Considerações sobre o procedimento de Regressão em Cristas

Robert Plant Pinto Santos

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Mestrado em Estatística

Orientador: Profa. Dr. Silvia Nagib Elian

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CNPq

São Paulo, Fevereiro de 2020

Considerações sobre o procedimento de Regressão em Cristas

Esta versão da dissertação contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 21 de Dezembro de 2020. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof[a]. Dr[a]. Silvia Nagib Elian (orientadora) - IME-USP
- Prof. Dr. João Ricardo Sato - UFABC
- Prof. Dr. Bruno Ramos dos Santos - UFES

Agradecimentos

A Deus.
À minha mãe por todo amor e apoio.
À professora Dr. Silvia Nagib Elian, por toda a ajuda, ensinamentos, paciência e disposição.
Ao Ademar, por todo apoio e ajuda em momentos difíceis.
Aos meus amigos do IME, que me apoiaram bastante, em especial para Marina, Pedro, e Simone.
Aos meus amigos da LexisNexis, em especial para Karina.
Aos meus professores do DEMA-UFC e IME-USP, por todo o conhecimento.
Aos meus amigos, por todo o apoio.

Resumo

SANTOS, Robert P. P. **Considerações sobre o procedimento de Regressão em Cristas**. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2020.

Em modelos de regressão, o método de Regressão em Cristas é uma alternativa ao método de mínimos quadrados em situações em que há multicolinearidade, consequência da existência de relações lineares entre as variáveis explicativas. Essa dissertação tem como objetivo apresentar atualizações ao trabalho realizado por Oishi (1983) sobre Regressão em Cristas. Inicialmente, é apresentado o procedimento de Regressão em Cristas, que consiste na adição de uma constante k, denotada por parâmetro das cristas, na diagonal principal da matriz $\mathbf{X}^\top\mathbf{X}$, as propriedades e uma generalização do método. Em seguida, serão apresentadas diferentes maneiras de estimação de k e uma discussão sobre inferência para os coeficientes de regressão e também é realizado um estudo de simulação para testar a eficiência dos intervalos de confiança para os coeficientes construídos através do método de *bootstrap*. Por último, é feita uma aplicação dos procedimentos descritos em um conjunto de dados reais.

Palavras-chave: Estimador em Cristas, Multicolinearidade, Erro Quadrático Médio, *Bootstrap*.

Abstract

SANTOS, Robert Plant Pinto **Consideration about the Ridge Regression procedure.**. 2020. Dissertation (Masters) - Institute of Mathematics and Statistics, University de São Paulo, São Paulo, 2020.

In regression models, the Ridge Regression method is an alternative to ordinary least squares in situations where there is multicolinearity, which is a consequence of the existence of linear relations between explanatory variables. This work has as objective to present updates on the work made by Oishi (1983) about Ridge Regression. Initially, it will be shown the Ridge Regression procedure that consists in adding the constant k, also known as ridge parameter, to the main diagonal of the matrix $\mathbf{X}^\top\mathbf{X}$, the method's properties and generalization. Then, it will be presented different ways for estimating the constant k and a discussion about inerence for the regression coefficients and also a simulation study is made to test the efficiency of confidence intervals for the coefficients built using the bootstrap method. Finally, the described methods will be applied to a real dataset.

Keywords: Ridge Estimator, Multicollinearity, Mean Squared Error, Bootstrap.

Sumário

Lista de Abreviaturas

CNO	Componente não-ortogonal (*non-orthogonal component*)
EQM	Erro quadrático médio (*Mean squared error*)
ENVVUM	Estimador não viciado de variância uniformemente mínima (*Uniformly minimum variance unbiased estimator*)
FIV	Fator da inflação da variância (*Variance inflation factor*)
MMQ	Método de mínimos quadrados (*Least square method*)
SQReg	Soma de quadrados de regressão (*Regression sum of squares*)
SQRes	Soma de quadrados dos resíduos (*Residual sum of squares*)

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Modelos de Regressão Linear são utilizados para prever valores de uma variável de interesse através dos valores de outras variáveis e também para estudar a relação dessas com a variável de interesse. Essa relação é estudada através dos coeficientes de regressão, que precisam ser estimados utilizando os dados disponíveis. O método de estimação mais conhecido e com melhores propriedades quando todas as suposições são válidas é o de mínimos quadrados.

Entretanto, na presença de multicolinearidade entre as variáveis explicativas, o estimador obtido pelo método de mínimos quadrados tem grande variância e é inadequado para estimar os coeficientes de regressão, essa afirmação é demonstrada no Capítulo 2. O procedimento de Regressão em Cristas é uma importante alternativa na presença de multicolinearidade. Este método teve grande destaque a partir dos trabalhos de Hoerl e Kennard (1970a), Hoerl e Kennard (1970b) e está descrito nos principais livros de Regressão Linear (Draper e Smith (1998), Montgomery *et al.* (2012)).

Oishi (1983) apresentou uma excelente dissertação de mestrado em Estatística no IME-USP, na qual descreve em detalhes o procedimento, apresenta sua interpretação geométrica, aplicações, utilizando os artigos de Hoerl e Kennard (1970a) que introduzem o método de Regressão em Cristas. Além disso, Hoerl *et al.* (1975), Wichern e Churchill (1978) e McDonald e Galarneau (1975) realizaram estudos de simulação comparando o método de mínimos quadrados e também fizeram comparações entre diferentes métodos de escolha do parâmetro de cristas k, que é um dos desafios do método de Regressão em Cristas. Os artigos de Lindley e Smith (1972) e Lawless e Wang (1976) discutem uma interpretação Bayesiana para o procedimento de Regressão em Cristas.

O presente trabalho tem como objetivo trazer uma atualização do tópico, visto que muitos artigos relativamente recentes encontram-se na literatura como Obenchain (1977) , Ullah *et al.* (1984), Hoerl e W. Kennard (1990), Halawa e El Bassiouni (2000), que apresentam técnicas de inferência em Regressão em Cristas, Walker e Page (2001) discutem um método de seleção de variáveis, Kibria (2003), Khalaf e Shukur (2005), Dorugade e Kashid (2010) fornecem vários métodos de escolha para o parâmetro k.

Além disso também serão discutidos métodos de re-amostragem e como eles podem auxiliar na construção de inferência para os coeficientes de regressão estimados pelo método de Regressão em Cristas.

1.1 Organização da dissertação

Esta dissertação é composta por oito capítulos. No Capítulo 2 são discutidos alguns tópicos como o método das cristas, o modelo de Regressão em Cristas, a forma e propriedades do estimador desse procedimento. Além disso, também são discutidos a forma canônica, uma generalização do modelo de Regressão em Cristas, um método de seleção de variáveis baseado na estatística C_p e o modelo de Regressão em Cristas sob o ponto de vista bayesiano.

Os critérios para a escolha do parâmetro de cristas k e comentários sobre os estudos de simulação realizados por diversos autores são discutidos ao longo do Capítulo 3. Alguns métodos de inferências construídos a partir do estimador do procedimento de Regressão em Cristas como o teste $t_{(k)}$ para um único coeficiente e o teste $F_{(k)}$ para todos os coeficientes de regressão, e devido a estatística $F_{(k)}$ não ter uma forma fechada são apresentados limites inferior e superior para essa estatística e valores de referência para esses limites utilizando a distribuição F são todos apresentados no Capítulo 4, onde também são apresentados a decomposição da soma de quadrados totais do modelo de Regressão em Cristas, um teste t não exato e a intervalos de confiança construídos através do método de re-amostragem por *bootstrap*.

O método de re-amostragem por *bootstrap* é apresentado com mais detalhes no Capítulo 5 assim como construir os intervalos de confiança utilizando esse procedimento, neste capítulo também é discutido como aplicar o método de *bootstrap* para inferência sobre os coeficientes de regressão, e em particular os coeficientes de Regressão em Cristas. Em seguida, no Capítulo 6 é discutido como foi feito um estudo de simulação para testar a eficiência dos intervalos de confiança construídos utilizando a técnica de *bootstrap* para os coeficientes do modelo de Regressão em Cristas e também são apresentados os resultados das simulações.

O modelo de Regressão em Cristas é ajustado, no Capítulo 7, a um conjunto de dados disponibilizado pelo Centro de Estatística Aplicado (CEA-IME-USP) para estudar a potência de uma droga em ratos, e para finalizar o Capítulo 8 apresenta as considerações finais com sugestões de tópicos complementares a serem abordados futuramente.

Capítulo 2

Análise em cristas

2.1 Regressão em Cristas

A solução através dos mínimos quadrados é bastante comum em problemas de Regressão Linear, entretanto, existem situações em que essa solução é não satisfatória, podendo apresentar instabilidades nas estimativas dos parâmetros do modelo.

Em aplicações de Regressão, uma dessas situações é quando há multicolinearidade, que se dá devido à existência de relações lineares entre as variáveis explicativas. O procedimento de análise de cristas foi primeiramente sugerido por Hoerl e Kennard (1970a) como alternativa para tentar solucionar esse problema.

Nessa seção será discutido como essa técnica, denominada Regressão em Cristas, pode ser aplicada em modelos de regressão. Serão discutidos ainda, os métodos de estimação, a relação dessa técnica com o método de mínimos quadrados, a forma canônica do modelo e o método de regressão em cristas generalizado.

2.1.1 Modelo de Regressão Linear

O modelo de regressão linear é definido como:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{2.1}$$

em que

$\mathbf{X}$: é a matriz de variáveis explicativas do modelo de dimensão $n \times p$ e posto p,

$\mathbf{y}$: é o vetor da variável resposta de dimensão n,

$\boldsymbol{\beta}$: é o vetor de coeficientes do modelo de dimensão p,

$\boldsymbol{\epsilon}$: é o vetor aleatório de erros de dimensão n, com $\mathrm{E}(\boldsymbol{\epsilon}) = \mathbf{0}$ e $\mathrm{Var}(\boldsymbol{\epsilon}) = \sigma^2\mathbf{I}_n$.

