Lista 6

MI406-Regressão

Caio Gomes Alves

1 Questão 1

1.1 Pergunta

Considere o problema de Regressão Linear com erros auto-regressivos da seguinte forma:

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon$$

onde $\epsilon \sim N(0, \sigma^2 V)$, e

$$V = \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{n-1} \\ \rho & 1 & \rho & \cdots & \rho^{n-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{n-1} & \rho^{n-2} & \rho^{n-3} & \cdots & 1 \end{bmatrix}$$

- **(a)** Simule um conjunto de dados com 2 covariáveis + intercepto desse modelo. Escolha os coeficientes da forma que achar apropriado, porém utilize $\sigma^2 = 1$ e $\rho = 0.75$.
- **(b)** Encontre a inversa da matriz V. O que você pode observar nesse caso? Teste para outros valores de ρ.
- **(c)** Com o conjunto de dados simulado e considerando ρ conhecido, encontre os estimadores de Mínimos Quadrados Generalizados.
- **(d)** Com o mesmo conjunto de dados, encontre os estimadores de Mínimos Quadrados Ordinários e compare com o item anterior.
- **(e)** Considerando os coeficientes β e σ^2 conhecidos, estime ρ por máxima verossimilhança. Dica: $\epsilon = \mathbf{Y} - \mathbf{X}\beta$ pode ser obtido nesse caso, e a verossimilhança é a densidade da normal multivariada apropriada do vetor ϵ.

1.2 Resposta

a)

Utilizaremos o seguinte código em R para gerar 10 observações provenientes do modelo considerando os valores de $\beta_0 = 1, \beta_1 = -1, \beta_2 = 2$:

```r
# Parâmetros:

# Número de observações:
n <- 10
```

```r
# Correlação entre as observações:
rho <- 0.75

# Variância:
sigma <- 1

# Valores verdadeiros dos betas:
beta_true <- c(1, -1, 2)

# Simulação:

# Seed para reprodutibilidade:
set.seed(1)

# Matriz do modelo:
(X <- cbind(1, matrix(rnorm(n * 2), ncol = 2)))
```

```
##       [,1]       [,2]        [,3]
##  [1,]    1 -0.6264538  1.51178117
##  [2,]    1  0.1836433  0.38984324
##  [3,]    1 -0.8356286 -0.62124058
##  [4,]    1  1.5952808 -2.21469989
##  [5,]    1  0.3295078  1.12493092
##  [6,]    1 -0.8204684 -0.04493361
##  [7,]    1  0.4874291 -0.01619026
##  [8,]    1  0.7383247  0.94383621
##  [9,]    1  0.5757814  0.82122120
## [10,]    1 -0.3053884  0.59390132
```

```r
# Função para gerar a matriz V:
V_func <- function(n, rho) {
  V <- matrix(0, nrow = n, ncol = n)
  for (i in 1:n) {
    for (j in 1:n) {
      V[i, j] <- rho^(abs(i - j))
    }
  }
  return(V)
}

V <- V_func(n, rho)

# Vejamos os valores da matriz V, arredondados em
# 4 casas decimais:
round(V, 4)
```

```
##         [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]  [,10]
##  [1,] 1.0000 0.7500 0.5625 0.4219 0.3164 0.2373 0.1780 0.1335 0.1001 0.0751
##  [2,] 0.7500 1.0000 0.7500 0.5625 0.4219 0.3164 0.2373 0.1780 0.1335 0.1001
##  [3,] 0.5625 0.7500 1.0000 0.7500 0.5625 0.4219 0.3164 0.2373 0.1780 0.1335
##  [4,] 0.4219 0.5625 0.7500 1.0000 0.7500 0.5625 0.4219 0.3164 0.2373 0.1780
##  [5,] 0.3164 0.4219 0.5625 0.7500 1.0000 0.7500 0.5625 0.4219 0.3164 0.2373
```

```
##  [6,] 0.2373 0.3164 0.4219 0.5625 0.7500 1.0000 0.7500 0.5625 0.4219 0.3164
##  [7,] 0.1780 0.2373 0.3164 0.4219 0.5625 0.7500 1.0000 0.7500 0.5625 0.4219
##  [8,] 0.1335 0.1780 0.2373 0.3164 0.4219 0.5625 0.7500 1.0000 0.7500 0.5625
##  [9,] 0.1001 0.1335 0.1780 0.2373 0.3164 0.4219 0.5625 0.7500 1.0000 0.7500
## [10,] 0.0751 0.1001 0.1335 0.1780 0.2373 0.3164 0.4219 0.5625 0.7500 1.0000
```

```r
# Gerar vetor de erros:

# Utilizaremos decomposição de Cholesky de sigma * V (que será simétrica)
# para obter a raiz quadrada da matriz de covariância:
L <- chol(sigma * V)
epsilon <- t(L) %*% rnorm(n)

# Gerar Y:

(Y <- X %*% beta_true + epsilon)
```

```
##             [,1]
##  [1,]  5.5689935
##  [2,]  2.8026107
##  [3,]  1.5473932
##  [4,] -5.6248287
##  [5,]  2.8802192
##  [6,]  1.6633743
##  [7,]  0.3267212
##  [8,]  1.0614346
##  [9,]  0.9344596
## [10,]  1.9204823
```

b)

