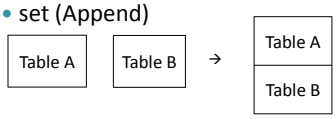
Go over

- Lab 4 answers (post now)
  - Check your own answers
  - Any particular questions?
- Mid point email
  - Who didn't submit ?
  - Not from Lab 4
  - Review together

Table Operations: multiple table → 1 table

- set (Append)



- merge (link)

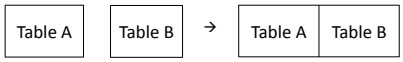


Table Operations: 1 table → 1 table (reshaping)

- Proc Transpose

1	2
a	d
b	e
c	f

→

1	a	b	c
2	d	e	f

- Proc Summary

A
B
C

→ D

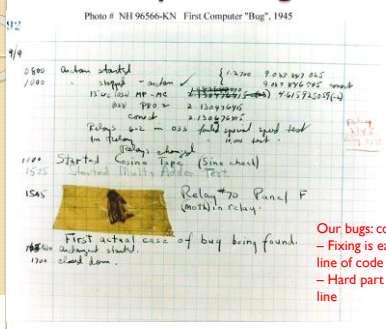
Where D=function(A,B,C)
Examples of function are
Sum(A,B,C) Mean(A,B,C) Max(A,B,C) Min(A,B,C)

Testing and Debugging

Bugs

- Glitches that cause unexpected behavior
- Range of severity
  - Program does not work at all (easy to fix)
  - Program does something, but not what I want
  - Program usually does what I want, but sometimes, … (difficult to fix)
- Debugging = removing the bugs

First computer bug



Our bugs: coding mistakes
– Fixing is easy – change one line of code
– Hard part is knowing which line

http://www.history.navy.mil/photos/images/h96000/h96566k.jpg

Syntax bugs

- easier to fix
- Usually gets ERROR message
- ERROR message
  - learn to skim useful information
  - Where the problem starts
- Incorrect grammar
- Spelling error
- Missing punctuation (; “)
- Indentation, syntax highlighting editors

Semantic bugs

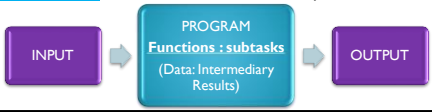
- Logical errors
- More difficult to detect and fix
- More detrimental
- ALWAYS check your table
- TRY to check for different situations
  - Multiple “print where/list if”
- Build your code modularly
- If not sure
  - binary search on line: detect from where the table is wrong

How to debug?

- good old prints, “print where”
  - `proc print; where cond`
- Use comments
  - Comment out sections of code
- “breakpoints”
  - When a line (or condition) is reached, suspend execution and check status of tables
  - Check table, before moving on to next steps

Programming: Hye-Chung Kum

- Take INPUT and turn it into OUTPUT
  - OUTPUT : Know what you want/goal
  - INPUT : figure out what you have to work with
- Change what you have (INPUT) to what you need (OUTPUT)
  - Break up the problems into small subproblems
  - Intermediate results (scrap paper)
  - Use functions to calculate the intermediary results



Think/Hypothesize output first

- Before running your code
  - Think about what you are expecting to see
    - in log (how many vars, obs)
    - Output (freq/print)
  - Run
  - Test that it is what you expected
  - If not, figure out why
    - What your hypothesis wrong?
    - If so where?
      - Program typo
      - Error in logic
      - Missing data (not located in the correct folder, in the correct form)

Programming

- Step at a time
- Jump to confirm before moving onto next step
- Know where you are going
- Check you are on track every step of the way

Preventing Bugs

- Follow best practices on small projects
  - KISS – Keep It Simple, Stupid
- Good programming practice. Helps debug
  - Small statements
  - Explicit parenthesis
  - Initialize variables: v='12345678901234567880';
  - Document assumptions

Testing Code

- Written code typically does not work at first!
- Code may also break when adding new "features."
- Keeping code simple (modular code) helps the testing
  - Easy to test a simple function
  - Not easy to test 100,000 lines of code
- Write small test-scripts to test program/function-output with respect to expected output.
- Keep and rerun the test scripts whenever changes to the code occur.
- Put "breakpoints" in the editor and step through, to catch subtle bugs. Check that the variables get the type, shape, and values that they should.
  - proc print; proc contents

Testing SAS example setup

```
* Setup data;
data test;
    num=1234;
    str=num;          * initialize str;
    newnum=.;         * setup newnum as number;
    newnum=str;       * str is converted back to newnum;
    output;

    num=123;            * initialize num;
    str=put(num, $3.);  * initialize str;
    newnum=.;         * setup newnum as number;
    newnum=str;       * str is converted back to newnum;
    output;       * writes out one obs;
```

Testing SAS example

```
* Test if conversion back and forth are correct;
data test;
    set test;

    if newnum=num then
      test=1;
    else
      test=0;

proc print data= test(obs=10);

proc freq data= test;
    tables test;
```

Good habits…

- Comments are essential – intent vs actual
- Build your program incrementally – build a little, test a little.
  - **Write modular code**
  - You will start to copy/paste/edit parts of your own code
- When there is a problem, investigate:
  - Methodically – one issue at a time!
  - Precisely – look at each variable and determine if it makes sense!
  - Creatively – e.g., can you find an example that would make the function fail, and test if it DOES?

Demog.sas

Feedback

- What have you learned?
- What do you like?
- What would you change?
- On average, how many hours per week?

Due Next Week

- Assignment 4
- Practice quiz posted
- Email me for an appointment, if you feel you are lost
- Record Linkage
  - AHRQ reading

Tips

- File I/O (resources)
- Tiny code to large code
- Grep
- Diff
- Folder structure
- Dr. Scott Long
  - http://www.indiana.edu/~jslsoc/

- Record Linkage
- OR
- Merge/reshape
  - Questions ?