O estimador de mínimos quadrados de $\boldsymbol{\beta}$ é obtido minimizando em $\boldsymbol{\beta}$ a soma de quadrados dos resíduos (SQRes) definida por:

$$\mathrm{SQRes} = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}), \tag{2.2}$$

a forma algébrica do estimador é:

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} \\ &= \mathbf{C}^{-1}\mathbf{X}^\top\mathbf{y}. \end{aligned} \tag{2.3}$$

O estimador $\hat{\boldsymbol{\beta}}$ é não viciado e de variância uniformemente mínima, como pode ser visto em Eicker *et al.* (1963). Nesse estudo, por conveniência, as variáveis explicativas serão consideradas em sua forma padronizada, de forma que a matriz $\mathbf{C} = \mathbf{X}^\top\mathbf{X}$ forneça as correlações entre as variáveis explicativas e o vetor $\mathbf{X}^\top\mathbf{y}$ as correlações entre as variáveis explicativas e a variável resposta.

Quando há presença de multicolinearidade entre as variáveis explicativas, a matriz $\mathbf{C}^{-1}$ se distancia de uma matriz identidade e os elementos de sua diagonal principal, c_{ii}^{-1}, para $i = 1, \ldots, p$ são inflacionados, e, como será discutido mais a frente, acarretará no aumento da variância dos estimadores de mínimos quadrados.

Denotando os autovalores da matriz $\mathbf{C}$ por: $\lambda_{\max} = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p = \lambda_{\min} > 0$, o número de condição (2.4) é definido por

$$\kappa(\mathbf{X}) = \frac{\lambda_{\max}}{\lambda_{\min}}. \tag{2.4}$$

Essa quantidade é limitada inferiormente por 1, que acontece no caso em que a matriz $\mathbf{C}$ for ortogonal. Se houver dependência linear entre as colunas da matriz $\mathbf{X}$ então $\lambda_{\min} \approx 0$, e consequentemente $\kappa(\mathbf{X})$ tende a ser grande.

Uma outra forma de avaliar a multicolinearidade existente entre as variáveis explicativas é através do cálculo do fator de inflação de variância (FIV) para cada variável explicativa. O FIV da i-ésima variável explicativa é obtido a partir do cálculo de R_i^2, o coeficiente de explicação do modelo de regressão linear em que $\mathbf{X}_i$ é usada como variável resposta e $\mathbf{X}_1, \ldots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_p$ como variáveis preditoras. Seja c_{ii}^{-1} o i-ésimo elemento da diagonal principal da matriz $\mathbf{C}^{-1}$ descrita em (2.3), o FIV_i é definido como:

$$\text{FIV}_i = c_{ii}^{-1} = \frac{1}{1 - R_i^2}, \tag{2.5}$$

dessa forma, situações em que $\mathbf{X}_i$ puder ser predita pela outras variáveis explicativa resultará em valores de $R_i^2 \approx 1$ e um valor FIV elevado, e portanto a estimativa do coeficiente do parâmetro correspondente a i-ésima variável preditora terá uma variância grande.

Para demonstrar os efeitos de um mal condicionamento da matriz $\mathbf{C}$, podemos analisar o valor esperado da quantidade L^2, definida como a distância euclidiana entre $\hat{\boldsymbol{\beta}}$ e $\boldsymbol{\beta}$,

$$L^2 = (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^\top(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}).$$

Então, o valor esperado dessa quantidade, que será denominado de erro quadrático médio (EQM), é obtido por:

$$\begin{aligned}\text{EQM}(\hat{\boldsymbol{\beta}}) = \text{E}(L^2) &= \text{E}[(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^\top(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})] \\ &= \text{E}(\hat{\boldsymbol{\beta}}^\top\hat{\boldsymbol{\beta}}) - \text{E}(\hat{\boldsymbol{\beta}}^\top)\boldsymbol{\beta} - \boldsymbol{\beta}^\top\text{E}(\hat{\boldsymbol{\beta}}) + \boldsymbol{\beta}^\top\boldsymbol{\beta} \\ &= \text{E}(\hat{\boldsymbol{\beta}}^\top\hat{\boldsymbol{\beta}}) - \boldsymbol{\beta}^\top\boldsymbol{\beta}.\end{aligned}$$

Devido à relação $\text{E}(\mathbf{y}^\top\mathbf{A}\mathbf{y}) = \text{tr}(\mathbf{A}\text{Var}(\mathbf{y})) + \text{E}(\mathbf{y})^\top\mathbf{A}\text{E}(\mathbf{y})$, ver Searle e Khuri (2017), segue que

$$\begin{aligned}\text{E}(L^2) &= \sigma^2\text{tr}(\mathbf{C}^{-1}) + \boldsymbol{\beta}^\top\boldsymbol{\beta} - \boldsymbol{\beta}^\top\boldsymbol{\beta} \\ &= \sigma^2\text{tr}(\mathbf{C}^{-1}) \\ &= \sigma^2\sum_{i=1}^{p} 1/\lambda_i\end{aligned}$$

ou equivalentemente

$$E(\hat{\boldsymbol{\beta}}^\top\hat{\boldsymbol{\beta}}) = \sigma^2\sum_{i=1}^{p} 1/\lambda_i + \boldsymbol{\beta}^\top\boldsymbol{\beta}. \tag{2.6}$$

Verifica-se ainda que a variância de L^2, sob a hipótese de normalidade do vetor aleatório $\boldsymbol{\epsilon}$, é dada por:

$$\begin{aligned}Var(L^2) &= 2\sigma^4\text{tr}(\mathbf{C})^{-2} \\ &= 2\sigma^4\sum_{i=1}^{p} 1/\lambda_i^2 \leq 2\sigma^4\sum_{i=1}^{p}\frac{1}{\lambda_p^2} = \frac{2\sigma^4 p}{\lambda_{\text{min}}^2}.\end{aligned}$$

Desse modo, os limites superiores para $E(L^2)$ e $Var(L^2)$ são, respectivamente, $\sigma^2 p/\lambda_{\text{min}}$ e $2\sigma^4 p/\lambda_{\text{min}}^2$. Portanto, como discutido anteriormente, na presença de alta multicolinearidade entre as variáveis explicativas, λ_{min} irá se aproximar de 0, e por consequência a distância média entre $\hat{\boldsymbol{\beta}}$ e $\boldsymbol{\beta}$, medida por $\text{E}(L^2)$, e o erro quadrático médio total de $\hat{\boldsymbol{\beta}}$ serão grandes. Contudo, como $\hat{\boldsymbol{\beta}}$ é um estimador não viciado de variância mínima, portanto, uma alternativa será procurar um bom estimador na classe dos estimadores viciados.

2.1.2 O estimador pelo método das Cristas

Considerando o modelo descrito em (2.1) e $\hat{\boldsymbol{\beta}}^*$ um estimador qualquer de $\boldsymbol{\beta}$, podemos denotar a soma de quadrados dos resíduos ao se estimar $\boldsymbol{\beta}$ por $\hat{\boldsymbol{\beta}}^*$ por

$$\Delta = (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}^*)^\top(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}^*). \tag{2.7}$$

Utilizando o método da análise de cristas, Hoerl (1959), para minimizar (2.7), é possível verificar a existência de soluções distintas para $\hat{\boldsymbol{\beta}}^*$ que possam ser mais estáveis a pequenas pertubações nos dados do que a solução pelo Método de Mínimos Quadrados ($\hat{\boldsymbol{\beta}}$).

Formalmente, é necessário minimizar Δ sujeito à restrição $\hat{\boldsymbol{\beta}}^{*\top}\hat{\boldsymbol{\beta}}^{*} = R^2$, o que é feito através do método de multiplicadores de Lagrange, definindo a função:

$$F = (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}^{*})^{\top}(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}^{*}) + k(\hat{\boldsymbol{\beta}}^{*\top}\hat{\boldsymbol{\beta}}^{*} - R^2),$$

em que k é o multiplicador de Lagrange. Ao derivar F em relação à $\hat{\boldsymbol{\beta}}^{*}$ e igualar a zero, obtemos as seguintes expressões

$$\begin{aligned} \partial F/\partial\hat{\boldsymbol{\beta}}^{*} &= -2\mathbf{X}^{\top}\mathbf{y} + 2\mathbf{C}\hat{\boldsymbol{\beta}}^{*} + 2k\hat{\boldsymbol{\beta}}^{*} = \mathbf{0} \\ (\mathbf{C} + k\mathbf{I}_p)\hat{\boldsymbol{\beta}}^{*} &= \mathbf{X}^{\top}\mathbf{y}, \end{aligned}$$

portanto, a expressão para $\hat{\boldsymbol{\beta}}^{*}$ que minimiza Δ sujeito a restrição $\hat{\boldsymbol{\beta}}^{*\top}\hat{\boldsymbol{\beta}}^{*} = R^2$ é:

$$\begin{aligned} \hat{\boldsymbol{\beta}}^{*}_{(k)} &= (\mathbf{C} + k\mathbf{I}_p)^{-1}\mathbf{X}^{\top}\mathbf{y} \\ &= \mathbf{W}\mathbf{X}^{\top}\mathbf{y}, \end{aligned} \tag{2.8}$$

em que $\mathbf{W} = (\mathbf{C} + k\mathbf{I}_p)^{-1}$.

