STAT580 HOMEWORK4

Xingche Guo

3/10/2017

Problem1

C Code:

```c
#include<stdio.h>
#include<stdlib.h>
#include<assert.h>
#include <string.h>

void dgesv_(int *N, int *NRHS, double *A, int *LDA, int *IPIV,
                 double *B, int *LDB, int *INFO);

void dgemm_(char *TRANSA, char *TRANSB, int *M, int *N, int *K,
        double *ALPHA, double *A, int *LDA, double *B, int *LDB,
        double *BETA, double *C, int *LDC);

int main(int argc, char *argv[]){
    int i, j;
    int n = 0;
    int p = 0;
    int c;
    FILE *f;
    double *Y, *X, *X1, *XX, *XY;
    int *ipiv;
    char trans = 'T', no_trans = 'N';
    double alpha = 1, beta = 0;
    int K, N_X, N_Y, info;


    if (argc != 3){
         printf("This program is designed to calculate the regression coefficients.\n");
         printf("You need to:\n");
         printf("1. input the name of data file.\n");
         printf("2. input 1 or 0------(1=intercept, 0=no intercept). \n");
         return (1);
       }


    f = fopen(argv[1],"r");

    if (f==NULL) {
        printf("Error opening file.\n");
        return (1);
    }
```

```
      c = fgetc(f);
      while (c != EOF){
          if (c == ' ') p++;
          if (c == '\n') n++;
          c = fgetc(f);
      }

      p = p/n;     /*line 42-49 used to calculate ncol & nrow: ncol = p; nrow = n. */
      Y = (double*) malloc(sizeof(double) * n);
      assert(Y != NULL);
      X = (double*) malloc(sizeof(double) * (n*(p+1)) );
      assert(X != NULL);

      rewind(f);  /* reset the file pointer.*/

      for (i=0; i<n; i++){
          X[i] = 1;
          fscanf(f, "%lf ", &(Y[i]) );
          for (j=1; j<p; j++){
              fscanf(f, "%lf ", &(X[j*n+i]) );
          }
          fscanf(f, "%lf\n", &(X[p*n+i]) );
      }           /*assign the first column of the data to Y; assign the rest of the columns to X.*/

    K = n;
    N_Y = 1;

    printf("Sample size and the number of predictors are %d and %d respectively.\n",n,p);

if (atoi(argv[2]) == 1){   /*solve the normal equation(with intercept):X'Xb=X'y  */
      N_X = p+1;
      XX = (double*) malloc(sizeof(double) * (p+1)*(p+1));
      assert(XX != NULL);
      XY = (double*) malloc(sizeof(double) * (p+1));
      assert(XY != NULL);
      ipiv = (int*) malloc(sizeof(int) * (p+1));
      assert(ipiv != NULL);

    dgemm_(&trans, &no_trans, &N_X, &N_X, &K, &alpha, X, &K, X,
            &K, &beta, XX, &N_X);

    dgemm_(&trans, &no_trans, &N_X, &N_Y, &K, &alpha, X, &K, Y,
            &K, &beta, XY, &N_X);

    dgesv_(&N_X, &N_Y, XX, &N_X, ipiv, XY, &N_X, &info);

    if (info == 0){
        printf("The regression coefficients (with intercept):\t");
        for (i = 0; i<N_X; i++){
            printf("%f\t", XY[i]);
        }
      printf("\n");
    }
```

