CLIENT.DEMOG has 0 observations and  
3 variables.  
NOTE: Statements not processed because of errors noted  
above.



```

proc append
  base = client.demog
  data = demog
  force;
run;

```

NOTE: Appending WORK.DEMOG to CLIENT.DEMOG.  
WARNING: Variable adddte was not found on BASE file.  
WARNING: Variable dob was not found on BASE file.  
WARNING: Variable sex has different lengths on BASE and  
DATA files (BASE 6 DATA 4).  
NOTE: FORCE is specified, so dropping/truncating will occur.  
NOTE: There were 1 observations read from the data set  
WORK.DEMOG.  
NOTE: 1 observations added.  
NOTE: The data set CLIENT.DEMOG has 1 observations and  
3 variables.



NOTE: Appending WORK.DEMOG to CLIENT.DEMOG.  
WARNING: Variable adddte was not found on BASE file.  
WARNING: Variable dob was not found on BASE file.  
WARNING: Variable sex has different lengths on BASE and DATA files (BASE 6 DATA 4).  
NOTE: FORCE is specified, so dropping/truncating will occur.  
NOTE: There were 1 observations read from the data set WORK.DEMOG.  
NOTE: 1 observations added.  
NOTE: The data set CLIENT.DEMOG has 1 observations and 3 variables.

pt 8. label='subject number' format=z6.	sex \$6 label='subject gender'	race 8. label='ethnic origin' format=race.
---	-----------------------------------	--



## CONCLUSION

- All variables read into the data step are available on the program data vector
- They may ultimately be dropped, kept, renamed, or retained



## CONCLUSION

- Knowing the difference between SAS statements and data set options and when they take affect
- Knowing how to use their strengths
- Knowing how to exploit their timing



## CONCLUSION

- You can change your cinder blocks into bricks, and your bricks into pennies
- You can improve your programming skills
- You can improve your program efficiency



## ? QUESTIONS ?

Jim Johnson

[jb.johnson@covance.com](mailto:jb.johnson@covance.com)

