# STAT580 HOMEWORK3

*Xingche Guo*

*2/17/2017*

## Problem1

### (a)

$$\because \quad P\{U \le r(X)\} = \int_{\chi} P\{U \le r(x),\ X = x\}dx$$

$$\because \quad P\{U \le r(x),\ X = x\} = P\{U \le r(x)|X = x\}g(x) = P\{U \le r(x)\}g(x)$$

$$\therefore \quad P\{U \le r(X)\} = \int_{\chi} P\{U \le r(x)\}g(x)dx = \int_{\chi} \frac{q(x)}{\alpha g(x)}g(x)dx = \frac{1}{\alpha}\int_{\chi} q(x)dx$$

### (b)

Similarly, we have:

$$P\{U \le r(X),\ X \in A\} = \int_{\chi} P\{U \le r(x),\ x \in A\}g(x)dx$$

$$\because \int_{\chi} P\{U \le r(x),\ x \in A\}g(x)dx = \int_{A} P\{U \le r(x)\}g(x)dx$$

$$\therefore \quad P\{U \le r(X),\ X \in A\} = \int_{A} P\{U \le r(x)\}g(x)dx = \int_{A} \frac{q(x)}{\alpha g(x)}g(x)dx = \frac{1}{\alpha}\int_{A} q(x)dx$$

### (c)

We've already proved in class that:

$$P\{Y \in A\} = P\{X \in A|U \le r(X)\}$$

$$\therefore \quad P\{Y \in A\} = \frac{P\{U \le r(X),\ X \in A\}}{P\{U \le r(X)\}}$$

$$\because \quad \frac{P\{U \le r(X),\ X \in A\}}{P\{U \le r(X)\}} = \frac{\frac{1}{\alpha}\int_A q(x)dx}{\frac{1}{\alpha}\int_{\chi} q(x)dx} = \frac{\int_A q(x)dx}{\int_{\chi} q(x)dx}$$

$$\therefore \quad \frac{\int_A q(x)dx}{\int_{\chi} q(x)dx} = \frac{\int_A cq(x)dx}{\int_{\chi} cq(x)dx} = \frac{\int_A f(x)dx}{\int_{\chi} f(x)dx} = \int_A f(x)dx$$

# Problem2

## (a)

Set:

$$g(x) = C(2x^{\theta-1}e^{-x} + x^{\theta-\frac{1}{2}}e^{-x})$$

Due to the property of pdf, we have:

$$\int_0^\infty g(x)dx \equiv 1$$

$$\therefore \quad C = \frac{1}{\int_0^\infty (2x^{\theta-1}e^{-x} + x^{\theta-\frac{1}{2}}e^{-x})dx} = \frac{1}{2\int_0^\infty x^{\theta-1}e^{-x}dx + \int_0^\infty x^{\theta-\frac{1}{2}}e^{-x}dx} = \frac{1}{2\Gamma(\theta) + \Gamma(\theta+\frac{1}{2})}$$

## (b)

Set: $X \sim Gamma(\theta, 1)$, $Y \sim Gamma(\theta + \frac{1}{2}, 1)$. Therefore, we have:

$$f_X(x) = \frac{1}{\Gamma(\theta)}x^{\theta-1}e^{-x}$$

$$f_Y(x) = \frac{1}{\Gamma(\theta+\frac{1}{2})}x^{\theta-\frac{1}{2}}e^{-x}$$

$$\therefore \quad g(x) = C_1 f_X(x) + C_2 f_Y(x)$$

where:

$$C_1 = \frac{2\Gamma(\theta)}{2\Gamma(\theta)+\Gamma(\theta+\frac{1}{2})} \quad ; \ C_2 = \frac{\Gamma(\theta+\frac{1}{2})}{2\Gamma(\theta)+\Gamma(\theta+\frac{1}{2})}$$

## (c)

$$\because \quad g(x) = C_1 f_X(x) + C_2 f_Y(x) \ , \quad C_1 + C_2 = 1$$

$$\therefore \quad \int_A g(x) = C_1 \int_A f_X(x) + C_2 \int_A f_Y(x)$$

$$\therefore \quad P(Z \in A) = C_1 P(X \in A) + C_2 P(Y \in A) \quad , where\ Z \sim g(x)$$

$$\therefore \quad P(Z \in A) = C_1 P(Z \in A | Z \overset{d}{=} X) + C_2 P(Z \in A | Z \overset{d}{=} Y)$$

Define:

$$C_1 = P(Z \overset{d}{=} X), \quad C_2 = P(Z \overset{d}{=} Y)$$

$$\because \quad P(Z \overset{d}{=} X) + P(Z \overset{d}{=} Y) = 1$$

Which means $Z$ can only obey to $Gamma(\theta, 1)$ or $Gamma(\theta + \frac{1}{2}, 1)$.