Podemos encontrar o valor da inversa de V por meio da função `solve` presente no R. Assim, teremos que:

```r
# Inversa de V:
V_inv <- solve(V)

# Muitos valores serão calculados numericamente
# como muito próximos de zero, quando na verdade serão
# zeros realmente. Por isso, usemos o valor arredondado da matriz:
print(round(V_inv, 4))
```

```
##          [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]    [,9]   [,10]
##  [1,]  2.2857 -1.7143  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
##  [2,] -1.7143  3.5714 -1.7143  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
##  [3,]  0.0000 -1.7143  3.5714 -1.7143  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
##  [4,]  0.0000  0.0000 -1.7143  3.5714 -1.7143  0.0000  0.0000  0.0000  0.0000  0.0000
##  [5,]  0.0000  0.0000  0.0000 -1.7143  3.5714 -1.7143  0.0000  0.0000  0.0000  0.0000
##  [6,]  0.0000  0.0000  0.0000  0.0000 -1.7143  3.5714 -1.7143  0.0000  0.0000  0.0000
##  [7,]  0.0000  0.0000  0.0000  0.0000  0.0000 -1.7143  3.5714 -1.7143  0.0000  0.0000
##  [8,]  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000 -1.7143  3.5714 -1.7143  0.0000
##  [9,]  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000 -1.7143  3.5714 -1.7143
## [10,]  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000 -1.7143  2.2857
```

```r
# Testar para outros valores de rho:
rho_teste1 <- 0.2
V_teste1 <- V_func(n, rho_teste1)
V_inv_teste1 <- solve(V_teste1)
print(round(V_inv_teste1, 4))
```

```
##          [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]    [,9]   [,10]
##  [1,]  1.0417 -0.2083  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
##  [2,] -0.2083  1.0833 -0.2083  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
##  [3,]  0.0000 -0.2083  1.0833 -0.2083  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
##  [4,]  0.0000  0.0000 -0.2083  1.0833 -0.2083  0.0000  0.0000  0.0000  0.0000  0.0000
##  [5,]  0.0000  0.0000  0.0000 -0.2083  1.0833 -0.2083  0.0000  0.0000  0.0000  0.0000
##  [6,]  0.0000  0.0000  0.0000  0.0000 -0.2083  1.0833 -0.2083  0.0000  0.0000  0.0000
##  [7,]  0.0000  0.0000  0.0000  0.0000  0.0000 -0.2083  1.0833 -0.2083  0.0000  0.0000
##  [8,]  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000 -0.2083  1.0833 -0.2083  0.0000
##  [9,]  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000 -0.2083  1.0833 -0.2083
## [10,]  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000 -0.2083  1.0417
```

```r
rho_teste2 <- 0.9
V_teste2 <- V_func(n, rho_teste2)
V_inv_teste2 <- solve(V_teste2)
print(round(V_inv_teste2, 4))
```

```
##          [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]    [,9]   [,10]
##  [1,]  5.2632 -4.7368  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
##  [2,] -4.7368  9.5263 -4.7368  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
##  [3,]  0.0000 -4.7368  9.5263 -4.7368  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
##  [4,]  0.0000  0.0000 -4.7368  9.5263 -4.7368  0.0000  0.0000  0.0000  0.0000  0.0000
##  [5,]  0.0000  0.0000  0.0000 -4.7368  9.5263 -4.7368  0.0000  0.0000  0.0000  0.0000
##  [6,]  0.0000  0.0000  0.0000  0.0000 -4.7368  9.5263 -4.7368  0.0000  0.0000  0.0000
##  [7,]  0.0000  0.0000  0.0000  0.0000  0.0000 -4.7368  9.5263 -4.7368  0.0000  0.0000
##  [8,]  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000 -4.7368  9.5263 -4.7368  0.0000
##  [9,]  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000 -4.7368  9.5263 -4.7368
## [10,]  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000 -4.7368  5.2632
```

Observa-se a estrutura de banda (ou matriz tri-diagonal) presente independente do valor escolhido para ρ, que é uma estrutura característica da especificação da matriz V com valores de correlação decrescente pelo valor absoluto da diferença entre os valores das linhas/colunas.

c)

Temos que os estimadores de mínimos quadrados generalizados serão dados por:

$$(X^\top V^{-1} X)^{-1} X^\top V^{-1} Y$$

Que, computacionalmente, serão calculados como:

```r
# Estimadores de Mínimos Quadrados Generalizados (GLS):
(beta_hat_GLS <- solve(t(X) %*% V_inv %*% X) %*% t(X) %*% V_inv %*% Y)
```

```
##            [,1]
## [1,]  0.9610649
## [2,] -1.3471742
## [3,]  2.0654359
```

Que são valores muito próximos dos valores reais de $\beta = (1, -1, 2)$.

d)

Temos que os estimadores de mínimos quadrados ordinários serão dados por:

$$(X^\top X)^{-1} X^\top Y$$

Que, computacionalmente, serão calculados como:

```
# Estimadores de Mínimos Quadrados Ordinários (OLS):
(beta_hat_OLS <- solve(t(X) %*% X) %*% t(X) %*% Y)
```

```
##           [,1]
## [1,]  1.070812
## [2,] -1.687576
## [3,]  1.850054
```

