



Coding Guidelines


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

License:
Data Science in the Health Domain by Hye-Chung Kum is licensed under a
[Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/)

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





1



What we are going to learn



- Big Picture
- How to avoid code confusion and associated programming errors.
- Common pitfalls.
- Programming Style Guidelines.
- Basic ideas behind good programming methodologies and good programming etiquette.



2

Why are your programming habits SO IMPORTANT?



- We'll talk about this over and over, so this is just a first assault!
- Programming done poorly is almost worthless:
 - You won't be able to understand what you programmed just last week,
 - Others won't be able to understand what you tried to accomplish,
 - And neither you nor anyone else can FIX your bad code. So
 - The time to develop good habits is NOW!



3

Outlining and Sentence Diagrams

- Remember when your English teacher...(Here it comes- this is one of those "when I was younger lectures...!")
- So, here are my notes for what I want to tell you:
 - Planning is important...to?
 - You and the people you interact with!
 - Planning saves time...why?
 - Outcomes trump effort
 - Planning is not easy...why?
 - Requires **crystal clear** thinking (computers only know 0/1)
 - Requires re-thinking
 - Sometimes requires throwing stuff away!
 - Planning can be irritating
 - Not making progress!



4

What does this (planning) mean vis a vis programming?

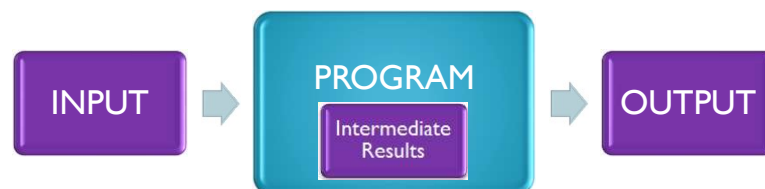
- Think “top-down”
 - Design the program before you code.
 - Break the problem down into small steps
 1. State the problem clearly.
 2. Define the inputs and outputs
 3. Describe the algorithm:
 - Psuedocode, flow charts, or even **comments!**
 4. Translate the steps to SAS code
 5. TEST EACH STEP on a small version
 - Look at memory (table) after each step
 - `proc print data=fn(obs=10);` where condition



5

Programming



- OUTPUT : Know what you want
- INPUT : what you have
- Intermediate results: What you need
- Program: change what you have (INPUT) to what you need (intermediate results. Often more than one level) to what you want (OUTPUT)



6

File format



- text vs word?
- html
- How do you create text files?
- What is a file extension?



7

SAS online manual

- https://documentation.sas.com/?cdclid=pgmsascdc&cdcVersion=9.4_3.5&docsetId=pgmsashome&docsetTarget=home.htm&locale=en
- Google to get help
- Stackoverflow
 - <https://stackoverflow.com/questions/52920619/proc-print-and-proc-means>



8

Basics of Programming: SAS

- **data** step
 - Row at a time
- **proc** step
 - Full table
- **libname**: directory location (folder)
 - No libname: temporary data
- **run;** (missing last results)
- **;** (I am done. Can be more than one line)
- **log & lst (html)**: computer communicating back with you what happened
 - Learn to READ the log
- **comments**
 - `/* comments */`
 - `* line comments;`
 - Length limit 256. If you are using it for long lines pay attention to log for messages.