Set:

$$W = \begin{cases} 1, & Z \sim Gamma(\theta, 1) \\ 0, & Z \sim Gamma(\theta + \frac{1}{2}, 1) \end{cases}$$

$$\therefore \quad P(Z \in A) = P(Z \in A | W = 1)P(W = 1) + P(Z \in A | W = 0)P(W = 0)$$

It means $P(Z)$ is the marginal distribution of $P(Z, W)$.

$$\because \quad P(Z,\ W) = P(Z|W)P(W)$$

Therefore, we can generated $(Z, W)$ by generating $W$ first and generating $Z$ by $P(Z|W)$.

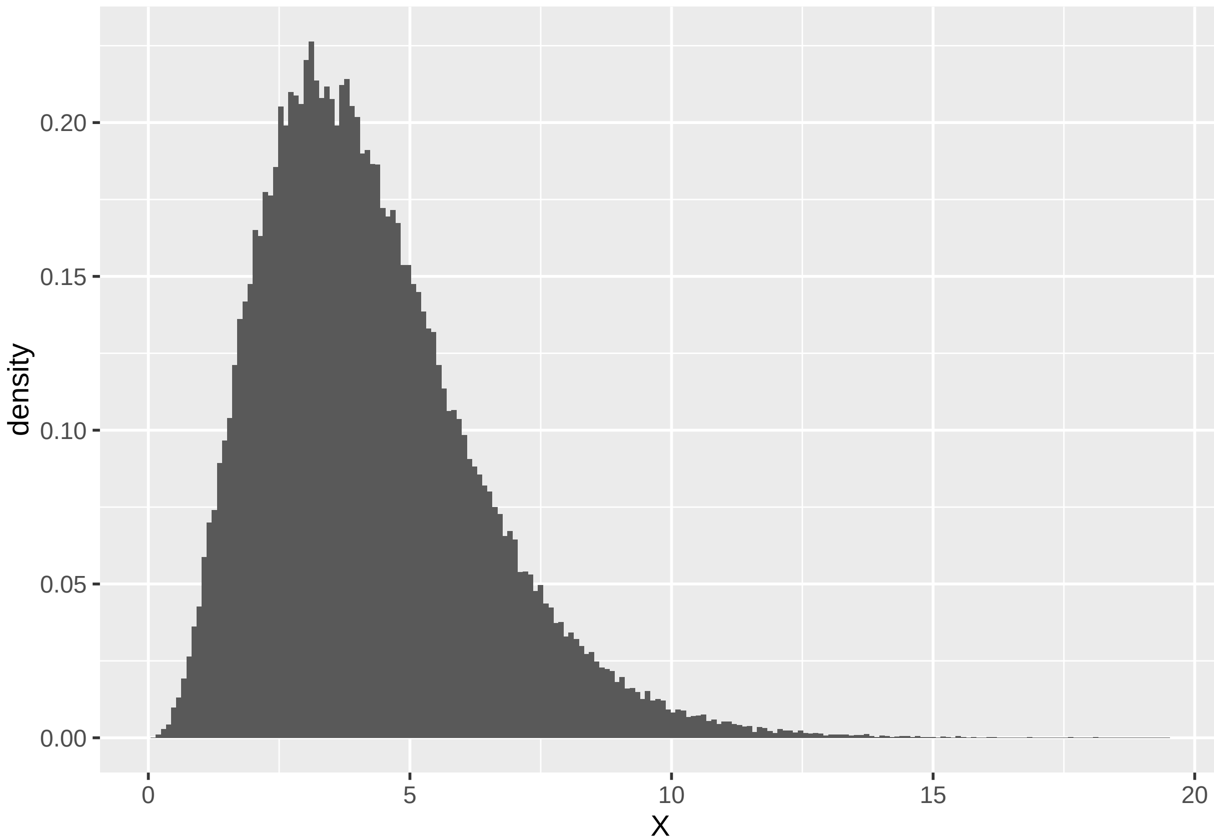
So, the algorithm is:

**Algorithm:**

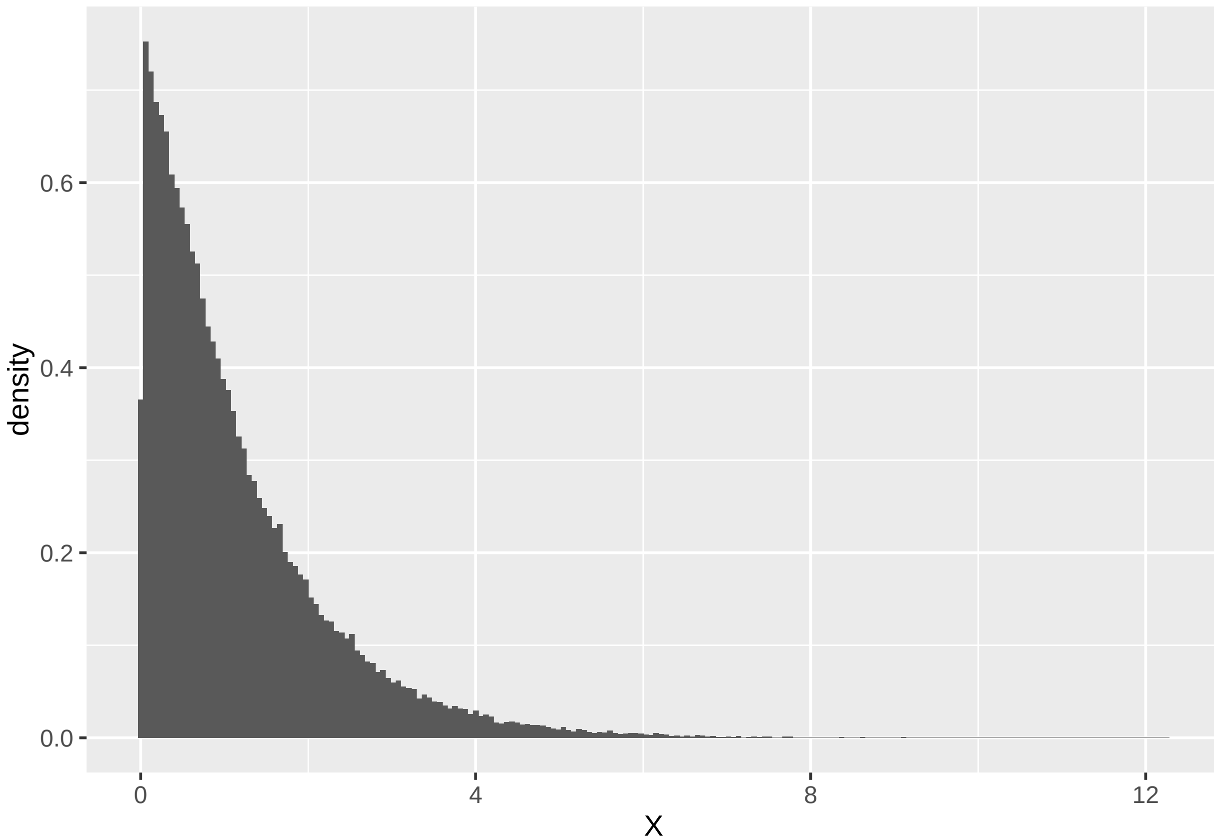
1. Generate $U \sim Unif(0, 1)$.
2. if $U < \frac{2\Gamma(\theta)}{2\Gamma(\theta)+\Gamma(\theta+\frac{1}{2})}$. Generate $Z \sim Gamma(\theta, 1)$. else, Generate $Z \sim Gamma(\theta + \frac{1}{2}, 1)$.

