Stat 302
Statistical Software and Its Applications
SAS: Working with Data

Fritz Scholz

Department of Statistics, University of Washington

Winter Quarter 2015

February 26, 2015

Outline

- Chapter 7 in Learning SAS by Example, A Programmer's Guide by Ron Cody
- Conditional Statements
- Subsetting
- Loops
- Start by saving the data sets `student.txt` and `student2.txt` from the course web site to the folder from which you usually import data into SAS, in my case `U:\data`.

Comparison Operators

Logical Comparison	Mnemonic	Symbol
Equal to	EQ	=
Not equal to	NE	^= or ~=
Less than	LT	<
Less than or equal to	LE	<=
Greater than	GT	>
Greater than or equal	GE	>=
Equal to one in a list	IN	

Using if/then Statement to Create Variables

- Using `student.txt` generate new variable `AgeGroup`.

```
data student;
   infile "U:\data\student.txt";
   input Age Major $ GPA;
   if Age le 22 then AgeGroup = 1;
   if Age gt 22 then AgeGroup = 2;
   * try it with <= and > in place of le and gt;
run; * Also try Agegroup instead of AgeGroup;

title "Student Data with Age Group";
proc print data = student noobs;
* noobs removes the observations column;
run;
```

The Output

Student Data with Age Group

Age	Major	GPA	AgeGroup
24	Math	2.3	2
21	Stat	2.6	1
20	Math	3.8	1
19	Bio	3.8	1
18	Bio	3.9	1
18	Nursing	2.4	1
19	Stat	2.6	1
23	Nursing	3.8	2
25	Bio	3.8	2
18	Math	3.9	1
19	Bio	2.4	1
27	Nursing	2.5	2
20	Nursing	2.7	1
20	Stat	2.4	1
22	Math	2.8	1
23	Stat	3.3	2
23	Nursing	3.8	2
19	Bio	3.9	1
20	Bio	3.4	1

Here I used the Snipping Tool in Accessories (not on server) to capture the output image and saved it as `student.JPG` in the appropriate folder for use in LaTeX on my machine. Not crisp.

Using if/then statement with missing values

- Be careful with missing values in if/then statements.
- SAS treats missing values logically as the most negative number on your computer.
- If you use $<$ or $<=$ statements, this will include missing values.

```
data student2;
   infile "U:\data\student2.txt";
   input Age Major $ GPA;
   if Age le 22 then AgeGroup = 1;
   if Age gt 22 then AgeGroup = 2;
run;

title "Student Data with Age Group
and Missing Values";
proc print data = student2 noobs;
run;
```

The Output

Student Data with Age Group and Missing Values

Age	Major	GPA	AgeGroup
24	Math	2.3	2
21	Stat	2.6	1
20	Math	3.8	1
.	Bio	3.8	1
18	Bio	3.9	1
18	Nursing	2.4	1
19	Stat	2.6	1
.	Nursing	3.8	1
25	Bio	3.8	2
18	Math	3.9	1
19	Bio	2.4	1
27	Nursing	2.5	2
20	Nursing	2.7	1
20	Stat	2.4	1
22	Math	2.8	1
.	Stat	3.3	1
23	Nursing	3.8	2
19	Bio	3.9	1
20	Bio	3.4	1

Here missing values are classified as ≤ 22, which is inappropriate.

Correct treatment of missing values

```
data student2;
infile "U:\data\student2.txt";
input Age Major $ GPA;
case = _N_;
if Age le 22 and Age ne . then AgeGroup = 1;
if Age gt 22 then AgeGroup = 2;
run; * Try AgeGroup = "A" or "B", then A or B;
title "Student Data with Age Group
and Missing Values";
proc print data = student2 noobs;
var Case Age Major GPA AgeGroup;
run;
```

- Missing values are now reflected as . in the AgeGroup column.
- Here we printed the output to Adobe PDF, for better graphics.
- Showed how to get a `Case` variable using `_N_`.
- Using `_N_` after `var` gives error.
- Showed how to control the variable order in `proc print`.

