Midterm Review

Hye-Chung Kum (kum@tamu.edu)
Associate Professor
Population Informatics Lab (https://pinformatics.org/)
Course URL: http://pinformatics.org/phpm672

2

Questions: Assignment 5?

SAS String functions

- SAS 9.1
- https://support.sas.com/publishing/pubcat/chaps/59343.pdf

4

Assignment 3: feedback sent back

- Elegance will count in this assignment! It does not mean complicated programs. Remember KISS. The beauty of great programs is its SIMPILICITY. What you are trying to learn is how to break down complex tasks into simple modular tasks.
- Indentation
- Blank lines
- 0.5 points
- Share two examples

5

Assignment 3: indentations

```
do i=1 to 12;
   if discharge[i]~=. then leave;
   end;
   if i=13 then first=.;
   else first=yearq[i];
drop i;
```

```
   do i = 1 to 12;
      if discharge(i)=. then discharge(i)=0;
end;
```

6

Not elegant:

```
array dc2010(*) dc20101- dc20104;
array dc2011(*) dc20111- dc20114;
array dc2012(*) dc20121- dc20124;
       do i= 1 to dim(dc2010);
       adc2010 =sum (of dc2010(*));
    end;
    do i= 1 to dim (dc2011);
       adc2011 =sum (of dc2011(*));
    end;
    do i = 1 to dim(dc2012);
    adc2012 =sum (of dc2012(*));
    end;
run;
```

Not ok: logical makes no sense

```
*Task 2 - Creating new binary variables by adding 'b' as a
prefix for the previous variable and recoding the binary
variable such that 1 = yes, and no = 0.;
   array aver(*) CVDINFR4 CVDCRHD4 CVDSTRK3 ASTHMA3
_DENVST2;

   array biever(*) bCVDINFR4 bCVDCRHD4 bCVDSTRK3 bASTHMA3
bDENVST2 ;

   do idxa= 1 to 5;
     if aver(idxa) = 2 then biever(idxa) = 0;
     else if aver(idxa) = 1 then biever(idxa) = 1;
   end;
```

What you should have learned in 8 weeks

- Do you know the question is asking?
- What is your answer before class started? Now?
- Questions
  - Do you know how to talk to a computer? (To get it to do what you want)
  - Do you know how to think data?
  - Can you use SAS to manipulate data into a format you need?
    - libname, variables, data steps, labels, formats, arrays, loops, conditionals, boolean expressions, proc summary, proc transpose
- What is left: 7 weeks
  - 1 week: midterm
  - 2 weeks: reusable code (macros)
  - 4 weeks: a project to try this out

Midterm format (20%)

- 5 questions (50 points): take home today
  - Open book / open notes / use SAS
  - Programming/debugging questions
  - submit by midnight Monday on E-Campus (one week)
- 25 questions (about 2*25=50 points)
  - On E-Campus
  - multiple choice similar to quiz
  - Closed book
  - Thur (in class): 1h 15min

Open Response: Due Monday midnight

- Write SAS code to (8*5=40pts)
  - Data Step 1
    - Q1.1 read in datasets X1..Xn and make new dataset Y
    - Q1.2 keep, rename, label variables v1-vn
    - Q1.3 code variable c1
    - Q1.4 use arrays and loops to recode variable c2
  - Proc Steps
    - Q2.5 convert dataset Y to dataset Z
    - Q2.6 Find and show descriptive (avg/max/median) (Must use SAS code)
  - Data Step 2
    - Q2.7 link in dataset L to dataset Y
    - Q2.8 Print observations meeting condition (Must use SAS code)
  - Typically few lines of code per question
  - Submit code/log/output
- Debug the following code (10pts)
  - Fix the program to run properly
  - Submit code/log/output
- Extra Credit (10pts)

Extra Credit (10pts=2+3+5)

- **PART 3.1: Extra Credit -**
- READ your assignment 2 (this is the first real program you submitted in class) that you submitted, and make is more elegant code now that you know more about coding.
- Submit FOUR files, the regular sas (the more elegant code you wrote)/log/lst AND the code annotated with the changes you made and why (you can do this in word so that you can use formatting, such as bold/color, to annotate.

```
******** Section 1: Frist Data Step *******;
code

* Q1.1;
code

* Q1.2;
code

******** Section 2: Proc Steps *******;
* Q2.7;
code

******** Section 3: Second Data Step *******;
```

Things to watch out for

- Make your code readable to people
  - Indent your code
  - Use newline
  - Use reason variable names
- Be efficient
  - No unnecessary data steps

Midterm: Responsible materials

- Readings from the Little SAS Book
  - All sections in chapter 1
  - All sections in chapter 2
  - All sections in chapter 3
  - Sections 4.1 to 4.10 in chapter 4
  - All sections in chapter 6
  - Note that some of the materials were not covered in class or assignment, but you are responsible for anything covered in the required reading from the book
- Other materials
  - All class notes up to this class (slides on the class website).
  - None of the articles are part of the midterm (except to the extent covered in class on the notes)

SAS Basics

- program/log/output (lst or html)
- libname
- ;
- setting up work environment
  - How you will use the software
  - How you will organize your files

SAS

- keywords
  - data, set, merge, obs, where, if, do, end, keep, drop, rename, label, in
  - array
  - proc
    - sort, print, summary, transpose, freq
- functions
  - put ()
  - compress ()
  - lowcase () / upcase ()

Boolean expression evaluation

- X || (Y & X)

X	Y		X||Y&X

Questions: midterm?