Que dão valores também próximos dos valores originais dos β's.

e)

Como β e σ^2 são dados como conhecidos, a função de log-verossimilhança para ρ será dada por:

$$\log(L(\rho)) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2}\log(|V|) - \frac{1}{2}\epsilon^\top(\sigma^2 V)^{-1}\epsilon$$

```
# Erros observados (já que beta e sigma^2 são conhecidos)
epsilon_obs <- Y - X %*% beta_true

# Função de log-verossimilhança para rho:
log_likelihood <- function(rho, epsilon_obs, sigma, n) {
    # Confere se rho está no espaço paramétrico:
    if (rho <= -1 || rho >= 1) {
        return(-Inf)
    }
    # Calcula a matriz V usando a função anterior:
    V_rho <- V_func(n, rho)
    # Lidar com possíveis erros de singularidade ao calcular a inversa e o determinante:
    tryCatch({
        V_rho_inv <- solve(V_rho)
        log_det_V_rho <- as.numeric(determinant(V_rho, logarithm = TRUE)$modulus)
        val <- -n/2 * log(2 * pi * sigma) - 1/2 * log_det_V_rho -
            1/2 * t(epsilon_obs) %*% (1/sigma * V_rho_inv) %*% epsilon_obs
        return(val)
    }, error = function(e) {
        # -Inf retornado para valores problemáticos de rho (-1, 0, 1):
        return(-Inf)
    })
}

# Otimizar a função de log-verossimilhança:
rho_hat_ML_optim <- optimize(f = log_likelihood, interval = c(-0.99, 0.99),
                             epsilon_obs = epsilon_obs, sigma = sigma, n = n,
                             maximum = TRUE)
```

```r
(rho_hat_ML <- rho_hat_ML_optim$maximum)
```

```
## [1] 0.7609039
```

Que é um valor próximo ao valor real de $\rho = 0.75$.

2 Questão 2

2.1 Pergunta

Considere o modelo de Regressão Linear Simples $Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$, onde $\epsilon_i \perp \epsilon_j$ para $i \neq j$, mas $\text{Var}(\epsilon_i) = \frac{\sigma^2}{x_i^2}$.

- **(a)** Descreva a matriz de covariâncias do vetor aleatório formado pelos erros ϵ na forma $\sigma^2 V$.
- **(b)** Calcule V^{-1}.
- **(c)** Encontre as expressões para os estimadores de Mínimos Quadrados Generalizados de β_0 e β_1.
- **(d)** Generalize o resultado anterior (expressão dos estimadores) para o caso onde $\text{Var}(\epsilon_i) = \sigma^2 g(x_i)$, onde g é uma função estritamente positiva.

2.2 Resposta

a)

Como $\epsilon_i \perp \epsilon_j$, temos que os valores que não estão na diagonal principal de $V serão iguais a 0. Por isso, a matriz de covariâncias de ϵ será:

$$\text{Cov}(\epsilon) = \begin{bmatrix} \frac{\sigma^2}{x_1^2} & 0 & \cdots & 0 \\ 0 & \frac{\sigma^2}{x_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\sigma^2}{x_n^2} \end{bmatrix} = \sigma^2 \begin{bmatrix} \frac{1}{x_1^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{x_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{x_n^2} \end{bmatrix}$$

Desse modo, a matriz V será dada por:

$$V = \begin{bmatrix} \frac{1}{x_1^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{x_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{x_n^2} \end{bmatrix}$$

b)

Por ser uma matriz diagonal, a inversa de V será:

$$V^{-1} = \begin{bmatrix} x_1^2 & 0 & \cdots & 0 \\ 0 & x_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_n^2 \end{bmatrix}$$

c)

Como o modelo possui somente o intercepto e uma covariável x, temos que a matriz do modelo X será:

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

Os estimadores de Mínimos Quadrados Generalizados são dados por $(X^\top V^{-1} X)^{-1} X^\top V^{-1} Y$, que por partes serão:

$$X^\top V^{-1} = \begin{bmatrix} x_1^2 & x_2^2 & \cdots & x_n^2 \\ x_1^3 & x_2^3 & \cdots & x_n^3 \end{bmatrix}$$

$$X^\top V^{-1} X = \begin{bmatrix} \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i^3 \\ \sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_i^4 \end{bmatrix}$$

$$X^\top V^{-1} Y = \begin{bmatrix} \sum_{i=1}^n x_i^2 Y_i \\ \sum_{i=1}^n x_i^3 Y_i \end{bmatrix}$$

A inversa de $(X^\top V^{-1} X)$ terá valor fechado, e dado por:

$$(X^\top V^{-1} X)^{-1} = \frac{1}{(\sum x_i^4)(\sum x_i^2) - (\sum x_i^3)^2} \begin{bmatrix} \sum_{i=1}^n x_i^4 & -\sum_{i=1}^n x_i^3 \\ -\sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_i^2 \end{bmatrix}$$

