Cody’s Collection of Popular SAS® Programming Tasks and How to Tackle Them

Ron Cody
Contents

List of Programs ........................................................................................................................ ix

About This Book ......................................................................................................................... xv

About The Author ..................................................................................................................... xix

Acknowledgments .................................................................................................................... xxi

Chapter 1 Tasks Involving Conversion: Character to Numeric, Specific Values to Missing, and Changing Case .............................................................. 1

Introduction ............................................................................................................................... 1

Task: Converting character values to numeric values ............................................................. 2

Task: Converting character values to numeric values using a macro .................................. 3

Task: Converting a specific value such as 999 to a missing value for all numeric variables in a SAS data set ................................................................. 5

Task: Converting a specific value such as 'NA' to a missing value for all character variables in a SAS data set .............................................................. 7

Task: Changing all character values to either uppercase, lowercase, or proper case .......... 8

Task: Reading a numeric value that contains units such as Lbs. or Kgs. in the value .......... 9

Task: Solving part of the previous task using a Perl regular expression ......................... 10

Conclusion ............................................................................................................................... 11

Chapter 2 Grouping Data ...................................................................................................... 13

Introduction ............................................................................................................................. 13

Task: Grouping values using IF-THEN-ELSE statements ................................................. 13

Task: Grouping values using user-defined formats ............................................................... 14
Task: Creating groups using PROC RANK ................................................................. 15
Conclusion .................................................................................................................. 19

Chapter 3 Summarizing Data ................................................................................. 21
Introduction ................................................................................................................ 21
Task: Using PROC MEANS to create a data set containing summary information .................................................. 22
Task: Computing the mean of a variable broken down by values of another variable: Using a BY variable ............................................................. 23
Task: Computing the mean of a variable broken down by values of another variable: Using a CLASS statement ........................................................... 24
Task: Have PROC MEANS name the variables in the output data set automatically (the AUTONAME option) ................................................................ 25
Task: Creating multiple output data sets from PROC MEANS, each with a different combination of CLASS variables ................................................................ 26
Task: Combining summary information (a single mean) with detail data: Using a conditional SET statement ...................................................................... 29
Task: Combining summary information (a single mean) with detail data: Using PROC SQL ............................................................................................. 31
Task: Combining summary information (a single mean) with detail data: Using PROC SQL without using PROC MEANS .................................................... 32
Task: Combining summary information (a single mean) with detail data: Using a macro variable ....................................................................................... 33
Task: Combining summary data with detail data—for each category of a BY variable ........................................................................................................ 34
Conclusion ................................................................................................................ 36

Chapter 4 Combining and Updating SAS Data Sets ........................................... 37
Introduction .............................................................................................................. 37
Task: Concatenating two SAS data sets—Using a SET statement ......................... 38
Task: Concatenating two SAS data sets—Using PROC APPEND .......................... 40
Task: Concatenating two SAS data sets with character variables of different lengths ..................................................................................................... 40
Task: Concatenating two SAS data sets that contain character variables of different lengths and controlling the length of the character variables .......... 42
Task: Developing a macro to concatenate two SAS data sets that contain character variables of different lengths ............................................................. 43

Task: Updating a SAS data set using a transaction data set ............................................................. 47

Task: Using a MODIFY statement to update a master file from a transaction file .... 50

Task: Updating several variables using a transaction file created with an INPUT method called named input ............................................................. 50

Task: Matching names from two SAS data sets where the names may not be spelled the same (fuzzy merge) ............................................................. 53

Conclusion ............................................................................................................................. 56

Chapter 5 Creating Formats from SAS Data Sets ............................................................. 57

Introduction ............................................................................................................................ 57

Task: Using a SAS data set to create a format (by creating a control data set) ........ 57

Task: Adding new format values to an existing format ............................................................. 62

Conclusion ............................................................................................................................. 63

Chapter 6 Table Lookup Techniques ........................................................................... 65

Introduction ............................................................................................................................ 65

Task: Performing a one-way table lookup using a MERGE statement ........ 65

Task: Performing a one-way table lookup using user-defined informats ........ 67

Task: Creating an INFORMAT using a control data set ............................................................. 69

Task: Performing a one-way table lookup using a temporary array ........................................ 70

Task: Performing a two-way table lookup using a temporary array ........................................ 71

Conclusion ............................................................................................................................. 73

Chapter 7 Restructuring (Transposing) SAS Data Sets ........................................... 75

Introduction ............................................................................................................................ 75

Task: Converting a data set with one observation per subject into one with multiple observations per subject (using a DATA step) ............................................................. 76

Task: Converting a data set with one observation per subject into one with multiple observations per subject (using PROC TRANSPOSE) ............................................................. 77

Task: Converting a data set with multiple observations per subject into one with one observation per subject (using a DATA step) ............................................................. 79
Task: Converting a data set with multiple observations per subject into one with one observation per subject (using PROC TRANSPOSE) .......... 80

Conclusion ........................................................................................................................................ 81

Chapter 8 Tasks Involving Dates ..................................................................................................... 83