**code:**

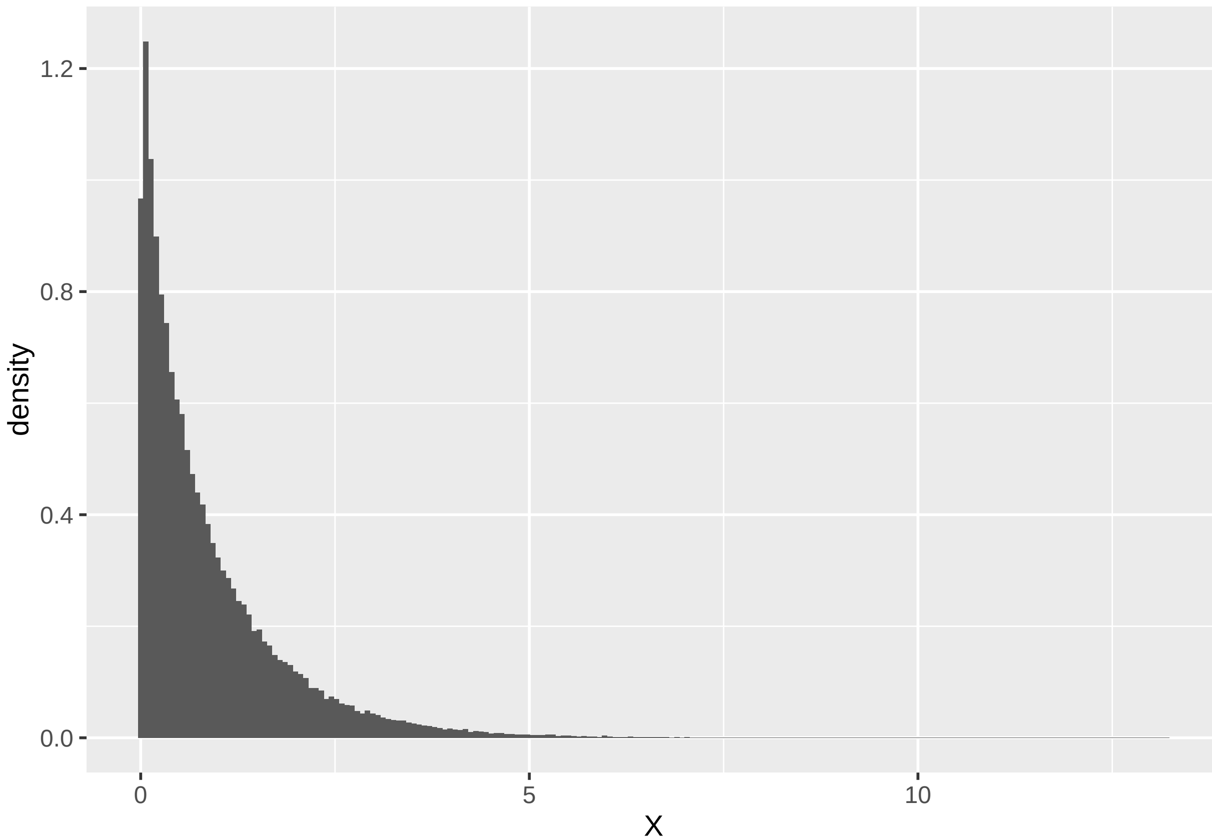
```
Rg <- function(n = 1, theta = 1){

  f<-function(x,a){
    x^(a-1) * exp(-x)
  }

  c1 <- integrate(f, a = theta, lower = 0, upper = Inf)[[1]]
  c2 <- integrate(f, a = theta+1/2, lower = 0, upper = Inf)[[1]]
  C <- 2*c1/(2*c1+c2)

  U <- runif(n)
  index <- ( U < C )
  m <- sum(index)
  X <- rep(0,n)
  X1 <- rgamma(m, theta, 1)
  X2 <- rgamma(n-m, theta+1/2, 1)
  X[index] <- X1
  X[!index] <- X2

  return(X)
}

#####theta=4
library(ggplot2)
ggplot(data = data.frame(X = Rg(100000, 4) )    , aes(x = X, ..density..) )+
  geom_histogram(bins = 200)
```




```
#####theta=1
ggplot(data = data.frame(X = Rg(100000, 1) )    , aes(x = X, ..density..) )+
  geom_histogram(bins = 200)
```




```
#####theta=3/4
ggplot(data = data.frame(X = Rg(100000, 3/4) )    , aes(x = X, ..density..) )+
  geom_histogram(bins = 200)
```