De modo que o Estimador de Mínimos Quadrados Generalizados será:

$$\hat{\beta} = \frac{1}{(\sum x_i^4)(\sum x_i^2) - (\sum x_i^3)^2} \begin{bmatrix} \sum_{i=1}^n x_i^4 & -\sum_{i=1}^n x_i^3 \\ -\sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_i^2 \end{bmatrix} \begin{bmatrix} \sum_{i=1}^n x_i^2 Y_i \\ \sum_{i=1}^n x_i^3 Y_i \end{bmatrix}$$

d)

Utilizando $\mathrm{Var}(\epsilon_i) = \sigma^2 g(x_i)$, teremos que a matriz V^{-1} será:

$$V^{-1} = \begin{bmatrix} \frac{1}{g(x_1)} & 0 & \cdots & 0 \\ 0 & \frac{1}{g(x_1)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{g(x_n)} \end{bmatrix}$$

De modo que:

$$X^\top V^{-1} = \begin{bmatrix} \frac{1}{g(x_1)} & \frac{1}{g(x_2)} & \cdots & \frac{1}{g(x_n)} \\ \frac{x_1}{g(x_1)} & \frac{x_2}{g(x_2)} & \cdots & \frac{x_n}{g(x_n)} \end{bmatrix}$$

$$X^\top V^{-1} X = \begin{bmatrix} \sum_{i=1}^n \frac{1}{g(x_i)} & \sum_{i=1}^n \frac{x_i}{g(x_i)} \\ \sum_{i=1}^n \frac{x_i}{g(x_i)} & \sum_{i=1}^n \frac{x_i^2}{g(x_i)} \end{bmatrix}$$

$$X^\top V^{-1} Y = \begin{bmatrix} \sum_{i=1}^n \frac{Y_i}{g(x_i)} \\ \sum_{i=1}^n \frac{x_i Y_i}{g(x_i)} \end{bmatrix}$$

Novamente, teremos que $(X^\top V^{-1} X)^{-1}$ terá forma fechada e será dado por:

$$(X^\top V^{-1} X)^{-1} = \frac{1}{\left(\sum \frac{1}{g(x_i)}\right)\left(\sum \frac{x_i^2}{g(x_i)}\right) - \left(\sum \frac{x_i}{g(x_i)}\right)^2} \begin{bmatrix} \sum \frac{x_i^2}{g(x_i)} & -\sum \frac{x_i}{g(x_i)} \\ -\sum \frac{x_i}{g(x_i)} & \sum \frac{1}{g(x_i)} \end{bmatrix}$$

E o estimador generalizado para β será:

$$\frac{1}{\left(\sum \frac{1}{g(x_i)}\right)\left(\sum \frac{x_i^2}{g(x_i)}\right) - \left(\sum \frac{x_i}{g(x_i)}\right)^2} \begin{bmatrix} \sum \frac{x_i^2}{g(x_i)} & -\sum \frac{x_i}{g(x_i)} \\ -\sum \frac{x_i}{g(x_i)} & \sum \frac{1}{g(x_i)} \end{bmatrix} \begin{bmatrix} \sum_{i=1}^n \frac{Y_i}{g(x_i)} \\ \sum_{i=1}^n \frac{x_i Y_i}{g(x_i)} \end{bmatrix}$$

3 Questão 3

3.1 Pergunta

Considere o problema de Regressão Linear:

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon$$

com $\epsilon \sim N(0, \sigma^2 V)$, com uma matriz V conhecida. Encontre o estimador de máxima verossimilhança do vetor de coeficientes β.

3.2 Resposta

Como $Y = X\beta + \epsilon \Rightarrow \epsilon = Y - X\beta$. Substituindo esse valor na verossimilhança da normal multivariada chegamos em:

$$L(\beta, \sigma^2; Y, X) = (2\pi)^{-n/2} |\sigma^2 V|^{-1/2} \exp\left(-\frac{1}{2}(Y - X\beta)^\top (\sigma^2 V)^{-1} (Y - X\beta)\right)$$

E sua log-verossimilhança será:

$$l(L(\beta, \sigma^2; Y, X)) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2}\log(|V|) - \frac{1}{2\sigma^2}(Y - X\beta)^\top V^{-1}(Y - X\beta)$$

Que deve ser maximizada para β. Verifique que os termos que não dependem de β podem ser suprimidos, pois não irão influenciar na maximização, de modo que ficamos apenas com o termo $(Y-X\beta)^\top V^{-1}(Y-X\beta)$, que será maximizado em:

$$\begin{aligned}
\frac{\partial}{\partial\beta}(Y - X\beta)^\top V^{-1}(Y - X\beta) &= 0 \\
-2X^\top V^{-1}(Y - X\hat{\beta}) &= 0 \\
X^\top V^{-1}Y - X^\top V^{-1}X\hat{\beta} &= 0 \\
X^\top V^{-1}X\hat{\beta} &= X^\top V^{-1}Y \\
\hat{\beta} &= (X^\top V^{-1}X)^{-1}X^\top V^{-1}Y
\end{aligned}$$

Que demonstra que o estimador de máxima verossimilhança é exatamente o Estimador de Mínimos Quadrados Generalizados quando consideramos a matriz V conhecida.