Introduction ....................................................................................................................................... 83
Task: Computing a person’s age, given his or her date of birth ......................................................... 83
Task: Computing a SAS date given a month, day, and year (even if the day value is missing) ........... 84

Conclusion ........................................................................................................................................ 85

Chapter 9 Data Cleaning Tasks ........................................................................................................ 87

Introduction ....................................................................................................................................... 87
Task: Looking for possible data errors using a given range ............................................................... 88
Task: Demonstrating a macro to report on outliers using fixed ranges .......................................... 89
Task: Demonstrating a macro that performs automatic outlier detection ......................................... 92
How the macro works ......................................................................................................................... 94

Conclusion ........................................................................................................................................ 96

Chapter 10 Reading Data with User-Defined INFORMATS ............................................................ 97

Introduction ....................................................................................................................................... 97
Task: Reading a combination of character and numeric data ............................................................. 97

Conclusion ........................................................................................................................................ 100

Chapter 11 Tasks Involving Multiple Observations per Subject ...................................................... 101

Introduction ....................................................................................................................................... 101
Task: Using PROC SORT to detect duplicate BY values or duplicate observations (records) .......... 102
Task: Extracting the first and last observation in a BY group ........................................................... 106
Task: Detecting duplicate BY values using a DATA step ............................................................... 108
Task: Identifying observations with exactly 'n' observations per subject ......................................... 109
Task: Computing differences between observations (for each subject) ......................................... 110
Task: Computing the difference between the first and last observation for each subject ................................................................. 112

Conclusion ............................................................................................................................................................................. 114

Chapter 12 Miscellaneous Tasks.......................................................... 115

Introduction .......................................................................................................................... 116

Task: Determining the number of observations in a SAS data set (using the NOBS= SET option) ......................................................... 116

Task: Determining the number of observations in a SAS data set and assigning the value to a macro variable ................................................................. 117

Task: Determining the number of observations in a SAS data set (using library tables) ................................................................. 118

Task: Determining the number of observations in a SAS data set (using SAS functions) ................................................................. 119

Task: Counting the number of a specific response in a list of variables .................. 120

Task: Computing a moving average ........................................................................ 122

Task: Presenting a macro to compute a moving average .............................................. 124

Task: Replacing the first eight digits of a credit card number with asterisks .......................... 126

Task: Sorting within an observation (using the ORDINAL function) ......................... 127

Task: Sorting within an observation (using CALL SORTN) ............................................. 128

Task: Computing the average of the 'n' highest scores ................................................. 129

Task: Extracting the first and last name (and possibly a middle name) from a variable containing the first and last name (and possibly a middle name) in a single variable ................................................................. 130

Index ............................................................................................................................................................................. 133

Chapter 1: Tasks Involving Conversion: Character to Numeric, Specific Values to Missing, and Changing Case

Introduction

This chapter contains programs to perform character-to-numeric conversion, one of the most common tasks you will face as a SAS programmer. You will see a sample program as well as a useful macro that accomplishes this goal.

Another task that you will probably face is converting a specific numeric value such as 999 or a specific text value such as 'NA' to a SAS missing value.

In this chapter, you will also see how to convert every character variable to a specific case, such as uppercase.
The last task in this chapter demonstrates how to read data values that contain units, such as 100Lbs. or 50Kgs. and create a numeric variable with all of the values using the same units.

**Task: Converting character values to numeric values**

**Keywords**
- Character-to-numeric conversion
- Swap and Drop

How many times have you been given a SAS data set with variables such as Height or Weight but, instead of being numeric variables, they are stored as character? The following example describes how to convert these character variables to numeric variables, maintaining the original variable names.

For this example, you start out with a SAS data set called Char_values. Here is a listing:

<table>
<thead>
<tr>
<th>Age</th>
<th>Weight</th>
<th>Gender</th>
<th>DOB</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>150</td>
<td>M</td>
<td>10/21/1983</td>
</tr>
<tr>
<td>67</td>
<td>220</td>
<td>M</td>
<td>09/12/2001</td>
</tr>
<tr>
<td>77</td>
<td>101</td>
<td>F</td>
<td>05/06/1977</td>
</tr>
</tbody>
</table>

If you run PROC CONTENTS on this data set, you see that Age and Weight are character variables. The following program performs the conversion:

**Program 1.1: Converting character values to numeric values**

```sas
*Converting character values to numeric;

data Num_Values;
  set Char_Values(rename=(Age = C_Age
                           Weight = C_Weight));
  Age = input(C_Age,best12.);
  Weight = input(C_Weight,best12.);
  drop C_:;
run;
```

The “trick” here is to rename the variables as they are read from the input data set. This allows you to use the original variable names for the resulting numeric variables. The character-to-numeric conversion is performed using the INPUT function. You don’t have to worry if the INFORMAT used in the INPUT function represents more digits than you need—unlike an INPUT statement, you can never read past the end of a character value when using the INPUT function.
Notice the variable list on the DROP statement _C_: The colon acts as a wildcard suffix. _C_: represents all variables that begin with the characters C followed by an underscore.