Assignment 1

- Setup work environment
- Use the SAS software
- SAS programming basics
  - data step & proc step
  - Libname (where is the folder with the data?)
  - Writing code & Reading logs

Assignment 2

- Understand variables (names, types, labels)
- To write conditional logic codes
- Subset columns (variables) from a table
- Subset rows (observations) from a table
- Recode, rename variables and calculate new variables
- Label variables and values

Assignment 3

- use for loops (iterative loops)
- use while loops (conditional loops)
- SAS: use one dimensional arrays

Assignment 4

- **Concatenate multiple tables (more rows)**
  - **stack tables on top of each other to increase the number of rows**
  - using **set**
  - Be sure to understand the different behavior given different situations (i.e. what happens to shared variables? What happens to not shared variables?)
- **Link up multiple tables using a shared key (more columns)**
  - **align the rows using the shared key, and link multiple tables to increase the number of variables in the tables**
  - using **merge**
  - Be sure to understand the different behavior given different situations (i.e. what happens to shared vars? What happens to not shared vars?)
  - What is a 1-to-1 link
  - What is a 1-to-N link
  - What is a N-to-N link (you will not be doing this, but need to understand what this is. This must be done with proc sql in SAS)
- New keyword **in=**

23

Assignment 4 continued

- Combine multiple rows into one row
  - by group processing **proc summary**
- Reshape table to flip rows & columns
  - using **proc transpose**
  - Also transpose (flip rows & columns) by groups or row

Table Operations: 1 table → 1 table (reshaping)

- Proc Transpose

1	2
a	d
b	e
c	f

→

1	a	b	c
2	d	e	f

- Proc Summary

A
B
C

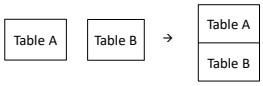
→

D

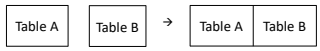
Where D=function(A,B,C)
Examples of function are
Sum(A,B,C) Mean(A,B,C) Max(A,B,C) Min(A,B,C)

Table Operations: multiple table → 1 table

- set (Append)



- merge (link)



lab 4

- proc transpose by

Formats

- Create using proc format
- Use Case 1: Labeling values
  - Assign using format statement (permanent, temporary)
  - Only used interpret the value (ie. printing, display)
- Use Case 2: Can be used to recode variables (know how different)
  - put(var, format)
  - new variable type? Value?

```
proc format;
   value gender
   1= 'Male'
   2= 'Female'
   other= 'Missing' ;
* In data step;
data outfn;
set infn;

csex $7.;
csex=put(sex, gender.);
```

Arrays

- Array n{*} n9-n23;
- Array a{*} $7. a11-a23;
- Name? n and a
- How many elements? N=15 a=13
- Type? N=number, a=string of length 7
- n15 index? 7

ever{1}	ever{2}	ever{3}	ever{4}	bever{1}	bever{2}	bever{3}	bever{4}
cigever	alcever	cocever	mjever	bcigever	balcever	bcocever	bmjever

```
* Brute Force:  Cut & Paste & Tweak
if cigever=1 then bcigever=1;
else if cigever=2 then bcigever=0;

if alcever=1 then balcever=1;
else if alcever=2 then balcever=0;

if cocever=1 then bcocever=1;
else if cocever=2 then bcocever=0;

if mjever=1 then bmjever=1;
else if mjever in (0,2) then bmjever=0;

* Using arrays is much more elegant and accurate;
array ever{4} cigever alcever cocever mjever;
array bever{4} bcigever balcever bcocever bmjever;
do i=1 to 4;
  if ever{i}=1 then bever{i}=1;
  else if ever{i} in (0,2) then bever{i}=0;
end;
```

loops

- How many times?
- Do while (cond)
  - correct expression

Record Linkage
Inherent Nature of Real Data

- Data are expressed differently
  - nick names
- Data change over time
  - person's last name
- Data are not unique attributes
  - John Smith
- Missing Data
  - ssn are often missing
- Errors in Data
  - Rule of thumb : 5% error in administrative data