4 Questão 4

4.1 Pergunta

Considere o modelo de regressão linear $\mathbf{Y} = \mathbf{X}\beta + \epsilon$, com $\epsilon \sim N(0, I_n)$ e a função objetivo:

$$L(\beta; \mathbf{Y}, \mathbf{X}) = (Y - X\beta)^\top(Y - X\beta) + \lambda\beta^\top\beta$$

- **(a)** Encontre $\hat{\beta}_r^\lambda = \operatorname{argmin}_\beta L(\beta; Y, X)$.
- **(b)** Calcule o viés de $\hat{\beta}_r^\lambda$.
- **(c)** Calcule $\mathrm{Var}(\hat{\beta}_r^\lambda)$.
- **(d)** Escolha uma matriz X com pelo menos 2 colunas. Calcule, numericamente, o viés e a variância de $\hat{\beta}_r^\lambda$ para um determinado valor de λ.
- **(e)** Refaça o item anterior para diferentes valores de λ, ilustrando a relação entre viés e variância para diferentes valores de λ.

4.2 Resposta

a)

Expandindo a função objetivo, temos que:

$$(Y - X\beta)^\top(Y - X\beta) = Y^\top Y - 2Y^\top X\beta + \beta^\top X^\top X\beta$$

De modo que a função objetivo será $L(\beta; Y, X) = Y^\top Y - 2Y^\top X\beta + \beta^\top X^\top X\beta + \lambda\beta^\top\beta$. Para encontrar o valor que a maximiza, derivaremos com relação a β e igualaremos a zero, com o qual teremos:

$$\begin{aligned}
\frac{\partial}{\partial\beta}L(\beta; Y, X) = -2X^\top Y + 2X^\top X\beta + 2\lambda\beta &= 0 \\
X^\top X\beta + \lambda\beta &= X^\top Y \\
(X^\top X + \lambda I_p)\beta &= X^\top Y \\
\hat{\beta} &= (X^\top X + \lambda I_p)^{-1}X^\top Y
\end{aligned}$$

Considerando que I_p é a matriz identidade com p (número de covariáveis em X) colunas. Assim, $\hat{\beta}_r^\lambda = (X^\top X + \lambda I_p)^{-1} X^\top Y$ será o estimador para a regularização *ridge*.

b)

Temos que Viés$(\hat{\beta}_r^\lambda) = E(\hat{\beta}_r^\lambda) - \beta$, que será:

$$
\begin{aligned}
E(\hat{\beta}_r^\lambda) &= E((X^\top X + \lambda I_p)^{-1} X^\top Y) \\
&= E((X^\top X + \lambda I_p)^{-1} X^\top (X\beta + \epsilon)) \\
&= (X^\top X + \lambda I_p)^{-1} X^\top X\beta + (X^\top X + \lambda I_p)^{-1} X^\top E(\epsilon) \\
&= (X^\top X + \lambda I_p)^{-1} X^\top X\beta
\end{aligned}
$$

Já que $E(\epsilon) = 0$. Assim, o viés será:

$$
\begin{aligned}
\text{Viés}(\hat{\beta}_r^\lambda) &= E(\hat{\beta}_r^\lambda) - \beta \\
&= (X^\top X + \lambda I_p)^{-1} X^\top X\beta - \beta \\
&= ((X^\top X + \lambda I_p)^{-1} X^\top X - I_p)\beta
\end{aligned}
$$

Que será um estimador viciado, a menos que $\lambda = 0$.

c)

Teremos que:

$$
\begin{aligned}
\text{Var}(\hat{\beta}_r^\lambda) &= \text{Var}((X^\top X + \lambda I_p)^{-1} X^\top Y) \\
&= (X^\top X + \lambda I_p)^{-1} X^\top \text{Var}(Y) X((X^\top X + \lambda I_p)^{-1})^\top \\
&= \sigma^2 (X^\top X + \lambda I_p)^{-1} X^\top X (X^\top X + \lambda I_p)^{-1}
\end{aligned}
$$

Que coincide com a variância usual de $\hat{\beta}$, quando $\lambda = 0$.

d)

