```
###Operações matriciais
##O modelo lm gera as estimativas pontuais a partir da operação matricial que
resulta da solução do problema de mínimos quadrados ordinários
#Tal operação matricial, conforme nota de aula 4, é a seguinte:
#b_mqo^ = [(X'X)^-1](X'Y)
#Sendo b_mqo^ o vetor N x 1 de estimadores para todos os parâmetros da FRP
(intercepto e coeficientes);
#X = matriz N x (K+1) de variáveis explicativas com uma primeira coluna contendo
apenas números unitários, para com isso incorporar o intercepto à estimação;
#Y = vetor N x 1 de valores da variável dependente.
#É possível obter diretamente o vetor de estimativas pontuais realizando a
operação matricial anterior no R.

#Definindo a matriz X
#Inicialmente-se seleciona-se estritamente as variáveis explicativas do modelo
#Para isso, basta utilizar o comando subset com opção select
mat_x<-
subset(db_min_15,select=c(anos_educ,exper_idade_trab,exper_idade_trab_sq,d_etnia_pre_par,d_hom,d_
#Agora é preciso adicionar o vetor com valores unitários
#Gerando um vetor 1 x N de 1's
vec_1<-matrix(1,nrow=nrow(db_min_15),ncol=1)
#Acoplando horizontalmente mat_x e vec_1
#importante: vec_1 tem de ser a primeira coluna da esquerda para a direita,
obrigatoriamente.
mat_x<-cbind(vec_1,mat_x)
#Visualizando
View(mat_x)
#De fato o vetor de 1's está na primeira coluna
#Verificando dimensão da matriz X
dim(mat_x)
#Notar que há exatamente 26 colunas, exatamente o número de parâmetros do modelo
até então estimado (contendo o intercepto)

#É necessário transfomar mat_x em um objeto matricial para poder fazer as
operações matriciais.
mat_x<-as.matrix(mat_x)

#Definindo o vetor Y
vec_y<-as.matrix(db_min_15$sal_hor,nrow=nrow(db_min_15),ncol=1)
#Verificando dimensão do vetor Y
dim(vec_y)
#Notar que o número de linhas é o mesmo da matriz X

#Realizando operações matriciais
#O objetivo é calcular b_mqo^ = [(X'X)^-1](X'Y)
#Vamos subdividir em quatro operações

#Operação 1: X'X
#Notar que a matriz à esquerda é a transposição da matriz X.
#A função t(.) transpõe no R.
#O operador %*% aplica o produto matricial.
#Não se trata do produto escalar em que são multiplicados os elementos em posições
equivalentes das duas matrizes.
op_1<-t(mat_x)%*% mat_x

#Operação 2: (X'X)^-1
#A função "solve" inverte matrizes no R. Essa é sem dúvida a operação que mais
exige do computador.
op_2<-solve(op_1)

#Operação 3: (X'Y)
#Trata-se do produto matricial abaixo. Atenção: nunca alterar a ordem dos fatores
de uma multiplicação matricial.
op_3<-t(mat_x) %*% vec_y
#Importante notar a transposição da matriz X, sem a qual a multiplicação seria
impossível.
```

```r
#Basta observar as dimensões de mat_x e vec_y abaixo
dim(mat_x)
dim(vec_y)
#Agora, com mat_x transposta:
dim(t(mat_x))
dim(vec_y)
#O número de colunas de mat_x transposta é equivalente ao número de linhas de
vec_y, como é necessário.

#Operação 4: operação 2 x operação 3
#Recordando, b_mqo^ = (X'X)^-1(X'Y) = (op_2)(op_3)
#Novamente trata-se de produto matricial
op_4<-op_2 %*% op_3

#Visualizando os resultados
op_4
#O que são exatamente os números associados a cada variável?
#O que significa o número associado a vec_1?

#Comparando com os coeficientes estimados por MQO
cbind(op_4,coef(mqo_2))

#Fazendo as quatro operações de uma só vez
solve(t(mat_x)%*%mat_x)%*%(t(mat_x)%*%vec_y)

#Notar o poder da álgebra matricial: com apenas uma operação matricial é possível
obter de uma só vez 25 estimativas pontuais.
#E, em um computador pessoal com configuração média o resultado é obtido em menos
de um segundo.
#Daí a importância de saber álgebra matricial.

#Obtendo a matrix de variância-covariância e as estimativas pontuais para os erros-
padrão dos estimadores
#Sob as hipóteses do modelo clássico de regressão linear, a matriz de variância-
covariância tem forma simples.
#Trata-se de: vcov = [(sigma^2)^](X'X)^-1
#Sendo [(sigma^2)^] a estimativa pontual para a variância do termo de perturbação.
#Esta corresponde à soma dos quadrados dos resíduos dividida pelo respectivo
número de graus de liberdade (N-K-1).
#Vamos calcular a partir de três operações

#Calculando [(sigma^2)^] = SQR/(N-K-1)
op_1<-sum(mqo_2$residuals^2)/(nrow(db_min_15)-length(coef(mqo_2)))

#Calculando (X'X)^-1
op_2<-solve(t(mat_x)%*%mat_x)

#Multiplicando op_1 e op_2
#Notar que, nesse caso, como [(sigma^2)^] é um escalar, deve-se utilizar o produto
escalar "*"
op_3<-op_1*op_2
#Dimensão da matriz gerada
dim(op_3)
#Está correta a dimensão?

#Para comparar com a matriz vcov gerada pelo comando lm, vamos fazer dois passos.
#Passo 1: subtração elemento a elemento das duas matrizes
#A matriz vcov gerada pelo comando lm(.) fica armazenada em vcov(mqo_2), conforme
laboratório 2.
#Para subtrair elemento a elemento, basta usar o sinal de subtração
vcov(mqo_2) - op_3
#Visualizando apropriadamente
View(vcov(mqo_2) - op_3)
#notar que as diferenças são desprezíveis, com ordem de magnitude de 10^-11 ou
menor
#De fato, a diferença mínima é
```

```r
min(vcov(mqo_2) - op_3)
#Com certeza, a operação matricial realizada é equivalente à utilizada pelo
comando lm(.)
#A soma das diferenças elemento a elemento atesta isso
sum(vcov(mqo_2) - op_3)
#Ou seja, as diferenças somam um número desprezível.

#Passo 2: comparar a diagonal da matriz op_3 com os erros-padrão dos estimadores
segundo comando lm(.)
#Faz sentido fazer iso?
#Extraindo a diagonal da matriz gerada
op_4<-diag(op_3)
cbind(manual=op_4,lm=diag(vcov(mqo_2)))

#Fica nítido pois a ausência de diferença entre as variância dos estimadores
obtidas com as operações manuais e via lm(.)
```