Loops & Arrays

efficiency
for statements
while statements

Hye-Chung Kum
Population Informatics Research Group
http://pinformatics.org/

Course URL:
http://pinformatics.org/phpm672

1

Objective

- use for loops (counting loops)
- use while loops (conditional loops)
- use one dimensional arrays
- Understand how to write reusable code
- Understand how to optimize your programming time: KISS (Keep it simple)

2

SAS: Arrays

array{1}	array{2}	array{3}	array{4}
rate2005	rate2006	rate2007	rate2008

- All variables in one array must be of the same type
- Variables specified within an array do not need to already exist
- array aname {dim} [$len] elements
  - array rate {4} rate2005-rate2008;
  - array rate {*} rate2005-rate2008;
  - array rate {4} ; *implicit: rate1-rate4;
- Dim(Dimension): how many elements
  - Can be implicit by using *
- $len: type and length of variables when strings
  - Omitted for numerical variables
  - Array name{3} $10.;
- elements: list of variables
- index: an integer pointer that identifies the element in the array
  - array {index} or array [index]
  - rate2006 is indexed by 2

POPULATION INFORMATICS

3

SAS: for loop statement

the **counted loop** solution

```
do <varindex> = <start> to <stop>;
     <Body: do some work with varindex>
end;

do <idx> = <start> to <stop> by <step>;
     <Body: do some work with varindex>
end;
```

POPULATION INFORMATICS

4

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

Indent
Why?

```
* 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;
```

Indent
Why?

```
* Even better, more extensible, using arrays;
array ever{*} cigever alcever cocever mjever;
array bever{*} bcigever balcever bcocever bmjever;
do i=1 to dim(ever); * uses the dimension of the array;
  if ever{i}=1 then bever{i}=1;
  else if ever{i} in (0,2) then bever{i}=0;
end;
```

5

do while loop statement

the **conditional loop** solution (SAS)

```
do while (<test>);
    <Body: do some work>
    <Update: make progress towards exiting loop>
end;
```

If we don't know ahead of time, how many times we need to loop but we can write a **test** for when we are done; Then the **while** loop is a great solution.

Note: For this to work properly, the <test> needs to evaluate to a logical value.

Note: The body of the **while** loop will continue to get executed as long as the <test> evaluates to `true`. The while loop is exited as soon as the condition evaluates to `false`.

6

Algorithms

- Common Idioms
  - Divide & Conquer
  - Iterate
  - Copying
  - Counting
  - Summing
  - Searching
  - Sorting

leave statement

Terminates `for` or `while` loops. breaks flow of control of inner most nested `while` or `for` loop containing `leave` statement.

```
array rate{*} rate2001 - rate2013;
idx = 1;
count = 0;

* What year was the 4th year when rate > 7;
do while ( idx <= dim(rate) );
  if rate(idx) > 7.0 then
      count = count + 1;

  * Jump out of while loop;
  if (count = 4) then leave;
  idx = idx + 1;
end;
* Control flow jumps to here after break;
if (count=4) then year4=2000+idx;
```

Breaking out of loop

- The LEAVE statement causes processing of the current loop to end.
- The CONTINUE statement stops the processing of the current iteration of a loop and resumes with the next iteration.

Nested loops

```
data test;
do i=1 to 10;
     do j=1 to 5;
          output;
     end;
end;
```

Multi Dimensional Arrays

- We only looked at one dimensional arrays
  - SAS: Two dimensional arrays (two indices)
  - array m{4,3} $3. month1-month12;
  - first month of each quarter: m{qtr,1} where 1<=qtr<=4
  - 4 rows & 3 columns
  - SAS places variables into a two-dimensional array by filling all rows in order, beginning at the upper-left corner of the array (known as row-major order).

month1 (Jan)	month2 (Feb)	month2 (Mar)
month4 (Apr)	month5 (May)	month6 (Jun)
month7 (Jul)	month8 (Aug)	month9 (Sep)
month10 (Oct)	month11 (Nov)	month12 (Dec)

Population Informatics

Population Informatics

Assignment 3

- Opened Lab 3 & Assignment 3
- Try to rewrite assignment 2 to be elegant code

POPULATION INFORMATICS

13

Assignment 2
Elegant code

14

Programming Etiquette

15

Readable Programs

- **Whitespace**
  - Grouping
  - Indentation
    - to show control flow
- **Documentation**
  - Naming
  - Comments
- **Modular Code**
  - Break large blocks into smaller pieces
  - Use sub-routines or functions (more later)

Write programs for people first, computers second.
-- Steve McConnell

Will you be able to read and understand your own code six months from now?

16

Whitespace
Use **indentation** to show logical structure

Which script is more readable?

```
x = 3; if x < 3 then y = 3; else y = 5;
```

or

```
x = 3;
if x < 3 then y = 3;
else y = 5;
```


Documentation
Use meaningful names

Which is more readable?

```
xx = yyy( x );
xxx = PinkFlamingo( xx );
x4 = max(find(xxx)~=0);
floyd = x4.balance;
```

or

```
currID = CustomerID( custName );
currAccounts = BankAcct( currID );
mainAcct = max(find(currAccounts)~=0);
currBalance = mainAcct.balance;
```

Meaningful?

```
proc print data=source.sasdata (obs=10);
```


Documentation

use **comments** to clarify meaning

- The first comment at the beginning of the script or function should describe what the script or function does.
- Approximately one comment per group of commands is about right.
- Avoid comments which just repeats what the associated code does.
- Use comments to document tricky code
- Use comments to give credits
- Did you see what google did on the csv file?

