Introduction to Programming
Logical Expressions & Conditionals

Hye-Chung Kum
Population Informatics Lab
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
Course URL:
http://pinformatics.org/phpm672

Operators
- Logical (\(~ / \|\), (\& / and), (\| / or)
- Relational (<, <=, ==, >)

Learn Conditional programming
- if then else end

Common Pitfalls

What we are going to learn

What we are going to learn

What we are going to learn

Relational Operators
Tests relationship between two objects

<table>
<thead>
<tr>
<th>Name</th>
<th>Operators</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalence</td>
<td>= (SAS)</td>
<td>5 == x, x == y</td>
</tr>
<tr>
<td>Equality</td>
<td>== (STATA)</td>
<td>x == y</td>
</tr>
<tr>
<td>Inequality</td>
<td>~== (SAS)</td>
<td>5 ~= x, 5 ~= (x^2 + y^2)</td>
</tr>
<tr>
<td>Binary Operators</td>
<td>&lt;</td>
<td>5 &lt; 3</td>
</tr>
<tr>
<td></td>
<td>&lt;=</td>
<td>4 &lt;= 6</td>
</tr>
<tr>
<td></td>
<td>&gt;=</td>
<td>7 &gt;= 10</td>
</tr>
<tr>
<td></td>
<td>&gt;</td>
<td>10 &gt; 7</td>
</tr>
</tbody>
</table>

Logical Operators

Boolean operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Operators</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unary Operators</td>
<td>~</td>
<td>~ (SAS) / ! (STATA)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>~ (3 == 5) = 1 (true)</td>
</tr>
<tr>
<td>Binary Operators</td>
<td>&amp;</td>
<td>T &amp; T = 1 (true)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Performs binary logic on two logical data type operands to return a logical result.

Logical Expressions

- Simple or complex expression whose final result is a single true/false logical result

Examples: Given x=3, y=4, z=5
- x == 3
- (x+y) < z
- Logical operators allow us to build up compound tests, piece by piece

Boolean Logic

Truth Tables (1=T, 0=F)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>NOT</th>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Operator Precedence (Full)

<table>
<thead>
<tr>
<th>Level</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Parentheses ( ) inner to outer</td>
</tr>
<tr>
<td>2</td>
<td>Transpose ’, Power ^</td>
</tr>
<tr>
<td>3</td>
<td>Unary plus +, Unary Minus -</td>
</tr>
<tr>
<td>4</td>
<td>Multiplication *, Division /</td>
</tr>
<tr>
<td>5</td>
<td>Addition +, Subtraction -</td>
</tr>
<tr>
<td>6</td>
<td>Comparisons &lt;, &lt;=, &gt;, &gt;=, ==</td>
</tr>
<tr>
<td>7</td>
<td>Logical ‘And’ &amp;</td>
</tr>
<tr>
<td>8</td>
<td>Logical ‘Or’</td>
</tr>
</tbody>
</table>

* Left to right rule applies

\[ x \& y \mid z = ? \text{ (put parenthesis)} \]

Boolean Logic

Truth Tables: \(x \& y \mid z\)

<table>
<thead>
<tr>
<th>(x)</th>
<th>(y)</th>
<th>(z)</th>
<th>(x &amp; y \mid z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>0 0</td>
<td>0 1</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0</td>
<td>1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1</td>
<td>1 0</td>
<td>0 0</td>
</tr>
<tr>
<td>1 0</td>
<td>0 1</td>
<td>1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>1 1</td>
<td>1 1</td>
<td>1 1</td>
<td>1 1</td>
</tr>
</tbody>
</table>

Logical Data Types

- **Data Range**
  - Conceptually: Takes on only two Values
    - true or false (1 or 0)
  - Actually:
    - false ↔ zero (0)
    - true ↔ any non-zero value (1 or greater)
  - This difference can cause subtle bugs if you are not careful.

- **Storage**
  - Conceptually: Uses a single binary bit
  - Physically/Actually: Takes a single byte

Other Logical Objects

- Functions which return logical data types as their output
- Test functions (is* functions)
  - Examples: isfloat(), isvarname(), iskeyword()
- String Comparison functions:
  - strcmp(), strcmpi(), strncmp(), strncmpi()

Motivation

- **Step by Step Programming**
  - All we have learned to do up to now...
  - Execute statements in order they occur
  - Single path through program script
- **Conditional Programming**
  - What if we only want to run the code only if some test is satisfied? (print if cond)
  - What if need to make a choice between 2 or more options?
  - How do we make the choice?

Example

```
SAS
* Initialize to default hourly rate;
* If MS, assign higher rate;
rate=10;
if edu>3 then rate=12;
proc print data=fn(obs=10);
  where gender='F' ;
```
If-end Statement
Single conditional path

- Syntax:

```plaintext
if <test> then [do;]
  commands; * 1 or more;
[end;]
```

- Tip: For the <test>, use logical expressions that evaluate to a single true/false value.