The resulting data set has exactly the same variables as the original data set except the two variables Age and Weight are now numeric. A partial listing from PROC CONTENTS confirms this:

<table>
<thead>
<tr>
<th></th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Age</td>
<td>Num</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DOB</td>
<td>Num</td>
<td>8</td>
<td>MMDDYY10.</td>
</tr>
<tr>
<td>1</td>
<td>Gender</td>
<td>Char</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Weight</td>
<td>Num</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

**Task: Converting character values to numeric values using a macro**

**Keywords**
- Character-to-numeric conversion
- Conversion macro

Because character-to-numeric conversion is required in so many situations, this chapter offers you a macro that performs the conversion automatically. As in the previous program, the resulting data set uses the same variable names as in the original data set that contains the character variables. Here is the macro, followed by an explanation:

**Program 1.2: Presenting a macro to perform character-to-numeric conversion**

```sas
*Macro to convert selected character variables to numeric variables;
%macro char_to_num(In_dsn=, /*Name of the input data set*/
Out_dsn=, /*Name of the output data set*/
Var_list= /*List of character variables that you want to convert from character to numeric, separated by spaces*/);

/*Check for null var list */
%if &var_list ne %then %do;
/*Count the number of variables in the list */
%let n=%sysfunc(countw(&var_list));
data &Out_dsn;
    set &In_dsn(rename=(
        %do i = 1 %to &n;
            %sysfunc(lowercase(%sysfunc(putn(&i,1w)))) &var_list
        %end;
    ));
run;
%mend;
```
Cody’s Collection of Popular SAS Programming Tasks and How to Tackle Them

```sas
/* break up list into variable names */
%let Var = %scan(&Var_list,&i);
/* Rename each variable name to C_ variable name */
&Var = C_&Var
%end;
%do i = 1 %to &n;
%let Var = %scan(&Var_list,&i);
&Var = input(C_&Var,best12.);
%end;
drop C_:;
run;
%mend char_to_num;
```

The calling arguments in this macro are the names of the input and output data sets and a list of the variables that you wish to convert from character to numeric. You enter the names of each variable in this list, separated by spaces.

The first task of the macro is to rename each of the original variable names by appending the prefix `C_` to each of the names. To determine how many variable names there are in `&Var_list`, you use the COUNTW function. This function computes the number of words in a string. To obtain each of the variable names, you use the `%SCAN` macro function. This function works in the same way as the regular non-macro SCAN function. The first argument is the list of variable names. The second argument specifies which “word” you want in the string. The macro uses a `%DO` loop to extract each of the individual variable names.

The next `%DO` loop performs the character-to-numeric conversion using the INPUT function. Notice that the first argument of the INPUT function is the original variable name with the `C_` prefix added. Finally, a DROP statement deletes all of the `C_` variables.

To test the macro, you can use the original data set `Char_values` and enter `Age` and `Weight` as the argument of `Var_List`. Here is the code:

**Program 1.3: Testing the character-to-numeric conversion macro**

```
*Test the macro;
%char_to_num(In_dsn=char_values, Out_dsn=Num_values, Var_list=Age Weight)
```

After you run the macro, the output data set (Num_values) is identical to the one created by the previous program.
**Task: Converting a specific value such as 999 to a missing value for all numeric variables in a SAS data set**

**Keywords**
- Numeric variables
- `_numeric_`
- Array

You will find numerous occasions where you need to perform an operation on all numeric (or character) variables in a SAS data set. For example, you may have a SAS data set where specific values, such as 999 or 9999, were used to represent a missing value. In the character domain, you may want to convert all character values to uppercase or convert a specific value such as ‘NA’ to a SAS missing value. The approach to all of these tasks is the same. You create an array of all numeric or character variables. Once you do this, you can then use a DO loop to perform any operation you desire on all of the variables in the array.

This first example converts a value of 999 to a SAS missing value for all the numeric variables in data set Demographic.

A listing of data set Demographic is shown here:

<table>
<thead>
<tr>
<th>Subj</th>
<th>Score</th>
<th>Weight</th>
<th>Heart_Rate</th>
<th>DOB</th>
<th>Gender</th>
<th>Party</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>70</td>
<td>999</td>
<td>76</td>
<td>04NOV1955</td>
<td>Male</td>
<td>NA</td>
</tr>
<tr>
<td>2</td>
<td>26</td>
<td>160</td>
<td>62</td>
<td>08APR1955</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>3</td>
<td>71</td>
<td>195</td>
<td>71</td>
<td>20JUL1955</td>
<td>male</td>
<td>na</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>132</td>
<td>74</td>
<td>08JAN1955</td>
<td>Male</td>
<td>Republican</td>
</tr>
<tr>
<td>5</td>
<td>999</td>
<td>181</td>
<td>62</td>
<td>15AUG1951</td>
<td>Female</td>
<td>Democrat</td>
</tr>
<tr>
<td>6</td>
<td>62</td>
<td>71</td>
<td>52</td>
<td>24JAN1950</td>
<td>Male</td>
<td>democrat</td>
</tr>
<tr>
<td>7</td>
<td>24</td>
<td>136</td>
<td>72</td>
<td>26NOV1950</td>
<td>Female</td>
<td>democrat</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>174</td>
<td>71</td>
<td>08NOV1950</td>
<td>Female</td>
<td>democrat</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>172</td>
<td>47</td>
<td>28DEC1951</td>
<td>Male</td>
<td>Democrat</td>
</tr>
<tr>
<td>10</td>
<td>94</td>
<td>173</td>
<td>999</td>
<td>06MAY1953</td>
<td>Male</td>
<td>republican</td>
</tr>
<tr>
<td>11</td>
<td>99</td>
<td>170</td>
<td>63</td>
<td>27FEB1950</td>
<td>na</td>
<td>NA</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>133</td>
<td>63</td>
<td>18MAR1954</td>
<td>Male</td>
<td>democrat</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>131</td>
<td>60</td>
<td>26MAR1951</td>
<td>Female</td>
<td>republican</td>
</tr>
</tbody>
</table>
You will use this data set for several of the tasks in this chapter. For this example, notice that there are several values of 999 for the variables Score, Weight, and Heart_Rate.

Here is the code that performs the conversion:

**Program 1.4: Converting a specific value such as 999 to a missing value for all numeric variables in a SAS data set**

```
*Converting a specific value such as 999 to a missing value for all numeric variables in a SAS data set;

data Num_missing;
  set Demographic;
  array Nums[*] _numeric_;
  do i = 1 to dim(Nums);
    if Nums[i] = 999 then Nums[i] = .;
  end;
  drop i;
run;
```

The key to this program, as well as several programs to follow, is to create an array using the keyword _NUMERIC_. When used in a DATA step, _NUMERIC_ represents all the numeric variables that have been defined up to that point in the DATA step. Since the ARRAY statement follows the SET statement, the Nums array contains all of the numeric variables in data set Demographic (Subj, Score, Heart_Rate, and DOB). To make this important point clear, had you placed the ARRAY statement before the SET statement, the array Nums would not contain any variables.

You certainly do not want to have to count all the numeric variables in a large data set. Therefore, you use an asterisk in the brackets following the array name. When you do this, SAS will count the number of variables for you. But, what value do you use in the DO loop? You can use the DIM (dimension) function to determine how many variables are in the array. Your work is almost finished. All you need to do now is to check for values of 999 and convert them to a SAS numeric missing value. Don’t forget to drop the DO loop counter.
The first five observations in data set Num_missing are shown next, to demonstrate that the program worked as expected:

<table>
<thead>
<tr>
<th>Subj</th>
<th>Score</th>
<th>Weight</th>
<th>Heart_Rate</th>
<th>DOB</th>
<th>Gender</th>
<th>Party</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>70</td>
<td>.</td>
<td>76</td>
<td>04NOV1955</td>
<td>Male</td>
<td>NA</td>
</tr>
<tr>
<td>2</td>
<td>26</td>
<td>160</td>
<td>62</td>
<td>08APR1955</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>3</td>
<td>71</td>
<td>195</td>
<td>71</td>
<td>20JUL1955</td>
<td>Male</td>
<td>na</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>132</td>
<td>74</td>
<td>08JAN1955</td>
<td>Male</td>
<td>Republican</td>
</tr>
<tr>
<td>5</td>
<td>.</td>
<td>181</td>
<td>62</td>
<td>15AUG1951</td>
<td>Female</td>
<td>Democrat</td>
</tr>
</tbody>
</table>

**Task: Converting a specific value such as 'NA' to a missing value for all character variables in a SAS data set**

**Keywords**
- Character variables
- character_Array

This task is similar to the previous task. The difference is that you want to convert a specified character value to a SAS character missing value. All you need to do is use the SAS keyword _CHARACTER_ to create an array of all character variables. Here is the program:

Program 1.5: Converting a specific value such as ‘NA’ to a missing value for all character variables in a SAS data set

```sas
*Converting a specific value such as "NA" to a missing value for all character variables in a SAS data set;
data Char_missing;
  set Demographic;
  array Chars[*] _character_;
  do i = 1 to dim(Chars);
    if Chars[i] in ('NA' 'na') then Chars[i] = ' ';
  end;
  drop i;
run;
```

Array Chars contains all the character variables in data set Demographic (in this case, Gender and Party). As in the previous task, the DIM function returns the number of variables in the array. To make the program more general, it looks for uppercase or lowercase values of 'NA'. Here is a listing of the first five observations in data set Char_missing:
Task: Changing all character values to either uppercase, lowercase, or proper case

Keywords

- Uppercase
- Lowercase
- Proper case

_In a similar manner to the previous program, you can use an array of all your character variables to convert them all to a unified case: uppercase, lowercase, or proper case. Please refer to the previous program if you would like an explanation of this program. As you can see, this program is converting all the character values in the Demographic data set to uppercase. The two other functions that convert character values to lowercase or proper case are LOWCASE and PROPCASE, respectively. Here is the program:

