

---

VALORES CRÍTICOS AJUSTADOS PARA INFERÊNCIA SOB  
HETEROSCEDASTICIDADE DE FORMA DESCONHECIDA

ANTONIO CARLOS RICARDO BRAGA JUNIOR

Orientador: Prof. Dr. Francisco Cribari Neto

Área de concentração: Estatística Matemática

Dissertação submetida como requerimento parcial para obtenção do  
grau de Mestre em Estatística pela Universidade Federal de Pernambuco

Recife, fevereiro de 2005

---

```

/*****
*
* PROGRAMA: vc_empirico.ox
*
*
* AUTORES: Antonio Carlos R. Braga Junior;
*          Francisco Cribari-Neto.
*
*
*****/

// Inclusao de bibliotecas
#include <oxstd.h>
#include <oxprob.h>
#include <oxfloat.h>

//Parametros para simulacao
const decl N = 20; //Tamanho da amostra INICIAL
const decl NREP = 100000; //Numero de replicas Monte Carlo
const decl alpha_1 = 0, alpha_2 = 0; //Controla o grau de heteroscedasticidade

main()
{

//Declaracao de variaveis
decl tempo,mu,i,beta,Y,MU;

decl vbhat,mtxinv,XtX,tau,VC10,VC5,VC1,VC1_HC2,VC5_HC2,VC10_HC2,VC10_HC3,VC5_HC3,
VC1_HC3,VC10_HC4,VC5_HC4,VC1_HC4;

decl P,Pt,H,resid2,var_white,var_HC2,var_HC3,var_HC4,whitevector=zeros(3,1),
HC2vector=zeros(3,1),HC3vector=zeros(3,1),HC4vector=zeros(3,1),tau_abs;

decl M,w,nvar,g,gnew,weight2,weight3,weight4,tau_abs_white,tau_abs_HC2,tau_abs_HC3,
tau_abs_HC4,hmax,hbarra,lambda,gama;

decl white=zeros(3,NREP),HC2=zeros(3,NREP),HC3=zeros(3,NREP),HC4=zeros(3,NREP),
mbhat=zeros(3,NREP);

decl txrej10white=0,txrej5white=0,txrej1white=0,txrej10HC2=0,txrej5HC2=0,txrej1HC2=0,
txrej10HC3=0,txrej5HC3=0,txrej1HC3=0;

decl txrej10HC4=0,txrej5HC4=0,txrej1HC4=0,sigma,x1 = zeros(N,1),x2 = zeros(N,1),X,
omega,varbetas,covbetas;

decl viesrelbetawhite,viesrelbetaHC2,viesrelbetaHC3,viesrelbetaHC4,teste,p,n;

decl stat_tau_white,stat_tau_HC2,stat_tau_HC3,stat_tau_HC4,median_tau;

//Registrando o tempo de inicio do programa

```

```

tempo = timer();

// valores criticos assintoticos
decl vc10 = quann(0.950);
decl vc5 = quann(0.975);
decl vc1 = quann(0.995);

//Selecao do gerador de numeros aleatorios
ranseed("GM"); // gerador Multiply-with-Carry de GM
ranseed({1111,1000});

print("*****\n");
print("Nome do programa: braga_tese.ox\n");
print("Autor: Antonio Carlos R. Braga Junior\n");
print("Data: ", date());
print("\nHora: ", time());
print("\n*****\n");
print("Modelo considerado:  $Y_t = \beta_1 + \beta_2 X_{2t} + \beta_3 X_{3t} + \text{erro}_t$ \n");
print("Funcao cedastica considerada:  $\sigma^2 = \exp\{\alpha_1 X_2 + \alpha_2 X_3\}$ \n");
print("Alpha_1 e Alpha_2 considerados iguais a : ", alpha_1, " e ", alpha_2, "\n");
print("Hipotese considerada:  $H_0: \beta_3 = 0$ \n");
print("Modelo sob  $H_0$ :  $Y_t = \beta_1 + \beta_2 X_{2t} + \text{erro}_t$ \n");
print("Numero de replicacoes Monte Carlo = ", NREP);
print("\nGerador de numeros pseudo-aleatorios utilizado: George Marsaglia");
print("\nSemente do gerador: ", ranseed(0));

//*****
x1 = ranu(N,1); //Gerando regressores da distribuicao Uniforme
x2 = ranu(N,1);
//*****
// x1 = ranlogn(N,1); //Gerando regressor da distribuicao LogNormal
// x2 = rant(N,1,2); //Gerando regressor da distrib. t-Student com 2 graus de liberdade.
//*****
// DEFININDO O TAMANHO AMOSTRAL. Replicar de 20 em 20
//(i.e. para N=60 fazer x1=x1|x1|x1 e x2=x2|x2|x2)
x1 = x1;
x2 = x2;
X = 1~x1~x2; //Criando a matriz do modelo incluindo o intercepto
n = rows(X);
nvar = columns(X);
print("\nTAMANHO AMOSTRAL: ", n);
print("\n");

beta = ones(nvar-1,1)|0; // Valores verdadeiros dos betas
MU = X*beta; //para testar a hipotese de beta3=0.

XtX = X'*X;
mxtxinvsym = invertsym(XtX);

P = mxtxinvsym*X';
Pt = P';

