Chapter 14 MODIFYING CHARACTER VARIABLE VALUES

WITH SAS FUNCTIONS

* The function **SCAN** returns a specified word from a character variable value. The syntax is

scan(*variable\_name*, *word\_number*, *delimiters*)

where **delimiters** are special characters used to separate character expressions. They must be enclosed in single quotation marks. Default delimiters are

blank , . < ( ) + | & ! $ \* ; ^ - / %

* Note that if delimiters in the function SCAN belong to the list of the default ones, then they may be omitted. The syntax then is scan(*variable\_name*, *word\_number*)

Example. The following code separates students’ names as written in an official class roster into last name and first name and drops the middle name. Note that the delimiters used to separate the names are comma and blank, which are default delimiters and may be omitted.

data roster (drop=name);

input student\_id$ 1-9 @11 name$ 21.;

last\_name=scan(name,1);

first\_name=scan(name,2);

cards;

001039133 Nguyen, Melany

679834566 Mari, Leana Adrienne

113816433 Cawley, Anne Margaret

;

proc print noobs;

run;

The output is

student\_ last\_ first\_

id name name

001039133 Nguyen Melany

679834566 Mari Leana

113816433 Cawley Anne

* The function **SUBSTR** is used to (i) extract a portion of a character variable value; (ii) replace the contents of a character variable value.

(i) To extract a portion of a character variable value, use the syntax

*new\_variable*=substr(*variable\_name*, *position*, *n*)

where *position* is the character position to start from, and *n* is the number of characters to extract. If *n* is omitted, all remaining characters are included in the substring.

Example. In the roster example, the following code extracts the last four digits of the student id numbers.

data roster1 (drop=student\_id);

set roster;

id\_last4=substr(student\_id, 6, 4);

run;

proc print noobs;

run;

The output is

last\_ first\_

name name id\_last4

Nguyen Melany 9133

Mari Leana 4566

Cawley Anne 6433

(ii) To replace the contents of a character variable value, use the syntax

substr(*variable\_name*, *position*, *n*)= '*new\_string* '

where *position* is the character position to start replacement from, *n* is the number of characters to replace, and the *new\_string* is a substring to replace with. If *n* is omitted, all remaining characters are replaced.

Example. In the roster example, the following code replaces the first five digits in the student id numbers by a string of X’s.

data roster2;

set roster;

substr(student\_id, 1, 5)='XXXXX';

run;

proc print noobs;

run;

The output is

student\_ last\_ first\_

id name name

XXXXX9133 Nguyen Melany

XXXXX4566 Mari Leana

XXXXX6433 Cawley Anne

* The function **TRIM** is used to remove trailing blanks from character variable values. The syntax is trim(*variable\_name*)

Example. The following data step defines student name, number and street name, city, state, and ZIP.

data address;

input name $21. @23 street $20. @44 city $12. @57 state $2. @60 zip $5.;

cards;

Nguyen, Melany 6249 ELM AVENUE CYPRESS CA 90630

Mari, Leana Adrienne 3199 MAIN STREET SANTA MONICA CA 90405

Cawley, Anne Margaret 277 N VERMONT AVENUE LOS ANGELES CA 90004

;

A one address variable may be created that combines the values of the variables street, city, state, and zip. The concatenation operator || may be used to join the four variables.

data address (drop=street city state zip);

set address;

new\_address=street||', '||city||', '||state||' '||zip;

run;

proc print noobs;

run;

The output is

name new\_address

Nguyen, Melany 6249 ELM AVENUE , CYPRESS , CA 90630

Mari, Leana Adrienne 3199 MAIN STREET , SANTA MONICA, CA 90405

Cawley, Anne Margaret 277 N VERMONT AVENUE, LOS ANGELES , CA 90004

Note that the new addresses contain embedded blanks. The function TRIM may be used to get rid of these blanks.

data address (drop=street city state zip);

set address;

new\_address=trim(street)||', '||trim(city)||', '||state||' '||zip;

run;

proc print noobs;

run;

The output is

name new\_address

Nguyen, Melany 6249 ELM AVENUE, CYPRESS, CA 90630

Mari, Leana Adrienne 3199 MAIN STREET, SANTA MONICA, CA 90405

Cawley, Anne Margaret 277 N VERMONT AVENUE, LOS ANGELES, CA 90004

* The function **CATX** concatenates character strings, removes leading and trailing blanks, and inserts delimiter. The syntax is

*new\_variable*=catx(*delimiter*, *variable\_1*, …, *variable\_n*)

Example. In the previous example, alternatively to the function TRIM and the concatenation operator, the function CATX may be used. The following program specifies that the variable new\_address contains street, city, state, and zip separated from each other by a comma and a blank.