Program 1.6: Changing case for all character variables in a SAS data set

```sas
*Converting all character values to uppercase (or lower- or proper-case);
data Upper;
set Demographic;
array Chars[*] _character_
  do i = 1 to dim(Chars);
    Chars[i] = upcase(Chars[i]);
  end;
drop i;
run;
```

If the character variables you are dealing with represent names and addresses, after you have converted all the values to a consistent case, you may want to take the additional step and use the COMPBL function to convert all multiple blanks to a single blank, to help standardize the names and addresses.
Task: Reading a numeric value that contains units such as Lbs. or Kgs. in the value

Keywords
- Character-to-numeric conversion
- Removing units from a value
- Extracting digits from a string
- COMPRESS function
- SCAN function

Data set Units contains a character variable called Weight that includes units such as Lbs. and Kgs. (pounds and kilograms). To add insult to injury, the variable Height also contains units and it is expressed in feet and inches (sometimes the inches value is missing when the inches value is zero). A listing of data set Units is shown here:

<table>
<thead>
<tr>
<th>Subj</th>
<th>Weight</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>80kgs</td>
<td>5ft 3in</td>
</tr>
<tr>
<td>002</td>
<td>190lbs</td>
<td>6' 1&quot;</td>
</tr>
<tr>
<td>003</td>
<td>70KG.</td>
<td>5ft 11in</td>
</tr>
<tr>
<td>004</td>
<td>177LbS.</td>
<td>5' 11&quot;</td>
</tr>
<tr>
<td>005</td>
<td>100kgs</td>
<td>6ft</td>
</tr>
</tbody>
</table>

Notice that the Weight units are not always in the same case and some of the units end in periods. For Height, the abbreviation 'ft' or 'in' is used; sometimes a single quote and double quote represent feet and inches.

You would like to create two new variables (Weight_Lbs and Height_Inches) that are numeric variables and are equal to the weight in pounds and the height in inches, respectively. Here is the program:

Program 1.7: Reading data values that contain units

```
*Reading data values that contain units;
data No_Units;
  set Units;
  Weight_Lbs = input(compress(Weight,'kd'),12.);
  if findc(Weight,'k','i') then Weight_lbs = Weight_lbs*2.2;
  Height = compress(Height,'kds');
  Feet = input(scan(Height,1,' '),12.);
  Inches = input(scan(Height,2,' '),12.);
```
You start by extracting the digits from Weight using the COMPRESS function with the modifiers 'kd' (keep digits). It is important to include two commas following the first argument of the COMPRESS function so that the function interprets 'kd' as modifiers and not the second argument to the COMPRESS function that is used to list the characters you want to compress from a string. Since the result of the COMPRESS function is a character value, you use the INPUT function to perform the character-to-numeric conversion. All you need to do is test the original variable (Weight) to see if it contains a 'K' in uppercase or lowercase. Use the FINDC function with the 'i' modifier (ignore case) to do this. If you find a 'K', you multiply by 2.2 to convert from kilograms to pounds.

The Height variable presents more of a challenge. You first use the COMPRESS function with three modifiers, 'kds' (keep digits and space characters). The variable Height now contains two sets of digits (or only a single digit if there are zero inches) and can use the SCAN function to extract the feet and inch values. The SCAN function returns a missing value for Inches if Height only contains a single number (feet). You can now add 12 times the feet plus the number of inches to obtain the height in inches. Here is the listing of the data set No_Units:

<table>
<thead>
<tr>
<th>Subj</th>
<th>Weight</th>
<th>Height</th>
<th>Weight_Lbs</th>
<th>Height_Inches</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>80kgs</td>
<td>5 3</td>
<td>176</td>
<td>63</td>
</tr>
<tr>
<td>002</td>
<td>190lbs</td>
<td>6 1</td>
<td>190</td>
<td>73</td>
</tr>
<tr>
<td>003</td>
<td>70KG.</td>
<td>5 11</td>
<td>154</td>
<td>71</td>
</tr>
<tr>
<td>004</td>
<td>177LbS.</td>
<td>5 11</td>
<td>177</td>
<td>71</td>
</tr>
<tr>
<td>005</td>
<td>100kgs</td>
<td>6</td>
<td>220</td>
<td>72</td>
</tr>
</tbody>
</table>

Solving this task without the COMPRESS and SCAN functions would certainly be a challenge—with these functions, it’s a snap.

---

**Task: Solving part of the previous task using a Perl regular expression**

**Keywords**
- Removing units from a value
- Extracting digits from a string
- Perl regular expression
My younger son, who is a wizard at programming, suggested I solve this problem using a Perl regular expression. This solution is not simpler than the previous solution, but it demonstrates the versatility of regular expressions.