The Output: Missing Treated Correctly

Student Data with Age Group and Missing Values

case	Age	Major	GPA	AgeGroup
1	24	Math	2.3	2
2	21	Stat	2.6	1
3	20	Math	3.8	1
4	.	Bio	3.8	.
5	18	Bio	3.9	1
6	18	Nursing	2.4	1
7	19	Stat	2.6	1
8	.	Nursing	3.8	.
9	25	Bio	3.8	2
10	18	Math	3.9	1
11	19	Bio	2.4	1
12	27	Nursing	2.5	2
13	20	Nursing	2.7	1
14	20	Stat	2.4	1
15	22	Math	2.8	1
16	.	Stat	3.3	.
17	23	Nursing	3.8	2
18	19	Bio	3.9	1
19	20	Bio	3.4	1

Using if statements alone to subset data

```
data stat_student;
   infile "U:\data\student.txt";
   input Age Major $ GPA;
   if Major eq "Stat";
* here "Stat" is case sensitive, "STAT" and "stat"
  will produce an empty set, no output.
  Inside the file student.txt Stat is used;
  * for data values case/Case matters! ;
run;

title "Stat Student Data";
proc print data = stat_student;
run;
```

The Output: Subsetting Stat Students

Stat Student Data

Obs	Age	Major	GPA
1	21	Stat	2.6
2	19	Stat	2.6
3	20	Stat	2.4
4	23	Stat	3.3

Using the IN Operator

- The IN operator can check for multiple conditions, can be used in place of multiple OR statements

```
if Quiz="A+" or Quiz="A" or Quiz="B+"
   then QuizRange=1;
```

- We can use the IN operator instead as follows

```
if Quiz in ("A+" "A" "B+") then QuizRange=1;
```

- The list values in (...) can be separated by spaces or commas.
- You can also use IN with numeric values

```
if Subject in (10,22:25,30);
```

- The above assume that `Quiz` and `Subject` are defined variables, character and numeric, respectively.

Using the IN Operator Selecting Observations

```
data stat_student;
   infile "U:\data\student.txt";
   input Age Major $ GPA;
   if _N_ in (2:5, 10,15);
* here we select observations 2,3,4,5,10,15;
run;

title "Selected Students 2,3,4,5,10,15";
ods PDF newfile=output
    file='U:\data\StudentSelect.pdf';
proc print data = stat_student;
run;
ods PDF close;
```

The Output: Selected Students

Obs	Age	Major	GPA
1	21	Stat	2.6
2	20	Math	3.8
3	19	Bio	3.8
4	18	Bio	3.9
5	18	Math	3.9
6	22	Math	2.8

Using the WHERE statement to subset data

- We can use the WHERE statement to subset data.
- This is only possible with SAS data sets.
- These must be brought in with SET command.
- See next example.
- There are more operators that can be used whith WHERE.
- WHERE can also be used inside SAS procs to subset data.
- The IF statement cannot be used inside procs.
- WHERE can select rows only from existing SAS data sets.
- The IF statement can select rows from existing SAS data sets or from raw data files being read with INPUT statements.
- WHERE is more efficient than IF, especially when applied directly in a proc (not in data step).

Using the WHERE Operator Selecting Observations

```
libname mydata "U:\data";
data stat_student;
   set mydata.student;
   where major eq "Stat";
   * here we select Stat majors;
   * where _N_ in (2:4,15);
   * works, but produces ERROR in log
     since _N_ is system variable and
     not a variable in the SAS data set;
   * if _N_ in (2:4,15); * works fine ;
run;
title "Stat Student Data";
proc print data = stat_student;
run;
```

- Same result as on slide 11.
- This assumes the presence of a permanent SAS data set `student.sas7bdat` in location `U:\data`.