Nessas condições, $\hat{\boldsymbol{\beta}}^{*}_{(k)}$ é o estimador encontrado pelo método das cristas para $\boldsymbol{\beta}$, e a análise de regressão construída utilizando esse estimador leva o nome de Regressão em Cristas. Note que se $k = 0$, então $\hat{\boldsymbol{\beta}}^{*}_{(k)} = \hat{\boldsymbol{\beta}}$, o quê implica que o estimador de mínimos quadrados é um caso especial dessa classe de estimadores.

Verifica-se que os autovalores da matriz $\mathbf{W}$ são $(\xi_{\mathbf{W}})_i = 1/(\lambda_i + k)$, para $i = 1, \ldots, p$, em que λ_i são os autovalores da matriz $\mathbf{C}$. Esse resultado é obtido diretamente pela definição da matriz $\mathbf{W}$ em (2.8) e a solução da equação característica $|\mathbf{W} - \boldsymbol{\xi}\mathbf{I}_p| = 0$.

Assim, devido a adição da constante k à diagonal principal da matriz $\mathbf{C}$, o número de condição desse novo sistema é:

$$\kappa(\mathbf{X}(k)) = \frac{\lambda_{\max} + k}{\lambda_{\min} + k}. \tag{2.9}$$

Note que, para $k > 0$, a quantidade (2.9) é sempre menor que $\lambda_{\max}/\lambda_{\min}$. Os valores de k podem ser tanto positivos quanto negativos, entretanto, se k se aproxima de $-\lambda_{\min}$, tanto o número condição como os valores de $\hat{\boldsymbol{\beta}}^{*}_{(k)}$ tendem a aumentar rapidamente em valor absoluto, isso acontece devido à matriz $\mathbf{W}$ ser singular quando $k = -\lambda_{\min}$. A quantidade k será adotada daqui para frente como uma quantidade maior ou igual a zero.

2.1.3 Propriedades do estimador de Regressão em Cristas

Nessa sub-seção, serão discutidas algumas propriedades do estimador de regressão em cristas.

(i) $\hat{\boldsymbol{\beta}}^{*}_{(k)}$ em termos de $\hat{\boldsymbol{\beta}}$

É possível denotar a relação entre o estimador de mínimos quadrados e o estimador de regressão em cristas por:

$$\begin{aligned}\hat{\boldsymbol{\beta}}^*_{(k)} &= (\mathbf{C}+k\mathbf{I}_p)^{-1}\mathbf{X}^\top\mathbf{y}\\ &= (\mathbf{C}+k\mathbf{I}_p)^{-1}\mathbf{C}\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{y}\\ &= [\mathbf{C}^{-1}(\mathbf{C}+k\mathbf{I}_p)]^{-1}\hat{\boldsymbol{\beta}}\\ &= [\mathbf{I}_p+k\mathbf{C}^{-1}]^{-1}\hat{\boldsymbol{\beta}}\\ &= \mathbf{Z}\hat{\boldsymbol{\beta}}, \end{aligned} \tag{2.10}$$

em que $\mathbf{Z} = [\mathbf{I}_p + k\mathbf{C}^{-1}]^{-1}$, logo $\hat{\boldsymbol{\beta}}^*_{(k)}$ é linear em $\hat{\boldsymbol{\beta}}$. Como as propriedades do estimador de mínimos quadrados já são mais conhecidas (ver Montgomery *et al.* (2012)), é viável utilizar seus resultados para determinar o valor esperado e a variância do estimador de regressão em cristas.

(ii) Valor esperado de $\hat{\boldsymbol{\beta}}^*_{(k)}$

Para $k > 0$, é possível mostrar que $\hat{\boldsymbol{\beta}}^*_{(k)}$ é um estimador viciado para $\boldsymbol{\beta}$, pois temos

$$\begin{aligned} E(\hat{\boldsymbol{\beta}}^*_{(k)}) &= E(\mathbf{Z}\hat{\boldsymbol{\beta}})\\ &= \mathbf{Z}E(\hat{\boldsymbol{\beta}})\\ &= \mathbf{Z}\boldsymbol{\beta} \neq \boldsymbol{\beta}. \end{aligned}$$

(iii) Variância do estimador $\hat{\boldsymbol{\beta}}^*_{(k)}$

$$\begin{aligned} \mathrm{Var}(\hat{\boldsymbol{\beta}}^*_{(k)}) &= \mathrm{Var}(\mathbf{Z}\hat{\boldsymbol{\beta}})\\ &= \mathbf{Z}\mathrm{Var}(\hat{\boldsymbol{\beta}})\mathbf{Z}^\top\\ &= \sigma^2\mathbf{Z}\mathbf{C}^{-1}\mathbf{Z}^\top, \end{aligned} \tag{2.11}$$

(iv) Comprimento de $\hat{\boldsymbol{\beta}}^*_{(k)}$

Para $k \neq 0$, a norma de $\hat{\boldsymbol{\beta}}^*_{(k)}$ é menor que a norma de $\hat{\boldsymbol{\beta}}$, ver Riley (1955), isso implica que

$$(\hat{\boldsymbol{\beta}}^*_{(k)})^\top(\hat{\boldsymbol{\beta}}^*_{(k)}) < \hat{\boldsymbol{\beta}}^\top\hat{\boldsymbol{\beta}}. \tag{2.12}$$

Uma matriz $\mathbf{A}$ é dita positiva definida se $\mathbf{x}^\top\mathbf{A}\mathbf{x} > 0$, para qualquer vetor $\mathbf{x} \neq \mathbf{0}$. Se $\mathbf{A}$ é positiva definida então $\mathbf{A}^{-1}$ também é positiva definida, pois fazendo $\mathbf{y} = \mathbf{A}\mathbf{x}$ então

$$\mathbf{y}^\top\mathbf{A}^{-1}\mathbf{y} = \mathbf{x}^\top\mathbf{A}^\top\mathbf{A}^{-1}\mathbf{A}\mathbf{x} = \mathbf{x}^\top\mathbf{A}\mathbf{x} > 0. \tag{2.13}$$

A soma entre duas matrizes, digamos $\mathbf{A}$ e $\mathbf{B}$, positivas definidas também resulta em uma matriz positiva definida, pois

$$\mathbf{x}^\top(\mathbf{A}+\mathbf{B})\mathbf{x} = \mathbf{x}^\top\mathbf{A}\mathbf{x} + \mathbf{x}^\top\mathbf{B}\mathbf{x} > 0, \tag{2.14}$$

para qualquer $\mathbf{x} \neq \mathbf{0}$.

Com os resultados (2.13) e (2.14) e como a matriz $\mathbf{C}$ é simétrica e positiva definida, $\mathbf{Z}$ também é simétrica e positiva definida para $k > 0$. Denotando $\boldsymbol{\xi}_\mathbf{Z}$ como o vetor com os autovalores de $\mathbf{Z}$, Riley (1955) demonstra que para $\mathbf{Z}$ positiva definida, a seguinte inequação é válida

$$\begin{aligned}(\hat{\boldsymbol{\beta}}^*_{(k)})^\top(\hat{\boldsymbol{\beta}}^*_{(k)}) &= (\mathbf{Z}\hat{\boldsymbol{\beta}})^\top(\mathbf{Z}\hat{\boldsymbol{\beta}}) \\ &\leq \max(\boldsymbol{\xi}_\mathbf{Z})\hat{\boldsymbol{\beta}}^\top\hat{\boldsymbol{\beta}}. \end{aligned} \tag{2.15}$$

De forma análoga à descrita para encontrar os autovalores de $\mathbf{W}$, é possível determinar $(\xi_\mathbf{Z})_i = \lambda_i/(\lambda_i + k)$, com $i = 1, \ldots, p$. Então, temos $\max(\boldsymbol{\xi}_\mathbf{Z}) = \lambda_{\max}/(\lambda_{\max} + k)$ e

$$(\hat{\boldsymbol{\beta}}^*_{(k)})^\top(\hat{\boldsymbol{\beta}}^*_{(k)}) \leq \left(\frac{\lambda_{\max}}{\lambda_{\max}+k}\right)\hat{\boldsymbol{\beta}}^\top\hat{\boldsymbol{\beta}}. \tag{2.16}$$

(v) A soma dos quadrados dos resíduos para $\hat{\boldsymbol{\beta}}^*_{(k)}$

Para definir a soma dos quadrados dos resíduos para $\hat{\boldsymbol{\beta}}^*_{(k)}$ será feito uso da seguinte relação

$$\begin{aligned}\mathbf{Z} &= \mathbf{I}_p - k(\mathbf{C} + k\mathbf{I}_p)^{-1} \\ &= \mathbf{I}_p - k\mathbf{W}. \end{aligned} \tag{2.17}$$

Essa relação é verificada escrevendo $\mathbf{Z}$ na sua forma alternativa, $\mathbf{Z} = (\mathbf{C} + k\mathbf{I}_p)^{-1}\mathbf{C} = \mathbf{W}\mathbf{C}$ (ver equação (2.10)), pré-multiplicando ambos os termos em (2.17) por $\mathbf{W}^{-1}$.

A soma de quadrados dos resíduos para $\hat{\boldsymbol{\beta}}^*_{(k)}$ é definida por:

$$\begin{aligned}\text{SQRes}(\hat{\boldsymbol{\beta}}^*_{(k)}) &= \left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}^*_{(k)}\right)^\top\left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}^*_{(k)}\right) \\ &= \left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right)^\top\left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right) + k^2\hat{\boldsymbol{\beta}}^\top_{(k)}\mathbf{C}^{-1}\hat{\boldsymbol{\beta}}_{(k)} \\ &= \text{SQRes}(\hat{\boldsymbol{\beta}}) + k^2\hat{\boldsymbol{\beta}}^\top\mathbf{Z}^\top\mathbf{C}^{-1}\mathbf{Z}\hat{\boldsymbol{\beta}}. \end{aligned} \tag{2.18}$$

A expressão (2.18) mostra que a soma dos quadrados dos resíduos de $\hat{\boldsymbol{\beta}}^*_{(k)}$ pode ser escrita como a soma dos quadrados dos resíduos do estimador de mínimos quadrados para $\boldsymbol{\beta}$ mais uma quantidade positiva que depende de k.