You start by using PRXPARSE to compile the regular expression:

```perl
/^\((\d+) (\D)\)/
```

This regex (this is what Perl programmers call regular expressions) is looking for one or more digits followed by a non-digit. The `^` in the beginning of the expression says to start the search at the beginning of the string. The digit and non-digit values will be placed in capture buffers because each of these expressions is in a set of parentheses. You use the PRXMATCH function to search for the pattern of a number followed by a non-number. The PRXPOSN function extracts the values in each of the capture buffers. The INPUT function performs the character-to-numeric conversion as in the previous task.

If the value in the second capture buffer is a 'K', you perform the kilogram to pound conversion.

**Program 1.8: Using a Perl regular expression to extract the digit and units part of a character value**

```sas
*Solution using Perl Regular expressions;
data No_Units;
set Units(drop=Height);
if _n_ = 1 then do;
  Regex = "/^\((\d+) (\D)\)/";
  re = prxparse(Regex);
end;
retain re;
if prxmatch(re,Weight) then do;
  Weight_Lbs = input(prxposn(re,1,Weight),8.);
  Units = prxposn(re,2,Weight);
  if upcase(Units) = 'K' then Weight_Lbs = Weight_Lbs*2.2;
end;
keep Subj Weight Weight_Lbs;
run;
```

The resulting data set contains values for Weight_Lbs that are identical to the values in the previous task.

**Conclusion**

It is quite likely that you will need to perform one or more of the tasks described in this chapter on a regular basis. Since the character-to-numeric conversion is one of the most common tasks, you may choose to store the conversion macro in your macro library.

Also keep in mind that using the special keywords _NUMERIC_ and _CHARACTER_ to define an array can save you immense time when you need to perform an operation on all character or numeric variables in a data set.
Index

A

age computation using birth date 83–84
_ALL_ sort option 105–106
APPEND procedure 37, 40, 41–42, 56, 90
arrays, temporary 70–73
ARRAY statement 6, 7, 70, 128
ATTRN function 120
Auto_Outlier macro 92–96
automatic outlier detection 92–96
AUTONAME option of MEANS procedure 25–26
average, computing moving 122–123, 124–125
average of `n' highest scores, computing 129

B

blanks, converting multiple to single 8
BY group, extracting first or last observation 106–107
BY statement 35, 49, 78, 106–107
BY values, detecting duplicate 102–106, 108

C

CALL SORTN call routine 128–129
CALL SYMPUTX statement 33, 117–118
Cartesian product 31, 53–54
case, changing character values 5, 8, 55, 121
CATS function 121, 126
_CHARACTER_ keyword 7–8, 11
character data
arrays of 7
converting case 5, 8, 55, 121
converting for all variables in data set 5
converting to numeric values 2–4, 5–7, 9–10, 11, 68
converting to specific value 5, 7–8
masking 126
reading with informats 98–100
reading with traditional approach 97–98
replacing in string 126
character variables, creating with PUT function 67
CHARTYPE option of MEANS procedure 26–29
CLASS statement 24, 34–35
CLOSE function 120
CNTLIN= option in FORMAT procedure 59–60, 70
CNTLOUT= option in FORMAT procedure 62
COALESCE function 85
COMPBL function 8
COMPRESS function, to extract Weight digits 10
concatenating data sets 37, 38–40, 42–47
conditional SET statement 29–31, 34
CONTENTS procedure 39, 43, 44
control data sets 57–61, 62–63, 69–70
converting characters
all variables in data set 5
to desired case 5, 8, 55, 121
to numeric values 2–4, 5–7, 9–10, 11, 68
to specific value 5, 7–8
converting macro variables, to numeric values 118
converting numeric variables
all variables in data set 5–7
from character values 2–4, 5–7, 9–10, 11, 68
from macro variables 118
COUNTC function 121–122
counting numeric variables 6
counting specific values in list of variables 120–122
COUNTW function 4
credit card number, masking 126

D

DATA= attribute for TRANSPOSE procedure 78
data cleaning 50–52, 56, 89–91, 92–96
See also data errors
Index

