Biostatistics 615/815
Implementing Fisher’s Exact Test

Hyun Min Kang

Januray 13th, 2011

Recap - helloWorld - Have all tried?

Writing helloWorld.cpp

```cpp
#include <iostream> // import input/output handling library
int main(int argc, char** argv) {
    std::cout << "Hello, World" << std::endl;
    return 0; // program exits normally
}
```

Installing Cygwin (Windows), Xcode (MacOS), or nothing (Linux).

```bash
user@host:~/$ g++ -o helloWorld helloWorld.cpp
```

Running helloWorld

```bash
user@host:~/$ ./helloWorld
Hello, World
```

Recap - precisionExample.cpp

```cpp
#include <iostream>
int main(int argc, char** argv) {
    float smallFloat = 1e-8; // a small value
    float largeFloat = 1.; // difference in 8 (>7.2) decimal figures.
    std::cout << "smallFloat = " << smallFloat << std::endl;
    smallFloat = smallFloat + largeFloat;
    smallFloat = smallFloat - largeFloat;
    std::cout << "smallFloat = " << smallFloat << std::endl;
    // similar thing happens for doubles (e.g. 1e-20 vs 1).
    return 0;
}
```

Running precisionExample

```bash
user@host:~/$ ./precisionExample
smallFloat = 1e-08
smallFloat = 0
```

Recap - Handling command line arguments

echo.cpp - echoes command line arguments to the standard output

```cpp
#include <iostream>
int main(int argc, char** argv) {
    for(int i=1; i < argc; ++i) { // i=1 : 2nd argument (skip program name)
        if (i > 1) // print blank if there is an item already printed
            std::cout << " ";
        std::cout << argv[i]; // print each command line argument
    }
    std::cout << std::endl; // print end-of-line at the end
}
```

Compiling and running echo.cpp

```bash
user@host:~/$ g++ -o echo echo.cpp
user@host:~/$ ./echo you need to try this out by yourself!
```

You need to try this out by yourself!
Announcements

- 815 Projects will be announced in the next lecture
- Midterm date will be scheduled on March 10 - any objections?
- Final exam date will be April 21st, 10:30am-12:30pm, just like the official schedule.
- Homework #1 will be announced today

Simple Math under Fisher’s Exact Test

**Possible 2 x 2 tables**

<table>
<thead>
<tr>
<th></th>
<th>Placebo</th>
<th>Treatment</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diseased</td>
<td>x</td>
<td>a+b-x</td>
<td>a+b</td>
</tr>
<tr>
<td>Cured</td>
<td>a+c-x</td>
<td>d-a+x</td>
<td>c+d</td>
</tr>
<tr>
<td>Total</td>
<td>a+c</td>
<td>b+d</td>
<td>n</td>
</tr>
</tbody>
</table>

Hypergeometric distribution

Given \( a+b, c+d, a+c, b+d \) and \( n = a+b+c+d \),

\[
\Pr(x) = \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{x!(a+b-x)!(a+c-x)!(d-a+x)!(d-a+x)!n!}
\]

Fishers’s Exact Test (2-sided)

\[
p_{FET}(a, b, c, d) = \sum_x \Pr(x)I[\Pr(x) \leq \Pr(a)]
\]
**intFishersExactTest.cpp**

**hypergeometricProb() function**

```cpp
int fac(int n) { // calculates factorial
    int ret;
    for(ret=1; n > 0; --n) { ret *= n; }
    return ret;
}

double hypergeometricProb(int a, int b, int c, int d) {
    int num = fac(a+b) * fac(c+d) * fac(a+c) * fac(b+d);
    int den = fac(a) * fac(b) * fac(c) * fac(d) * fac(a+b+c+d);
    return (double)num/(double)den;
}
```

**Running Examples**

```
user@host:~/$ ./intFishersExactTest 2 7 8 2
Two-sided p-value is 0.4  // correct
Two-sided p-value is 4.41018 // INCORRECT
```