Subsetting Print by Case Number

```
data student;
   infile "U:\data\student.txt";
   input Age Major $ GPA;
   case = _N_; * _N_ is a system variable;
run;

title "Student Data";
proc print data = student;
where case in (1:4  9:10 16);
run;
```

Output: Subsetting Print by Case

Student Data

Obs	Age	Major	GPA	case
1	24	Math	2.3	1
2	21	Stat	2.6	2
3	20	Math	3.8	3
4	19	Bio	3.8	4
9	25	Bio	3.8	9
10	18	Math	3.9	10
16	23	Stat	3.3	16

Subsetting Data Set by Case Number

```
data student;
   infile "U:\data\student.txt";
   input Age Major $ GPA;
   case = _N_;
   if case in (1:4  9:10 16);
   * could use WHERE in place of IF,
   but get ERROR in log;
run;

title "Student Data";
proc print data = student;
run;
```

The Output: Subsetting Data Set by Case

Student Data

Obs	Age	Major	GPA	case
1	24	Math	2.3	1
2	21	Stat	2.6	2
3	20	Math	3.8	3
4	19	Bio	3.8	4
5	25	Bio	3.8	9
6	18	Math	3.9	10
7	23	Stat	3.3	16

Using the WHERE operator

Operator	Description	Example
IS MISSING	is missing the stated value	`where Age is missing`
IS NULL	Equivalent to IS MISSING	`where Age is null`
BETWEEN_AND_	An inclusive range	`where age is between 20 and 25`
CONTAINS	Matches a substring	`where name contains Mac`
LIKE	Matching with wildcards	`where name like R_n%`

- The LIKE expression contains 2 wildcard operators.
- The underscore _ is a place holder for any character (use as many as you like)
- The % matches nothing or a string of any length.
- In the above example `R_n%` matches Ron, Ronald, Running, Run, etc.

Using the WHERE operator

```
libname mydata  "U:\data";
data nursing_student;
set mydata.student;
where Major eq "Nursing";
run;
title "Nursing Student Data";
proc print data = nursing_student;
run;
```

- This assumes the presence of a permanent SAS data set `student.sas7bdat` in location `U:\data`.
- This code creates a new, temporary SAS data set, `nursing_student`, consisting of just the nursing students.
- Using `mydata.nursing_student` in place of `nursing_student` throughout creates a permanent SAS data set `nursing_student.sas7bdat` in `U:\data`.

The Output

Nursing Student Data

Obs	ID	Major	Grade
1	18	Nursing	2.4
2	23	Nursing	3.8
3	27	Nursing	2.5
4	20	Nursing	2.7
5	23	Nursing	3.8

Subsetting by Columns

```
libname mydata  "U:\data";
data nursing_student;
set mydata.student (keep=Major Grade);
where Major eq "Nursing";
run;
title "Nursing Student Data, Major and Grade only";
proc print data = nursing_student;
run;
```

- This assumes the presence of a permanent SAS data set `student.sas7bdat` in location `U:\data`.
- This code creates a new, temporary SAS data set, `nursing_student`, consisting of just the nursing students.

The Output

Nursing Student Data, Major and Grade only

Obs	Major	Grade
1	Nursing	2.4
2	Nursing	3.8
3	Nursing	2.5
4	Nursing	2.7
5	Nursing	3.8

Multiple Operations for Same Condition

```
libname learn "U:\learn";

data grades; * This will be in WORK;
set learn.grades; * This comes from learn;
if missing(Age) then delete;
if Age le 39 then AgeGrp = "Younger Group";
if Age le 39 then Grade = .4*Midterm+.6*FinalExam;
if Age gt 39 then AgeGrp = "Older Group";
if Age gt 39 then Grade = (Midterm+FinalExam)/2;
run;

title "Listing of Grades";
proc print data=grades;
run;
```

Doing the Same Using a Do Group

```
libname learn "U:\learn";
data grades; * This will be in WORK;
set learn.grades; * This comes from learn;
if missing(Age) then delete;
if Age le 39 then do;
  AgeGrp = "Younger Group";
  Grade = .4*Midterm+.6*FinalExam;
end;
if Age gt 39 then do;
  AgeGrp = "Older Group";
  Grade = (Midterm+FinalExam)/2;
end;
run;
title "Listing of Grades Using Do Group";
proc print data=grades;
run;
```

The Output

Listing of Grades Using Do Group

Obs	Age	Gender	Midterm	Quiz	FinalExam	AgeGrp	Grade
1	21	M	80	B-	82	Younger Group	81.2
2	35	M	87	B+	85	Younger Group	85.8
3	48	F	.		76	Older Group	.
4	59	F	95	A+	97	Older Group	96.0
5	15	M	88		93	Younger Group	91.0
6	67	F	97	A	91	Older Group	94.0
7	35	F	77	C-	77	Younger Group	77.0
8	49	M	59	C	81	Older Group	70.0

R Analog to SAS Do Group

```
if(condition1){
...
...
}
if(condition2){
...
...
}
```

- `condition1` and `condition2` are two logic variables, with values `TRUE` or `FALSE`.