Do in one/two data steps

- **P1. Cleaning and Manipulating the Data**
- Your program should do the following. Please indicate BEFORE the code in comments where each of these items occurs in your code.
- P1.1 subset vars (You need at least one of each continuous, categorical, binary, id var. If you are missing a type, create a new variable of that type.)
- P1.2 subset obs (be creative if you do not need this for your analysis)
- P1.3 rename at least 3 variables
- P1.4 label at least 3 variables
- P1.5 label values for at least 3 variables (at least one must be permanently labeled, and one must be temporarily label)
- P1.6 Recode at least 3 variables (use your imagination, if not essential to your analysis)
- P1.7 Construct at least 3 new variables (use your imagination, if not essential to your analysis)
- P1.8 Save out your new data permanently

21

Proc steps

- **P2. Learning Your Data (Descriptive Analysis)**
- Your program should also do the following. Please indicate BEFORE the code in comments where each of these items occurs in your code.
- P2.1 List each type of variable (continuous, categorical, binary, id). (see P3.3)
- P2.2 Create summary statistics for all your continuous & binary variables
- P2.3 Create tabulations for each categorical variables
- P2.4 Answer one interesting question using at least 3 variables (see P3.4)

22

- What line is not needed?

```
DATA data.CHR2019st;
        set data.CHR2019st;
        g_pctrural = .;
        if n_pctrural = '.' then g_pctrural = '.';
        if n_pctrural <= 0.3259913864 then g_pctrural = 1;
        else if n_pctrural <= 0.5885717839 then g_pctrural = 2;
        else if n_pctrural <= 0.8625272424 then g_pctrural = 3;
        else g_pctrural = 4;
```


23

```
/********P1.7 Construct 3 new variables**********/

data dataout.brfss2017_8;
set dataout.brfss2017_7;
if insured = 0 and employed = 0 then empins=0;
if insured = 1 and employed = 1 then empins = 1;
if insured = 1 and employed = 0 then empins = 2;
if insured = 0 and employed = 1 then empins = 3;
if insured = . or employed = . then empins = .;
run;


data dataout.brfss2017_9;
set dataout.brfss2017_8;
if insured = 0 and greenvgg = 0 then insgreen= 0;
if insured = 1 and greenvgg = 1 then insgreen = 1;
if insured = 1 and greenvgg = 0 then insgreen= 2;
if insured = 0 and greenvgg = 1 then insgreen = 3;
run;

data dataout.brfss2017_10;
set dataout.brfss2017_9;
if employ1 = 1 or employ1 = 2 then employs= 1;
if employ1 = 3 or employ1 = 4 or employ1 = 5 or employ1 = 6 or employ1 = 7 or employ1 = 8 then employs=
0;
if employ1 = 9 or employ1 = . then employs= .;
run;
```


24

```
/********P1.7 Construct 3 new variables**********/

data dataout.brfss2017_8;
set dataout.brfss2017_7;
if insured = 0 and employed = 0 then empins=0;
else if insured = 1 and employed = 1 then empins = 1;
else if insured = 1 and employed = 0 then empins = 2;
else if insured = 0 and employed = 1 then empins = 3;
else if insured = . or employed = . then empins = .;
run;


data dataout.brfss2017_9;
set dataout.brfss2017_8;
if insured = 0 and greenvgg = 0 then insgreen= 0;
else if insured = 1 and greenvgg = 1 then insgreen = 1;
else if insured = 1 and greenvgg = 0 then insgreen= 2;
else if insured = 0 and greenvgg = 1 then insgreen = 3;
run;

data dataout.brfss2017_10;
set dataout.brfss2017_9;
if employ1 = 1 or employ1 = 2 then employs= 1;
else if employ1 = 3 or employ1 = 4 or employ1 = 5 or employ1 = 6 or employ1 = 7 or employ1 = 8 then
employs= 0;
else if employ1 = 9 or employ1 = . then employs= .;
run;
```


- What could be better?

```
libname source 'C:\PHPM 672\Assign2.SAS Documents\Data';
proc print data=source.sasdata (obs=10);
run;
```

- How can we do this better?
  - c_educ=educa;
- What does c_ mean?
  - categories

```
c_educ=.;
        if educa=1 then c_educ=1;
        else if educa=2 then c_educ=2;
        else if educa=3 then c_educ=3;
        else if educa=4 then c_educ=4;
        else if educa=5 then c_educ=5;
        else if educa=6 then c_educ=6;
```

27

- What is easier to read? What is better?

```
b_diabetes=.;  * option 1;
if diabete2=1 then b_diabetes=1;
else if diabete2=2 or diabete2=3 or diabete2=4 then b_diabetes=0;

b_diabtes=.; * option 2;
if diabete2=1 then b_diabetes=1;
else if diabete2 in (2,3,4) then b_diabetes=0;

b_diabtes=.; * option 3;
if diabete2=1 then b_diabetes=1;
else if diabete2>=2 then b_diabetes=0;
```

28

Can we improve?
Add sort. Use By.

```
PROC means data=data.CHR2019st_rev;
      vars n_pcpratio;
      where g_pctrural=1 and g_medhhi>3;
      run;
PROC means data=data.CHR2019st_rev;
      vars n_pcpratio;
      where g_pctrural=2 and g_medhhi>3;
      run;
PROC means data=data.CHR2019st_rev;
      vars n_pcpratio;
      where g_pctrural=3 and g_medhhi>3;
      run;
PROC means data=data.CHR2019st_rev;
      vars n_pcpratio;
      where g_pctrural=4 and g_medhhi>3;
      run;
```

POPULATION INFORMATICS