# ECON7333: Assignment 3

Thomas Doyle

**Setup**

```
options(scipen=999)
attach(ISLR::Wage)
```

## Exercise 1

### Exercise 1.1

```
loocv_tmse <- function(d){
  #' @d a data.frame returned by `model.frame()`
  #
  n <- dim(d)[1]
  p <- dim(d)[2]

  MSE <- rep(0,n)

  for(i in 1:n) {
    #
    lm_i <- lm(d[-i,],y=TRUE) # leave i out
    MSE_i <- (lm_i$y[i]-predict(lm_i,d[i,]))^2
    MSE[i] <- MSE_i
  }

  return(
    #
    CV_n <- mean(MSE,na.rm = TRUE)
  )
}
```

### Exercise 1.2

1. $logwage = \beta_0 + \beta_1 age$
2. $logwage = \beta_0 + \beta_1\ age + \beta_2\ age^2$
3. $logwage = \beta_0 + \beta_1\ age + \beta_2\ education$

```
Wage.models <- list(
  model.frame("logwage~age",ISLR::Wage),
  model.frame("logwage~age+I(age^2)",ISLR::Wage),
  model.frame("logwage~age+education",ISLR::Wage)
```

```
)
sapply(Wage.models, loocv_tmse)
```

```
## [1] 0.1306018 0.1381353 0.1531854
```

### Exercise 1.3

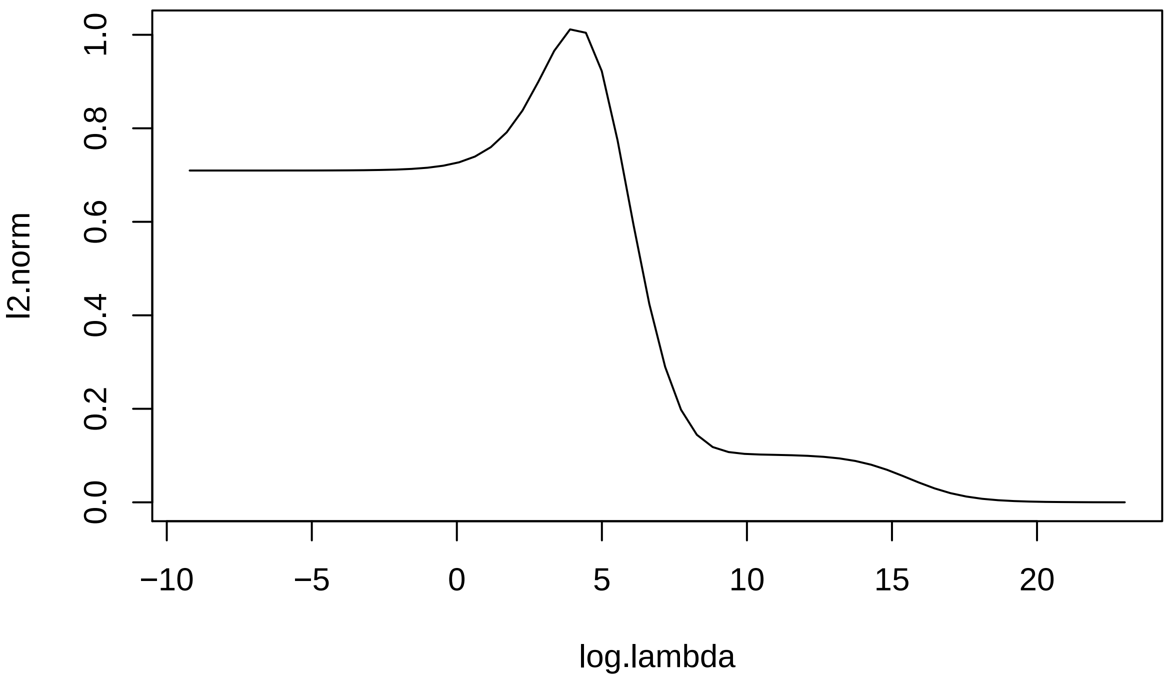
```
lm.ridge <- function(lambda=0) {

  m <- model.frame(logwage~age+education,ISLR::Wage)
  Terms <- attr(eval.parent(m),"terms")
  Y <- model.response(m)
  X <- model.matrix(object = Terms,data = m,contrasts.arg = list(education="contr.treatment"))
  n <- nrow(X)
  p <- ncol(X)

  # https://arxiv.org/pdf/1509.09169.pdf
  # Hoerl, A. E. and Kennard, R. W. (1970).
  ridge.betas <- solve( t(X) %*% X + lambda*diag(p) ) %*% t(X) %*% Y

  ## predict Y
  # resid <- (Y-X%*%ridge.betas)
  # penalty <- lambda*sum(ridge.betas[-1]^2)
  # rss <- sum(resid^2)

  l2.norm <- sqrt(sum(ridge.betas[-1]^2)) # drop intercept

  return(
    list(
      log.lambda=log(lambda),
      l2.norm=l2.norm,
      coefficients=ridge.betas,
      lambda=lambda
    )
  )
}
```

```
# MLE
lm.ridge.loocv <- function(limits) {

  m <- model.frame(logwage~age+education,ISLR::Wage)
  Terms <- attr(eval.parent(m),"terms")
  Y <- model.response(m)
  X <- model.matrix(object = Terms,data = m,contrasts.arg = list(education="contr.treatment"))

  loocv <- function(lambda, X, Y, Delta){
    n <- nrow(X)
    p <- ncol(X)
    loss <- 0

    for (i in 1:n) {
      loo_beta <- solve(t(X[-i,])%*%X[-i,]+lambda*diag(p))%*%t(X[-i,])%*%Y[-i]
      loss <- loss+(Y[i]-X[i,1]*loo_beta[1]-X[i,-1]%*%loo_beta[-1])^2
```

```
    }

    return(loss)
  }

  # optimize penalty parameter
  # minimize RSS
  opt <- optimize(loocv, limits, X=X, Y=Y)
  l2.norm <- lm.ridge(opt$minimum)$l2.norm

  return(
    data.frame(opt,l2.norm)
  )
}
```

