# HW5_Salem_Mohamed

*Mohamed Salem*

*September 29, 2019*

Problem 2

```r
# Seed Setting and Data Generation
set.seed(12345)
y <- seq(from = 1, to = 100, length.out = 1e+08) + rnorm(1e+08)
Ey <- mean(y)

# Using a for loop to compute Sum of Squares
system.time({
    Sy <- numeric(length(y))
    for (i in seq(1:length(y))) {
        Sy[i] <- (y[i] - Ey)^2
    }
    SSy <- data.frame(sum(Sy))
    rm(Sy)
})
```

```
##    user  system elapsed
##    7.90    0.24    8.14
```

```r
# Using Matrix algebra to achieve the same result
system.time({
    y_vec <- y - Ey
    SSy_mat <- t(y_vec) %*% y_vec
})
```

```
##    user  system elapsed
##    1.05    0.36    1.43
```

Problem 3

```r
# Seed setting ad generating data
set.seed(1256)
theta <- as.matrix(c(1, 2), nrow = 2)
X <- cbind(1, rep(1:10, 10))
h <- X %*% theta + rnorm(100, 0, 0.2)

# Setting up the computation
m <- length(X[, 1])
eps <- 1e-06
alpha = 0.02
theta_hat <- as.matrix(c(0, 0), nrow = 2)
y <- X %*% theta_hat
theta_hat = theta_hat - alpha * (1/m) * (t(X) %*% (y - h))

while (abs(-alpha * (1/m) * (t(X[, 1]) %*% (y - h))) > eps &&
    abs(-alpha * (1/m) * (t(X[, 2]) %*% (y - h))) > eps) {
    y <- X %*% theta_hat
    theta_hat = theta_hat - alpha * (1/m) * (t(X) %*% (y - h))
}
```

```r
coef(lm(h ~ 0 + X))
```

```
##        X1        X2
## 0.9695707 2.0015630
```

```r
theta_hat
```

```
##          [,1]
## [1,] 0.967923
## [2,] 2.001800
```

Rather than inverting matrices, we can solve:

$$\hat{\beta} = (X'X)^{-1}X'\underline{y}$$

by using the "b_solving" code displayed below. We show that this leads to the same result as inverting.

```r
n <- 5e+05
X <- runif(n, min = 1, max = 50)
Y <- 7 + 5 * X

system.time({
    b_solving <- solve(t(X) %*% X, t(X) %*% Y)
})
```

```
##    user  system elapsed
##    0.02    0.00    0.03
```

```r
system.time({
    b_inverting <- solve(t(X) %*% X) %*% t(X) %*% Y
})
```

```
##    user  system elapsed
##    0.02    0.00    0.01
```

```r
print(b_solving)
```

```
##         [,1]
## [1,] 5.20978
```

```r
print(b_inverting)
```

```
##         [,1]
## [1,] 5.20978
```

```r
set.seed(12456)
G <- matrix(sample(c(0, 0.5, 1), size = 1600, replace = T), ncol = 10)
R <- cor(G)  # R: 10 * 10 correlation matrix of G
rm(G)
C <- kronecker(diag(1600), R)  # C is a 16000 * 16000 block diagonal matrix
id <- sample(1:16000, size = 932, replace = F)
C <- C[, -id]
A <- C[id, ]  # matrix of dimension 932 * 15068
ASize_old <- object.size(A)
A <- as(A, "sparseMatrix")
B <- C[-id, ]  # matrix of dimension 15068 * 15068
BSize_old <- object.size(B)
```

```r
rm(C)  #save some memory space
B <- as(B, "sparseMatrix")
system.time({
    q <- sample(c(0, 0.5, 1), size = 15068, replace = T)  # vector of length 15068
    p <- runif(932, 0, 1)
    r <- runif(15068, 0, 1)
    Bi <- solve(B)
    y = p - A %*% Bi %*% (q - r)
})
```

```
##    user  system elapsed
##    2.05    0.00    2.56
```

```r
ASize <- object.size(A)
BSize <- object.size(B)
print(paste("A size:", ASize, "bytes", "|  B size:", BSize, "bytes"))
```

```
## [1] "A size: 156504 bytes |  B size: 1775208 bytes"
```

```r
print(paste("old A size:", ASize_old, "bytes", "|  old B size:",
    BSize_old, "bytes"))
```

```
## [1] "old A size: 112347224 bytes |  old B size: 1816357208 bytes"
```