```

```

M = X*P;
H = unit(n) - M;
//retorna a diagonal do produto como um vetor linha
w = diagonal(M);

hmax = max(w); //Grau maximo de alavancagem
hbarra = meanr(w); //Grau medio de alavancagem
gama = hmax/hbarra; // Razao entre os graus maximo e medio de alavancagem

print("\n");
println("GRAUS DE ALAVANCAGEM:", "%c", {"Maximo", "Medio", "Razao"}, "%12.5f",
hmax~hbarra~gama);

g = (n*w) ./ nvar; //usado no HC4
gnew = g .> 4 .? 4 .: g; //minimo entre 4 e g (HC4)

weight2 = (1.0 ./ (1.0-w')) ); // usado para o HC2
weight3 = (1.0 ./ ((1.0-w') .^ 2) ); // usado para o HC3
weight4 = (1.0 ./ ((1.0-w') .^ gnew')) ); // usado para o HC4

tau = zeros(NREP,4); //Definindo a dimensao da matriz que armazena os valores
//das estatisticas de teste. Cada coluna representa um HC
//e cada linha representa uma replica.

sigma = sqrt(exp(alpha_1*x1 + alpha_2*x2)); //Funcao Cedastica
omega = diag(sigma.^2); //matriz omega n X n utilizada no calculo do vies da
//variancia dos betas.
covbetas = mtxtinv*X'*omega*X*mtxtinv; //matriz de covariancias esperadas(VERDADEIRAS)
//dos betas.
varbetas = diagonal(covbetas); //matriz de variancias esperadas(VERDADEIRAS)
//dos betas.

//Inicio do laco Monte Carlo
for (i = 0; i < NREP; ++i)
{
// resposta simulada
Y = MU + sigma.*rann(n,1); //geracao dos dados

// estimativa de MQO
vbhat=P*Y; //obtencao das estimativas dos parametros
mbhat[] [i] = vbhat;

// residuos ao quadrado
resid2 = (H*Y) .^ 2;

// calculo da variancia estimada de betachapeu via HCO
var_white = P * (resid2 .* Pt);
whitevector = diagonal(var_white); //usado em cada repeticao Monte Carlo
white[] [i] = whitevector'; //armazenando os valores da variancia para
//tomarmos a media posteriormente.

```

```

// calculo da variancia estimada de betachapeu via HC2
var_HC2 = P * ( resid2 .* weight2) .* Pt);
HC2vector = diagonal(var_HC2); //usado em cada repeticao Monte Carlo
HC2[] [i] = HC2vector'; //armazenando os valores da variancia para
//tomarmos a media posteriormente.

// calculo da variancia estimada de betachapeu via HC3
var_HC3 = P * ( resid2 .* weight3) .* Pt);
HC3vector = diagonal(var_HC3); //usado em cada repeticao Monte Carlo
HC3[] [i] = HC3vector'; //armazenando os valores da variancia para
//tomarmos a media posteriormente.

// calculo da variancia estimada de betachapeu via HC4
var_HC4 = P * ( resid2 .* weight4) .* Pt);
HC4vector = diagonal(var_HC4); //usado em cada repeticao Monte Carlo
HC4[] [i] = HC4vector'; //armazenando os valores da variancia para
//tomarmos a media posteriormente.

//ESTADISTICAS DE TESTE
//testa beta3=0 -> modelo com intercepto VIA HC0.
tau[i] [0]=vbhat [2]/sqrt(whitevector [2]);
//testa beta3=0 -> modelo com intercepto VIA HC2.
tau[i] [1]=vbhat [2]/sqrt(HC2vector [2]);
//testa beta3=0 -> modelo com intercepto VIA HC3.
tau[i] [2]=vbhat [2]/sqrt(HC3vector [2]);
//testa beta3=0 -> modelo com intercepto VIA HC4 .
tau[i] [3]=vbhat [2]/sqrt(HC4vector [2]);

} //Final do laço de Monte Carlo

// taxas de rejeicao
txrej10white = double(sumc(fabs(tau[] [0]) .> vc10));
txrej5white = double(sumc(fabs(tau[] [0]) .> vc5));
txrej1white = double(sumc(fabs(tau[] [0]) .> vc1));

txrej10HC2 = double(sumc(fabs(tau[] [1]) .> vc10));
txrej5HC2 = double(sumc(fabs(tau[] [1]) .> vc5));
txrej1HC2 = double(sumc(fabs(tau[] [1]) .> vc1));

txrej10HC3 = double(sumc(fabs(tau[] [2]) .> vc10));
txrej5HC3 = double(sumc(fabs(tau[] [2]) .> vc5));
txrej1HC3 = double(sumc(fabs(tau[] [2]) .> vc1));

txrej10HC4 = double(sumc(fabs(tau[] [3]) .> vc10));
txrej5HC4 = double(sumc(fabs(tau[] [3]) .> vc5));
txrej1HC4 = double(sumc(fabs(tau[] [3]) .> vc1));

//calculo do vies
viesrelbetawhite = (meanr(white)-varbetas') ./varbetas';
viesrelbetaHC2 = (meanr(HC2)-varbetas') ./varbetas';
viesrelbetaHC3 = (meanr(HC3)-varbetas') ./varbetas';
viesrelbetaHC4 = (meanr(HC4)-varbetas') ./varbetas';