The Sum Statement (First Attempt)

```
data revenue;
input Day : $3.
      Revenue  : dollar6. ;
      * $3. character variable, length 3;
      * Dollar amount, length 6;
    Total = Total + Revenue; * this won't work;
format Revenue Total dollar8. ;
datalines;
Mon $1,000
Tue $1,500
Wed .
Thu $2,000
Fri $3,000
; run;
title "Listing of Revenue";
proc print data=revenue; run;
```

- This does not work since `Total` is initialized as missing value and `Total = Total + Revenue` ⇒ `Total = .`
- For more on formatted INPUT see Cody 3.10-3.14.

The Output

Obs	Day	Revenue	Total
1	Mon	$1,000	.
2	Tue	$1,500	.
3	Wed	.	.
4	Thu	$2,000	.
5	Fri	$3,000	.

The Sum Statement (Work Around 1)

```
data revenue;
retain Total 0; * Initializes Total;
input Day : $3.
      Revenue  : dollar6.;
    Total = Total + Revenue;
    * this won't work either;
format  Revenue  Total dollar8. ;
datalines;
Mon $1,000
Tue $1,500
Wed .
Thu $2,000
Fri $3,000
run;
title "Listing of Revenue";
proc print data=revenue; run;
```

The Output

Obs	Total	Day	Revenue
1	$1,000	Mon	$1,000
2	$2,500	Tue	$1,500
3	.	Wed	.
4	.	Thu	$2,000
5	.	Fri	$3,000

The Sum Statement (Work Around 2)

```
data revenue;
retain Total 0;
input Day : $3.
      Revenue  : dollar6.;
    if not missing(Revenue) then
     Total = Total + Revenue;
format  Revenue  Total dollar8.2 ;
datalines;
Mon $1,000
Tue $1,500
Wed .
Thu $2,000
Fri $3,000
run;
title "Listing of Revenue";
proc print data=revenue;
var Day Revenue Total; run;
```

The Output

Obs	Day	Revenue	Total
1	Mon	$1000.00	$1000.00
2	Tue	$1500.00	$2500.00
3	Wed	.	$2500.00
4	Thu	$2000.00	$4500.00
5	Fri	$3000.00	$7500.00

The Sum Statement Solution

```
data revenue;
input Day : $3.
      Revenue  : dollar6.;
    Total + Revenue;
format  Revenue  Total dollar8.2 ;
* the Dollar amount with cents ;
datalines;
Mon $1,000
Tue $1,500
Wed .
Thu $2,000
Fri $3,000
run;
title "Listing of Revenue";
proc print data=revenue;
```

- Same output as on previous slide.

Effect of the Sum Statement Solution

Form of the Sum Statement:

`variable+increment`

- `variable` is retained from data step to data step
- `variable` not automatically initialized as `.` (missing)
- `variable` is initialized at 0 on first data step
- Data steps with missing value in `increment` are ignored

Compound Interest

```
data compound;
Interest = .0125;
Total = 100;
Year+1; * A SUM statement;
Total + Interest*Total; * ditto;
output; * writes observation to the output;
Year+1;
Total + Interest*Total;
output; * same here;
format Total dollar10.2;
run;

title "Listing of Compound";
proc print data=compound noobs;
run;
```

- The compounding statements can be repeated, or use a loop.

The Output

Interest	Total	Year
0.0125	$101.25	1
0.0125	$102.52	2

The `output;` Statement

- The `output;` statement instructs SAS to write out an observation to the output data set, here output = `compound`.
- Here we want to output `Year` and `Total` each time you compute new values for them.
- An output usually occurs at the bottom of the data step.
- When you include an `output;` statement anywhere within the data step, SAS does not execute an automatic output at the bottom of the data step.

