HW4 - STAT 580 - Sp 2015

Yet T Nguyen

03/24/2015

1.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N  150 /* number of observations */
#define P  2 /* number of predictors */

void dgesv_(int *NN, int *NRHS, double *A, int *LDA, int *IPIV,
            double *B, int *LDB, int *INFO);
int main(int argc, char * argv[]){
    /* longley dataset from R: Employed (Y) GNP.deflator and Population (X) */
        double Y[N];
    double X[N][P];

    double XtX1[(P+1)*(P+1)], XtX0[P*P];
    double XtY1[P+1], XtY0[P];

    int ipiv1[P+1], ipiv0[P];
    int i=0, j, k, n1, n2, info;
    int itc = atoi(argv[2]);

    FILE *f;
    f = fopen(argv[1], "r");
    while(fscanf(f, "%lf%lf%lf\n",
                 &Y[i], X[i], X[i]+1)==3){
        i++;
    }
    fclose(f);
    switch(itc){
        case 1:

            XtX1[0] = N;

        for(i = 1;i<P+1;i++){
            XtX1[i*(P+1) + 0] = 0;
            for(j=0;j<N;j++)
                XtX1[i*(P+1) + 0] += X[j][i-1];
            XtX1[0*(P+1)+i] = XtX1[i*(P+1) + 0];
        }

        for(i=1;i<P+1;i++){
            for(j=1;j<P+1;j++){
                XtX1[i*(P+1)+j] = 0;
                for(k=0;k<N;k++){
                    XtX1[i*(P+1)+j]+=X[k][i-1]*X[k][j-1];
                }
            }
        }
        /* Calculate (1,X)'*Y which is a 3x1 matrix*/
        XtY1[0] = 0;
        for(i=0;i<N;i++)
        XtY1[0] +=Y[i];

        for(i=1;i<P+1;i++){
        XtY1[i] = 0;
        for(j=0;j<N;j++)
        XtY1[i] +=X[j][i-1]*Y[j];
        }
        n1 = P+1;
        n2 = 1;
        dgesv_(&n1, &n2, XtX1, &n1, ipiv1, XtY1, &n1, &info);

        if (info != 0)
        printf("dgesv error %d\n", info);
        for (i=0; i<P+1; i++)
        printf("%f\t", XtY1[i]);
        printf("\n");
        break;

        case 0:

        for(i=0;i<P;i++){
        for(j=0;j<P;j++){
        XtX0[i*P+j] = 0;
        for(k=0;k<N;k++){
        XtX0[i*P+j]+=X[k][i]*X[k][j];
        }
        }
        }

        /* Calculate (1,X)'*Y which is a 3x1 matrix*/
            for(i=0;i<P;i++){
                XtY0[i] = 0;
                for(j=0;j<N;j++)
                    XtY0[i] +=X[j][i]*Y[j];
            }
        n1 = P;
        n2 = 1;
        dgesv_(&n1, &n2, XtX0, &n1, ipiv0, XtY0, &n1, &info);

        if (info != 0)
            printf("dgesv error %d\n", info);
        for (i=0; i<P; i++)
            printf("%f\t", XtY0[i]);
        printf("\n");
        break;

    }
    /* Calculate (1, X)'*(1, X) which is a 3X3 matrix*/

    return 0;
    }
    /* gcc -pedantic -Wall -ansi  hw33.c -llapack -lblas -lgfortran */
    /* 26.851352  0.240842    0.119026*/#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N  150 /* number of observations */
```

```
#define P  2 /* number of predictors */

void dgesv_(int *NN, int *NRHS, double *A, int *LDA, int *IPIV,
            double *B, int *LDB, int *INFO);
int main(int argc, char * argv[]){
  /* longley dataset from R: Employed (Y) GNP.deflator and Population (X) */
    double Y[N];
double X[N][P];

double XtX1[(P+1)*(P+1)], XtX0[P*P];
double XtY1[P+1], XtY0[P];

int ipiv1[P+1], ipiv0[P];
int i=0, j, k, n1, n2, info;
int itc = 0;
FILE *f;
itc = atoi(argv[2]);
if (argc != 3){
    printf("This program have 2 arguments: data intercept \n");
    printf("data: data file\n");
    printf("intercept : 1 = intercept, 0 = no intercept\n");
    return 1;
  }


  f = fopen(argv[1], "r");
  while(fscanf(f, "%lf%lf%lf\n",
               &Y[i], X[i], X[i]+1)==3){
                i++;
               }
fclose(f);
printf("Sample size and number of predictors are %d and %d respectively.\n", N, P);
switch(itc){
  case 1:

XtX1[0] = N;

for(i = 1;i<P+1;i++){
  XtX1[i*(P+1) + 0] = 0;
  for(j=0;j<N;j++)
  XtX1[i*(P+1) + 0] += X[j][i-1];
  XtX1[0*(P+1)+i] = XtX1[i*(P+1) + 0];
}

for(i=1;i<P+1;i++){
  for(j=1;j<P+1;j++){
    XtX1[i*(P+1)+j] = 0;
    for(k=0;k<N;k++){
      XtX1[i*(P+1)+j]+=X[k][i-1]*X[k][j-1];
    }
  }
}

/* Calculate (1,X)'*Y which is a 3x1 matrix*/
        XtY1[0] = 0;
    for(i=0;i<N;i++)
        XtY1[0] +=Y[i];

    for(i=1;i<P+1;i++){
        XtY1[i] = 0;
        for(j=0;j<N;j++)
            XtY1[i] +=X[j][i-1]*Y[j];
    }
    n1 = P+1;
    n2 = 1;
    dgesv_(&n1, &n2, XtX1, &n1, ipiv1, XtY1, &n1, &info);

    if (info != 0)
        printf("dgesv error %d\n", info);
    printf("The regression coefficients: ");
    for (i=0; i<P+1; i++)
        printf("%f ", XtY1[i]);
    printf("\n");
    break;

    case 0:

        for(i=0;i<P;i++){
            for(j=0;j<P;j++){
                XtX0[i*P+j] = 0;
                for(k=0;k<N;k++){
                    XtX0[i*P+j]+=X[k][i]*X[k][j];
                }
            }
        }

    /* Calculate (1,X)'*Y which is a 3x1 matrix*/
for(i=0;i<P;i++){
  XtY0[i] = 0;
  for(j=0;j<N;j++)
  XtY0[i] +=X[j][i]*Y[j];
}
n1 = P;
n2 = 1;
dgesv_(&n1, &n2, XtX0, &n1, ipiv0, XtY0, &n1, &info);

if (info != 0)
printf("dgesv error %d\n", info);
printf("The regression coefficients: ");
for (i=0; i<P; i++)
printf("%f ", XtY0[i]);
printf("\n");
break;