Façamos todo o processo computacionalmente:

```r
# Parâmetros:

# Número de observações:
n <- 100
# Número de covariáveis (com intercepto):
p <- 3
# Valores verdadeiros dos coeficientes:
beta_true <- c(1, -1, 2)
# Variância do erro:
sigma_sq_erro <- 1

# Geração dos dados:

# Seed para reprodutibilidade:
set.seed(2)
```

```r
# Matriz do modelo:
X_num <- cbind(1, matrix(rnorm(n * (p - 1)), ncol = (p - 1)))

# Erros:
epsilon_num <- rnorm(n, mean = 0, sd = sqrt(sigma_sq_erro))

# Valores de Y:
Y_num <- X_num %*% beta_true + epsilon_num

# Escolher um valor de lambda:
lambda_val <- 0.5

# Calcular beta_hat_ridge para o lambda escolhido:
I_p <- diag(p)
(beta_hat_ridge <- solve(t(X_num) %*% X_num + lambda_val * I_p) %*% t(X_num) %*% Y_num)
```

```
##           [,1]
## [1,]  1.137674
## [2,] -1.028609
## [3,]  1.955217
```

```r
# Calcular o viés teórico:
(bias_ridge_teo <- -lambda_val * solve(t(X_num) %*% X_num + lambda_val * I_p) %*% beta_true)
```

```
##              [,1]
## [1,] -0.004588228
## [2,]  0.003154863
## [3,] -0.009994393
```

```r
# Calcular a variância teórica:
(var_ridge_teo <- solve(t(X_num) %*% X_num + lambda_val * I_p) %*%
     t(X_num) %*% X_num %*% solve(t(X_num) %*% X_num + lambda_val * I_p))
```

```
##               [,1]         [,2]          [,3]
## [1,]  0.0099154365 0.0002130918 -0.0002840332
## [2,]  0.0002130918 0.0074707722  0.0004828790
## [3,] -0.0002840332 0.0004828790  0.0103273978
```

```r
# Calculo aproximado via simulação:
num_simul <- 1000

beta_hat_ridge_sim <- matrix(0, nrow = p, ncol = num_simul)

for (i in 1:num_simul) {
  epsilon_sim <- rnorm(n, mean = 0, sd = sqrt(sigma_sq_erro))
  Y_sim <- X_num %*% beta_true + epsilon_sim
  beta_hat_ridge_sim[, i] <- solve(t(X_num) %*% X_num + lambda_val * I_p) %*% t(X_num) %*% Y_sim
}

# Calcular o viés numérico:
mean_beta_hat_ridge_sim <- rowMeans(beta_hat_ridge_sim)
```

```r
bias_ridge_num <- mean_beta_hat_ridge_sim - beta_true

# Comparação entre viés teórico e simulado:
cbind(Teorico = bias_ridge_teo, Simulado = bias_ridge_num)
```

```
##                       Simulado
## [1,] -0.004588228 -0.001756481
## [2,]  0.003154863  0.006415098
## [3,] -0.009994393 -0.007805414
```

```r
# Calcular a variância numérica:
(var_ridge_num <- cov(t(beta_hat_ridge_sim)))
```

```
##              [,1]          [,2]          [,3]
## [1,] 0.0110202346  0.0000879251  0.0003225708
## [2,] 0.0000879251  0.0075592204 -0.0001953029
## [3,] 0.0003225708 -0.0001953029  0.0107731694
```

```r
# Comparação entre variâncias teórica e simulada:
cbind(Teorica = diag(var_ridge_teo), Simulada = diag(var_ridge_num))
```

```
##          Teorica   Simulada
## [1,] 0.009915437 0.01102023
## [2,] 0.007470772 0.00755922
## [3,] 0.010327398 0.01077317
```

e)

Testemos para diferentes valores de λ, sendo eles $\{0, 0.1, 0.5, 1, 5, 10\}$:

```r
# Diferentes valores de lambda:
lambdas <- c(0, 0.1, 0.5, 1, 5, 10)

bias_simulados <- matrix(NA, nrow = length(lambdas), ncol = p,
                         dimnames = list(paste("lambda =", lambdas),
                                         paste("Vies_beta", 0:(p-1))))
var_simulados <- matrix(NA, nrow = length(lambdas), ncol = p,
                        dimnames = list(paste("lambda =", lambdas),
                                        paste("Var_beta", 0:(p-1))))

# Número de simulações para calcular:
num_simulacoes <- 1000

# Loop para calculo de viés e variância simulados para cada valor de lambda:
for (k in 1:length(lambdas)) {
  lambda_atual <- lambdas[k]
  beta_hat_ridge_sim_k <- matrix(0, nrow = p, ncol = num_simulacoes)
  for (i in 1:num_simulacoes) {
      epsilon_sim_k <- rnorm(n, mean = 0, sd = sqrt(sigma_sq_erro))
      Y_sim_k <- X_num %*% beta_true + epsilon_sim_k
      beta_hat_ridge_sim_k[, i] <- solve(t(X_num) %*% X_num +
                                         lambda_atual * I_p) %*%
```

```r
        t(X_num) %*% Y_sim_k
  }
  # Calcular viés numérico:
  mean_beta_hat_ridge_sim_k <- rowMeans(beta_hat_ridge_sim_k)
  bias_simulados[k, ] <- mean_beta_hat_ridge_sim_k - beta_true
  # Calcular variância numérica:
  var_simulados[k, ] <- diag(cov(t(beta_hat_ridge_sim_k)))
}

# Resultados:
print(bias_simulados)
```

```
##                  Vies_beta 0  Vies_beta 1  Vies_beta 2
## lambda = 0    0.0016764741 0.0013222347 -0.000430384
## lambda = 0.1 -0.0066027357 0.0008409709 -0.004162527
## lambda = 0.5 -0.0006269312 0.0049912395 -0.006581314
## lambda = 1   -0.0097723074 0.0052310355 -0.016970511
## lambda = 5   -0.0429961380 0.0331486900 -0.090756839
## lambda = 10  -0.0829301401 0.0620845333 -0.181001875
```