(vi) O erro quadrático médio de $\hat{\boldsymbol{\beta}}^*_{(k)}$

O EQM de $\hat{\boldsymbol{\beta}}^*_{(k)}$ é definido por

$$
\begin{aligned}
E(L^2(k)) &= E[(\hat{\boldsymbol{\beta}}^*_{(k)} - \boldsymbol{\beta})^\top (\hat{\boldsymbol{\beta}}^*_{(k)} - \boldsymbol{\beta})] \\
&= E[(\mathbf{Z}\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^\top (\mathbf{Z}\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})] \\
&= E[(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^\top \mathbf{Z}^\top \mathbf{Z} (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})] + (\mathbf{Z}\boldsymbol{\beta} - \boldsymbol{\beta})^\top (\mathbf{Z}\boldsymbol{\beta} - \boldsymbol{\beta}) \\
&= \sigma^2 \mathbf{tr}(\mathbf{Z}^\top \mathbf{Z}\mathbf{C}^{-1}) + \boldsymbol{\beta}^\top (\mathbf{Z} - \mathbf{I}_p)^\top (\mathbf{Z} - \mathbf{I}_p)\boldsymbol{\beta} \\
&= \sigma^2 \mathbf{tr}(\mathbf{W} - k\mathbf{W}^2) + k^2 \boldsymbol{\beta}^\top \mathbf{W}^2 \boldsymbol{\beta} \\
&= \sigma^2 [\mathrm{tr}(\mathbf{W}) - k\mathrm{tr}(\mathbf{W}^2)] + k^2 \boldsymbol{\beta}^\top \mathbf{W}^2 \boldsymbol{\beta} \\
&= \sigma^2 \sum_{i=1}^{p} \frac{\lambda_i}{(\lambda_i + k)^2} + k^2 \boldsymbol{\beta}^\top \mathbf{W}^2 \boldsymbol{\beta} \\
&= \gamma_1(k) + \gamma_2(k). \qquad (2.19)
\end{aligned}
$$

A expressão do erro quadrático médio de $\hat{\boldsymbol{\beta}}^*_{(k)}$ foi decomposta em duas partes. O primeiro termo de (2.19), $\gamma_1(k)$ é a soma das variâncias (também conhecida como variância total) dos estimadores de β na regressão em cristas, em outras palavras, é a soma dos elementos da diagonal principal de (2.11) . O segundo termo de (2.19), $\gamma_2(k)$ é a distância quadrática de $\mathbf{Z}\boldsymbol{\beta}$ até $\boldsymbol{\beta}$, e portanto, é o quadrado do viés ao utilizar $\hat{\boldsymbol{\beta}}^*_{(k)}$ ao invés de $\hat{\boldsymbol{\beta}}$. Note que $\gamma_2(0) = 0$.

Analisando $\gamma_1(k)$, é possível observar que essa quantidade é uma função monótona decrescente de k, por outro lado, $\gamma_2(k)$, é uma função monótona crescente de k. Entretanto, ao observar o comportamento da derivada de ambas funções quando $k \to 0^+$, tem-se

$$\lim_{k \to 0^+} (\partial \gamma_1 / \partial k) = -2\sigma^2 \sum_{i=1}^{p} (1/\lambda_i^2) \qquad (2.20)$$

$$\lim_{k \to 0^+} (\partial \gamma_2 / \partial k) = 0, \qquad (2.21)$$

dessa forma, $\gamma_1(k)$ tem derivada negativa que, para $\mathbf{C}$ ortogonal, se aproxima de $-2p\sigma^2$ e, para $\mathbf{C}$ mal condicionada e $\lambda_p \to 0$, se aproxima de $-\infty$ quando $k \to 0^+$. Além disso, outro resultado relevante é que o limite de $\gamma_1(k)$ é zero quando $k \to \infty$.

Esses resultados são importantes pois é possível concluir que é possível escolher $k > 0$ que adiciona pouco viés as estimativas dos coeficientes mas que reduz a variância total significativamente e consequentemente diminuindo o erro quadrático médio do estimador $\hat{\boldsymbol{\beta}}^*_{(k)}$.

(vii) Teorema da existência

O teorema da existência, ver Hoerl e Kennard (1970a), garante que, se $\boldsymbol{\beta}^\top \boldsymbol{\beta} < \infty$, existe pelo menos um valor de $k > 0$, tal que

$$\mathrm{EQM}(\hat{\boldsymbol{\beta}}^*_{(k)}) < \mathrm{EQM}(\hat{\boldsymbol{\beta}}) = \sigma^2 \sum \frac{1}{\lambda_i}. \qquad (2.22)$$

Para demonstrar o teorema, temos,

$$\begin{aligned}\text{EQM}(\hat{\boldsymbol{\beta}}^*_{(k)}) &= E(L^2(k)) = \gamma_1(k) + \gamma_2(k) \\ &= \sigma^2 \sum_{i=1}^{p} \frac{\lambda_i}{(\lambda_i + k)^2} + \sum_{i=1}^{p} \frac{k^2 \sigma_i^2}{(\lambda_i + k)^2},\end{aligned} \tag{2.23}$$

logo,

$$\frac{\partial \text{EQM}(\hat{\boldsymbol{\beta}}^*_{(k)})}{\partial k} = -2\sigma^2 \sum_{i=1}^{p} \frac{\lambda_i}{(\lambda_i + k)^3} + 2k \sum_{i=1}^{p} \frac{\lambda_i}{(\lambda_i + k)^3}, \tag{2.24}$$

assim,

$$\lim_{k \to 0} \frac{\partial \text{EQM}(\hat{\boldsymbol{\beta}}^*_{(k)})}{\partial k} = -2\sigma^2 \sum_{i=1}^{p} \frac{1}{\lambda_i}^2 < 0 \tag{2.25}$$

que indica a inclinação negativa de $\partial \text{EQM}(\hat{\boldsymbol{\beta}}^*_{(k)})/\partial k$ nas vizinhanças de $k = 0$. Portanto, isso implica na existência de valores de $k > 0$, que tornam $\text{EQM}(\hat{\boldsymbol{\beta}}^*_{(k)}) < \text{EQM}(\hat{\boldsymbol{\beta}}^*_{(0)}) = \text{EQM}(\hat{\boldsymbol{\beta}})$. Alguns desses valores são encontrados resolvendo a inequação

$$\frac{\partial \text{EQM}(\hat{\boldsymbol{\beta}}^*_{(k)})}{\partial k} \leq 0, \tag{2.26}$$

da qual é obtido

$$k \leq \frac{p\sigma^2}{\boldsymbol{\beta}^\top \boldsymbol{\beta}}. \tag{2.27}$$

2.2 Generalização do modelo de Regressão em Cristas

2.2.1 Forma canônica

Considerando novamente o modelo linear descrito em (2.1), é possível transformá-lo na forma canônica (Goldstein e Smith (1974)). Para isso, é preciso encontrar as matrizes ortogonais $\mathbf{U}$, de dimensão $n \times n$, e $\mathbf{V}$, de dimensão $p \times p$, tal que

$$\mathbf{X}^\top \mathbf{X} = \mathbf{V} \boldsymbol{\Lambda} \mathbf{V}^\top \tag{2.28}$$

e

$$\mathbf{U}^\top \mathbf{X} \mathbf{V} = \mathbf{D} = \begin{pmatrix} \boldsymbol{\Lambda}^{1/2} \\ \mathbf{0} \end{pmatrix}, \tag{2.29}$$

em que $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_p)$ é uma matriz diagonal com os autovalores de $\mathbf{X}^\top\mathbf{X}$ na diagonal principal. Estas matrizes são obtidas através da decomposição de $\mathbf{X}^\top\mathbf{X}$ em valores singulares.

Tomando $\mathbf{r} = \mathbf{U}^\top\mathbf{y}$, $\boldsymbol{\alpha} = \mathbf{V}^\top\boldsymbol{\beta}$ e $\boldsymbol{\nu} = \boldsymbol{U}^\top\boldsymbol{\epsilon}$, é possível obter uma expressão equivalente para o modelo linear

$$\mathbf{r} = \mathbf{D}\boldsymbol{\alpha} + \boldsymbol{\nu}, \tag{2.30}$$

em que $\text{E}(\boldsymbol{\nu}) = \mathbf{0}$ e $\text{Var}(\boldsymbol{\nu}) = \sigma^2\mathbf{I}_p$.