}
/* Calculate (1, X)'*(1, X) which is a 3X3 matrix*/

        return 0;
}
/* gcc -pedantic -Wall -ansi  hw41.c -llapack -lblas -lgfortran */
  /* ./a.out reg.dat 1*/
```

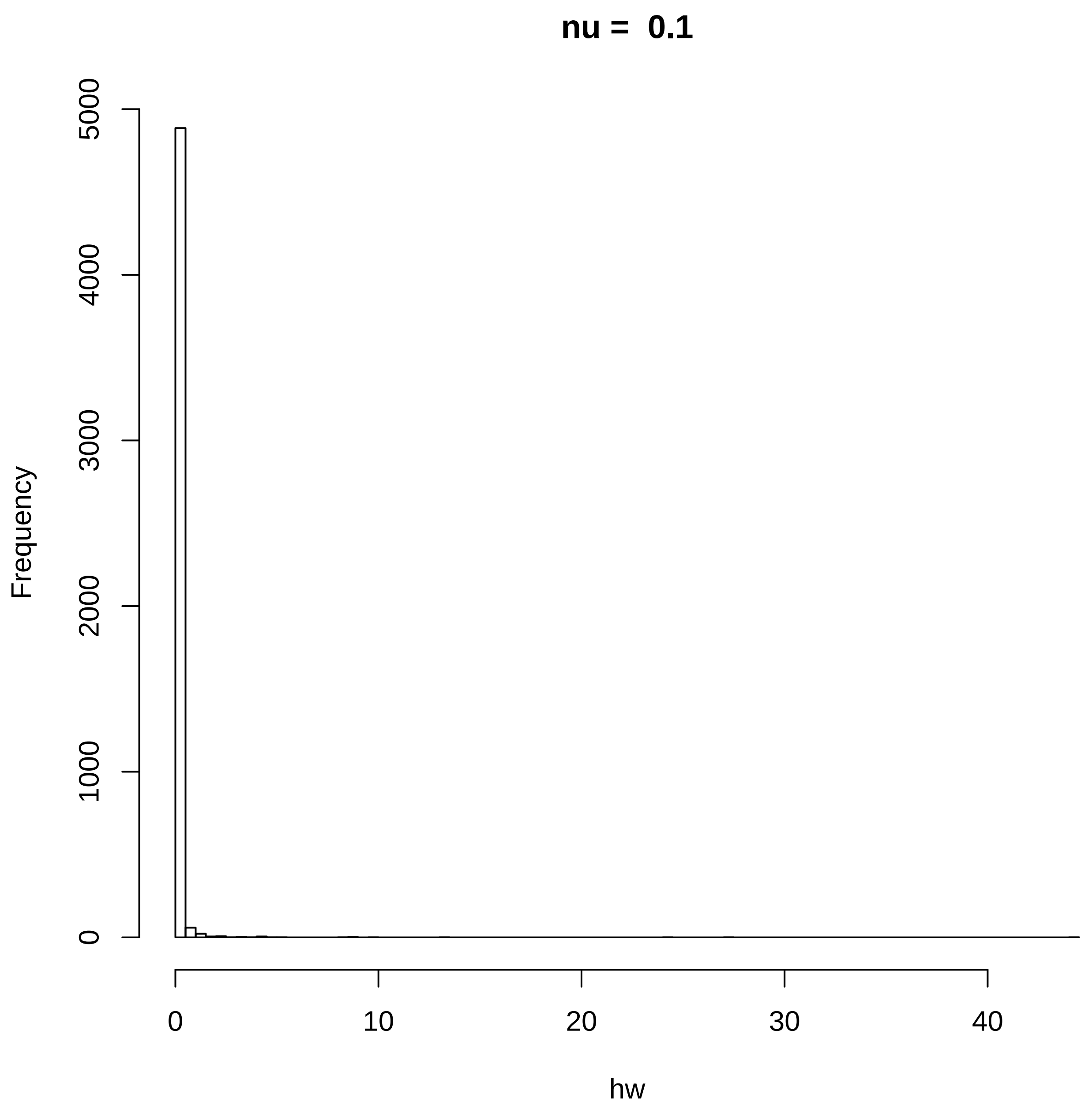
2.

```
#a.
set.seed(1)
n <- 5000
x <- runif(n, 0, 1)
```

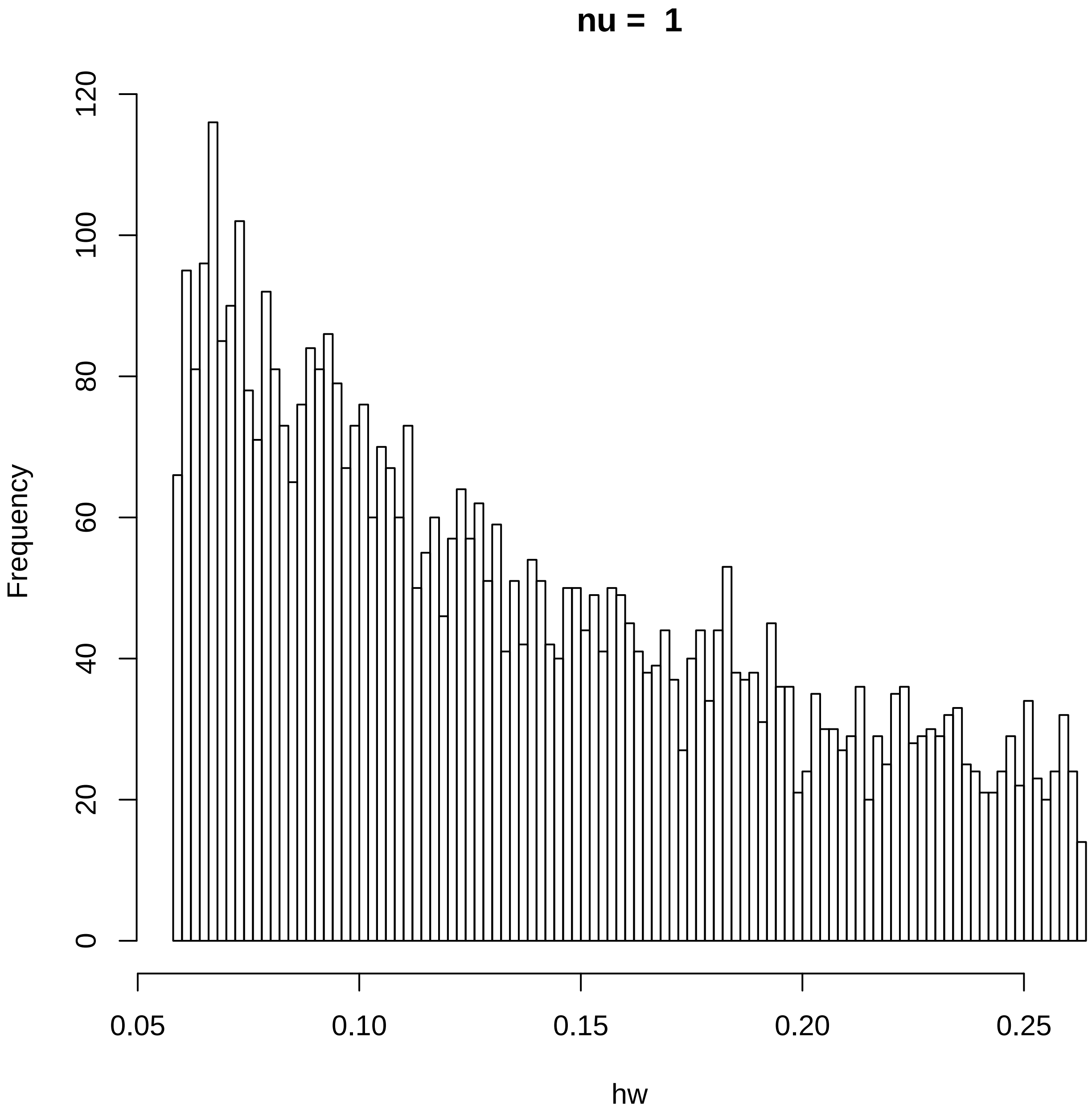
```
h <- x^2
mu <- mean(h)
varmu <- var(h)/n
cbind(mu, varmu)

