

# STAT 580 Homework 4

Yifan Zhu

March 19, 2017

1.

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

void dgesv_(int *n, int *nrhs, double *a, int *lda, int *ipiv, double *b,
            int *ldb, int *info);

int main(int argc, char *argv[])
{
    if (argc != 3)
    {
        printf("This program performs a linear regression and returns the regression coefficients.\n");
        printf("Arguments: data intercept\n");
        printf("data: data_file\n");
        printf("intercept: 1 = intercept, 0 = no intercept\n");
        return 1;
    }

    FILE *f;
    int N = 0, P = 0;
    int i, j, k;
    char cursor;

    f = fopen(argv[1], "r");
    while (((cursor = fgetc(f)) != EOF) && (cursor != '\n'))
    {
        if (cursor == '_') P++;
    }
    rewind(f);
    while ((cursor = fgetc(f)) != EOF)
    {
        if (cursor == '\n') N++;
    }
    rewind(f);

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

    double data[N * (P + 1)];

    for (i = 0; i < N * (P + 1); i++)
    {
```

```

        fscanf(f, "%lf", &data[i]);
    }

    fclose(f);

    double Y[N];
    for (i = 0; i < N; i++)
    {
        Y[i] = data[i * (P + 1)];
    }

    if (atoi(argv[2]) == 1)
    {
        double X[N][(P + 1)];
        int n1 = P + 1, n2 = 1, ipiv[P + 1], info;
        double XtX[n1 * n1];
        double XtY[n1];
        for (i = 0; i < N; i++)
        {
            X[i][0] = 1;
            for (j = 1; j < n1; j++)
            {
                X[i][j] = data[i * (P + 1) + j];
            }
        }

        for (i = 0; i < n1; i++)
        {
            for (j = 0; j < n1; j++)
            {
                XtX[i * n1 + j] = 0;
                for (k = 0; k < N; k++)
                    XtX[i * n1 + j] += X[k][i] * X[k][j];
            }
        }

        for (i = 0; i < n1; i++)
        {
            XtY[i] = 0;
            for (j = 0; j < N; j++)
            {
                XtY[i] += X[j][i] * Y[j];
            }
        }

        /* XtX is symmetric, no transpose needed before passing to
           Fortran subroutine */
        dgesv_(&n1, &n2, XtX, &n1, ipiv, XtY, &n1, &info);
        if (info != 0) printf("failure_with_error_%d\n", info);

        /* print beta */
        printf("The_regression_coefficients:_");
        for (i = 0; i < n1; i++)
        {
            printf("%f_", XtY[i]);
        }
        printf("\n");
    }

```

```

}
else if (atoi(argv[2]) == 0)
{
    double X[N][P];
    int n1 = P, n2 = 1, ipiv[P], info;
    double XtX[n1 * n1];
    double XtY[n1];
    for (i = 0; i < N; i++)
    {
        for (j = 0; j < n1; j++)
        {
            X[i][j] = data[i * (P + 1) + j + 1];
        }
    }

    for (i = 0; i < n1; i++)
    {
        for (j = 0; j < n1; j++)
        {
            XtX[i * n1 + j] = 0;
            for (k = 0; k < N; k++)
                XtX[i * n1 + j] += X[k][i] * X[k][j];
        }
    }

    for (i = 0; i < n1; i++)
    {
        XtY[i] = 0;
        for (j = 0; j < N; j++)
        {
            XtY[i] += X[j][i] * Y[j];
        }
    }

    /* XtX is symmetric, no transpose needed before passing to
       Fortran subroutine */
    dgesv_(&n1, &n2, XtX, &n1, ipiv, XtY, &n1, &info);
    if (info != 0) printf("failure_with_error_%d\n", info);

    /* print beta */
    printf("The_regression_coefficients:_");
    for (i = 0; i < n1; i++)
    {
        printf("%f_", XtY[i]);
    }
    printf("\n");
}

return 0;
}

```

2. (a) Let  $X \sim \text{Exponential}(1)$ ,

$$\int_0^{\infty} (x^2 + 5)xe^{-x}dx = E((X^2 + 5)X) \approx 10.61867$$