Compound Interest Using Do Loop

```
data compound;
   Interest = .0125;
   Total = 100;
   do Year = 1 to 5;
      Total + Total * Interest;
      output;
   end;
   format Total dollar10.2;
run;

title "Listing of Compound";
proc print data=compound noobs;
run;
```

- A lot more compact.

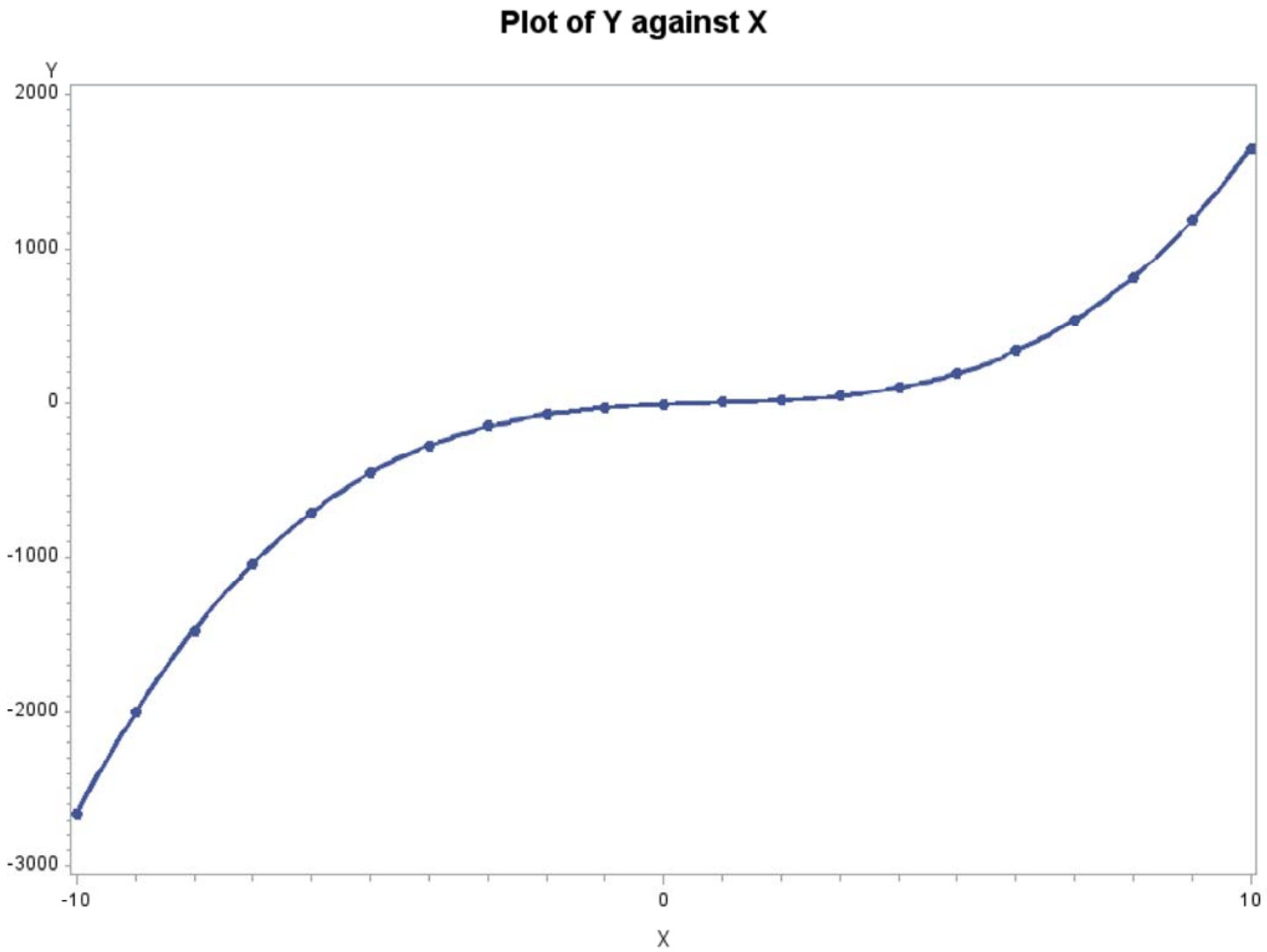
The Output

Interest	Total	Year
0.0125	$101.25	1
0.0125	$102.52	2
0.0125	$103.80	3
0.0125	$105.09	4
0.0125	$106.41	5