**factorial.cpp**

```cpp
int fac(int n) { // calculates factorial
    int ret;
    for(ret=1; n > 0; --n) { ret *= n; }
    return ret;
}

int main(int argc, char** argv) {
    int n = atoi(argv[1]);
    std::cout << n << fac(n) << std::endl;
    return 0;
}
```

**Running Examples**

```
user@host:~/$ ./factorial 12
12! = 479001600 // correct
user@host:~/$ ./factorial 13
13! = 6227020800 // INCORRECT : 13! > INT_MAX == 2147483647
```

**doubleFishersExactTest.cpp**

**new hypergeometricProb() function**

```cpp
double fac(int n) { // main() function remains the same
    double ret; // use double instead of int
    for(ret=1; n > 0; --n) { ret *= n; }
    return ret;
}

double hypergeometricProb(int a, int b, int c, int d) {
    double num = fac(a+b) * fac(c+d) * fac(a+c) * fac(b+d);
    double den = fac(a) * fac(b) * fac(c) * fac(d) * fac(a+b+c+d);
    return num/den; // use double to calculate factorials
}
```

**Running Examples**

```
user@host:~/$ ./doubleFishersExactTest 2 7 8 2
Two-sided p-value is 0.823841
Two-sided p-value is 4.41018 // INCORRECT
```

**Summary**

**Considering Precision Carefully**

**Problem - Limited Precision**

- int handles only up to fac(12)
- double handles only up to fac(170)

**Solution - Calculate in logarithmic scale**

\[
\log p_{FET} = \log \left[ \sum_x Pr(x) \frac{I(Pr(x) \leq Pr(a))}{Pr(a)} \right]
\]

\[
= \log Pr(a) + \log \left[ \sum_x \exp(\log Pr(x) - \log Pr(a)) I(\log Pr(x) \leq \log Pr(a)) \right]
\]
logFishersExactTest.cpp - main() function

```cpp
#include <iostream> // for calculating log() and exp()
#include <cmath>    // for calculating log() and exp()

double logHypergeometricProb(int a, int b, int c, int d); // defined later

int main(int argv, char** argv) {
    int a = atoi(argv[1]), b = atoi(argv[2]), c = atoi(argv[3]), d = atoi(argv[4]);
    int n = a + b + c + d;
    double logCutoff = logHypergeometricProb(a, b, c, d);
    double pFraction = 0;
    for(int x=0; x <= n; ++x) { // among all possible x
        if (a+b-x >= 0 && a+c-x >= 0 && d-a+x >= 0) { // consider valid x
            double l = logHypergeometricProb(x, a+b-x, a+c-x, d-a+x);
            if (l <= logCutoff) pFraction += exp(l - logCutoff);
        }
    }
    double logValue = logCutoff + log(pFraction);
    std::cout << "Two-sided log10-p-value is " << logValue/log(10.) << std::endl;
    return 0;
}
```

Summary:

- **Computational speed for large dataset**
  - ```time ./logFishersExactTest 2000 7000 8000 2000```
  - Two-sided log10-p-value is -1466.13, p-value is 0
  - ```real 0m42.614s```

- **Running Examples**
  - ```user@host:~/$ ./logFishersExactTest 200 700 800 200```
  - Two-sided log10-p-value is -15.2289, p-value is 5.90393e-16
  - ```user@host:~/$ ./logFishersExactTest 20 70 80 20```
  - Two-sided log10-p-value is -1.63801, p-value is 0.0230141
  - ```user@host:~/$ ./logFishersExactTest 2 7 8 2```

Even faster

