# Homework 4

Ashutosh Ekade

### Question 1: Linear Regression with Normal Errors

Load the `BostonHousing2` dataset from the `mlbench` package, which has data on 506 census tracts from the 1970's. Assume the model is $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + e$ with $e \sim N(0, \sigma^2)$, observations are independent, and where $Y$ denotes `medv` (the median value of homes in 1000's of dollars), $X_1$ denotes `rm` (the average number of rooms per home), $X_2$ denotes `age` (the proportion of older homes), and $X_3$ denotes `crim` (the crime rate in the area). See `?mlbench::BostonHousing2` for definitions of these variables.

```
#install.packages('mlbench')
library(mlbench)
data('BostonHousing2')
```

#### 1.a) Write, mathematically, the joint log-likelihood function. What link function (or inverse link function) is required to "connect" $\mu_i$ to $x_i'\beta$?

$$

$$
l(\beta, \sigma^2) = \sum_{i=1}^{n} \log f(Y_i | Y_i, \beta, \sigma^2) \tag{2}
$$
$$
= -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i - X_i'\beta)^2 \tag{3}
$$
$$
= -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y - X\beta)'(Y - X\beta) \tag{4}
$$

(1)

(5)

$$

**1.b) Write an R function to calculate the log-likelihood function from 1a above. Your function should take 3 arguments: (1) $\theta = (\beta, \sigma^2)$ a vector of all (five) parameters, (2) an $n \times k$ matrix $X$, and (3) a vector or $n \times 1$ matrix $y$.**

```
log_likelihood <- function(theta, X, y) {
      beta <- theta[1:4]
      sigma_sq <- theta[5]
      # log-likelihood
      n <- length(y)
      residuals <- y - X %*% beta
      log_likelihood_value <- -n/2 * log(2 * pi) - n/2 * log(sigma_sq) - 1/(2 * sigma_sq)
    return(log_likelihood_value)
  }
```

**1.c) Use optim() and your function from 1b to find $\hat{\beta}_{\text{MLE}}$ and $\hat{\sigma}^2_{\text{MLE}}$ as well as their standard errors. You may find it helpful to initialize your search at 0 for the $\beta$ parameters and var$(y)$ for $\sigma^2$. You will likely find some differences in your standard errors compared to the output from lm() in 1d below. This is because lm() uses analytical expressions for the standard errors, whereas optim() uses numerical approximations to the Hessian matrix.**

```
set.seed(1234)
X <- cbind(1,BostonHousing2$rm,BostonHousing2$age,BostonHousing2$crim)
Y <- matrix(BostonHousing2$medv,ncol=1)
initial_parameters <- c(1,1,1,1,var(Y))
#optimization function is called
result <- optim(par = initial_parameters, fn = log_likelihood,  X=X, y=Y,control = list(fn
```

```
Warning in log(sigma_sq): NaNs produced

Warning in log(sigma_sq): NaNs produced
```

```
print(result)
```

```
$par
[1] -23.60366982   8.03253919  -0.05224289  -0.21102912  36.83582886

$value
[1] -1630.435
```

```
$counts
function gradient
      58       23

$convergence
[1] 0

$message
NULL

$hessian
              [,1]          [,2]          [,3]          [,4]          [,5]
[1,] -1.373663e+01 -8.632967e+01 -9.419878e+02 -4.963762e+01 -5.229595e-06
[2,] -8.632967e+01 -5.493184e+02 -5.854902e+03 -2.937887e+02 -8.458301e-05
[3,] -9.419878e+02 -5.854902e+03 -7.545954e+04 -4.574757e+03 -3.916512e-05
[4,] -4.963762e+01 -2.937887e+02 -4.574757e+03 -1.193684e+03 -3.902301e-05
[5,] -5.229595e-06 -8.458301e-05 -3.916512e-05 -3.902301e-05 -1.864794e-01
```

```r
#MLE estimates
beta_mle <- result$par[1:4]
sigma_square_mle <- result$par[5]
std_error <- sqrt(-diag(solve(result$hessian)))
print("Beta MLE")
```

```
[1] "Beta MLE"
```

```r
print(beta_mle)
```

```
[1] -23.60366982   8.03253919  -0.05224289  -0.21102912
```

```r
print("Sigma Square MLE")
```

```
[1] "Sigma Square MLE"
```

```r
print(sigma_square_mle)
```

```
[1] 36.83583
```

```
print("Standard Error")
```

```
[1] "Standard Error"
```

```
print(std_error)
```

```
[1] 2.75832954 0.40040222 0.01042100 0.03392967 2.31571187
```