POPULATION INFORMATICS

CC BY NC SA

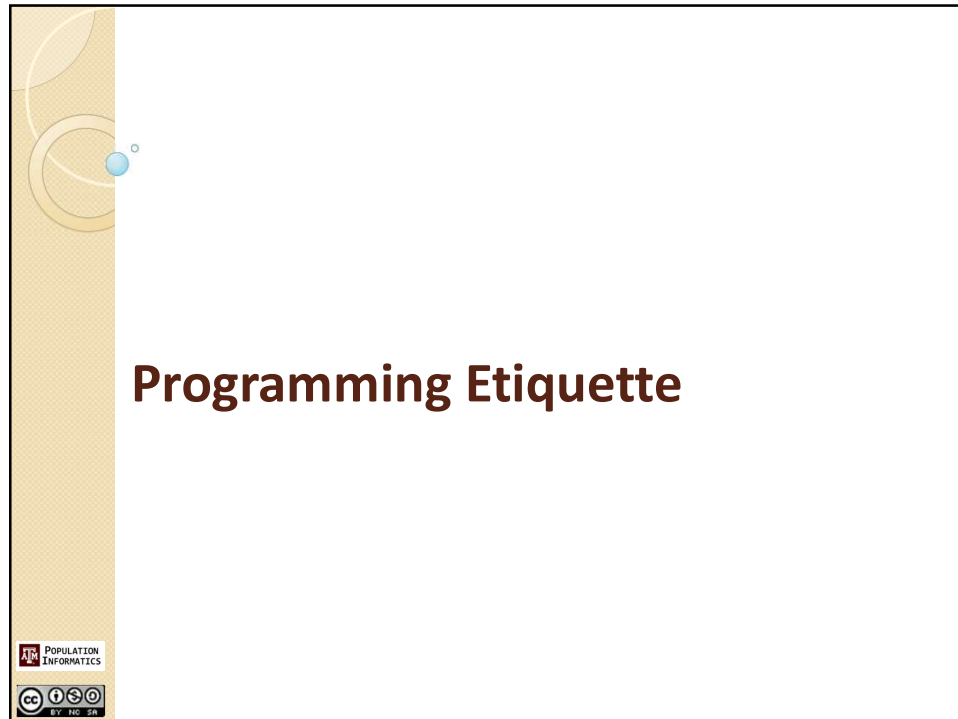
9



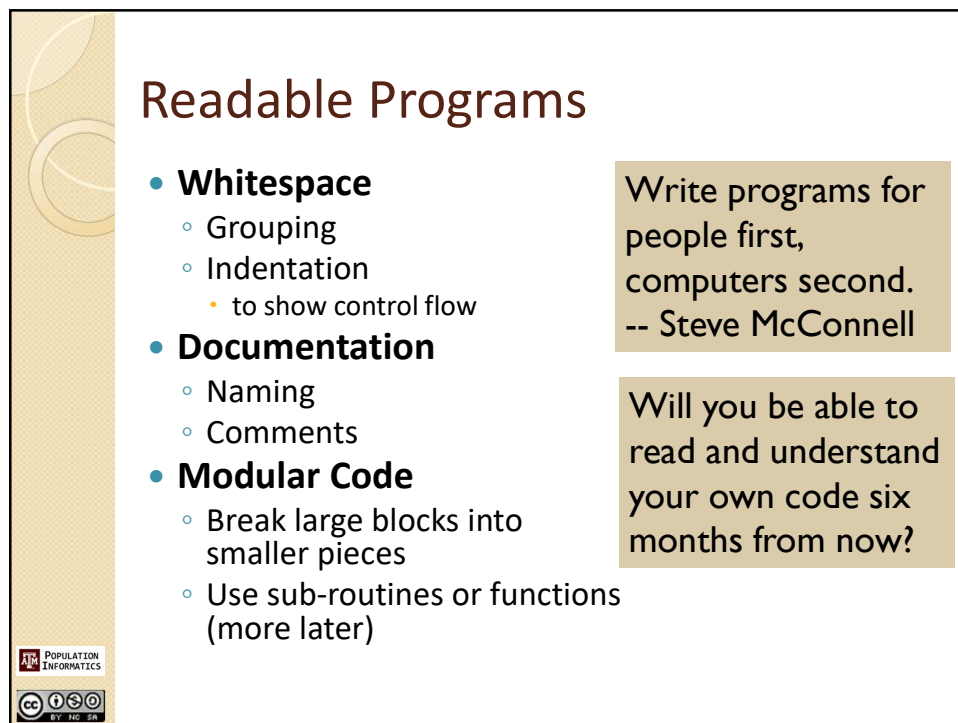
POPULATION INFORMATICS

CC BY NC SA

10



11



Slide 12: 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?