Record Linkage

- When merging data
  - Use numeric codes whenever possible
  - Remember to use uniform formatting
  - Use string functions to standardize variables
  - Check if the key provides unique rows
    - 1-to-1 or 1-to-N mapping
- Pay attention to what rows link and what do not
- Consider how many rows should link
  - Example: 20% expected 18% achieved
- Validate by printing
  - Links made
  - Links not made

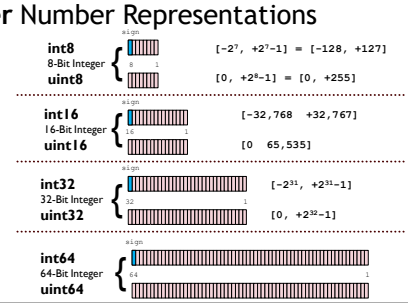
Common log messages

- NOTE: Variable yea is uninitialized
- ERROR: Array subscript out of range at line 45 column 3
- NOTE: MERGE statement has more than one data set with repeats of BY values.
- ERROR: BY variables are not properly sorted on data set WORK.FN

What is a Variable?

- A user defined name to represent a piece of memory for storing evaluated value(s).
  - A variable consists of 5 items
- Name: How the user refers to variable. Understandable by both human and computer
- Label: meaningful human friendly descriptions of the variable
- Data Type: number or string (character=string of length 1)
  - How to interpret variable for data representation
- Size: How much storage memory is needed to store data value
  - Can be inferred from data type
- Value: Actual value associated with variable
  - stored in memory
- Storage location: How the computer refers to a variable
  - Usually hidden from user by the interpreter or compiler
- For Our Purposes: Columns
  - Many variables. A columns of variables

Integer Number Representations



Variable naming rules

- Starts with a single letter or underscore followed by any number of letters, digits, or underscores.
- Digits `[0-9]`, Letters `[a-zA-Z]`, Underscore '_'
- No special characters
- Small or Large does not matter in SAS

Variable Types

Type	Stored value	Interpreted value	Label Interpreted Value
int	1000001 (65)	65	65 or older
Char/string (ASCII)	1000001 (65)	A	Asian
date	1000001 (65)	1960/3/6 (SAS)	

- `1 0 0 0 0 0 1 =64+1=65`
- `64 32 16 8 4 2 1`

Real Number Representations
IEEE 754 Floating point standard

- Reals (http://kipirvine.com/asm/workbook/floating_tut.htm)
  - Sign bit (1 bit) : + / -
  - Exponent (7 or 11 bits) : biased by 127 = exp-127
  - Mantissa (fraction) (23 bits or 52 bits): $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ ...
  - (+/-) (1+fraction) * 10 (exp-127) = + (1.01) * 10 (124-127)
  - = (1.01)*10 (-3) = (0.00101)=1/8+1/32=0.15625
  - Single 

  - Double 

Binary Numbers

1001 1 0 0 1

8 4 2 1 1/2 1/4 1/8 1/16

8*1+1*1=9 0.5+0.0625=0.5625

Declare a variable

- Tell the computer I need room in memory for a certain variable
  - A certain length
  - A certain type
  - With a certain name
  - Optional: Set to an initial value (initialize)
- Length: static vs dynamic
- SAS
  - SAS: implicit when used for the first time
  - Not one variable, but column of variables

Conversion between types

- Conversion: Use cast function
- Upcast to larger data type, no issue
- Downcast to smaller data type,
  - Truncation & clamping problems
  - Conversion between signed and unsigned as an example
- Conversion from real to integer,
  - truncation to closest integer
- Conversion from integer to real,
  - approximation by nearest real
- Conversion from number to/from string
  - Pay attention

Integer Issues

- **Overflow,** expression tries to create an integer value larger than allowed valid range `[min,max]`
  - `x = int8( 127 ) + 1`
- **Truncation,** fractions not supported
  - `int16(23)/int16(5) = 5` not `4.6`
  - Rounds result to nearest whole number

Real Issues (single, double)

- Precision Error $Error = |actual - representation|$
  - Most numbers don't get represented exactly
  - Finite precision of IEEE floating point
  - Represented by nearest real number
  - Separation between two closest numbers varies over entire range
- Numeric Stability (does error overwhelm?)
  - Truncation Errors $Error = |true\_answer - computed|$
  - Accumulated error from repeated calculations
- Don't compare real numbers
  - 3.0 == 3.0 (NOT GOOD)

Real Number Representations
IEEE 754 Floating point standard

- Reals (http://kipirvine.com/asm/...)
  - Sign bit (1 bit) : + / -
  - Exponent (7 or 11 bits) : biased by
  - Mantissa (fraction) (23 bits or 52 bi
  - (+/-) (1+fraction) * 10 (exp-127) = +
  - = (1.01)*10 (-3) = (0.00101)=
  - Single
  - Double

Decimal fraction to Binary fraction
Lose precision

0.200000000000
= .0011001100110011001100
+ remainder 0.000000071526