Solutions to Qishi Quiz Two

Si Chen, Yupeng Li, Jie Wang, Xian Xu and Yuanda Xu

November 1, 2015

1 Math/Stat

Problem 1

The total number position is 16, for each position, the combination is boy-girl or girl boy, and the total expected value is the sum of each position:
$P = \sum P_i = 16 \times \frac{7\times 9}{16\times 15} \times 2 = \frac{42}{5}$

Problem 2

Uncorrelated variable only mean $Cov(X,Y) = (X - E(X))(Y - E(Y)) = 0$, independent variables mean that $P(XY) = P(X)P(Y)$, X,Y have no relation. So independent relation is stronger and can derive the uncorrelated relation, e.g. Y=X^2.

Problem 3

Dynamic Programing from the last roll:
$E(3) = \frac{1}{6}(1+2+3+4+5+6) = 3.5,$
$E(2) = \frac{1}{6}(3E(3)+4+5+6) = 4.25,$
$E(1) = \frac{1}{6}(4E(2)+5+6) \approx 4.67,$ (1)

Problem 4 (Yupeng Li)

Assume random uniform distribution $U(0,L), X_1, X_2 \dots X_{n-1}$, and $X_1 < X_2 < \dots < X_{n-1}$ according to symmetry.
$L_1 = X_1, L_2 = X_2 - X_1, \dots L_n = 1 - X_{n-1}$
According to David and Nagaraja's Order Statistics, the probably of k particular piece is large than $l_1, l_2, l_3 ... l_k$, is $(1 - l_1 - l_2 \cdots - l_k)^{n-1}$
So, the maximum of X distribution is:
$P(V_n > x) = (1-x)^{n-1}$
The expected value for kth piece is:
$E(V_k) = \frac{1}{n}\sum \frac{1}{n-j+1}$
The maximum expected

Problem 5

Covariant Matrix and correlation matrix are semi-positive, so
$\Sigma = \begin{vmatrix} 1 & r & 0 \\ r & 1 & r \\ 0 & r & 1 \end{vmatrix}$, using Sylvester's criterion, $|\Sigma| > 0$, $\begin{vmatrix} 1 & r \\ r & 1 \end{vmatrix} > 0$, we have $1-r^2 > 0, 1-2r^2 > 0, -\frac{\sqrt{2}}{2} < r < \frac{\sqrt{2}}{2}$

Problem 6

(a)

It is a simple symmetric random walk, us the martingal: $E(Sn) = 0, E(S_n^2 - N) = 0$
$P(-1) \times -1 + P(99) \times 99 = 0,\ P(-1) + P(99) = 1 \Longrightarrow P(-1) = 0.99, P(99) = 0.01$
$E(N) = E(S_n^2) = (-1)^2 P(-1) + 99^2 P(99) = 99$

(2)

Use the mirror symmetry for the locked left door, the virtual left door should be in position -1-(99- -1)=-101, where the man reach right door after he reach left door. So:
$P(-101) \times -101 + P(99) \times 99 = 0,\ P(-101) + P(99) = 1 \Longrightarrow P(-101) = 0.495, P(99) = 0.505$
$E(N) = E(S_n^2) = (-101)^2 P(-101) + 99^2 P(99) = 9999$

Problem 7

Ridge Regression is defined mathematically:
$\min(\sum(y_i - x_i^T \beta_i)^2 - \lambda \sum \beta_i^2)$, giving the large β_i ols a square penalty term, to avoid large variation of the estimation β, all parameters β will be shrined by some ratio according to λ.

Problem 8

Ridge Regression is defined mathematically:
$\min(\sum(y_i - x_i^T \beta_i)^2 - \lambda \sum \mid \beta_i \mid)$, giving the large β_i ols a absolute penalty term, to avoid large variation of the estimation, some parameters β will be eliminated according to λ.

