Functions (Macros)  
Functions and Workspaces: Variables  
Functions (Macros)  
Why Functions (Macros)

Hye-Chung Kum  
Population Informatics Research Group  
http://research.tamhsc.edu/pinformatics/  
http://pinformatics.web.unc.edu/

License:  
Data Science in the Health Domain by Hye-Chung Kum is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Course URL:  
http://pinformatics.tamhsc.edu/phpm672

Programming

- Reusable code
- If you could not reuse code, writing exact steps for doing anything reasonable (usually takes MANY MANY lines of code) would take too much effort
- Programming works because
  - you write functions, small building blocks, that do small defined tasks correctly given certain input (parameters)
  - Then compose these functions together to carry out the complex task

Example mini-computer

CPU (Processor)
- Instruction set (2 bit)
  - 00: Save to
  - 01: Retrieve from
  - 10: Add
  - 11: Subtract

\[ 5 \times 3 = ? \]
- Add 5
- Add 5
- Add 5

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>10</td>
<td>0101</td>
</tr>
<tr>
<td>01</td>
<td>10</td>
<td>0101</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0101</td>
</tr>
</tbody>
</table>

RAM
- 00100101
- 01100101
- 10100101

Why use Functions?

- Top-down design
  - Break a complex problem into simpler manageable problems
  - Solve simpler problems
  - Connect simple solutions to solve original problem
- Testing strategy
  - Call function with different inputs to find bugs in algorithm
  - Small components tested individually
  - Connect components later (system integration)
  - Try testing 10,000 lines of script code without functions ??!

Why use Functions?

- Encapsulation
  - Black box programming
  - Hides internal details of algorithm from users
  - Users typically only care about using the function to get results.
  - Isolates computations, protects variables
  - Interaction through arguments
  - Separates interface and implementation
  - Interface: what a function does
  - Implementation: how a function does it
Why use Functions?

- Code reuse
  - Solve a problem once
  - Reuse your solution for similar problems
- Avoids repetitive typing
  - Consistency
  - Reduce Mistakes
  - Maintenance
    - Easier to fix one function than find and fix all locations of cut & paste code.

Why use Functions?

- Code sharing
  - Share your solution to a problem with others.
  - Collaboration
    - Team, organization, world
    - Another programmer only needs to know your function interface and behavior to use it.
    - Get solution from someone else
      - (and get caught easily if it’s an assignment)

Reusable Code Types

- Invocation (calls/runs the function)
  - Resolves variables (use value of the named variable) at run time
  - When the variable is resolved matters
    - SAS built in functions: month(date);
      - Parameter (input): date
      - Function name: month
      - Return value (output): month of the given date
  - Textual find & replace
    - SAS Macros (macro preprocessor)

SAS Macro (%)

Macro Preprocessor

SAS code with Macro Statements ➔ Standard SAS statements

- Macro variables
- Macro functions (macros) : not normally called functions

Assignment 6 Objectives

- Read and write SAS macro variables
- Read, use, and modify SAS macro functions

What is a workspace?

- The workspace is the set of variables that has been collected or instantiated during a session
- Session: one run of SAS (the time that you have been using SAS)
  - Batch mode: during the one run
- The two main workspace in SAS
  - SAS tables
  - Macro variables
Local vs Global Variables

- Based on scope of variable
  - Scope = workspace
- Global variables
  - Valid in all workspace
- Local variable
  - Valid in only the local workspace
  - For example inside a function or Macro

Macro Variables (older version)

- The name of a macro variable can be from one to eight characters.
- The name must begin with a letter or an underscore.
- Only letters, numbers, or underscores can follow the first letter.
- The content of macro variable can be up to 32K (in version 7, the limit is 64K).
- No macro variable can begin with SYS.
- No macro variable can have the same name as a SAS-supplied macro or macro function

Macro Variables

- Define a global macro variable:
  \%let varname = value;
- Use a defined macro variable:
  keep &varname;
  title "&varname": = must be double quotes;
- Resolves to be identical to:
  keep value;
  title "value":
- Try examples:

Evaluating Expressions

- Integer arithmetic:
  \%let macro_var = \%eval(expression);
- If float:
  \%let macro_var = \%sysvaluate(expression):

http://www.ats.ucla.edu/stat/sas/seminars/sas_macros_introduction/

Moving data between Macro Variable & SAS Tables

CALL SYMPUT ("macro_var_name", value);
CALL SYMGET ("macro_var_name");

- Create/reassign macro_var_name
- Same as \%let except, can take values from sas table
- Value could be
  - A variable from a sas dataset
  - Constant
- Assigns the value at the end of the step
  - Run
  - Proc & Data
  - Symget vs &
  - When the variable is resolved

Macro Functions

- Pro: Reusable code
  - Allows you to write a set of sas statements once, and then use them over and over again
- Con: more complicated code can lead to more difficulty in debugging
  - You MUST write modular code
  - First, write your program in normal SAS code
  - Test that it works
  - Then convert to SAS Macro
  - Test that the macro works
Macro Functions

* Define a macro:
  - The macro parameters are LOCAL macro variables to the macro function:
    ```sas
    %macro macro_name [macro_parameters];
    macro_body
    %end [macro-name];
    
    * Invoke a macro that has been defined:
    %macro_name [macro_parameters];
    
    * Both syntax is OK:
    %macro_name [value];
    
    * Try examples, Assignment 4:
    
    %age (dob, dt);
    . body of macro function;
    %mend;
    %age (birth, mdy(1/1/2014));
    
    Macro Conditional Logic

* Inside the macro function:
    ```sas
    %if condition %then %do:
    * if body code;
    %end;
    %else %if condition %then %do:
    * else if body code;
    %end;
    
    * Try examples:
    ```

Jargon

* Function Parameters
  - The variables declared in the function interface
  - dob & dt are local macro variable names
* Function Arguments
  - The actual values supplied when the function is called.
  - birth is a variable name from an actual table

```sas
%macro age (dob, dt):
  . Input Parameters
  %mend;
  %age (dob=birth, dt=mdy(1/1/2014));
```

Macro Loops

* Inside the macro function:
  ```sas
  %do istart %to iend:
  * if body code;
  %end;
  
  * Try examples:
  ```

Debugging Macros

* MPRINT
* SYMBOLGEN
* MLOGIC
* %put
* %include
  ```sas
  - config.sas
  ```

Options MPRINT MLOGIC SYMBOLGEN;
* Look at log:
**Built in Macro Variables**

- SAS supplied Macro variables
  - %put_all;
  - %put_automatic;
  - %put_user;
  - %put_local;
  - %put_global;
- SAS supplied variables
  - _numeric;
  - _character;
  - _all;

**Function Review**

- Functions
  - Creating a function
  - Writing a function
    - Function Rules
    - Calling a function
      - Parameters vs. Arguments
      - Scope
      - Functions
      - Variables

**Function Review**

- Functions
  - Creating a function
  - Writing a function
    - Function Rules
    - Calling a function
      - Parameters vs. Arguments
      - Scope
      - Functions
      - Variables

**From Assignment 6 on ...**

- Grading for style
  - Consistent style
  - Readable beautiful code
  - Good indentation
  - Good line breaks
  - Variable names
  - Comments
- For full grade: when you are done, go back and “EDIT” to make it readable and consistent before submission

**Recoding**

- It is perfectly fine to overwrite variable value in recoding.
  - acceptable and RECOMMENDED coding
    - `county=compress(county)`
  - It means take value from county, compress it, than save the new value into the county variable and overwrite what was there.

```plaintext
*clear blanks in county names:*
ncounty=compress(county);
drop county;
rename ncounty=county;
```

**Plans for final project?**

- Separate 4 points solely on style for final project
  - Indent, line break, comments : readable beautiful code

Programming ... Read. Watch. Do. Repeat doing until you get the hang of it.
Review Midterm (next week)

- Go over multiple choice together
- Go over open ended answers

Assignment 6

- Objectives
  - Read and write SAS macro variables
  - Read, use, and modify SAS macro functions
- Lab 6
  - Start doing in class