Transformações ortogonais consistem em uma rotação dos eixos de $\boldsymbol{\beta}$ e portanto preservam distâncias, em vista disso qualquer estimador linear para $\boldsymbol{\beta}$ será equivalente e terá o mesmo EQM de um estimador para $\boldsymbol{\alpha}$. Considerando a forma canônica do modelo de regressão linear, o estimador de mínimos quadrados de $\boldsymbol{\alpha}$ é definido por

$$\hat{\boldsymbol{\alpha}} = \mathbf{\Lambda}^{-1}\mathbf{D}^\top\mathbf{r}, \tag{2.31}$$

logo, $\hat{\alpha}_i = r_i/\lambda_i^{1/2}$, para $i = 1, \ldots, p$. O estimador pelo método de regressão em cristas para $\boldsymbol{\alpha}$ é definido por

$$\hat{\boldsymbol{\alpha}}^*_{(k)} = (\mathbf{\Lambda} + k\mathbf{I}_p)^{-1}\mathbf{D}^\top\mathbf{r}. \tag{2.32}$$

Assim,

$$\hat{\alpha}^*_{i(k)} = \frac{\lambda_i^{1/2} r_i}{\lambda_i + k} = \frac{\lambda_i}{\lambda_i + k}\hat{\alpha}^*_{i(0)}, \tag{2.33}$$

em que $\hat{\alpha}^*_{i(0)} = \hat{\alpha}_i$, o estimador de mínimos quadrados de α_i. O erro quadrático médio de $\hat{\alpha}^*_{i(k)}$ é dado por:

$$\text{EQM}(\hat{\alpha}^*_{i(k)}) = \frac{\lambda_i\sigma^2 + k^2\alpha_i^2}{(\lambda_i + k)^2}. \tag{2.34}$$

Analisando a forma do estimador $\hat{\alpha}^*_{i(k)}$ é possível perceber que existe a possibilidade de usar diferentes valores de k, digamos k_i, para cada $i = 1, \ldots, p$. Esse método é chamado de regressão em cristas generalizado e será estudado na próxima sub-seção.

2.2.2 Método de Regressão em Cristas Generalizado

A generalização do método de regressão em cristas consiste em, ao invés de usar um único k, usar $\mathbf{K} = \text{diag}(k_i)$ e $k_i > 0$, para $i = 1, \ldots, p$, como discutido em Goldstein e Smith (1974) e mencionado em Hoerl e Kennard (1970a).

Existem dois principais motivos para preferir o uso do método regressão em cristas generalizado, o primeiro é a maior facilidade de encontrar valores ótimos de k_i, ou seja, valores em que o EQM do estimador de regressão em cristas é mínimo. O segundo motivo é, em caso em que os valores de

k_i diferem significativamente um do outro, a possibilidade de um menor erro quadrático médio do que o estimador de regressão em cristas original.

O estimador de regressão em cristas generalizado é dado por

$$\hat{\boldsymbol{\alpha}}^*_{(\mathbf{K})} = (\boldsymbol{\Lambda} + \mathbf{K})^{-1}\mathbf{D}^\top \mathbf{r}, \tag{2.35}$$

e Hoerl e Kennard (1970b) mostraram que o valor de k_i, para $i = 1, \ldots, p$, que minimiza a função do erro quadrático médio é

$$k_i = \frac{\sigma^2}{\alpha_i^2}. \tag{2.36}$$

Entretanto, o valor ótimo para k_i depende de σ^2 e α_i que são desconhecidos e Hoerl e Kennard (1970a) sugerem substitui-los pelos seus respectivos estimadores não viciados, de forma que

$$\hat{k}_i = \frac{\hat{\sigma}^2}{\hat{\alpha}_i{}^2}, \quad i = 1, \ldots, p. \tag{2.37}$$

Walker e Page (2001) denotaram uma expressão para quantidade máxima de redução do EQM do estimador de regressão em cristas em comparação com o do estimador de mínimos quadrados. Essa expressão é dada por

$$\frac{\sigma^4}{\lambda_i(\alpha_i^2\lambda_i + \sigma^2)}. \tag{2.38}$$

É possível ver em (2.38) que a redução no EQM é maior quando λ_i é pequeno, ou seja, situações em que há multicolinearidade das variáveis explicativas. Em casos em que λ_i é grande, a redução no EQM é bem pequena e não há tanta vantagem em usar o estimador pelo método das cristas.

2.3 Método para seleção de variáveis em modelos de Regressão em Cristas

2.3.1 Estatística C_p

A estatística C_P é definida como

$$C_P = \frac{(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}_P)^\top(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}_P)}{\hat{\sigma}^2} - n + 2p. \tag{2.39}$$

em que

$$\hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^\top(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})}{n - p - 1}. \tag{2.40}$$

A quantidade (2.39) é uma medida de adequação de $\hat{\boldsymbol{\beta}}_P$, definido como o vetor de estimativas

de mínimos quadrados em que $P \leq p$ dos elementos são não nulos, e $p-P$ são nulos. O subconjunto P dentre os p elementos do vetor $\hat{\boldsymbol{\beta}}$ é escolhido minimizando (2.39). Mallows (1973) discute melhor sobre C_P e como utilizá-la para selecionar um subconjunto de variáveis em que o estimador pelo método de mínimos quadrados é adequado.

Mallows (1973) também sugeriu uma generalização da estatística C_P, denominada C_L, a qual pode ser aplicada no método de regressão em cristas na forma não generalizada. O trabalho desse autor será discutido na seção seguinte.

2.3.2 Generalização da Estatística C_P

Walker e Page (2001) abordaram uma outra generalização que considera o método de regressão em cristas generalizado discutido em (2.2.2). Considerando o estimador

$$\hat{\boldsymbol{\alpha}}^*_{(\mathbf{K})} = (\boldsymbol{\Lambda} + \mathbf{K})^{-1}\mathbf{D}^\top \mathbf{r} = \mathbf{L}\mathbf{r} \tag{2.41}$$

em que $\mathbf{K}$ é uma matriz diagonal com os os valores de k_i, para $i = 1, \ldots, p$, na sua diagonal principal, uma estimativa da soma dos quadrados dos resíduos é dada por

$$\hat{\zeta}_L = \frac{1}{\hat{\sigma}^2}(\hat{\boldsymbol{\alpha}}^*_{(\mathbf{K})} - \boldsymbol{\alpha})^\top \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top(\hat{\boldsymbol{\alpha}}^*_{(\mathbf{K})} - \boldsymbol{\alpha}) \tag{2.42}$$

Quanto menor for $\hat{\zeta}_L$, melhor será o estimador $\hat{\boldsymbol{\alpha}}^*_{(\mathbf{K})}$ para estimar os valores $\mathbf{Z}\boldsymbol{\alpha}$. Denotando C_L uma estimativa para $\mathrm{E}(\hat{\zeta}_L)$, então, é possível tomar um valor de $\mathbf{K}$ de forma que minimize C_L, garantindo que $\hat{\boldsymbol{\alpha}}^*_{(\mathbf{K})}$ é um bom estimador para $\boldsymbol{\alpha}$.

Temos que uma estimativa para o valor esperado de $\hat{\zeta}_L$ (ver Walker e Page (2001)), é dada por

$$C_L = \frac{1}{\hat{\sigma}^2}\sum_{i=1}^{p}\left(r_i - \lambda_i^{1/2}\hat{\alpha}^*_{i(k)}\right)^2 - n + 2\sum_{i=1}^{p}\frac{\lambda_i}{\lambda_i + k_i}, \tag{2.43}$$

e C_L é minimizado quando

$$k_i = \begin{cases} \frac{\lambda_i\hat{\sigma}^2}{r_i^2 - \hat{\sigma}^2}, & \text{se } r_i^2 > \hat{\sigma}^2 \\ \infty, & \text{se } r_i^2 \leq \hat{\sigma}^2, \end{cases} \tag{2.44}$$

para $i = 1, \ldots, p$.

Portanto, a escolha de k_i que minimiza C_L pode ser justificada em termos de minimização da soma de quadrado dos resíduos ponderada. É possível observar que, quando $k_i = \infty$, como consequência, $\hat{\alpha}^*_{i(k)} = 0$, dessa forma a estimativa do coeficiente da i-ésima variável explicativa é nula e então fazendo automaticamente uma seleção de variáveis.

Esse critério de escolha para k resulta em estimadores para $\hat{\alpha}^*_{i(k)}$ que coincidem com os estimadores Bayesianos empíricos para regressão em cristas, discutidos na próxima seção.

2.4 Regressão em Cristas sob o ponto de vista Bayesiano

Walker e Page (2001) apresentam uma interpretação bayesiana para o estimador das cristas considerando o modelo de regressão em cristas na forma canônica descrito em 2.2.1.

A interpretação dos autores consiste em atribuir distribuições à priori para cada parâmetro α_i, $i = 1, \ldots, p$, para o vetor aleatório $\boldsymbol{\nu}$, e para as observações da variável resposta r_i.

Considerando a suposição de que o $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$, então o modelo Bayesiano para essa distribuição a priori é dado por

$$r_i \sim \begin{cases} \mathcal{N}(\lambda_i^{1/2}\alpha_i, \sigma^2), & \text{se } i = 1, \ldots, p \\ \mathcal{N}(0, \sigma^2), & \text{se } i = p+1, \ldots, n. \end{cases} \tag{2.45}$$

Atribuindo, para todo i, a distribuição à priori $\alpha_i \sim \mathcal{N}(0, \sigma^2_{\alpha_i})$, ver Lindley e Smith (1972), em que $\sigma^2_{\alpha_i} = \sigma^2/\lambda_i$, o objetivo da análise Bayesiana é encontrar a distribuição à posteriori de α_i, para $i = 1, \ldots, p$.

Como as distribuições de r_i e α_i são normais, então a distribuição à posteriori também é normal, a média dessa distribuição é o estimador Bayesiano para α_i dado por

$$\alpha_i^* = \frac{\lambda_i^{1/2} r_i}{\lambda_i + k_i}, \tag{2.46}$$

em que $k_i = \sigma^2/\sigma^2_{\alpha_i}$.

Se a distribuição à priori de α_i fosse $\mathcal{N}(0, \sigma^2_\alpha)$, então todos α_i teriam a mesma distribuição, e $k = \sigma^2/\sigma^2_\alpha$ seriam iguais para qualquer $i = 1, \ldots, p$, coincidindo com o estimador descrito em (2.33). Entretanto, não há razões para supor que a distribuição de todo α_i seja a mesma, justificando o uso do estimador de regressão em cristas generalizado.