## (d)

Set:

$$q(x) = C\sqrt{4+x}\ x^{\theta-1}e^{-x}$$

$$\therefore \quad \frac{q(x)}{\alpha g(x)} = \frac{C\sqrt{4+x}\ x^{\theta-1}e^{-x}}{\alpha C(2x^{\theta-1}e^{-x} + x^{\theta-\frac{1}{2}}e^{-x})} = \frac{\sqrt{4+x}}{2+\sqrt{x}}\frac{1}{\alpha}$$

$$\because \quad \alpha = sup\frac{q(x)}{g(x)}$$

$$\because \quad (\sqrt{4+x})^2 = 4 + x \leq 4 + x + 4\sqrt{x} = (2+\sqrt{x})^2$$

$$\therefore \quad \frac{(\sqrt{4+x})^2}{(2+\sqrt{x})^2} \leq 1$$

$$\therefore \quad \frac{\sqrt{4+x}}{2+\sqrt{x}} \leq 1$$

When $x = 0$, $\alpha$ gets to the maximum, 1. Therefore, $\alpha = 1$.

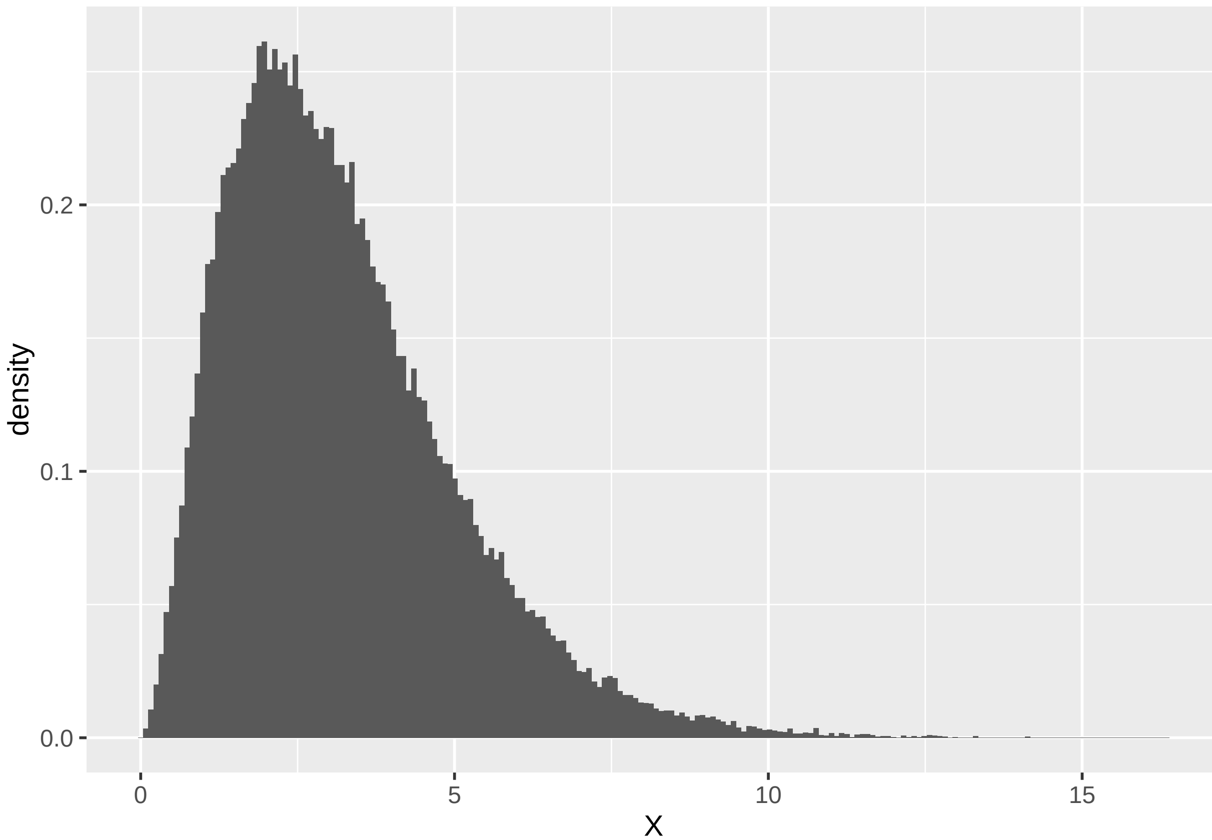
So, the algorithm is:

**Algorithm:**

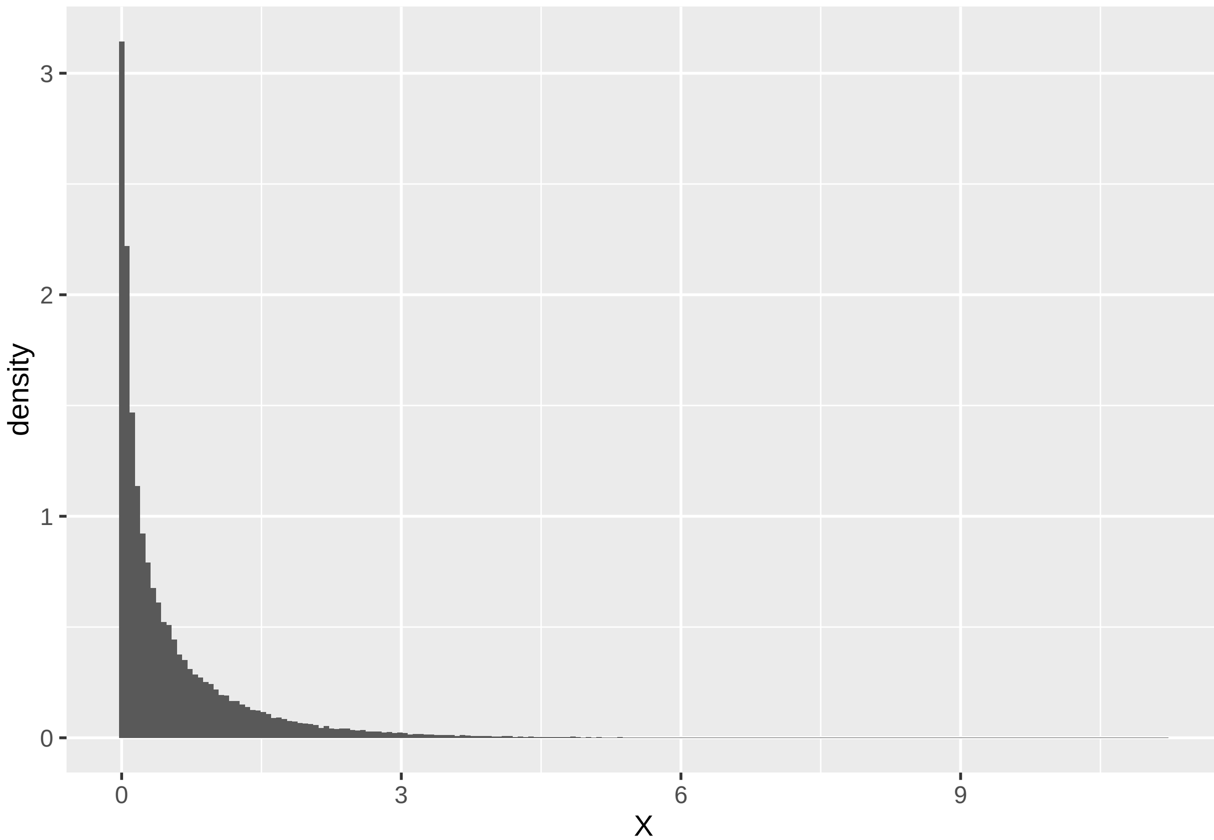
1. Generate $U \sim Unif(0,1)$ and $X \sim g(x)$ independently.
2. If $U > \frac{\sqrt{4+X}}{2+\sqrt{X}}$, then go to step 1. Otherwise, return X.

**Code:**

```
###theta=3
U <- runif(100000)
X <- Rg(100000, theta=3)

Index <- (   U <= sqrt(4+X)/(sqrt(X)+2)   )
Xf <- X[Index]
ggplot(data = data.frame(X = Xf )    , aes(x = X, ..density..) )+
  geom_histogram(bins = 200)
```




```
###theta=1/2
U <- runif(100000)
X <- Rg(100000, theta=1/2)

Index <- (   U <= sqrt(4+X)/(sqrt(X)+2)   )
Xf <- X[Index]
ggplot(data = data.frame(X = Xf )    , aes(x = X, ..density..) )+
  geom_histogram(bins = 200)
```