Plotting a Function Using a Do Loop

```
data equation;
   do X = -10 to 10 by 1;
      Y = 2*x**3-5*x**2+15*x-8;
      output;
   end;
run;
symbol value=dot interpol=sm;
title "Plot of Y against X";
proc gplot data = equation;
   plot Y * X;
run;
```

The Output



Other Forms of Iterative Do Loop

```
do x = 1,2,5,10;

( values of x are: 1, 2, 5, 10 )

do month = 'Jan' 'Feb' 'Mar';

( values of month are: 'Jan', 'Feb', 'Mar' )

do n = 1,3, 5 to 9 by 2, 100 to 200 by 50

( values of n are: 1, 3, 5, 7, 9, 100, 150, 200 )
```

Nested Do Loops

```
data treat;
   do Group = 'Placebo', 'Active';
     do Subj = 1 to 5;
          input Score @;
* @ ==> keep reading from same line until done;
          output;
     end;
   end;
datalines;
250 222 230 210
199 166 183 123 129 234
;
run;
title "Score by Treatment";
  proc print data = treat;
run;
```

The Output

Score by Treatment

Obs	Group	Subj	Score
1	Placebo	1	250
2	Placebo	2	222
3	Placebo	3	230
4	Placebo	4	210
5	Placebo	5	199
6	Active	1	166
7	Active	2	183
8	Active	3	123
9	Active	4	129
10	Active	5	234

Do Until Loop

```
data double;
   Interest = .0225;
   Total = 100;
   do until (Total ge 200);
      year+1;
      Total = Total+Interest*Total;
         * could drop Total =;
         * Total was initialized;
      output;
   end;
   format Total Dollar10.2;
run;
title "Doubling Capital";
proc print data = double;
where Total ge 180;
run;
```

- `do until` always executes at least once.

The Output

Doubling Capital

Obs	Interest	Total	year
27	0.0225	$182.35	27
28	0.0225	$186.45	28
29	0.0225	$190.65	29
30	0.0225	$194.94	30
31	0.0225	$199.33	31
32	0.0225	$203.81	32

While Loop

```
data double;
    Interest = .0225;
    Total = 100;
    do while (Total lt 200);
       year+1;
   Total = Total+Interest*Total;
   output;
    end;
    format Total Dollar10.2;
run;
title "Doubling Capital";
proc print data = double;
where Total ge 180;
run;
```

- `while` condition is tested at the start of loop (same result).
- May not execute at all. (Change `Total = 200;`)

Avoid Infinite Loops

- You can stop a SAS program by clicking the exclamation point on the task bar.
- Often the program stops itself because some number overflows.
- Combine `do until` with upper limit `do` loop.

```
data double;
   Interest = .0225;
   Total = 100;
   do Year = 1 to 100 until (Total gt 200);
       Total = Total+Interest*Total;
       output;
   end;
   format Total Dollar10.2; run;
title "Doubling Capital";
proc print data = double;
where Total ge 180; run;
```

Leave and Continue

- Break out of a loop by the `LEAVE` statement or go back to top of the loop by the `CONTINUE` statement.

```
data double;
   Interest = .0225;
   Total = 100;
   do Year = 1 to 100;
       Total = Total+Interest*Total;
       output;
       if Total ge 200 then leave;
       * this breaks us out of the loop;
   end;
   format Total Dollar10.2;
run;
title "Doubling Capital";
proc print data = double;
where Total ge 180;
run;
```

Leave and Continue (continued)

```
data double;
   Interest = .0225;
   Total = 100;
   do Year = 1 to 100 until (Total ge 200);
       Total = Total+Interest*Total;
       if Total le 190 then continue;
       * this makes us go back to the top
       of the loop, incrementing Year;
       output;
   end;
   format Total Dollar10.2;
run;
title "Doubling Capital";
proc print data = double;
where Total ge 180;
run;
```

The Output

Doubling Capital

Obs	Interest	Total	Year
1	0.0225	$190.65	29
2	0.0225	$194.94	30
3	0.0225	$199.33	31
4	0.0225	$203.81	32