##              mu        varmu
## [1,] 0.3333861 1.848547e-05

#b.

x <- runif(n, -2,2)
y <- runif(n, 0,1)
h <- 4*x^2*cos(x*y)
mu <- mean(h)
varmu <- var(h)/n
cbind(mu, varmu)

##             mu       varmu
## [1,] 3.449034 0.002507705

#c.
# let $x^3/4 = t$ then $3x^2/4dx = dt$ and
# $\int_0^\infty 3x^4/4 e^{-x^3/4}dx = 4^{2/3}\int_0^\infty t^{2/3} e^{-t}dt$
x <- rexp(n, 1)
h <- (4*x)^(2/3)
mu <- mean(h)
varmu <- var(h)/n
cbind(mu, varmu)

##             mu        varmu
## [1,] 2.279612 0.0004694551
```

3.

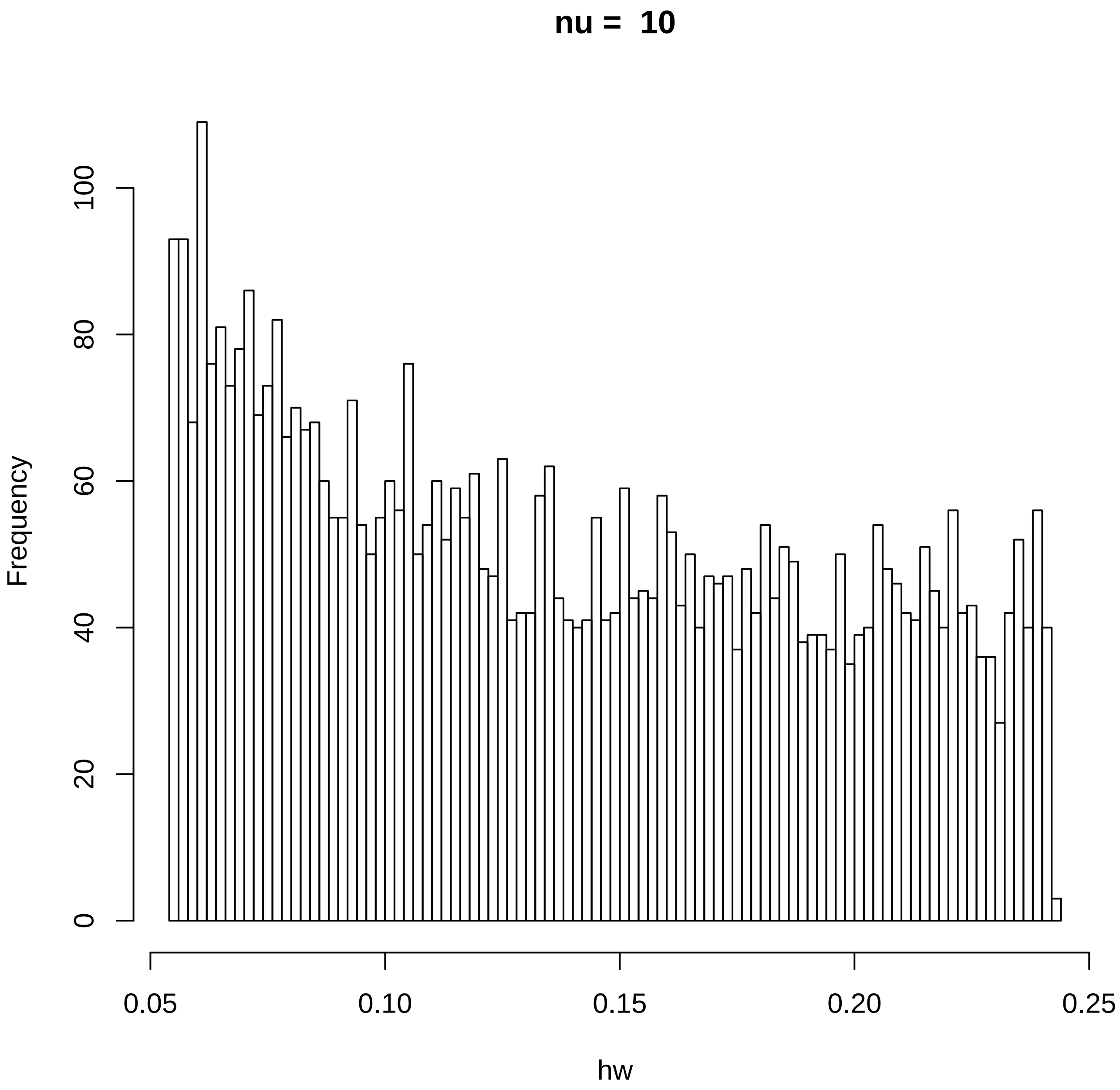
```
#
# install.packages("msm")
set.seed(1)
library(msm)
iss <- function(muu, nu){
  n <- 5000
  x <- rtnorm(n, mean = muu, sd = nu, lower = 1, upper = 2)
  h <- 1/sqrt(2*pi)*exp(-x^2/2)
  q <- 1
  g <-dtnorm(x, mean = muu, sd = nu,  lower = 1, upper = 2)
  w <- q/g
  hw <- h*w
  mu <- mean(hw)
  varmu <- var(hw)/n
  hist(hw, nclass = 100, main = paste("nu = ", nu))
  res <- cbind(mu,varmu)
  return(res)
}

iss(1.5,.1)
```



```
##               mu        varmu
## [1,] 0.1192351 0.0001633399

iss(1.5,1)
```



```
##              mu        varmu
## [1,] 0.1364263 6.504126e-07

iss(1.5,10)
```



```
##              mu        varmu
## [1,] 0.1368361 6.226023e-07
```

The case $\nu = .1$, there are some extreme values, the other cases do not.

4.

```
#4.
#(a)
n <- 1500
u <- runif(n, 0, 1)
hu <- 1/(u+1)
Ihat <- mean(hu)
Ihat

## [1] 0.690369
```

```
#sum((hu-Ihat)^2)/(n*(n-1))

#(b)
ecu <- 3/2
cu <- (u+1)
mcu <- mean(cu)
hu <- 1/(u+1)

b <- (1-log(2)*(1+1/2))/(1/12)

Icv <- mean(hu) - b*(mcu - ecu)
Icv

## [1] 0.6936498

#(c)
Ihat

## [1] 0.690369

Icv

## [1] 0.6936498

vIhat <- var(hu)/n
vIhat

## [1] 1.318234e-05

rho2 <- abs((1-log(2)*(1+1/2)))/sqrt((1/12)*(1/2-(log(2))^2))
rho2

## [1] 0.9841661

vIcv <- vIhat*(1-rho2)
# the variance of Icv less than that of Ihat 98.4%

#(d)
# If we can chose the another function of x, i.e., g(x), such that correlation coefficient of
# g(x) and h(x) is larger than that one of the previous part, then we can obtain a new
# Icv_g which has smaller variance comparing to the Icv above
```