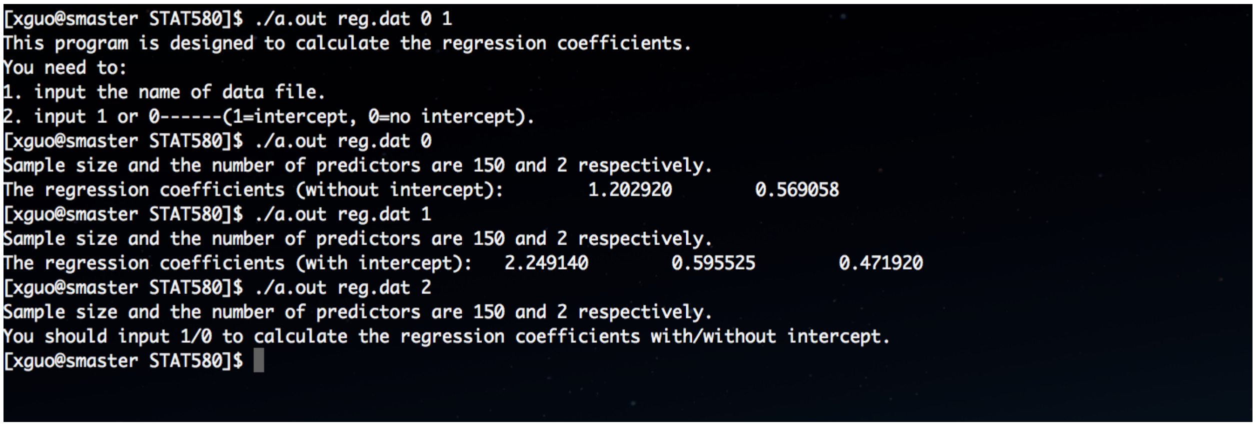
```
    else printf("dgesv error %d\n", info);
}
else if (atoi(argv[2]) == 0){  /*solve the normal equation(without intercept):X'Xb=X'y  */
      N_X = p;
      XX = (double*) malloc(sizeof(double) * p*p);
      assert(XX != NULL);
      XY = (double*) malloc(sizeof(double) * p);
      assert(XY != NULL);
      ipiv = (int*) malloc(sizeof(int) * p);
      assert(ipiv != NULL);
      X1 = (double*) malloc(sizeof(double) * n*p);
      assert(X1 != NULL);

    for (i=0; i<n*p; i++){
        X1[i] = X[i+n];
    }

    dgemm_(&trans, &no_trans, &N_X, &N_X, &K, &alpha, X1, &K, X1,
            &K, &beta, XX, &N_X);

    dgemm_(&trans, &no_trans, &N_X, &N_Y, &K, &alpha, X1, &K, Y,
            &K, &beta, XY, &N_X);

    dgesv_(&N_X, &N_Y, XX, &N_X, ipiv, XY, &N_X, &info);

    if (info == 0){
        printf("The regression coefficients (without intercept):\t");
        for (i = 0; i<N_X; i++){
            printf("%f\t", XY[i]);
        }
      printf("\n");
    }
    else printf("dgesv error %d\n", info);
}
else{
    printf("You should input 1/0 to calculate the regression coefficients with/without intercept.\n");
}

fclose(f);
return (0);
}
```

Output:

Problem2

Set n = 100000.

```
n <- 100000
```



```
[xguo@smaster STAT580]$ ./a.out reg.dat 0 1
This program is designed to calculate the regression coefficients.
You need to:
1. input the name of data file.
2. input 1 or 0------(1=intercept, 0=no intercept).
[xguo@smaster STAT580]$ ./a.out reg.dat 0
Sample size and the number of predictors are 150 and 2 respectively.
The regression coefficients (without intercept):        1.202920        0.569058
[xguo@smaster STAT580]$ ./a.out reg.dat 1
Sample size and the number of predictors are 150 and 2 respectively.
The regression coefficients (with intercept):   2.249140        0.595525        0.471920
[xguo@smaster STAT580]$ ./a.out reg.dat 2
Sample size and the number of predictors are 150 and 2 respectively.
You should input 1/0 to calculate the regression coefficients with/without intercept.
[xguo@smaster STAT580]$
```

Figure 1: Output of (4)

(a)

Generate $X_1, ..., X_n$ from $exp(1)$. The integral can be approximated by:

$$I_1 = \frac{1}{n}\sum_{i=1}^{n} h(X_i), \quad where\ h(x) = (x^2+5)x$$

```
X <- rexp(n,1)
h_X <- (X^2 +5)*X
I1 <- mean(h_X)
I1
```

```
## [1] 11.03003
```

```
var1 <- var(h_X)/n
var1
```

```
## [1] 0.008910866
```

We can also generate $X_1, ..., X_n$ from $Gamma(2,1)$. The integral can be approximated by:

$$I_2 = \frac{1}{n}\sum_{i=1}^{n} g(X_i), \quad where\ g(x) = x^2+5$$

```
X <- rgamma(n,2,1)
g_X <- (X^2 +5)
I2 <- mean(g_X)
I2
```

```
## [1] 10.9829
```

```
var2 <- var(g_X)/n
var2
```

```
## [1] 0.0008381288
```

Compared the variance of the two methods, we find that method2 is better.

