Overview

- Understand how SAS distinguishes between character and numeric variables
- Identify character handling functions to clean and prepare character variables for linkage
- Apply these functions to actual situations
- Choose the right functions for each situation

Basic Principles

- Data linkage success is best when data are clean
- By clean we mean:
  - Free from unnecessary symbols and characters
  - Broken down into the most desirable units
  - Both sets of data are as similar as possible
  - Apply cleaning steps to both datasets to achieve this
- Identify any systematic issues with your data that may impact linkage success
Character Data

- Data that are expressed in alphanumeric characters rather than solely as numbers
  - AKA strings or string data
- Includes uppercase and lowercase letters, symbols, and even numbers (more on this later)
- Lengths can vary
  - Default length for numeric data is 8 bytes
  - Character variables can be set to anything
- Special functions in SAS to work with character data

Examples of Character Data

- Names
- Addresses (including house number and street, city, state)
- ICD Codes
- Numbers that have leading zeros or have symbols in them:
  - Social Security Numbers
  - Phone Numbers
  - Zip Codes

Character Data Cleaning Issues

- Symbols used as delimiters
- Symbols used in phone numbers and SSNs
  - SSNs with dashes are fine for matching as long as both datasets have them in the same format
- Punctuation
- Numbers stored as characters
- Words in the wrong order
- Information split across many fields
- Inappropriate characters
Cleaning Issue Examples

- Symbols used as delimiters:
  - Caroline*Stampfel*C
- Symbols used in phone numbers and SSNs
  - 121-32-3544
  - 804.555.1212 or 804-555-1212
- Punctuation
  - 21-09 252nd St.
  - Richmond, VA

Cleaning Issue Examples

- Numbers stored as characters
  - Dates that are stored as text: 08-21-09
  - Numeric values with symbols: 2002-2005
  - Zip Codes: 05626 – the leading zero would be lost if this zip code were stored as a number. Same for SSN, some ICD codes
- Words in the wrong order
  - Stampfel, Caroline
- Information split across many fields
  - Addresses, names, etc.
- Inappropriate characters
  - Carol O'Toole (zero instead of O)

Character Handling Functions

- Allow you to manipulate character data
- Clean out values, symbols that are not needed
- Can be nested within each other
  - Use in combination to minimize steps
- Always make new variables
- In addition to cleaning, you can also use character functions to search character fields
### Selected Character Functions

<table>
<thead>
<tr>
<th>Cleaning functions</th>
<th>Useful functions</th>
<th>Search Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCAN()</td>
<td>TRIM()</td>
<td>INDEX()</td>
</tr>
<tr>
<td>SUBSTR()</td>
<td>TRANSLATE()</td>
<td>FIND()</td>
</tr>
<tr>
<td>COMPRESS()</td>
<td>TRANWRD()</td>
<td>FINDW()</td>
</tr>
<tr>
<td>TRIM()</td>
<td>Concatenate:</td>
<td></td>
</tr>
<tr>
<td>Concatenate:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### LENGTH Statement

- **What does it do?**
  - Allows you to set the length of character values so they do not default to something too large (200)
  - Syntax:
    - Length varname $ lengthinnumbers;
    - Creates the new variable with the desired length so you can “fill” it with your extracted pieces

### SCAN()

- **What does it do?**
  - Allows you to extract a “word” from a character variable based its location using delimiters
  - Syntax:
    - SCAN(charvar, n, 'list of delimiters')
    - Looks for the Nth “word” (anything between two delimiters) in a variable based on the delimeter(s) you specify in the list
    - Use this when you know the delimiters, the length of the “words” in the variable can vary
    - If you know the max length, use a length statement
SCAN() Example

- For a variable called NAME, which is set up as Firstname*MiddleName*LastName, the * is the delimiter, max length is 15
  - Length firstname $ 15 lastname $ 15;
- To separate out the first name, create a new variable called Firstname and use SCAN():
  - Firstname = SCAN(Name,1, " ");
- To separate out the last name, create a new variable called Lastname:
  - Lastname = SCAN(Name,3, " ");

SUBSTR()

- What does it do?
  - Allows you to extract one or more characters from a character variable based on the location of the string.
- Syntax:
  - SUBSTR(charvar, start position, length)
  - Extracts characters from the start position and goes the number of places specified in "length"
- Use this when you know the know the exact length and position of the words you want to extract, or there are no delimiters