data conversion
  all numeric variables in data set 5–7
  character to numeric 2–4, 5–7, 9–10, 11, 68
  macro variables to numeric 118
  to desired case 5, 8, 55, 121
  to specific value 5, 7–8
data errors 38, 87, 88, 89–91
data sets
  adding data to existing 37
  concatenating with APPEND procedure 42
  concatenating with different character
    variable lengths 37, 38
  concatenating with different character
    variable lengths using APPEND
    procedure 40–42
  concatenating with different character
    variable lengths using macro 43–47
  concatenating with different character
    variable lengths using SET and LENGTH
    statements 42–43
  concatenating with SET statement 37, 38–40, 42–43
  control 57–61, 62–63, 69–70
  correcting data errors 38
  restructuring 76–77, 79–80, 81
  updating using transaction data set 47–49
  updating with new values 38
DATA step
  concatenating data sets with different variable
    lengths 45
  _NUMERIC_ keyword 6, 11
  restructuring data sets 76–77, 79–80, 81
  stopping 118
  summarized data, combining with detail data 34
  summarizing data 30, 31
  dates 83–85
  deciles, dividing values into 15
    See also RANK procedure
  Delobs variable in library tables 119
  differences between observations, computing
    110–112
  DIF function 111
  digits, extracting from strings 9–10, 11
  DIM (dimension) function, use in DO loop 6, 7
  DO loops 4, 6, 7
  DROP= data set option 79
  DROP= option in MEANS procedure 28
  DROP statement, use of colon in variable list 2, 3
  duplicate BY values, detecting 102–106, 108
  duplicates 102–106, 108
  dynamic naming of variables 128
E
  END= option in MERGE statement 45
  END= option in SET statement 61, 62
  End variable in control data set 58, 69
  enhanced numeric informat 99
  ERROR MERGENOBY option 44, 45
  errors macro for range checking 89–91
  %EVAL function 118
  EXIST function 120
  extracting digits from string 9–10, 11
F
  file I/O functions 119–120
  FINDC function 10
  FIRSTOBS= data set option 118
  first observation 106–107, 112–113
  First. variables 106–107, 108
  Fmtname variable in control data set 58, 59
  FORCE option, use when combining data sets
    with different variable lengths 41–42
  FORMAT procedure 59–63, 70, 98–100
  formats
    See also informats
    adding formats to 62–63
    creating 57, 68, 99
    grouping values 14–15
    using control data set 57–61, 69–70
  _FREQ_ variable of MEANS procedure 23, 26
  fuzzy matches, between two data sets 38, 53–56
  fuzzy merge 53–56
G
  grand mean 22, 24, 27, 29–31
  grouping values 13–17, 19
GROUPS= option in RANK procedure  16, 17–18

H
Height variable in Units data set  9–10
HISTOGRAM statement  96
HLO variable in control data set  58, 60–61, 62

I
if-then-else statements, grouping values  13–14
%INCLUDE statement, for concatenating data sets with different variable lengths  43
inexact matching  38, 53–56
informat  57–61, 67–70, 98–100
INFORMAT statement, in INPUT statement  51
INPUT function
  creating informat  68
  informats in  67
  named input method  50–52, 56
  reading past end of character, inability  2
  to convert character values to numeric  10, 11
input/output (I/O) functions  119–120
INVALUE statement, to create informat  68, 99

J
Jordan, Mark  33, 85
JUST option, in INVALUE statement  99

K
KEEP= option in RANK procedure  17
KEEP= option of MEANS procedure  22, 23
kilograms, converting to pounds  10

L
Label variable in control data set  58, 61
LAG function  111, 113, 122, 124–125
Langston, Rick  129
last observation  106–107, 112–113
Last. variables  106–107, 108
LENGTH statement  14, 43, 45, 51, 62
%LET statement  90, 118
libnames format  119
library tables, determining number of observations  118–119
LOWCASE function  8
lowercase, converting characters to  8

M
macros
  concatenate data sets with different variable lengths  43–47
  convert character values to numeric  3–4
  converting program to  46
  moving average  124–125
  MPRINT option to view code  125
  outlier detection  92–96
  range checking  89–91
  reporting  89, 90–91
  running, example  4
  %SCAN function  4
macro variables  33–34, 117–118
masking characters  126
master file, updating from transaction file  50–52
MDY function  84–85
mean  32–33
MEAN function  32–33, 125, 129
MEANS procedure
  automatic naming of variables in output  25–26
  BY statement  35
  BY variable  23
  CLASS statement  24, 34–35
  computing mean by values of another variable  24–25
  creating multiple output data sets  26–29
  options  22, 23, 24–29, 30, 35
  OUTPUT statement  22, 25–26
  sorting  35
  SQL procedure as alternative  32–33
  summarizing data  22, 23, 24–29
  variables  23, 24, 26–27, 28
  VAR statement  22
memname format  119
merge, fuzzy  53–56
MERGENOBY system option  44–45
metadata, getting data set 118
MISSING function 14, 85, 122
missing values 5–8, 30, 127
MODIFY statement, updating using transaction file 50, 52
moving average, computing 122–123, 124–125
MPRINT option to view macro generated code 125
multiple observations per subject 76–81, 101, 110–113

N
named input method 50–52, 56
names, parsing 130–131
NLOBS argument of ATTRN function 120
NOBS= option in SET statement 116–118, 119
"n" observations per subject 109–110
Nobs variable in library tables 119
NODUPKEY option in SORT procedure 102–103
NODUPECS option in SORT procedure 103–106
NOPRINT option of MEANS procedure 22, 30
NOWARN MERGENOBY option 44–45
number of observations per subject, detecting 109–110, 116–118, 119
_NUMERIC_ keyword 6, 11
numeric informat, enhanced 99
numeric values
  converting all variables in data set 5–7
  converting from character values 2–4, 5–7, 9–10, 11, 68
  converting from macro variables 118
  creating with informats 68
  extracting from strings 9–10, 11
  reading 97–100
numeric variables, counting 6
NVAR argument of ATTRN function 120
NWAY option of MEANS procedure 24–25, 26, 35