É possível ainda determinar k_i utilizando a abordagem de estimadores Bayesianos empíricos, para isso, considere o modelo de regressão em cristas Bayesiano descrito na seção 2.4. Sob as mesmas suposições feitas nesse modelo, é possível encontrar uma expressão para a verossimilhança marginal $f(r_i|k_i)$ e determinar valores de k_i que maximizam essa verossimilhança.

A verossimilhança marginal de r_i dado k_i é

$$f(r_i|k_i) \sim \mathcal{N}\left(0, \sigma^2\left(1 + \frac{\lambda_i}{k_i}\right)\right), \tag{2.47}$$

e, portanto, k_i dado por

$$k_i = \begin{cases} \frac{\lambda_i \sigma^2}{r_i^2 - \sigma^2}, & \text{se } r_i^2 > \sigma^2 \\ \infty, & \text{se } r_i^2 \leq \sigma^2, \end{cases} \tag{2.48}$$

usando a propriedade da invariância e substituindo σ^2 pelo seu estimador $\hat{\sigma}^2$, o estimador de máxima verossimilhança para k_i é dado por

$$\hat{k}_i = \frac{\lambda_i \hat{\sigma}^2}{\hat{\phi}_i^2 - \hat{\sigma}^2} = \frac{\lambda_i \hat{\sigma}^2}{r_i^2 - \hat{\sigma}^2}, \tag{2.49}$$

em que $\phi_i^2 = \hat{\sigma}^2(1 + \lambda_i/k_i)$. O estimador de regressão em cristas para α_i dado $\hat{k}_i$ é

$$\hat{\alpha}_i^* = \frac{\lambda_i \hat{\alpha}_i}{\lambda_i + \frac{\lambda_i \hat{\sigma}^2}{r_i^2 - \hat{\sigma}^2}} = \frac{(r_i^2 - \hat{\sigma}^2)\hat{\alpha}_i^2}{r_i^2 - \hat{\sigma}^2 + \hat{\sigma}^2} = \hat{\alpha}_i - \frac{\hat{\sigma}^2}{r_i \lambda_i^{1/2}} = \frac{r_i^2 - \hat{\sigma}^2}{r_i \lambda_i^{1/2}}. \tag{2.50}$$

Note que $\hat{k}_i > 0$ quando $r_i^2 > \hat{\sigma}^2$. Se $r_i^2 \leq \sigma^2$, a verossimilhança é maximizada à medida que $\hat{k}_i \to \infty$, satisfazendo o critérios de Regressão em Cristas de que k_i seja positivo.

Assim, obtemos as seguintes estimativas para os coeficientes da regressão em cristas

$$\hat{\alpha}_i^* = \begin{cases} \frac{r_i^2 - \hat{\sigma}^2}{r_i \lambda_i^{1/2}}, & \text{se } r_i^2 > \hat{\sigma}^2 \\ 0, & \text{se } r_i^2 \leq \hat{\sigma}^2. \end{cases} \tag{2.51}$$

Dessa forma, o estimador Bayesiano em modelos de Regressão em Cristas generalizados na forma canônica são capazes de fazer seleção de variáveis explicativas, isso acontece quando a estimativa do coeficiente α_i é 0.

2.5 Outros tópicos em Regressão em Cristas

Um outro tópico não discutido nessa dissertação mas que vem sido bastante desenvolvido é a análise de influência quando é utilizado o estimador de Regressão em Cristas. Uma análise mais detalhada pode ser encontrada em Oikawa (2008) e Oikawa e Elian (2012).

Mason e Gunst (1985) alertam para o fato que muitas vezes, a existência de pontos discrepantes no conjunto de dados pode gerar multicolinearidade. Assim, a determinação da natureza da multicolinearidade é fundamental.

Walker e Birch (1988) propõe inúmeras medidas de de influência para a análise de Regressão em Cristas. Quando o estimador de Regressão em Cristas $\hat{\beta}_{(k)}^*$ definido em (2.10) é utilizado, o vetor de valores ajustados é dado por

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{W}\mathbf{X}^\top \mathbf{y}, \tag{2.52}$$

de modo que a matriz

$$\mathbf{H}^* = \mathbf{X}\mathbf{W}\mathbf{X}^\top \tag{2.53}$$

é a matriz "chapéu"e a previsão associada ao i-ésimo elemento amostral é da forma

$$\hat{y}_i^* = \sum_{j=1}^{n} h_{ij}^* y_j, \tag{2.54}$$

em que h_{ij}^* é o elemento de ordem ij de $\mathbf{H}^*$.

Utilizando a decomposição de valor singular da matriz $\mathbf{X}$ dada em (2.28), os autores obtém o valor h_{ii}^*, conhecido como ponto de alavanca,

$$h_{ii}^* = \sum_{j=1}^{p+1} \frac{\lambda_j}{\lambda_j + k} u_{ij}^2, \tag{2.55}$$

em que u_{ij} é o elemento de ordem ij da matriz $\mathbf{U}$.

Observa-se que, para $k > 0$, $h_{ii}^* < h_{ii}$, sendo h_{ii} o valor da alavanca associado ao estimador de mínimos quadrados. Verifica-se ainda que este valor cai à medida que k aumenta.

Por outro lado, o resíduo do modelo de Regressão em Cristas é dado por

$$e_i^* = e_i + k \sum_{j=1}^{n} y_j \left[\sum_{m=1}^{p} \frac{u_{im} u_{jm}}{\lambda_m + k} \right], \tag{2.56}$$

sendo que, a segunda parcela de e_i^* pode ser tanto positiva quanto negativa, de modo que o resíduo em cristas pode ser maior ou menor que o correspondente de mínimos quadrados.

São ainda definidos em Walker e Birch (1988) versões das distância de Cook e DFFITs para o contexto de Regressão em Cristas.

Billor e Loynes (1993), escrevendo o estimador em cristas como um estimador de máxima pseudo-verossimilhança, desenvolvem uma análise de Regressão em Cristas.

Finalizando, o método robusto de estimação em cristas foi introduzido em Lawrence e Marsh (1984), como alternativa à retirada de observações discrepantes.

2.6 Comentários sobre Regressão em Cristas

O método de Regressão em Cristas parece ser apropriado em situações em que há a presença de multicolinearidade nos dados. As propriedades dos estimadores pelo métodos das cristas destacadas nesse capítulo são desenvolvidas assumindo que k é não-estocástico. Entretanto, é preciso definir qual valor de k a ser utilizado e isso é feito através dos dados, e dessa forma k é uma quantidade estocástica.

Alguns estudos de simulação mostram que mesmo k sendo estocástico, o método da Regressão em Cristas traz melhoria no erro quadrático médio comparado ao estimador de mínimos quadrados.

A redução no erro quadrático médio quando utilizado o método de Regressão em Cristas depende da orientação do vetor $\boldsymbol{\beta}$. A maior redução acontece quando $\boldsymbol{\beta}$ coincide com o autovetor correspondente ao maior autovalor da matriz $\mathbf{C}$.

Outro fator que influencia na redução do EQM é a escolha da quantidade k a partir dos dados. Existem diversas formas de escolher k e algumas delas serão discutidas no Capítulo 3.

Capítulo 3

Critérios para seleção de k

3.1 Introdução

Nessa seção serão abordadas as diferentes formas de obtenção de k. Devido à dificuldade de comprovar os resultados teoricamente, os estimadores para k foram comparados em vários estudos de simulação na literatura, como por exemplo Hoerl *et al.* (1975), Lawless e Wang (1976), McDonald e Galarneau (1975), Kibria (2003).

Uma das primeiras formas de determinar k foi através do traço da crista. O traço da crista consiste em um gráfico dos valores estimados dos parâmetros e os valores para k. A ideia geral é usar k tal que as estimativas dos coeficientes estejam estabilizadas, para mais detalhes sobre o traço da cristas ver Hoerl e Kennard (1970a).

3.2 Escolha de k através dos dados

Hoerl e Kennard (1970a) sugerem que k seja dado por

$$\hat{k}_{HK} = \frac{\hat{\sigma}^2}{\hat{\alpha}^2_{\max}}, \tag{3.1}$$

em que $\hat{\alpha}_{\max}$ é o máximo elemento de $\hat{\boldsymbol{\alpha}}$. A justificativa para esse estimador é que, quando σ^2 e α são conhecidos, $\hat{k}_{\text{HK}}$ será o estimador que resultará em um menor EQM do que o estimador de mínimos quadrados, em situações de multicolinearidade.

Os mesmos autores sugeriram o método do traço da crista. Esse método é subjetivo e pode fornecer mais de um valor para k por ser tratar de uma análise gráfica que tem como objetivo encontrar intervalos em que as estimativas dos coeficientes através do método de Regressão em Cristas são mais estáveis.

O traço da crista é o gráfico dos coeficientes $\hat{\beta}_{i(k)}$, para $i = 1, \ldots, p$ contra diversos valores para k. Para cada $\hat{\beta}_{i(k)}$ temos uma curva que se iniciará em $\hat{\beta}_i$, quando $k = 0$, e a curva tende a 0 quando $k \to \infty$. Ao analisar o gráfico, é possível perceber o comportamento das estimativas dos coeficientes com relação à pertubação feita na diagonal de $\mathbf{C}$. Segundo Hoerl e Kennard (1970b), boas práticas para escolha de k através do método do traço da crista são quando os coeficientes já se estabilizaram e terão comportamento semelhante ao de um sistema ortogonal, ou seja, decrescem suavemente a medida que k aumenta.