```

```

print("\n");
print("VALORES VERDADEIROS DE BETA1, BETA2 e BETA3:", "%c",
{"Beta1", "Beta2", "Beta3"}, "%15.4f", beta');
print("\nESTIMATIVAS MEDIAS DE BETA1, BETA2 e BETA3:", "%c",
{"Beta1", "Beta2", "Beta3"}, "%15.4f", (meanr(mbhat))');

print("\n");
print("MEDIAS DAS VARIANCIAS ESTIMADAS:");
print("%12.5f", "%c", {"HC0", "HC2", "HC3", "HC4"},
meanr(white)~meanr(HC2)~meanr(HC3)~meanr(HC4));

println("\nVARIANCIAS VERDADEIRAS:", "%12.5f", varbetas');

print("\nVIESES RELATIVOS:");
print("%12.4f", "%c", {"HC0", "HC2", "HC3", "HC4"},
viesrelbetawhite~viesrelbetaHC2~viesrelbetaHC3~viesrelbetaHC4);

print("\nVIESES RELATIVOS TOTAIS:");
print("%12.4f", "%c", {"HC0", "HC2", "HC3", "HC4"}, sumc(fabs(viesrelbetawhite))~
sumc(fabs(viesrelbetaHC2))~sumc(fabs(viesrelbetaHC3))~sumc(fabs(viesrelbetaHC4)));

// valores absolutos das estatisticas de teste
tau_abs = fabs(tau); //forcando simetria
tau_abs_white = tau_abs [] [0];
tau_abs_HC2 = tau_abs [] [1];
tau_abs_HC3 = tau_abs [] [2];
tau_abs_HC4 = tau_abs [] [3];

// valores criticos empiricos
VC1      = quantilec(tau_abs_white, 0.99);
VC5      = quantilec(tau_abs_white, 0.95);
VC10     = quantilec(tau_abs_white, 0.90);
VC1_HC2  = quantilec(tau_abs_HC2, 0.99);
VC5_HC2  = quantilec(tau_abs_HC2, 0.95);
VC10_HC2 = quantilec(tau_abs_HC2, 0.90);
VC1_HC3  = quantilec(tau_abs_HC3, 0.99);
VC5_HC3  = quantilec(tau_abs_HC3, 0.95);
VC10_HC3 = quantilec(tau_abs_HC3, 0.90);
VC1_HC4  = quantilec(tau_abs_HC4, 0.99);
VC5_HC4  = quantilec(tau_abs_HC4, 0.95);
VC10_HC4 = quantilec(tau_abs_HC4, 0.90);

// grau de heteroscedasticidade
lambda = double(maxc(sigma.^2)/minc(sigma.^2)); //Grau de heteroscedasticidade

print("\n");
print("VALORES CRITICOS ESTIMADOS E ASSINTOTICOS:");
println("%11.4f", "%r", {"10%", " 5%", " 1%"}, "%c",
{"HC0", "HC2", "HC3", "HC4", "ASSINT."}, (VC10|VC5|VC1)~(VC10_HC2|VC5_HC2|VC1_HC2)~
(VC10_HC3|VC5_HC3|VC1_HC3)~(VC10_HC4|VC5_HC4|VC1_HC4)~(vc10|vc5|vc1) );

println(" ");

```

```

print("TAXAS DE REJEICAO OBTIDAS VIA SIMULACAO MONTE DE CARLO");
println("%11.2", "%r", {"10%", " 5%", " 1%"}, "%c", {"HC0", "HC2", "HC3", "HC4"},
((txrej10white|txrej5white|txrej1white)~(txrej10HC2|txrej5HC2|txrej1HC2)~
(txrej10HC3|txrej5HC3|txrej1HC3)~(txrej10HC4|txrej5HC4|txrej1HC4))/NREP*100 );

//estatisticas da estatistica de teste, TAU
stat_tau_white=moments(tau[][0]);
stat_tau_HC2=moments(tau[][1]);
stat_tau_HC3=moments(tau[][2]);
stat_tau_HC4=moments(tau[][3]);
median_tau=quantilec(tau,0.5);
print("\n");
print("ESTATISTICAS DA ESTATISTICA DE TESTE, TAU:");
println("%14.4f", "%r", {"# Replicas", " Media", " Desvio Padrao", " Assimetria",
" Curtose", " Mediana"}, "%c", {"HC0", "HC2", "HC3", "HC4"}, (stat_tau_white)~
(stat_tau_HC2)~(stat_tau_HC3)~(stat_tau_HC4)|(median_tau));

println("");
println("\nGRAU DE HETEROSCEDASTICIDADE:", "%10.4f", double(lambda));

print("\nTEMPO DE EXECUCAO: ", timespan(tempo));
print("\n");

print("\n*****\n");
}

```