The slide features a decorative vertical bar on the left with a blue circle and a white dot. The title "Readable Programs" is centered in a large, bold, dark red font. The content is organized into a bulleted list with sub-bullets. Two text boxes on the right contain quotes. At the bottom left, there is a logo for "POPULATION INFORMATICS" and a Creative Commons license icon (CC BY-NC-SA).

12


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;
```



13


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;
```





14

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?

 POPULATION INFORMATICS


15



 POPULATION INFORMATICS


16

• **How is lab 2 coming along?**

17



Lab 2 & Assignment 2: Objective

- 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**

18

Label variables

- SAS
 - `label var1 = "LABEL" ;`





19

Label values

- SAS: define format, then use in data step

```
proc format;  
value fname  
    val1= "LAB1"  
    val2= "LAB2" ;  
* inside data step;  
format var1 fname.
```





20

Label Var vs Value

Name	Type	Size	Value
bcigever	int8	1 byte	1 or 0

```
label bcigever= "Ever smoked" ;
```

- Labeling variable
 - Give a more human friendly name to the variable name.
 - Same as **bcigever** (the computer friendly name for the variable used in the programs)
 - Stored in the header information for the table

21

Label Var vs Value

Name	Type	Size	Value
bcigever	int8	1 byte	1 or 0

```
proc format;
  value bool
    1= "TRUE"
    0= "FALSE" ;



  * inside data step;
  data outfile;
  set infile;

  format bcigever bool. ;

  * Removing a format;
  data outfile;
  set infile;

  format bcigever;
```

- labeling value
 - Give a more human friendly name to the variable value.
 - Same as **1(=TRUE) or 0(=FALSE)**
 - internally, the computer stores 0 or 1
 - But, when printing the values for humans, the computer uses the format you created and designated to use for this variable.
 - Can be used on multiple variables
 - It can be permanent (if done in the data step) or temporary (if done in proc steps)
 - The format must be created BEFORE use
 - Stored in the header information for the table

22



23

Data Step

```
libname data "D:\HPM-Users\kum\phpm672\lab2\data";  
data outfn;  
  set infn;  
  ...code...  
  
data mynsduh;  
  set data.nsdh;  
  ...code...
```

24

Subset columns (variables)

- SAS
 - Three places possible
 - Reading in, writing out, during datastep
 - **keep**, **drop**

```
data mynsduh;  
  set data.nsdh (keep=var);
```

```
data mynsduh;  
  set data.nsdh (drop=var);
```



25

Calculate new variable (assignment)

- SAS (in data step)
 - `var1 = 1 ;` * assignment;
 - `num1=.` ; * numeric missing value;
 - `str1= ""` ; * string missing value;



 - `Gender=1;`
 - `Gender=F;`
 - `Gender= "F"` ;
 - `Gender= 'F'` ;



26

Rename existing variable



- SAS (in data step)
 - Depending on where you do this, different behavior
 - `rename oldvar=newvar`



27

Reminder


- Make sure to understand lab 2
 - You MUST submit programs, logs, and output along with assignment 2
 - This is how you will LEARN
 - Most IMPORTANT part of class
- Dataset(s) you want to use through out the class
 - Flu dataset
 - Texas Inpatient Public Use Data File (PUDF)
 - <http://www.dshs.state.tx.us/thcic/hospitals/Inpatientpudf.shtm>



28

Swap x1 & x2


- Write the code in SAS



29

Few tricks

- Divide & Conquer
 - Write code to do small things.
 - Combine the small pieces
- Look at memory (table) after each step
 - `proc print data=fn(obs=10);` where condition
- Test your code!
 - THINK about your expected output. Then check.
- Become good with an editor
 - emacs, vi, internal editors
 - copy & paste/find & replace
- Regular expression/ wild card
 - *.sas; [optional]
- grep expression files: find things in text files
- diff fn1 fn2: compare two programs



30