Problem 9

$E(N) = \frac{1}{2}2^1 + \frac{1}{2}2^2 + \frac{1}{2}2^3 + \frac{1}{2}2^4 + \cdot\cdot + \frac{1}{2}2^n = 2^n - 1/2$
But people will feel not much gain if the total number of gain has already been too much, e.g. over 100million. In such case, the intuitive feeling will stop the game, even though the actual expected value is infinite. The theory is provided by St. Peterburg Paradox, and we quote the claim: "The first proposal of this sort was due to Bernoulli himself. The same paper in which he proposed this problem contains the first published exposition of the "Principle of Decreasing Marginal Utility," which he developed to deal with St. Petersburg. This principle, later widely accepted in the theory of economic behavior, states that marginal utility (the utility obtained from consuming an extra increment of the good) decreases as the quantity consumed increases; in other words, that each additional good consumed is less satisfying than the previous one. He went on to suggest that a realistic measure of the utility of money might be given by the logarithm of the amount."

2 Programming

Problem 10

In computer programming languages the term default constructor can refer to a constructor that is automatically generated by the compiler in the absence of any programmer-defined constructors (e.g. in Java), and is usually a nullary constructor. In other languages (e.g. in C++) it is a constructor that can be called without having to provide any arguments, irrespective of whether the constructor is auto-generated or user-defined. Note that a constructor with formal parameters can still be called without arguments if default arguments were provided in the constructor's definition. (Wiki)When you do not explicitly write a no-argument constructor for a class, but want the construction of the class has some other functions, you need define your own default constructor with the function inside, though no argument included.

Problem 11

```
package com.crunchify.tutorials;
    public class ThreadSafeSingleton {
        private static final Object instance = new Object();
        protected ThreadSafeSingleton() {}// Runtime initialization // By defualt ThreadSafe
            public static Object getInstance(){
            return instance;
        }
    }
```

Through this approach we provide the necessary thread-safety, as the Singleton instance is created at class-load time. Any subsequent calls to the getInstance() method will return the already created instance. Furthermore, the implementation is optimized as we've eliminated the need for checking the value of the Singleton instance, i.e. instance == null. http://crunchify.com/thread-safe-and-a-fast-singleton-implementation-in-java/

Problem 12

Use dynamic programming,

local(k,i) have k transaction before i day and must sell at day i,

global(k,i) have k transaction before i day and can hold or sell at day i

```
  int maxProfit(int k, vector¡int¿ &prices) {
      if (prices.empty())
          return 0;
      int ans = 0;
      if (k ¿= prices.size())
    {
          for (int i = 1; i ¡ prices.size(); ++i) {
              if (prices[i] - prices[i - 1] ¿ 0) {
                  ans += prices[i] - prices[i - 1];
              }
          }
      } else {
          vector¡int¿ local(k+1);
    vector¡int¿ global(k+1);
          for (int i = 0; i ¡ prices.size() - 1; ++i) {
              int increase = prices[i + 1] - prices[i];                    for (int j = k; j ¿= 1; –j) {
```

```
                local[j] = max(global[j - 1] + max(increase, 0), local[j] + increase);
            global[j] = max(global[j], local[j]);
                                }
        }
        ans = global[k];
    }
    return ans;
}
```

Problem 13

Use ˆ to get the only number once:

```
int singleNumber(vector¡int¿ &A) {
    if (A.empty() or A.size() == 0) {
        return 0;
    }
    int rst = 0;
    for (int i = 0; i ¡ A.size(); i++) {
        rst ˆ= A[i];
    }
    return rst;
}
```

Problem 14

Yes. It's practically ok to do that. For example, the base class have one pure virtual function to generate the color of some clothes, and there is another function in the base class which generate the tab of the clothes, including the color also. In this case, the non-virtual tab function will need to call the virtual function of the color in the base class.

Problem 15

int x = rand() % range + start; check the probability of each number in large sample, a good random generator should have equal probability.

Problem 16

Use the combination of coin toll to approximate the dice:
2^n to simulate 6^m, when $2^n > 6m$, the remainder of $2^n\%6$ will be abandoned, we can get $m = n \log_3 2$, waste ratio: $r = 1 - \frac{3^m}{2^n}$

Problem 17

Let the Robert reach zero first, then allow the Robert wandering around zero with -1, 0, 1, do the same strategy to the other Robert. Then will final meet around the zero point.

```
while(){
if at-zero(R1) {Go left()}
else {Go Right()}
if at-zero(R2) {Go right()}
else {Go Left()}
}
```