```cpp
#include <iostream> // everything remains the same except for lines marked with ***
#include <cmath>

double logHypergeometricProb(double* logFacs, int a, int b, int c, int d); // ***
void initLogFacs(double* logFacs, int n); // *** New function ***

int main(int argv, char** argv) {
    int a = atoi(argv[1]), b = atoi(argv[2]), c = atoi(argv[3]), d = atoi(argv[4]);
    int n = a + b + c + d;
    double* logFacs = new double[n+1]; // *** dynamically allocate memory logFacs[0..n] ***
    initLogFacs(logFacs, n); // *** initialize logFacs array ***
    double logCutoff = logHypergeometricProb(logFacs, a, b, c, d); // *** logFacs added
    double pFraction = 0;
    for(int x=0; x <= n; ++x) { // among all possible x
        if (a+b-x >= 0 && a+c-x >= 0 && d-a+x >= 0) {
            double l = logHypergeometricProb(x, a+b-x, a+c-x, d-a+x);
            if (l <= logCutoff) pFraction += exp(l - logCutoff);
        }
    }
    double logValue = logCutoff + log(pFraction);
    std::cout << "Two-sided log10-p-value is " << logValue/log(10.) << std::endl;
    return 0;
}
```

Summary:

- **Computational speed for large dataset**
  - ```time ./fastFishersExactTest 2000 7000 8000 2000```
  - Two-sided log10-p-value is -1466.13, p-value is 0
  - ```real 0m42.614s```

- **Running Examples**
  - ```time ./fastFishersExactTest 2000 7000 8000 2000```
  - Two-sided log10-p-value is -147.563, p-value is 2.73559e-148

How to make it faster?

- Most time consuming part is the repetitive computation of factorial
  - # of logHypergeometricProbs calls is ≤ a + b + c + d = n
  - # of logFac call ≤ 9n
  - # of log calls ≤ 9n^2 - could be billions in the example above
- Key Idea is to store logFac values to avoid repetitive computation
function initLogFacs()

```cpp
void initLogFacs(double* logFacs, int n) {
    logFacs[0] = 0;
    for (int i=1; i < n+1; ++i) {
        logFacs[i] = logFacs[i-1] + log((double)i); // only n times of log() calls
    }
}
```

function logHyperGeometricProb()

```cpp
double logHyperGeometricProb(double* logFacs, int a, int b, int c, int d) {
}
```

Problem #1

Implement a program `fullFastFishersExactTest` which

- Prints out an error message and return -1 when number of input arguments are not 4 (excluding the program name itself)
- and outputs the two-sided p-value, and one-sided p-values
  
  - \( p_{2\text{sided}}(a, b, c, d) = \sum_{x} Pr(x)I[Pr(x) \leq Pr(a)] \)
  
  - \( p_{\text{greater}}(a, b, c, d) = \sum_{x > a} Pr(x) \)
  
  - \( p_{\text{lesser}}(a, b, c, d) = \sum_{x \leq a} Pr(x) \)

- based on `fastFishersExactTest.cpp`

Summary so far

- Algorithms are computational steps
- towerOfHanoi utilizing recursions
- Data types and floating-point precisions
- Operators, if, for, and while statements
- Arrays and strings
- Pointers and References
- Functions
- Fisher’s Exact Test
  - `intFishersExactTest` - works only tiny datasets
  - `doubleFishersExactTest` - handles small datasets
  - `logFishersExactTest` - handles hundreds of observations
  - `fastFishersExactTest` - equivalent to `logFishersExactTest` but faster
More Homework Problems

- Several example codes asking for the outputs (with explanations)
- Absolutely no discussion with your classmates.
- But you can consult to your computer (to compiler, not google!)
  - If you are certain what the answer is, you can just write your answer
  - If you are not certain, you can write the code, and see what happens
- Turn in your hard copy of your answers before Thursday (19th) lecture
- And email the source code (.cpp only) for Homework #1 to hmkang@umich.edu with title "BIOSTAT 615/815 Homework #1 - [your name]"

Next Lecture

More on C++ Programming

- Standard Template Library
- User-defined data types

Divide and Conquer Algorithms

- Binary Search
- Insertion Sort (skipped in lecture 1)
- Merge Sort