Hoerl *et al.* (1975) propuseram um estimador para k usando a média harmônica de $\hat{k}_i$ em (2.37), isso é

$$\hat{k}_{\text{HKB}} = \frac{p\hat{\sigma}^2}{\sum_{i=1}^{p}\hat{\alpha}_i^2} = \frac{p\hat{\sigma}^2}{\hat{\boldsymbol{\alpha}}^\top\hat{\boldsymbol{\alpha}}} = \frac{p\hat{\sigma}^2}{\hat{\boldsymbol{\beta}}^\top\hat{\boldsymbol{\beta}}}. \tag{3.2}$$

Esse critério apresentou bons resultados em um estudo de simulação dos mesmo autores, quando comparado em termos de EQM com o estimador obtido através do método de mínimos quadrados. Entretanto, em outros trabalhos comparativos, não apresentou o mesmo rendimento, sendo considerado como um critério que deve ser melhor estudado antes de ser aplicado, ver Wichern e Churchill (1978).

O principal problema desse critério é o fato de que às vezes $\hat{\boldsymbol{\beta}}^\top\hat{\boldsymbol{\beta}}$ pode ser muito grande devido à multicolinearidade, fazendo com que $\hat{k}_{\text{HKB}}$ seja muito pequeno, e consequentemente $\hat{\boldsymbol{\beta}}^*(\hat{k}_{\text{HKB}})$ seja próximo de $\hat{\boldsymbol{\beta}}$. Essa afirmação tem como base o resultado

$$E(\hat{\boldsymbol{\beta}}^\top\hat{\boldsymbol{\beta}}) = \sigma^2\sum_{i=1}^{p} 1/\lambda_i + \boldsymbol{\beta}^\top\boldsymbol{\beta},$$

visto em (2.6).

Definindo Q como

$$Q = \hat{\boldsymbol{\beta}}^\top\hat{\boldsymbol{\beta}} - \hat{\sigma}^2\sum_{i=1}^{p}\frac{1}{\lambda_i}, \tag{3.3}$$

McDonald e Galarneau (1975) propuseram escolher $\hat{k}_{\text{MDG}}$ tal que

$$\begin{aligned} \hat{\boldsymbol{\beta}}^*(\hat{k}_{\text{MDG}})^\top\hat{\boldsymbol{\beta}}^*(\hat{k}_{\text{MDG}}) = Q, \quad &\text{se} \quad Q > 0 \\ \hat{\boldsymbol{\beta}}^*(\hat{k}_{\text{MDG}}) = \hat{\boldsymbol{\beta}}, \quad &se \quad Q \leq 0. \end{aligned} \tag{3.4}$$

Desta forma, se através dos dados encontramos evidências que $\hat{\boldsymbol{\beta}}$ superestima $\boldsymbol{\beta}$, ou seja, $\hat{\boldsymbol{\beta}}^\top\hat{\boldsymbol{\beta}}$ é muito maior que $\boldsymbol{\beta}^\top\boldsymbol{\beta}$, então $\hat{k}_{\text{MDG}}$ pode ser um bom valor para k. Esse critério apresentou bons resultados nos trabalhos de McDonald e Galarneau (1975) e Wichern e Churchill (1978) em casos em que a multicolinearidade é alta.

Lawless e Wang (1976) propuseram um estimador que foi motivado pela interpretação Bayesiana da Regressão em Cristas. Como discutido na Seção 2.4, se considerar a princípio a distribuição à priori de $\boldsymbol{\alpha}$ como sendo normal multivariada com vetor de médias $\mathbf{0}$ e matriz de covariâncias $\sigma_\alpha^2\mathbf{I}_p$, então o estimador Bayesiano para cada α_i é dado em (2.46).

As quantidades σ^2 e σ_α^2 são desconhecidas, e precisam ser estimadas. Se os estimadores para essas quantidades forem, respectivamente, $\hat{\sigma}^2$ e $\sum_{i=1}^{p}\hat{\alpha}_i^2/p$, então o estimador para k coincidiram com $\hat{k}_{\text{HKB}}$. Entretanto, Lawless e Wang (1976) propuseram a seguinte estimativa para k

$$\hat{k}_{\text{LW}} = \frac{p\hat{\sigma}^2}{\sum_p^{i=1} \lambda_i \hat{\alpha}_i^2}, \tag{3.5}$$

em que $\hat{\boldsymbol{\alpha}} = \mathbf{V}^\top \hat{\boldsymbol{\beta}}$ e $\mathbf{V}$ é a matriz de autovetores de $\mathbf{C}$. Os autores também realizaram um estudo de simulação com as mesmas características do estudo de Hoerl *et al.* (1975), e concluíram que apesar de $\hat{k}_{\text{LW}}$ apresentar resultados um pouco melhores que $\hat{k}_{\text{HKB}}$, ambos os estimadores para k apresentam melhorias no erro quadrático médio comparados ao estimador de mínimos quadrados em situações com presença de multicolinearidade.

Outro método para encontrar um valor para k através dos dados, o procedimento de validação cruzada funciona da seguinte forma: suponha que $\hat{\boldsymbol{\beta}}_{(k)}^{(i)*}$ é o estimador para $\boldsymbol{\beta}$ pelo método de Regressão em Cristas com a i-ésima observação y_i omitida, e se k escolhido for uma boa escolha, então o i-ésimo elemento de $\mathbf{X}\hat{\boldsymbol{\beta}}_{(k)}^{(i)*}$, digamos $[\mathbf{X}\hat{\boldsymbol{\beta}}_{(k)}^{(i)*}]_i$, é um bom preditor para y_i. Portanto, o estimador para k através da validação cruzada é $\hat{k}_{\text{vc}}$ que minimiza

$$\text{CV}(k) = \frac{1}{n}\sum_{i=1}^{n}([\mathbf{X}\hat{\boldsymbol{\beta}}_{(k)}^{(i)*}]_i - y_i)^2. \tag{3.6}$$

Kibria (2003) propôs três novos métodos para estimar o valor de k. Os métodos consistem em utilizar a média aritmética, média geométrica e a mediana de $\hat{k}_i$ em (2.37). Os estimadores são

$$\hat{k}_{\text{ma}} = \frac{1}{p}\sum_{i=1}^{p}\frac{\hat{\sigma}^2}{\hat{\alpha}_i^2} \tag{3.7}$$

$$\hat{k}_{\text{mg}} = \frac{\hat{\sigma}^2}{(\prod_{i=1}^{p}\hat{\alpha}_i^2)^{1/p}} \tag{3.8}$$

$$\hat{k}_{\text{med}} = \text{mediana}\left[\frac{\hat{\sigma}^2}{\hat{\alpha}_i^2}\right], \quad i = 1, 2, \ldots, p. \tag{3.9}$$

Em estudos de simulação, o autor obteve resultados em que $\hat{k}_{\text{mg}}$ e $\hat{k}_{\text{LW}}$ tiveram performances melhores que $\hat{k}_{\text{HKB}}$. O estimador $\hat{k}_{\text{mg}}$ teve melhor resultados quando o número de condição era pequeno ou moderado e $\hat{k}_{\text{LW}}$ teve melhores resultados quando o número condição era grande.

Khalaf e Shukur (2005) sugeriram uma modificação de $\hat{k}_{\text{HK}}$ denotada por

$$\hat{k}_{KS} = \frac{\lambda_{\max}\hat{\sigma}^2}{(n-p)\hat{\sigma}^2 + \lambda_{\max}\hat{\alpha}_{\max}^2}, \tag{3.10}$$

em que $\lambda_{\max}$ é o máximo autovalor da matriz $\mathbf{C}$, e $\hat{\alpha}_{\max}^2$ é o quadrado do valor máximo de $\hat{\boldsymbol{\alpha}}$. A ideia por trás desse estimador é separar a variação causada pelo tamanho da amostra n da causada pelo grau de multicolinearidade. Os autores também realizaram estudos de simulação comparando esse estimador com $\hat{k}_{\text{HK}}$ e o também com o estimador de mínimos quadrados, $\hat{k} = 0$. Os resultados obtidos mostram que o estimador proposto por Khalaf e Shukur (2005) era melhor que $\hat{k}_{\text{HK}}$ e $\hat{k} = 0$ em termos de erro quadrático médio e inclusive era mais robusto quando σ^2 era grande.

Com base nos trabalhos de Kibria (2003) e Khalaf e Shukur (2005), Alkhamisi *et al.* (2006)

propuseram os seguintes estimadores para k

$$\hat{k}_{\text{ma}}^{\text{AKS}} = \frac{1}{p}\sum_{i=1}^{p}\frac{\lambda_i\hat{\sigma}_i^2}{(n-p)\hat{\sigma}_i^2+\lambda_i\hat{\alpha}_i^2} \tag{3.11}$$

$$\hat{k}_{\text{max}}^{\text{AKS}} = \max\left[\frac{\lambda_i\hat{\sigma}_i^2}{(n-p)\hat{\sigma}_i^2+\lambda_i\hat{\alpha}_i^2}\right] \tag{3.12}$$

$$\hat{k}_{\text{med}}^{\text{AKS}} = \text{mediana}\left[\frac{\lambda_i\hat{\sigma}_i^2}{(n-p)\hat{\sigma}_i^2+\lambda_i\hat{\alpha}_i^2}\right], \quad i=1,2,\ldots,p. \tag{3.13}$$

Os autores compararam esses estimadores com $\hat{k}_{\text{HK}}$ em estudo de simulação realizado considerando diferentes distribuições para o erro do modelo. Os resultados desse estudo mostraram uma pequena vantagem de $\hat{k}_{\text{max}}^{\text{AKS}}$ comparado a $\hat{k}_{\text{ma}}^{\text{AKS}}$, $\hat{k}_{\text{med}}^{\text{AKS}}$ e $\hat{k}_{\text{HK}}$ em termos de EQM.