## Problem3

### C Code:

```
#include<stdio.h>
#define n 16  /* number of observations */
#define p 2   /* number of predictors */

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(){
/* longley dataset from R: Employed (Y) GNP.deflator and Population (X) */
    double Y[n] = {60.323,61.122,60.171,61.187,63.221,63.639,64.989,
                   63.761,66.019,67.857,68.169,66.513,68.655,69.564,
                   69.331,70.551};
    double X[n][p] =
    {{83,107.608},
    {88.5,108.632},
```

```
    {88.2,109.773},
    {89.5,110.929},
    {96.2,112.075},
    {98.1,113.27},
    {99,115.094},
    {100,116.219},
    {101.2,117.388},
    {104.6,118.734},
    {108.4,120.445},
    {110.8,121.95},
    {112.6,123.366},
    {114.2,125.368},
    {115.7,127.852},
    {116.9,130.081}};

    int i,j,k;
    double X1[n*(p+1)];
    char trans = 'T', no_trans = 'N';
    double alpha = 1, beta = 0;
    int K = n;
    int N_X = p+1;
    int N_Y = 1;
    double XX[(p+1)*(p+1)], XY[(p+1)];
    int info;
    int ipiv[p+1];


    for (k=0; k<n; k++){
        X1[k] = 1;
    }

    for (i=1; i<p+1; i++){
        for (j=0; j<n; j++){
            X1[i*n+j] = X[j][i-1];
        }
    }


    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:\t");
        for (i = 0; i<N_X; i++){
            printf("%f\t", XY[i]);
        }
      printf("\n");
    }
```

```c
    else printf("dgesv error %d\n", info);

return (0);

}
```

**Output:**

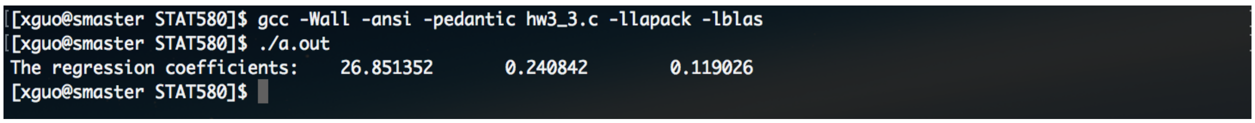



Figure 1: Output of (3)

## Problem4

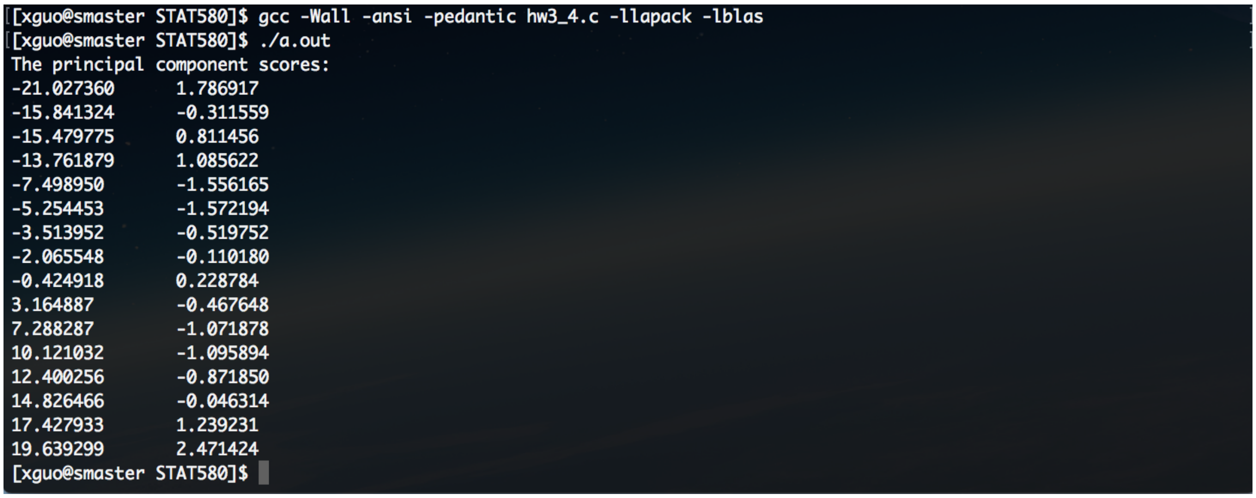
### C Code:

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

#define n 16  /* number of observations */
#define p 2   /* number of predictors */

void dgesvd_(char *JOBU, char *JOBVT, int *M, int *N, double *A,
        int *LDA, double *S, double *U, int *LDU, double *VT,
        int *LDVT, double *WORK, int *LWORK, int *INFO);


int main(){

    /* longley dataset from R */
    double X[n][p] =
    {{83,107.608},
    {88.5,108.632},
    {88.2,109.773},
    {89.5,110.929},
    {96.2,112.075},
    {98.1,113.27},
    {99,115.094},
    {100,116.219},
    {101.2,117.388},
    {104.6,118.734},
    {108.4,120.445},
    {110.8,121.95},
    {112.6,123.366},
    {114.2,125.368},
    {115.7,127.852},
```

```c
    {116.9,130.081}};

    int i, j, info;
    double X_sum[p], Z[n*p];
    char jobu = 'S';
    char jobvt = 'A';
    int MA = n;
    int NA = p;
    double S[p], U[n*p], VT[p*p];
    int lwork = -1;
    double wkopt;
    double *work;
    double UD[n][p];

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

    for (j=0; j<p; j++){
        for (i=0; i<n; i++){
            Z[i+n*j] = X[i][j] - X_sum[j]/n;
        }
    }

    dgesvd_(&jobu, &jobvt, &MA, &NA, Z, &MA, S, U, &MA, VT, &NA, &wkopt,
            &lwork, &info);

     lwork = (int)wkopt;
     work = (double*)malloc( lwork*sizeof(double) );
     assert(work != NULL);

     dgesvd_(&jobu, &jobvt, &MA, &NA, Z, &MA, S, U, &MA, VT, &NA, work,
            &lwork, &info);


    if (info == 0){
        printf("The principal component scores:\n");
        for (i=0; i<n; i++){
            for (j=0; j<p; j++){
                UD[i][j] = U[i+j*n]*S[j];
                printf("%f\t", UD[i][j]);
            }
            printf("\n");
        }

    }
    else printf("dgesvd error %d\n", info);

     return (0);
```

```
}
```

**Output:**



```
[xguo@smaster STAT580]$ gcc -Wall -ansi -pedantic hw3_4.c -llapack -lblas
[xguo@smaster STAT580]$ ./a.out
The principal component scores:
-21.027360      1.786917
-15.841324      -0.311559
-15.479775      0.811456
-13.761879      1.085622
-7.498950       -1.556165
-5.254453       -1.572194
-3.513952       -0.519752
-2.065548       -0.110180
-0.424918       0.228784
3.164887        -0.467648
7.288287        -1.071878
10.121032       -1.095894
12.400256       -0.871850
14.826466       -0.046314
17.427933       1.239231
19.639299       2.471424
[xguo@smaster STAT580]$
```

Figure 2: Output of (4)

# Problem5

## C Code:

```c
#include<stdio.h>
#define N 10

int main(){

    double x[N] = {3.1, -1.2, 5.3, 1, 4.4, 21, 3, 7, -1.2, 3.2};
    double *p, *q, *r;
    int i;
    double temp;

    printf("Original data:\n");
    for (i=0; i<N; i++){
        printf("%f\n",x[i]);
    }


    for (i=1; i<N; i++){
        r = &x[i];
        p = r;
        q = r-1;
        while (*p < *q){
                temp = *p;
                *p = *q;
```

```
                *q = temp;
                if (q == x) break;
                p--;
                q--;
        }
    }

    printf("Sorted data:\n");
    for (i=0; i<N; i++){
        printf("%f\n",x[i]);
    }
    printf("\n");

    printf("median = ");
    if ( (double)N/2 - (int)(N/2) == 0 ){
        printf("%f\n", (x[(N/2)-1]+x[N/2])/2 );
    }
    else{
            printf("%f\n", x[(N-1)/2] );
        }

    return (0);
}
```

**Output:**

```
[xguo@smaster STAT580]$ gcc -Wall -ansi -pedantic hw3_5.c
[xguo@smaster STAT580]$ ./a.out
Original data:
3.100000
-1.200000
5.300000
1.000000
4.400000
21.000000
3.000000
7.000000
-1.200000
3.200000
Sorted data:
-1.200000
-1.200000
1.000000
3.000000
3.100000
3.200000
4.400000
5.300000
7.000000
21.000000

median = 3.150000
[xguo@smaster STAT580]$
```

Figure 3: Output of (5)