STAT 5110/6110: SAS Programming and Applications

2-D. Manipulating SAS Data Sets

Peng Zeng

Department of Mathematics and Statistics

Auburn University

Manipulating SAS Data Sets

```
data new-SAS-data;
  set existing-SAS-data;
  /* more statements */
run;
```

In a data step, we can

- drop unwanted variables
- create or modify a variable
- execute statements conditionally
- specify a variable's length
- subset data

Keep or Drop Variables

Use keep=/drop= option or keep/drop statement to select or remove variables. (pay attention to the difference)

```
data selected1;
   set sashelp.baseball (keep = name team Salary);
run;
```

```
data selected2 (keep = name team Salary);
   set sashelp.baseball;
run;
```

```
data selected3;
   set sashelp.baseball;
   keep name team Salary;
run;
```

Comments

- keep/drop statement only works for data step
- keep=/drop= option combined with the names of data sets can be used in any other procedures

For example:

```
proc print data = sashelp.baseball (keep = name team Salary);
run;
```

```
proc print data = sashelp.baseball;
   var name team Salary;
run;
```

Define New Variables

An assignment statement can

- evaluate an expression
- assign the resulting value to a variable

variable = *expression*;

where expression can involve

- numbers, characters, parentheses
- addition (+), subtract (−), multiplication (∗), division (/), exponentiation (∗∗), negative (−)
- mathematical function such as sin(), exp(), log(), log10(), ...
- other functions supported by SAS

Example: Major League Baseball Players

```
data salary;
   set sashelp.baseball;
   /* change unit from thousand $ to $ */
   salary = salary * 1000;
   /* percent of 1986 homerun out of career homerun */
   rate = nHome / CrHome;
run;
```

How to create salary groups?

salary < 190	less than 190K
$190 \leq$ salary < 425	between 190K and 425K
$425 \leq$ salary < 750	between 425K and 750K
salary ≥ 750	larger than 750K

Conditional Execution

if *expression* then *statement*; else *statement*;

- An expression usually involves a logical operation.
- The else statement can be omitted.
- Only one statement is allowed in an if-then or else statement.
- Use do and end to include a group of statements.

```
if expression then do;
   multiple-executable-statements;
end;
else do;
   multiple-exectuable-statements;
end;
```

Logical Operator

Each comparison operator yields a value of T (true) or F (false).

operator	Example
EQ (=)	Region = 'Spain'
NE (~= or ^=)	Region ~= 'Spain'
GT (>)	Rainfall > 20
LT (<)	Rainfall < AvgRain
GE (>=)	Rainfall >= AvgRain + 5
LE (<=)	Rainfall <= AvgRain / 1.25

It is equivalent to write as follows.

```
region eq "Spain"
```

```
rainfall gt 20
```

Comments

- SAS use = for both assignment and equality.

```
if high = "T" then score = 1;
```

- Character comparison is case-sensitive. Use functions upcase() and lowcase() to convert letters to uppercase or lowercase.

```
lowcase(Region) = "spain"
```

```
upcase(Region) = "SPAIN"
```

- Any numeric value other than 0 or missing is true, and a value of 0 or missing is false.

```
if score then grade = "valid";
```

Comparison and Logic Operators

A logic operator (such as `and`, `or`, `not`) can link two comparisons.

True and True = True	True or True = True
True and False = False	True or False = True
False and False = False	False or False = False
not True = False	not False = True

For example

```
(lowcase(Region) = "spain") and (Rainfall > 20)
```

```
not (lowcase(Region) = "spain")
```

Example: Blood Pressure

Create a new SAS dataset bloodnew from a existing dataset blood. Define three new variables ave (numeric), high (character), and selected (numeric).

```
data bloodnew;
   set blood;
   ave = (systolic + diastolic) / 2;
   if (systolic >= 140) or (diastolic >= 90)
   then high = "T";
   else high = "F";
   if patient in ("CP", "GS", "SB") then selected = 1;
run;
```

The in operator is convenient for character variables. It allows commas or blanks to separate values.

```
selected = (patient in ("CP", "GS", "SB")); /* compare codes */
```

Comments

- When assigning a character string to a categorical variable, make sure to use quotation marks.

```
high = "T";
```

- We can update the value of an existing variable

```
score = score * 2;
```

- We can also assign values as missing explicity.

```
age = .;          /* numeric variable */
color = "";       /* character variable */
```

- After you make changes to a dataset, make sure to check the contents of the dataset using proc print.
- It is possible to defining new variables at the same time when we create a new SAS dataset. Simply write the statements between input and datalines statements.

Example: Major League Baseball Players

```
data baseball;
   set sashelp.baseball;
   if salary < 190 then group = "less than 190K";
   else if salary < 425 then group = "between 190K and 425K";
   else if salary < 750 then group = "between 425K and 750K";
   else group = "larger than 750K";
run;
```

Questions

- Is baseball and sashelp.baseball the same data set?
- What happens if the salary contains missing values?
- Will the values of group be correctly assigned?

Example: Updated Codes

```
data baseball2;
   length group $25;
   set sashelp.baseball;
   if missing(salary) then group = "";
   else if salary < 190 then group = "less than 190K";
   else if salary < 425 then group = "between 190K and 425K";
   else if salary < 750 then group = "between 425K and 750K";
   else group = "larger than 750K";
run;
```

By default, SAS sets the length of a character variable by the first value it encounters for that variable. Use the length statement to specify a length to avoid truncation of your values.

```
   length Address1 Address2 Address3 $200;
```

Generate a Subset

We can generate a subset using the following methods

- if *expression*: select observations to keep

```
   if systolic > 120;
```

- if *expression* then delete: select observations to remove

```
   if high = "T" then delete;
```

- where statement or where option

```
 data complete;
   set sashelp.baseball (where = (salary is not missing));
run;
```

```
data complete;
   set sashelp.baseball;
   where salary is not missing;
run;
```

Frequently Used Operators in Where

operator	Example
IS NOT MISSING	Region IS NOT MISSING;
BETWEEN AND	Age BETWEEN 30 AND 50;
CONTAINS	Region CONTAINS 'ain';
IN (list)	Region IN ('Rain', 'Spain', 'Plain');
AND (&)	Rainfall > 20 AND Temp < 90;
OR (|)	Rainfall > 20 OR Temp < 90;
NOT	Region NOT IN ('Rain', 'Spain');

- Character comparisons are case sensitive.
- The in operator allows commas or blanks to separate values.

In-Class Exercise

The sashelp library has a data set named cars, which contains information on some cars in 2004.

- create a data set dropping two variables (Cylinders, Horsepower)
- define a new variable named diff, which is the difference between MSRP and Invoice
- define a new variable named expensive, whose value is yes for MSRP $\geq$ \$30,000 and no otherwise.
- define a new variable named imported, whose value is 1 for non-USA cars and 0 otherwise
- How many different origins? What are the percentages?
- create a subset containing only European sedans
- create a subset for cars whose model names contain 4dr