**1.d) Use `lm()` and `summary()` to check your work in 1c above.**

```
summary(lm(BostonHousing2$medv ~ BostonHousing2$rm+BostonHousing2$age+BostonHousing2$crim)
```

```
Call:
lm(formula = BostonHousing2$medv ~ BostonHousing2$rm + BostonHousing2$age +
    BostonHousing2$crim)

Residuals:
    Min      1Q  Median      3Q     Max
-19.959  -3.143  -0.633   2.150  39.940

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         -23.60556    2.76938  -8.524  < 2e-16 ***
BostonHousing2$rm     8.03284    0.40201  19.982  < 2e-16 ***
BostonHousing2$age   -0.05224    0.01046  -4.993 8.21e-07 ***
BostonHousing2$crim  -0.21102    0.03407  -6.195 1.22e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.094 on 502 degrees of freedom
Multiple R-squared:  0.5636,    Adjusted R-squared:  0.561
F-statistic: 216.1 on 3 and 502 DF,  p-value: < 2.2e-16
```

**1.e) Do the signs on the 3 estimated slope coefficients make sense? Why or why not?**

The estimated slope coefficients align with expectations for the median home value determination. As generally observed, the number of rooms per home shows a positive relationship,

while both age and crime rate exhibit inverse correlations with home values. Older houses typically depreciate in value over time, reflecting a negative correlation, while higher crime rates often decrease the desirability of an area, leading to lower demand and subsequently lower home values. Therefore, the estimated coefficients in the model are anticipated to be negative, reflecting the adverse effects of house age and crime rate on the estimated median home value.

**1.f) In 1c, `optim()` returned the hessian matrix, which enabled you to calculate the standard errors of the MLEs. In 1d, `lm()` provided the standard errors from analytical expressions we derived in class. Suppose, however, that you were unable to calculate the standard errors and needed to use a bootstrap to estimate them. Write `R` code to perform a bootstrap to estimate the standard errors of the 3 $\hat{\beta}_{\text{MLE}}$ parameters. Each time through the loop, you may use `lm()` and `coef()` if you would like, or you may calculate $\hat{\beta}$ directly from matrices; you may not use the `boot()` function from the `boot` package.**

```
set.seed(1234)
B <- 1000
n <- nrow(X)
res <- matrix(NA_real_, nrow=B, ncol=ncol(X))
X_test <- X[,c(2,3,4)]

for(b in 1:B) {
  draws <- sample(1:n, size=n, replace=T)
  Y_boot <- Y[draws,]
  X_boot <- X_test[draws,]
  res[b,] <- lm(Y_boot ~ X_boot)$coef
}

serr <- apply(res, 2, function(x) sqrt(var(x)))

print("The standard errors estimated from the bootstrap are:")
```

```
[1] "The standard errors estimated from the bootstrap are:"
```

```
serr[cbind(2,3,4)]
```

```
[1] 0.67701162 0.00927441 0.03502999
```

**1.g) Use your estimates and standard errors from 1c above to test whether the 3 $\beta$ slope coefficients are each (separately) statistically significantly different from zero at a 95% confidence level. Check your results against the output in 1d above.**

```
alpha <- 0.05
critical_value <- qt(1-alpha/2,n-3)
beta_stat <- beta_mle/std_error[1:4]

print("Beta Statistics")
```

```
[1] "Beta Statistics"
```

```
beta_stat
```

```
[1] -8.557233 20.061175 -5.013230 -6.219605
```

```
critical_value <- qt(1 - alpha / 2, n-3)
print("Critical Value")
```

```
[1] "Critical Value"
```

```
print(critical_value)
```

```
[1] 1.964691
```

```
print("All the 3 slope coefficients are statistically significant (at 95% confidence level
```

```
[1] "All the 3 slope coefficients are statistically significant (at 95% confidence level) be
```