```
n <- 1500;
X <- rexp(n);
mean((X^2 + 5)*X)
```

- (b) Let  $X \sim N(0, 1/2)$ ,  $Y \sim \text{Unif}(0, 1)$  be independent. Then

$$\int_0^1 \int_{-\infty}^{\infty} e^{-x^2} \cos(xy) dx dy = \int_0^1 \int_{-\infty}^{\infty} \sqrt{\pi} \cos(xy) \frac{1}{\sqrt{\pi}} e^{-x^2} dx dy = E(\sqrt{\pi} \cos(XY)) \approx 1.698413$$

```
n <- 1500;
X <- rnorm(n, mean = 0, sd = 1/2);
Y <- runif(n);
mean(sqrt(pi)*cos(X*Y))
```

- (c) Let  $X \sim \text{Weibull}(3, \sqrt[3]{4})$ , then

$$\int_0^{\infty} \frac{3}{4} x^4 e^{-x^3/4} dx = \int_0^{\infty} x^2 \frac{3}{\sqrt[3]{4}} \left( \frac{x}{\sqrt[3]{4}} \right)^2 e^{-\left( \frac{x}{\sqrt[3]{4}} \right)^3} dx = E(X^2) \approx 2.253528$$

```
n <- 1500;
X <- rweibull(n, shape = 3, scale = 4^(1/3));
mean(X^2)
```

3.

$$I = \int_1^2 \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx = \int_{-\infty}^{\infty} h(x)f(x)dx$$

Here

$$h(x) = \begin{cases} 1 & , x \in (0, 1) \\ 0 & , \text{otherwise} \end{cases}$$

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

$g(x) = \frac{1}{\sqrt{2\pi\nu}} e^{-(x-1.5)^2/(2\nu^2)}$ , then

$$h(x)w^*(x) = h(x) \frac{f(x)}{g(x)} = \begin{cases} \nu e^{-\frac{x^2}{2} + \frac{(x-1.5)^2}{2\nu^2}} & , x \in (0, 1) \\ 0 & , \text{otherwise} \end{cases}$$

Thus let  $X \sim N(1.5, \nu^2)$

$$\frac{1}{2\pi} \int_1^2 e^{-x^2/2} = E(h(X)w^*(X)) \approx \begin{cases} 0.1235178 & , \nu = 0.1 \\ 0.1354527 & , \nu = 1 \\ 0.1339445 & , \nu = 10 \end{cases}$$

```
mcint <- c(0,0,0);
i <- 1;

for (v in c(0.1, 1, 10)){
  n <- 150000;
  X <- rnorm(n, mean = 1.5, sd = v)
  mcint[i] <- mean(v * exp(-X^2/2 + (X - 1.5)^2/(2*v^2)) * ifelse(X > 1
    & X < 2, 1, 0))
  i <- i+1
}
mcint
```

4. (a)  $\hat{I}_{MC} = 0.6918192$

(b)  $E(c(U)) = E(1 + U) = 1.5$ ,  $\hat{b} = \frac{\widehat{Cov}(h(U), c(U))}{\widehat{Var}(c(U))} = 12\widehat{Cov}(1/(1 + U), 1 + U)$ .

$$\hat{I}_{CV} = 0.692778$$

(c)

$$\widehat{Var}(\hat{I}_{MC}) = 1.289725 \times 10^{-5}, \widehat{Var}(\hat{I}_{CV}) = 4.309791 \times 10^{-7}$$

Hence  $\widehat{Var}(\hat{I}_{CV}) < \widehat{Var}(\hat{I}_{MC})$ .

(d) Let  $c_1(U) = e^{-U}$ , then  $E(c_1(U)) = 1 - \frac{1}{e}$ .  $\hat{b}_1 = \widehat{Cov}(h(U), c_1(U))/\widehat{Var}(c_1(U))$ . Then

$$\hat{I}_{CV_1} = 0.6931038, \widehat{Var}(\hat{I}_{CV_1}) = 3.678175 \times 10^{-8} < \widehat{Var}(\hat{I}_{CV})$$

```
n <- 1500;
U <- runif(n);
h <- 1/(1 + U);
```

```

IMC <- mean(h);

IMC

c <- 1 + U;
b <- 12 * cov(h,c);

ICV <- mean(h) - b * (mean(c) - 1.5);

ICV

VIMC <- var(h)/n;

VIMC

VICV <- var(h - b*c)/n

VICV

c1 <- exp(-U);
b1 <- cov(h, c1)/var(c1);

ICV1 <- mean(h) - b1 * (mean(c1) - (1 - exp(-1)));

ICV1

VICV1 <- var(h - b1 * c1)/n;

VICV1

```