O
observations 109–113, 116–120
  See also restructuring data sets
ODS OUTPUT statement 94
one-way table lookup, using temporary arrays 70–71
OPEN function 120
ORDINAL function 127–128, 129
OUT= attribute for TRANSPOSE procedure 78
OUT= keyword for MEANS procedure 22, 78
outliers, data 87, 88, 89, 92–96
OUTPUT statement 22, 25–26, 44

P
parsing strings 130–131
PDV (program data vector), effect of observation processing on 106
Perl regular expressions (regex), to extract digits 11
PREFIX= option for TRANSPOSE procedure 80–81
PRINT procedure, REPORT procedure compared 60
program data vector (PDV), effect of observation processing on 106
PROPCLASS function 8
proper case, converting characters to 8
PRXMATCH function 11
PRXPARSE function 11
PRXPOSN function 11
PUT function 14–15, 67, 68

Q
questionnaires, counting specific values in list of variables 120–122

R
range checking 88, 89
RANK procedure 15–18
RANKS statement, in RANK procedure 16, 17
reading data 2, 97–100
regex (Perl regular expressions) 11
removing units from value 9–10, 11
RENAME= data set option 59, 79
renaming variables 2, 3–4
Index 137

report macro for reporting range errors 89, 90–91
REPORT procedure 60
resources on SAS, web site 125
restructuring data sets
  multiple observations per subject into one observation per subject 79–81
  need for 75–76
  one observation per subject into multiple observations per subject 76–79
RETAI N statement 59, 106, 112–113

S
SAME_ keyword in INVALUE statement 99
SAS 9.3, version changes 84
SAS dates, computing with MDY function 84–85
SAS Global Forum 29
SASHELP.VTABLE, getting metadata from 118–119
SAS Users Group International (SUGI) 29
SCAN function 4, 10, 126, 130–131
SELECT statement, for documentation 60
SET statement
  adding to existing data sets 37
  APPEND procedure compared 56
  ARRAY statement with 6
  concatenating data sets 37, 38–40, 42–43
  conditional 29–31, 34
  followed by BY statement 106–107
  options 61, 62, 116–118, 119
SET SUMMARY statement 30, 31
single observations per subject
  converting from multiple observations per subject data set to 79–81
  converting to multiple observations per subject data set 76–79
SMALLEST function 127
sorting 35, 66–67, 127–129
SORTN call routine 128–129
SORT procedure 35, 102–106, 108
SPEDIS (spelling distance) function 53, 55
SQL procedure 31–34, 53–54
Start variable in control data set 58, 69
statistical methods, for identifying possible outliers 87, 92
STOP statement in DATA step 118
strings 9–10, 11, 126, 130–131
  See also character data
SUBSTR function 126
SUGI (SAS Users Group International) 29
summarized data, combining with detail data conditional SET statement 29–31
  DATA step 34
  macro variable 33–34
  MERGE statement 34–36
  SQL procedure 31–34
summarizing values
  in DATA step 30, 31
  macro variable 33–34
  MEANS procedure 22, 23, 24–29
  SQL procedure 31–33
  uses 22
SUM statement 110
swap and drop 2–3
SYMPUTX call statement 33, 117–118

T
table lookup 65–68, 70–73
tables, creating with SQL procedure 31
TABULATE procedure 21
_TEMPORARY_ keyword in ARRAY statement 70
temporary arrays 70–73
transaction data set, updating data sets with 47–49
transaction file, updating master file using 50–52
TRANSPOSE procedure 77–81
transposing data sets
  See restructuring data sets
trends, calculating 122–123, 124–125
trimmed statistics for automatic outlier detection 92, 93
trim value for outlier detection 96
two-way table lookup, using temporary arrays 71–73
_TYPE_ variable in MEANS procedure  23, 24, 26–27, 28
Type variable in control data set  58, 59

U
units, removing from value  9–10, 11
Units data set  9
UNIVARIATE procedure  21, 92, 96
UPCASE function  8, 55, 121
UPCASE option in INVALUE statement  99
UPDATE statement  49, 52
updating master file  50–52, 56
uppercase, converting characters to  5, 8, 55, 121

V
values, grouping  13–17, 19
VALUE statement, to create formats,  68, 99
variables
  macro  33–34, 117–118
  naming  2, 3–4, 128
  representing groups  13–17, 19
VAR statement in MEANS procedure  22
VAR statement in RANK procedure  16, 17
VAR statement in TRANSPOSE procedure  80
VNAME function  94

W
WARN MERGENOBY option  44, 45
web site, SAS  125
Weight variable in Units data set  9–10
WHERE= data set option  44, 79

Y
YRDIF function  84
About The Author

Ron Cody, EdD, is a retired professor from the Robert Wood Johnson Medical School who now works as a private consultant and a national instructor for SAS. A SAS user since 1977, Ron’s extensive knowledge and innovative style have made him a popular presenter at local, regional, and national SAS conferences. He has authored or co-authored numerous books.

Learn more about this author by visiting his author page at support.sas.com/cody. There you can download free chapters, access example code and data, read the latest reviews, get updates, and more.
Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.