Plot $\ln(\lambda)$ against $\ell_2$ norm.

```
lambdas <- 100^seq(-2, 5, length = 60)
plot(t(sapply(lambdas,lm.ridge))[,1:2],type="l")
```




```
# mle
lm.ridge.loocv(c(10^-10, 10^10))
```

```
##    minimum objective   l2.norm
## 1 0.020457  275.9958 0.7100099
```

## Exercise 2

### Exercise 2.1

```r
dgf <- function(n,p) {
  #' @n
  #' @p

  e <- rnorm(n,0,1)
  X <- matrix(runif(n*p),n,p)
  y <- 2*X[,1]+4*X[,2]+e

  output <- list(y=y,X=X)
  return(list(y,X))
}
d.5 <- dgf(100,5)
y <- d.5[[1]]
X <- d.5[[2]]
```

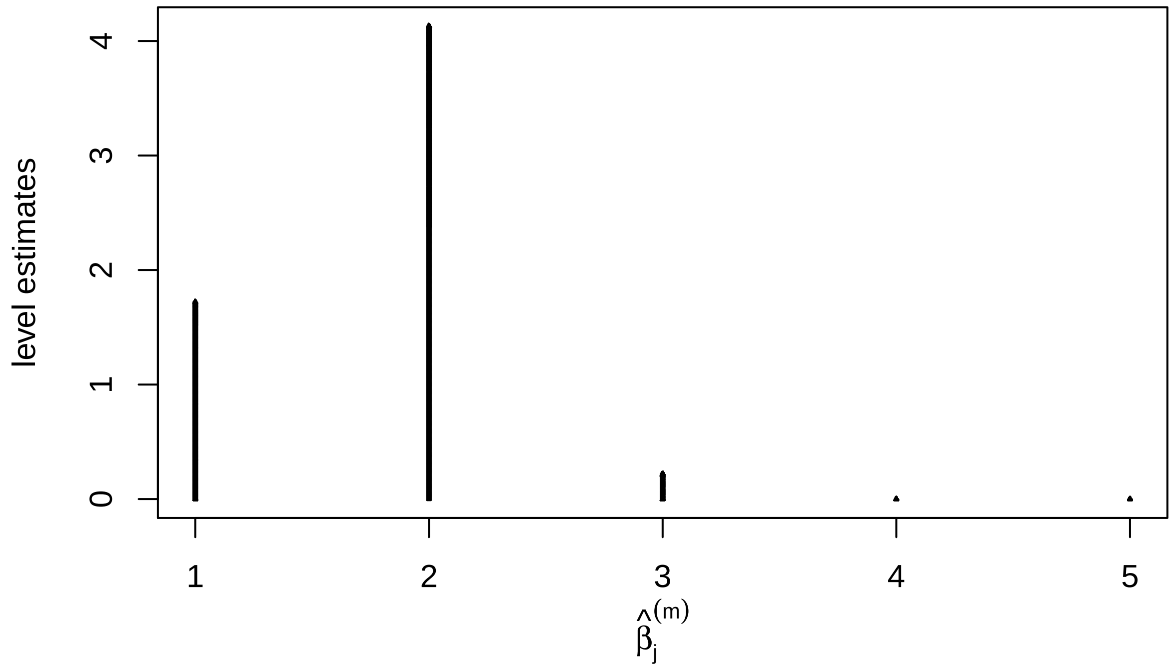
### Exercise 2.2

```r
subset_selection <- function(y,X,M) {
  #' @y
  #' @X
  #' @M

  p <- dim(X)[2]
  n <- dim(X)[1]

  # Initialise
  #
  b <- matrix(rep(NA,p*M),p,M)
  b[,1] <- 0
  r <- as.vector(y)
  # e <- as.vector(rep(0.01,n))
  e <- 0.01

  for(m in 2:M) {
    z <- b[,m-1]

    ## select x_j with highest correlation with r
    j <- which.max(cor(x=X,y=r))
    xj <- X[,j]

    a <- suppressWarnings(e*sign(t(xj)%*%r))
    z[j] <- z[j]+a
    b[,m] <- z
    r <- as.vector(r-a%*%xj)
  }

  return(b)
}
```