SUBSTR() Example

- For a variable called DOB_txt, which is a date stored as text, and has the variable structure: MMDDYYYY
- To extract out the day (DD)
  - Day = SUBSTR(DOB_txt,3, 2);
  - Starts at position 3 and extracts the characters two places out
Left hand SUBSTR()

- Used on the left side of the =, SUBSTR() allows you to substitute specific characters at a specific position
- TRANWRD() does this anywhere in the string.
- SUBSTR() is good if you know that some things always have undesired characters in the same location
  - Example: BB and BG preceding infant names with dashes, use COMPRESS() to remove the dashes
- If you do not specify a symbol, SUBSTR() leaves blanks, which may not be desirable

COMPRESS()

- What does it do?
  - Removes stuff from a character var. Stuff can include symbols, numbers, letters, blanks.
- Syntax:
  - COMPRESS(charvar, ‘List of characters’)
- Default is to remove blanks if no characters specified
- COMPRESS() also has options to remove whole sets of characters

COMPRESS() options

COMPRESS(charvar, ‘List of characters’)
  d – Adds numerals (digits) to the list of characters to be deleted
  a – Adds upper and lowercase letters to the list of characters to be deleted
  i – Ignores case
  k – Keeps listed characters instead of removing them
  s – Adds blanks, tabs, line-feeds, or carriage returns to the list of characters to be deleted
  p – Adds punctuation to the list of characters to be deleted
COMPRESS() Examples

- For a variable called SSN, which has a variable structure: NNN-NN-NNNN
  - To remove the dashes
    - SSN_new1 = COMPRESS(SSN, '-');
  - To remove the all punctuation
    - SSN_new2 = COMPRESS(SSN, 'p');
  - To keep the numbers instead of removing punctuation
    - SSN_new2 = COMPRESS(SSN, 'dk');

TRIM()

- What does it do?
  - Removes trailing blanks that occur after the last valid character in a string
- Syntax:
  - TRIM(charvar)
- Useful when you want to use the variable for other character functions, particularly concatenation

Concatenate Function

- What does it do?
  - Combines two or more character variables
- Syntax:
  - Symbolized by two pipes: ||
    - Variable1 || Variable2
- Useful when you have pieces of information that need to be displayed or stored together
  - Addresses, area code and phone numbers
  - ID constructed from two or more fields
**Concatenate and TRIM() Examples**

- To combine variables name1 and name2
  - `Fullname = Name1||Name2;`
- With TRIM():
  - `Fullname = TRIM(Name1)||Trim(Name2);`
- With a blank space between names:
  - `Fullname = Name1 ' '||Name2`

**TRANSLATE() and TRANWRD()**

- What do they do?
  - TRANSLATE: allows you to substitute one character value for another
  - TRANWRD: substitute one “word” for another
  - Both functions allow you to sub in blanks
- Syntax:
  - `TRANSLATE(charvar, 'character(s) to sub in', 'character(s) to replace);`
  - `TRANWRD(charvar, 'word(s) to replace, 'word(s) to sub in');` *Note the different order!*

**TRANSLATE() and TRANWRD() Examples**

- In a name, numbers are used instead of letters. To replace the number with the letter, rather than compress it out
  - `Lastname1 = TRANSLATE(lname,'O','0');`
  - "O'Toole becomes O'Toole;
- Multi-character replacements
  - If the order of the characters does not matter, or the characters are not related, use TRANSLATE
  - `Lastname2 = TRANSLATE (lname, 'OLESG', '01386');`
  - 0 replaced by O, 1 replaced by L, 3 replaced by E, etc.
  - If the order matters (as in a word or phrase), use TRANWRD
  - `address1 = TRANWRD(address, 'STREET','ST.');`
  - STREET replaced with ST.
Other Techniques (1)
- Get to know your data so you can identify any systematic issues
  - Run frequencies on last and first names and scan beginning and end of list
  - Research cultural norms in naming for ethnic groups that may be present in your data
- Examples:
  - Standard words or phrases used to denote when an infant’s name is unavailable (unknown, BB, BG, etc)
  - Naming conventions for children taking parent names

Other Techniques (2)
- Remove values that are clearly not valid names and replace them with a character missing (""")
- Examples:
  - FATHER NOT LISTED
  - BABYBOYADOPTED
  - INFANTFEMALE
  - UNKNOWNASYET
  - BABYFEMALE
  - BABYGIRL
  - RECORD
  - MALE
  - BB