data address (drop=street city state zip);

set address;

new\_address=catx(', ', street, city, state, zip);

run;

proc print noobs;

run;

The output is

name new\_address

Nguyen, Melany 6249 ELM AVENUE, CYPRESS, CA, 90630

Mari, Leana Adrienne 3199 MAIN STREET, SANTA MONICA, CA, 90405

Cawley, Anne Margaret 277 N VERMONT AVENUE, LOS ANGELES, CA, 90004

* The function **INDEX** searches a character variable value for a specified string. It returns the position of the string’s first character. If the string is not found, it returns a zero. The INDEX function is case sensitive, therefore, the string that this function searches for must be specified exactly as it is recorded in the data set. The syntax is

index(*variable\_name*, '*string*')

Example. The following program selects students in the address data set who live in Los Angeles.

data los\_angeles;

set address;

if index(city, 'LOS ANGELES')>0;

run;

proc print noobs;

run;

The output is

name street city state zip

Cawley, Anne Margaret 277 N VERMONT AVENUE LOS ANGELES CA 90004

* The function **FIND** searches a character variable value for a specified string. It is similar to the function INDEX, but has more options. The syntax is

find(*variable\_name*, '*string*', '*modifiers*', *start\_position*)

where the modifier '***i***' causes the FIND function to **ignore** character case during the search, and the modifier **'*t*'** **trims trailing blanks** from *variable\_name* and *string*. The *start\_position* specifies the position at which the search should commence and the direction of the search. If *start\_position* is omitted, the search starts at the beginning and goes to the right. If *start\_position* is **positive**, the search is conduced **left to right**; if *start\_position* is **negative**, the search goes **right to left**. The absolute value of *start\_position* determines the position at which the search starts. If the string is not found, FIND returns zero.

Example. In the above example, an alternative program that produces the same output might be as follows

data los\_angeles;

set address;

if find(city, 'los angeles', 'i')>0;

run;

proc print noobs;

run;

* The function **UPCASE** coverts all letters in a character variable value to uppercase. The syntax is upcase(*variable\_name*)

Example. The following program converts all names in the address data set to uppercase.

data uppercase (drop=name);

set address;

upcase\_name=upcase(name);

run;

proc print noobs;

var upcase\_name street city state zip;

run;

The output is

upcase\_name street city state zip

NGUYEN, MELANY 6249 ELM AVENUE CYPRESS CA 90630

MARI, LEANA ADRIENNE 3199 MAIN STREET SANTA MONICA CA 90405

CAWLEY, ANNE MARGARET 277 N VERMONT AVENUE LOS ANGELES CA 90004

* The function **LOWCASE** coverts all letters in a character variable value to lowercase. The syntax is lowcase(*variable\_name*)

Example. The following program converts all values of street variable in the address data set to lowercase.

data lowercase (drop=street);

set address;

lowcase\_street=lowcase(street);

run;

proc print noobs;

var name lowcase\_street city state zip;

run;

The output is

name lowcase\_street city state zip

Nguyen, Melany 6249 elm avenue CYPRESS CA 90630

Mari, Leana Adrienne 3199 main street SANTA MONICA CA 90405

Cawley, Anne Margaret 277 n vermont avenue LOS ANGELES CA 90004

* The function **PROPCASE** coverts all letters in a character variable value to proper case, so that the initial letter in each word is capitalized. The syntax is

propcase(*variable\_name*)

Example. The following program converts all values of street variable in the address data set to proper case.

data propcase (drop=street);

set address;

propcase\_street=propcase(street);

run;

proc print noobs;

var name propcase\_street city state zip;

run;

The output is

name propcase\_street city state zip

Nguyen, Melany 6249 Elm Avenue CYPRESS CA 90630

Mari, Leana Adrienne 3199 Main Street SANTA MONICA CA 90405

Cawley, Anne Margaret 277 N Vermont Avenue LOS ANGELES CA 90004

* The function **TRANWRD** replaces or removes all occurrences of a particular string within a character variable value. The syntax is

tranwrd(*variable\_name*,'*string*','*replacement*')

Example. The following program replaces all ‘Graduate degree’ strings by ‘Masters degree’ strings.

data enrollment;

input @1 name$ 21. @23 degree$ 20. ;

cards;

Nguyen, Melany Undergraduate degree

Mari, Leana Adrienne Graduate degree

Cawley, Anne Margaret Graduate degree

;

data enrollment;

set enrollment;

degree=tranwrd(degree, 'Graduate degree', 'Masters degree');

run;

proc print noobs;

run;

The output is

name degree

Nguyen, Melany Undergraduate degree

Mari, Leana Adrienne Masters degree

Cawley, Anne Margaret Masters degree