(b)

Generate $X_i \sim N(0, \frac{1}{2}), Y_i \sim Unif(0,1)$, where each pair of X_i and Y_i are independent. The integral can be approximated by:

$$I = \frac{1}{n}\sum_{i=1}^{n} h(X_i, Y_i), \quad where\ h(x,y) = \sqrt{\pi}cos(xy)$$

```
X <- rnorm(n,0,sqrt(1/2))
Y <- runif(n)
h_XY <- sqrt(pi)*cos(X*Y)
I <- mean(h_XY)
I
```

```
## [1] 1.634568
```

(c)

Generate $X_1, ..., X_n$ from $exp(1)$. The integral can be approximated by:

$$I = \frac{1}{n}\sum_{i=1}^{n} h(X_i), \quad where\ h(x) = \frac{3}{4}x^4 e^{x - \frac{x^3}{4}}$$

```
X <- rexp(n,1)
h_X <- (3/4)*(X^4)*exp( X - (X^3)/4 )
I <- mean(h_X)
I
```

```
## [1] 2.275536
```

```
var1 <- var(h_X)/n
var1
```

```
## [1] 0.0001322616
```

Problem3

Generate $X_1, ..., X_n$ from $N(1.5, \nu^2)$. In this case:

$$h(x) = I(1 < x < 2)$$

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

$$w(x) = \frac{f(x)}{g(x)} = \nu \exp(-\frac{x^2}{2} + \frac{(x-1.5)^2}{2\nu^2})$$

The integral can be approximated by:

$$I = \frac{1}{N}\sum_{i=1}^{N} h(X_i)w(X_i)$$

when $\nu = 1$:

```r
N <- 100000
e <- 1
X <- rnorm(N,1.5,e)
H <- (X>1 & X<2)
W <- (e)*exp( -X^2/2 + (X-3/2)^2/(2*e^2) )
I <- mean(H*W)
I
```

```
## [1] 0.1356631
```

```r
var1 <- var(H*W)/N
var1
```

```
## [1] 3.825059e-07
```

when $\nu = 10$:

```r
e <- 10
X <- rnorm(N,1.5,e)
H <- (X>1 & X<2)
W <- (e)*exp( -X^2/2 + (X-3/2)^2/(2*e^2) )
I <- mean(H*W)
I
```

```
## [1] 0.1370361
```

```r
var2 <- var(H*W)/N
var2
```

```
## [1] 5.295599e-06
```

when $\nu = 0.1$:

```r
e <- 0.1
X <- rnorm(N,1.5,e)
H <- (X>1 & X<2)
W <- (e)*exp( -X^2/2 + (X-3/2)^2/(2*e^2) )
I <- mean(H*W)
I
```

```
## [1] 0.1322973
```

```r
var3 <- var(H*W)/N
var3
```

```
## [1] 0.0002147837
```

Problem4

(a)

Generate $U_1, ..., U_n$ from $Unif(0,1)$. The integral can be approximated by:

$$I_{MC} = \frac{1}{n}\sum_{i=1}^{n} h(U_i), \quad where\ h(x) = \frac{1}{1+x}$$

```r
U <- runif(1500)
I_mc <- mean( 1/(1+U) )
I_mc
```

```
## [1] 0.6917592
```

```r
var1 <- var(1/(1+U))/1500
var1
```

```
## [1] 1.280004e-05
```

(b)

Let $c(x) = 1 + x$ be a control variable and estimate I with:

$$I_{CV} = \frac{1}{n}\sum_{i=1}^{n} h(U_i) - b[\ \frac{1}{n}\sum_{i=1}^{n} c(U_i) - E\{c(U)\}\ ]$$

It's esay to show that:

$$E\{c(U)\} = \int_0^1 (1+u)du = \frac{3}{2}$$

$\hat{b}$ is the slope of the least-square regression line for $(h(U_i), c(U_i))$:

```r
theta_mc <- mean(1+U)
theta <- 3/2
Y <- 1/(1+U)
X <- 1+U
b <- as.numeric( coef(lm(Y~X))[2] )
I_cv <- I_mc - b*(theta_mc - theta)
I_cv
```

```
## [1] 0.6925726
```

(c)

$$Var(I_{CV}) = Var(I_{MC})(1-\rho^2), \quad where\ \rho = corr(h(U), c(U))$$

```r
var2 <- var1 * (1-cor(X,Y)^2)
var2
```

```
## [1] 4.065206e-07
```

From both formula and solution of $Var(I_{CV})$, we can see that:

$$Var(I_{CV}) < Var(I_{MC})$$

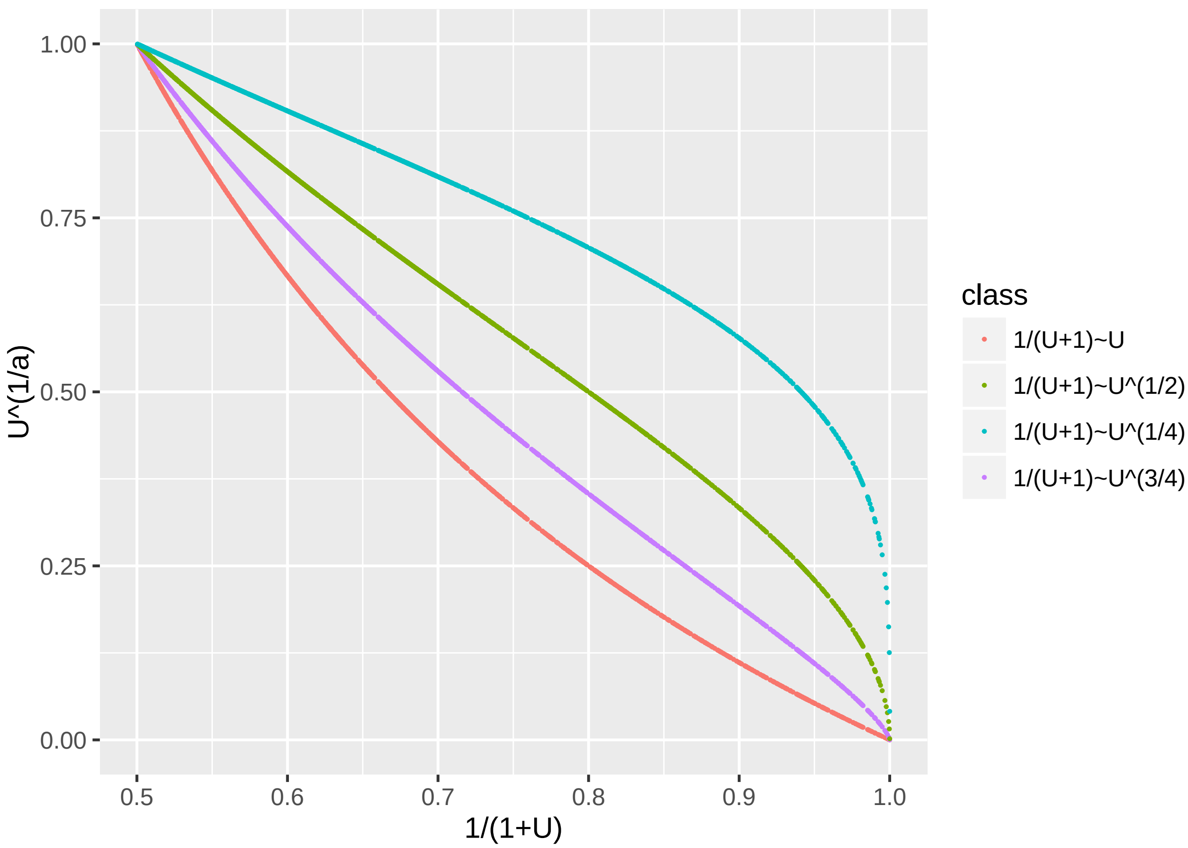
(d)

We first draw graphics and calculate the linear correlation between:

1. $(\frac{1}{1+U_i}, U_i)$.
2. $(\frac{1}{1+U_i}, {U_i}^{\frac{3}{4}})$.
3. $(\frac{1}{1+U_i}, \sqrt{U_i})$.

4. $(\frac{1}{1+U_i}, U_i^{\frac{1}{4}})$.

```
library(ggplot2)
y1 <- U
y2 <- U^(3/4)
y3 <- sqrt(U)
y4 <- U^(1/4)
Y <- c(y1,y2,y3,y4)
x <- 1/(1+U)
X <- rep(x, 4)
class <- rep(c("1/(U+1)~U","1/(U+1)~U^(3/4)","1/(U+1)~U^(1/2)","1/(U+1)~U^(1/4)"), each=1500)
d <- data.frame(X = X, Y = Y, class = as.factor(class) )

ggplot(data = d) +
  geom_point(aes(x = X, y = Y, colour = class),  size = 0.2) +
  ylab("U^(1/a)") +
  xlab("1/(1+U)")
```




```
cor(y1,x)
```

```
## [1] -0.9839922
```

```
cor(y2,x)
```

```
## [1] -0.9958827
```

```
cor(y3,x)
```

```
## [1] -0.9982002
```

```r
cor(y4,x)
```

```
## [1] -0.9807573
```

It turns out that $\sqrt{U}$ has the greatest linear correlation with $\frac{1}{1+U}$. Thus, we can use $c(x) = \sqrt{x}$ be a control variable and use the control variable method again:

```r
theta_mc1 <- mean( sqrt(U) )
theta1 <- 2/3
Y <- 1/(1+U)
X <- sqrt(U)
b <- as.numeric( coef(lm(Y~X))[2] )
I_cv1 <- I_mc - b*(theta_mc1 - theta1)
I_cv1
```

```
## [1] 0.69298
```

```r
var3 <- var1 * (1-cor(X,Y)^2)
var3
```

```
## [1] 4.603328e-08
```

The variance shows that it's a better method than in (b).