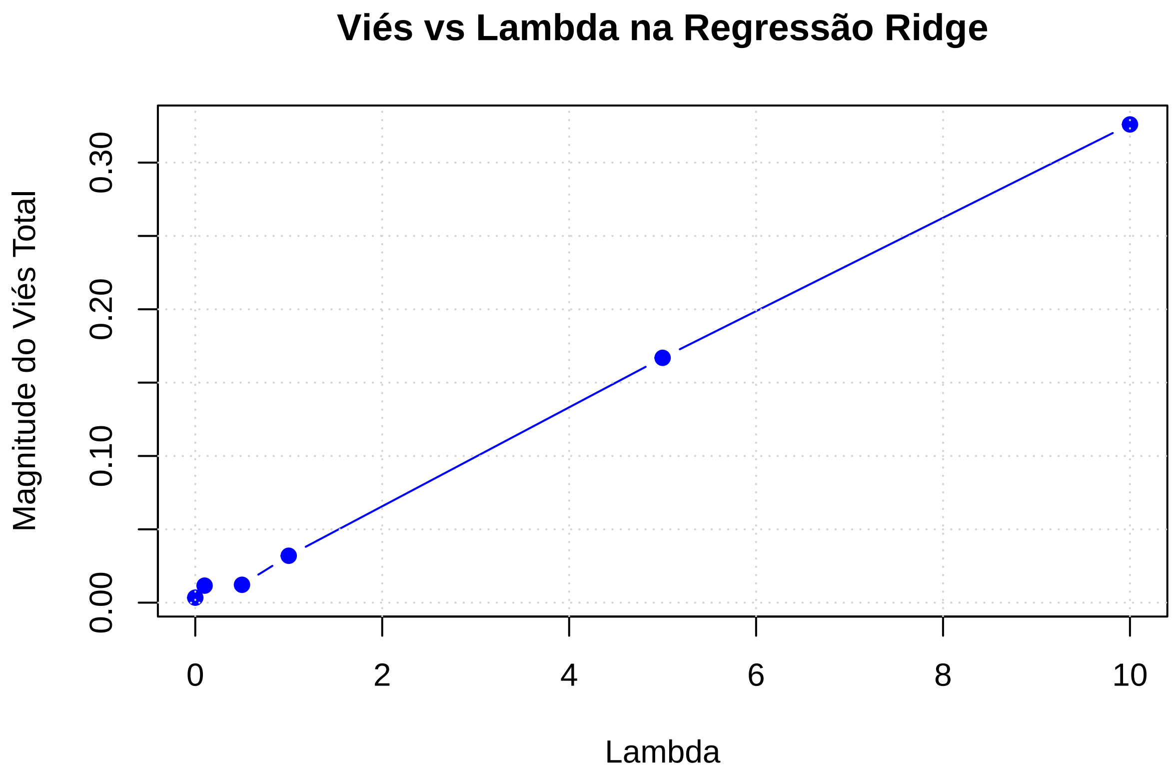
```r
print(var_simulados)
```

```
##                Var_beta 0  Var_beta 1  Var_beta 2
## lambda = 0   0.009688176 0.007712212 0.010625387
## lambda = 0.1 0.010711004 0.007782597 0.010431431
## lambda = 0.5 0.010722131 0.007197142 0.010930421
## lambda = 1   0.009396149 0.007106766 0.010882028
## lambda = 5   0.008872161 0.007301274 0.009789282
## lambda = 10  0.008507000 0.006446608 0.008230587
```

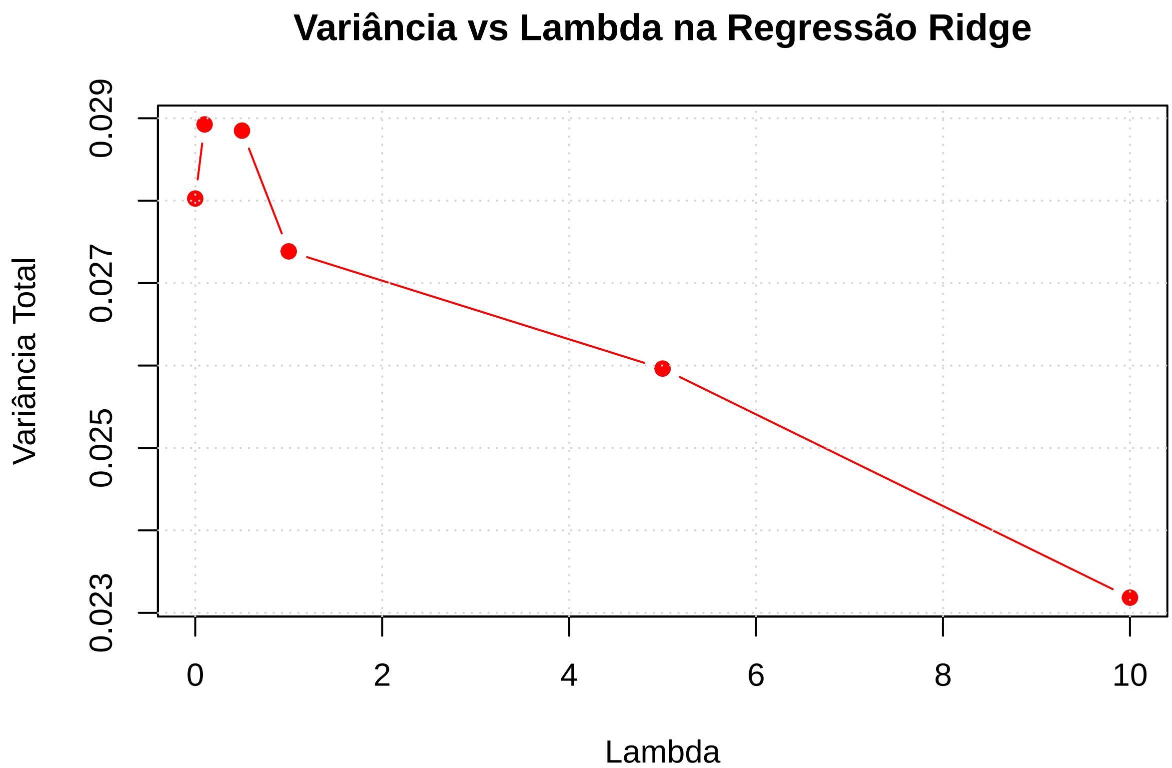
Podemos assim perceber que, conforme aumentamos o valor de λ o viés aumenta, ao passo que a variância diminui. Para verificar, vejamos graficamente as relações:

```r
# Visualização da relação entre viés e variância:

# Plot do viés (em valor absoluto) vs lambda:
plot(lambdas, rowSums(abs(bias_simulados)), type = "b", col = "blue",
     xlab = "Lambda", ylab = "Magnitude do Viés Total",
     main = "Viés vs Lambda na Regressão Ridge")
points(lambdas, rowSums(abs(bias_simulados)), col = "blue", pch = 16)
grid()
```