Other Techniques (3)
- Use a standard naming convention for your variables so you can distinguish which variables belong to which dataset
- Examples:
  - Birth certificate variable names prefixed with B
  - Linkage dataset variable names prefixed with L
Things to Remember

When using character functions:
- Order Matters
- Make New Variables
- Check to see if you got what you expected

Code Example: Length

Length statement is used to create "shell" variables which will receive cleaned names
- Creates interim variables for cleaning steps
  - Final variables are prefixed with a b
  - All interim variables have the same length as final variables
  - Length set to 35 for all names after determining maximum length for my dataset

Code Example: Child's Last Name

1) Tranwrd() replaces hyphens with spaces
   Since hyphens separate last names that could be listed separately, this step substitutes a space rather than compressing it and making the last names appear as one word
2) Tranwrd() replaces the word "DUPLICATE" with a question mark
3) Compress() with "p" options removes all punctuation, including the question mark from step 2
4 & 5) create two new variables containing the first and second last names
   Scan() uses a space as a delimiter (empty quotes) to find the names
Resources

- UCLA Academic Technology Services
  - Resources to help you learn and use SAS
  - http://www.ats.ucla.edu/stat/sas/
  - Search individual functions
- If you want the book:
- If you are too cheap to buy the book:
  - Cody, SAS Global Forum 2007: An Introduction to SAS® Character Functions

Supplemental Slides

INDEX() and INDEXC()

- What do they do?
  - Search a variable for a specific string, return location of the first instance of the string
  - INDEX() will look for the whole string if multiple characters
  - INDEXC() allows you to search for multiple characters individually and give you first location of any in the list
- Syntax:
  - INDEX(charvar, 'string'); INDEXC(charvar, 'string(s)');
  - Case-sensitive: use with UPCASE() or LOWCASE() for best results (converts all characters to upper or lowercase)
  - INDEXC(upcase(charvar, 'string'));
- Useful for identifying values that need cleaning or that will get a recoded value
FIND(), FINDW(), FINDC()

- What do they do?
  - Search variable for specific string, word, or character
    - Returns the position where it finds the first qualifying string, word, or character, 0 if not found
  - Unlike INDEX, has option to ignore case 'i'
  - FINDW can use delimiters and starting positions
- Syntax:
  - FIND(charvar, 'string')
  - FINDW(charvar, 'word', 'delimiter or start position'),
  - FINDC(charvar, 'character(s)')
- Useful for identifying values that need cleaning or that will get a recoded value

FIND(), FINDW(), FINDC() Examples

<table>
<thead>
<tr>
<th>Function Used</th>
<th>Phrase = “Over there is the dog.”</th>
<th>Result Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX(phrase, 'the')</td>
<td>Finds “the” in there, upcase used</td>
<td>6</td>
</tr>
<tr>
<td>INDEX(upcase(phrase), 'THE')</td>
<td>Finds “the” in there, upcase used</td>
<td>6</td>
</tr>
<tr>
<td>FIND(phrase, 'the')</td>
<td>Finds “the” in there, ‘i’ ignores case</td>
<td>6</td>
</tr>
<tr>
<td>FIND(phrase, 'THE', 'i')</td>
<td>Finds “the” in there, ‘i’ ignores case</td>
<td>6</td>
</tr>
<tr>
<td>INDEXC(phrase, 'the')</td>
<td>Finds “e” in Over</td>
<td>3</td>
</tr>
<tr>
<td>INDEXC(upcase(phrase), 'THE')</td>
<td>Finds “e” in Over</td>
<td>3</td>
</tr>
<tr>
<td>FINDC(phrase, 'the')</td>
<td>Finds “e” in Over, order of search strings does not matter</td>
<td>3</td>
</tr>
<tr>
<td>FINDC(phrase, 'eth')</td>
<td>Finds “e” in Over, order of search strings does not matter</td>
<td>3</td>
</tr>
<tr>
<td>FINDC(phrase, 'thE', 'i')</td>
<td>Finds “e” in Over, order of search strings does not matter</td>
<td>3</td>
</tr>
<tr>
<td>FINDW(phrase, 'the')</td>
<td>Finds “the” as a whole word, ignores “there”</td>
<td>15</td>
</tr>
</tbody>
</table>