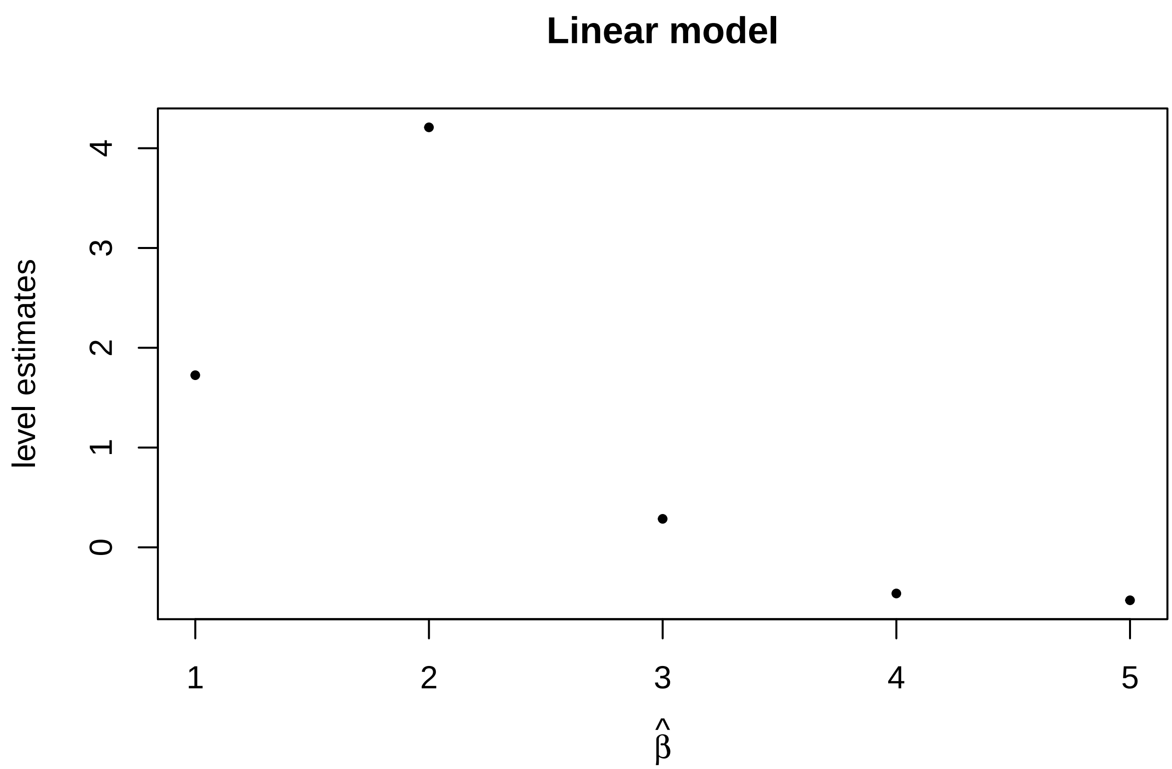
### Exercise 2.3

Plot $\hat{\beta}^{(m)}$ using `matplot`.

```
ss.b <- subset_selection(y,X,1000)
lm.b <- lm(y~X)

matplot(ss.b,pch=2, cex=.2, col="black",
        ylab="level estimates",
        xlab=expression(hat(beta)[j]^(m)),
        main="")
```




```
plot(lm.b$coefficients[-1], pch=20, cex=.8,
     ylab="level estimates",
     xlab=expression(hat(beta)),
     main="Linear model")
```




### Exercise 2.4

```
s1 <- model.frame("y~X-1",data=data.frame(y,X),subset=1:50)
s2 <- model.frame("y~X-1",data=data.frame(y,X),subset=51:100)

mean(apply(ss.b,2, function(x) {
  sum((s1$y - x%*%t(s1$X))^2)
}))
```

```
## [1] 153.2914
```

```
mean(apply(ss.b,2, function(x) {
  sum((s2$y - x%*%t(s2$X))^2)
}))
```

```
## [1] 173.4859
```

Curry `dgf` and `subset_selection` so that `p` is only variable.