Simple Example

* One way:
  rate=10;
  if (edu > 3) then do;
    rate=12;
  end;

* Another way:
  rate=10;
  if (edu > 3) then rate=12;

If-else-end statement
Two alternatives, if <true> else <false> end

- Syntax:

```plaintext
if <test> then [do;]
  commands1; * True;
end; else do;
  commands2; * False;
end;
```

Simple Example

* One way:
  if (edu > 3) then do;
    rate=12;
  end; else do;
    rate=10;
  end;

* Another way:
  if (edu > 3) then rate=12;
  else rate=10;

If-elseif-else-end Conditional Execution
Multiple chained tests

```plaintext
if <Test1> then do;
  commands1; * T1 true;
end; else if <Test2> then do;
  commands2; * T2 true;
end; else if <Test3> then do;
  commands3; * T3 true;
end; else do;
  commands4; * all false;
end;
```

Example:

if (edu > 5) then do;
  rate=16;
end; else if (edu > 4) then do;
  rate=14;
end; else if (edu > 3) then do;
  rate=12;
end; else do;
  rate=10;
end;
Conditional Execution
Nested conditions

```plaintext
if <Test1> then do;
  if <Test2> then do;
    commands1; * T1,T2 both true;
  end; else do;
    commands2; * T1=1, T2=0;
  end;
else do;
  if <Test3> then do;
    commands3; * T1=0, T3=1;
  end; else do;
    commands4; * T1,T3 both false;
  end;
end;
```

Common Pitfalls

- Using = instead of == and vice versa.
  - SAS: same, STATA: different
    - `if x = 5 ... % Error, use if x == 5`
- Confusing & (and) and | (or)
- Inserting an extra space in a 2 character relational operator
  - `if x < = y % Error, note extra space`
  - `if x <= y % Correct`
Common Pitfalls, cont.

- Using multiple comparisons properly
  - 10 <= x <= 100  \% Error (OK in SAS)
  - (10 <= x) & (x <= 100)  \% Correct

- Forgetting the quotes when working with characters or strings
  - if letter ==y  \% Error (y is the name of var)
  - if letter =="y"  \% Correct (y is value of var)

- Comparing characters / strings (be careful)
  - 'c' < 'Z'  \% OK, compatible sizes
  - 'cat' < 'catch'  \% Error, size problem
  - strcmp('cat', 'catch')  \% Use strcmp

Logical Expressions & Conditional Programming

Reminder

- Practice using conditional logic
  - Learn logical operators ~, &, |
  - Learn relational operators <, <=, ==, >, >=
  - Logical expressions
  - If statement
  - Practice writing conditional code
  - Do the online modules

Learn to fish

- Reading: READ sections in the recommended book & modules I give you before class
- Give you good problems (lab & assignment) to learn to fish on your own
  - Lab: Read my/TA code
  - Assignment: Now write your code
- Available when you get stuck
- Top (problem) down(data) vs bottom up
  - Need to iterate

Before we start

- I will do more coding in class so you can see how coding is done
  - Remember this is just ONE way of doing it. I have very old habits from when computers were very different. So pick and choose what you think works for you
- LAB: I will share code I write, so you learn to read code
- Assignment: now try to write code to do similar things with your own data
- Computing environment is important
  - Does everyone have a stable environment?
  - Any question?
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

Lab 2: done?

Recommended Reading

- Carefully read each of the modules below. Each has very good explanations of exactly how to do certain things.
  - http://www.ats.ucla.edu/stat/sas/modules/vars.htm
  - http://www.ats.ucla.edu/stat/sas/modules/subset.htm
  - http://www.ats.ucla.edu/stat/sas/modules/missing.htm
  - http://www.ats.ucla.edu/stat/sas/modules/labels.htm
- Little SAS book
  - Sections in Chapter 3

Label variables

- SAS
  - label var1 = “LABEL”;

Label values

- SAS: define format, then use in data step

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

Label Var vs Value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bcigever</td>
<td>in8</td>
<td>1 byte</td>
<td>1 or 0</td>
</tr>
</tbody>
</table>

- 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
Label Var vs Value

- 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

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.

Type of variables (from analysis perspective)

- Var Types
  - Continuous (discrete is continuous in computers)
  - Categorical
  - Boolean
  - ID: no other information but to link tables together. i.e. random patient ID used in two tables.
  - Helps you start thinking about what you can do with the information
  - Not all variables types exist in datasets.
  - Just state NA.

Basic descriptive analysis

- Numerical
  - N, mean, max, min, std dev, unique values (mode)
  - SAS: proc means
- Categorical
  - Frequencies, cross tabulation
  - SAS: proc freq;
    - tables var1list/nocol norow nopercent;
    - tables var1*var2/nocol norow nopercent;

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

Swap x1 & x2

- Write the code in SAS