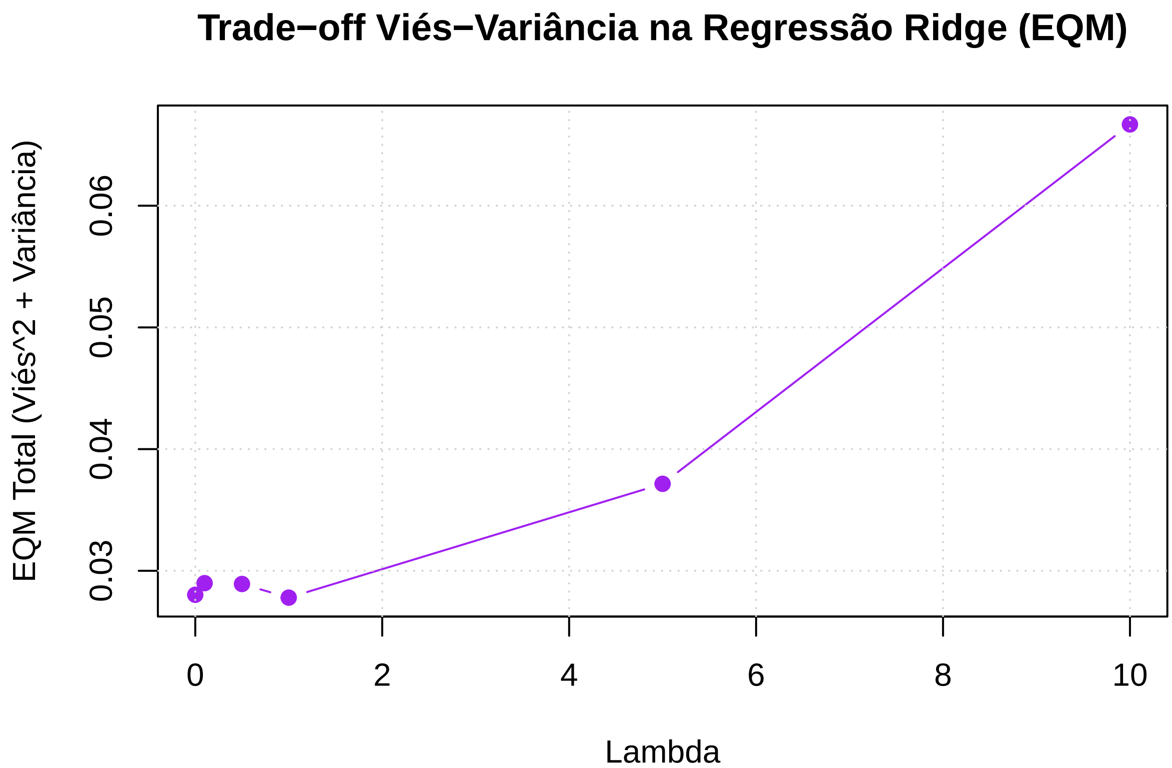



```r
# Plot da variância total vs lambda:
plot(lambdas, rowSums(var_simulados), type = "b", col = "red",
     xlab = "Lambda", ylab = "Variância Total",
     main = "Variância vs Lambda na Regressão Ridge")
points(lambdas, rowSums(var_simulados), col = "red", pch = 16)
grid()
```




```r
# Para verificar o trade-off entre viés e variância,  podemos nos
# focar no Erro Quadrático Médio:
eqm <- rowSums(bias_simulados^2) + rowSums(var_simulados)

plot(lambdas, eqm, type = "b", col = "purple",
     xlab = "Lambda", ylab = "EQM Total (Viés^2 + Variância)",
     main = "Trade-off Viés-Variância na Regressão Ridge (EQM)")
points(lambdas, eqm, col = "purple", pch = 16)
grid()
```


Trade−off Viés−Variância na Regressão Ridge (EQM)
EQM Total (Viés^2 + Variância)
0.03
0.04
0.05
0.06
0
2
4
6
8
10
Lambda