Dorugade e Kashid (2010) sugeriram um estimador para k baseado em (3.2) dado por

$$\hat{k}_{\text{DK}} = \max\left(0, \hat{k}_{\text{HKB}} - \frac{1}{n(\text{FIV}_i)_{\text{max}}}\right), \tag{3.14}$$

em que $(\text{FIV}_i)_{\text{max}}$ é o máximo FIV considerando todas as variáveis explicativas no modelo. Esse método de estimação de k apresentou bons resultados no estudo de simulação feito por Dorugade e Kashid (2010). Outros critérios de escolha de $\hat{k}$ podem ser vistos em Hocking *et al.* (1976) e Nomura (1988).

3.3 Comentários sobre os critérios de escolha de $\hat{k}$

Em geral, os estudos de simulação mostram que, em situações em que a matriz $\mathbf{C}$ é mal condicionada, qualquer método de escolha de $\hat{k}$ é superior ao método de mínimos quadrados em termos de erro quadrático médio.

Os estudos mostram que mesmo com k estocástico, é possível ter melhores resultados utilizando o método de Regressão em Cristas. Entretanto, os resultados também depende de outras quantidades como σ^2, o grau de multicolinearidade, o tamanho da amostra n e também a orientação do vetor $\boldsymbol{\beta}$.

O estudo de simulação de Kibria (2003) mostra que quando valer a relação $\boldsymbol{\beta}^\top\boldsymbol{\beta}/\sigma^2 \geq 10000$, o método de mínimos quadrados apresenta melhores resultados, caso contrário, é recomendado usar o método de Regressão em Cristas se o número de condição for maior que 100.

Firinguetti-Limone e Pereira-Barahona (2019) utilizam a abordagem bayesiana para obter estimativas para o parâmetro k.

Admitindo um modelo bayesiano hierárquico para os parâmetros $(\boldsymbol{\beta}|\sigma^2,k)$, $(\sigma^2|k)$ e k, os autores obtêm a distribuição a posteriori de k. No entanto, devido à grande complexidade de sua expressão, a média dessa distribuição a posteriori só pode ser calculada através de procedimentos numéricos.

No próximo capítulo, serão apresentadas técnicas de inferências para o vetor $\boldsymbol{\beta}$ construídas utilizando o estimador pelo método de Regressão em Cristas. Quase todas as técnicas consideram k como quantidade não-estocástica e algumas abordam k como estocástico.

Capítulo 4

Inferência em modelos de Regressão em Cristas

4.1 Introdução

Inferência Estatística é o procedimento de tirar conclusões sobre as propriedades ou sobre os parâmetros de uma distribuição dessa população. Esses parâmetros são geralmente desconhecidos, e por isso há a necessidade de procedimentos de inferência. A inferência estatística consiste em ir além estimativas pontuais, como testar hipóteses ou criar intervalos de confiança para os parâmetros.

O estimador de mínimos quadrados em modelos de Regressão Linear, $\hat{\boldsymbol{\beta}}^*_{(0)} = \hat{\boldsymbol{\beta}}$, é o estimador não viciado de variância uniformemente mínima (ENVVUM) para $\boldsymbol{\beta}$. Inferências para esse estimador são construídas através da quantidade pivotal dada por

$$t_{(0)} = \frac{\hat{\beta}^*_{i(0)} - \beta_i}{S\left(\hat{\beta}^*_{i(0)}\right)}, \tag{4.1}$$

em que $\hat{\beta}^*_{i(0)}$ é o i-ésimo componente do vetor $\hat{\boldsymbol{\beta}}^*_{(0)}$ e $S(\hat{\beta}^*_{i(0)}) = \sqrt{\hat{\sigma}^2 c_{ii}^{-1}}$, nos quais c_{ii}^{-1} é o i-ésimo elemento da diagonal principal da matriz $\mathbf{C}^{-1}$ e $\hat{\sigma}^2$ é definido em (2.40). Além disso, a medida $S^2(\hat{\boldsymbol{\beta}}^*_{i(0)})$ em (4.1) pode ser reescrita como

$$S^2(\hat{\boldsymbol{\beta}}^*_{i(0)}) = \hat{\sigma}^2 \sum_{j=1}^{p} \mathrm{v}_{ij}^2/\lambda_{jj}, \tag{4.2}$$

em que v_{ij} representa o elemento da i-ésima linha e j-ésima coluna da matriz $\mathbf{V}$ de autovetores de $\mathbf{C}$ e λ_{jj} é o j-ésimo elemento da diagonal principal de $\mathbf{\Lambda}$, a matriz diagonal com os autovalores de $\mathbf{C}$.

Por $\mathbf{C}$ se tratar de matriz de correlação associada às variáveis correspondentes às colunas de $\mathbf{X}$, seus autovetores têm a propriedade $|\mathrm{v}_{ij}| < 1$ para quaisquer i e j, e portanto o valor de $S(\hat{\boldsymbol{\beta}}^*_{i(0)})$ será grande se algum λ_{jj} for próximo de 0. Consequentemente, (4.1) será pequeno e não será capaz de detectar o distanciamento de $\hat{\boldsymbol{\beta}}^*_{i(0)}$ de β_i de forma precisa, implicando em inferências errôneas.

Em casos de mau condicionamento de C temos λ_{ii} bem próximos de 0, e portanto a quantidade

pivotal $t_{(0)}$ não é apropriada para conduzir inferência para os parâmetros β_i, para todo $i = 1, \ldots, p$.

Nesse capítulo iremos considerar técnicas de inferência construídas com base no método das cristas por serem mais adequadas em situação de multicolinearidade. Existe uma série de artigos na literatura que discutem estimação pontual em Regressão em Cristas, entretanto poucos abordam o assunto de inferência para o vetor de coeficientes $\boldsymbol{\beta}$. Alguns desses artigos, como Coutsourides *et al.* (1979), Obenchain (1977), Ohtani (1985), Hoerl e W. Kennard (1990) e Halawa e El Bassiouni (2000), serão discutidos nas próximas seções.

4.2 Estatísticas $t_{(k)}$ e $F_{(k)}$

Baseando-se na quantidade pivotal descrita em (4.1) e considerando que o estimador pelo método das cristas $\hat{\beta}^*_{i(k)}$ é um estimador viciado para β_i, para $k > 0$, Obenchain (1977) derivou uma estatística sob a hipótese nula, $H_0 : \boldsymbol{\beta} = 0$. A estatística , denotada por $t_{(k)}$, e dada por

$$t_{(k)} = \frac{(\hat{\boldsymbol{\beta}}^*_{i(k)} - b_i)}{\sqrt{\widehat{\mathrm{Var}}(\hat{\boldsymbol{\beta}}^*_{i(k)} - b_i)}}, \tag{4.3}$$

em que $\widehat{\mathrm{Var}}(\hat{\boldsymbol{\beta}}^*_{i(k)} - b_i)$ é um estimador não viciado da variância de $(\hat{\boldsymbol{\beta}}^*_{i(k)} - b_i)$ e b_i é definido por

$$b_i = \mathbf{v}_i^\top \boldsymbol{\Delta} \mathbf{V} \left[\mathbf{I}_v - \mathbf{C}^{-1} \mathbf{e}_i^\top (\mathbf{e}_i \mathbf{C}^{-1} \mathbf{e}_i{}^\top)^{-1} \mathbf{e}_i \right] \hat{\boldsymbol{\beta}}^*_{(0)}, \tag{4.4}$$

em que $\mathbf{v}_i$ é a i-ésima linha da matriz $\mathbf{V}$, $\boldsymbol{\Delta}$ é uma matriz diagonal dos autovalores da matriz $\mathbf{Z}$ definida em (2.10), ou seja, com os elementos $(\xi_{\mathbf{Z}})_i = \lambda_i/(\lambda_i + k)$ na diagonal principal, e $\mathbf{e}_i$ é a i-ésima linha da matriz identidade de ordem p, para $i = 1, \ldots, p$. O autor mostra que a quantidade escalar b_i é o estimador não viciado de variância mínima de $\mathrm{E}(\hat{\boldsymbol{\beta}}^*_{i(k)})$ sob a hipótese nula $H_0 : \boldsymbol{\beta} = 0$.

Obenchain (1977) também mostra que $(\hat{\boldsymbol{\beta}}^*_{i(k)} - b_i)$ é uma quantidade múltipla de $\hat{\boldsymbol{\beta}}^*_{i(0)}$ e como consequência $t_{(k)}$ reduz-se a $t(0)$ que tem distribuição t com $n-p-1$ graus de liberdade, denotada por $t_{(n-p-1)}$.

O autor conclui que apesar do estimador pelo método de Regressão em Cristas ser viciado, as técnicas inferenciais para o método de mínimos quadrados são válidas para o método da Regressão em Cristas. Os resultados de Coutsourides *et al.* (1979) são similares mas foram estendidos para outros estimadores da classe dos estimadores viciados.

Além de inferência para um único elemento de $\boldsymbol{\beta}$, β_i, para $i = 1, \ldots, p$, baseada na quantidade pivotal t, também é possível inferir sobre o vetor $\boldsymbol{\beta}$ fazendo o uso da estatística F. Para o estimador pelo método de mínimos quadrados, essa estatística e bem conhecida na literatura e é definida por

$$F = \frac{(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^\top \mathbf{C} (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})}{p\hat{\sigma}^2} \sim F_{(p,n-p-1)}, \tag{4.5}$$

em que $F_{(p,n-p-1)}$ representa a distribuição F com p e $n-p-1$ graus de liberdade.

Já para o estimador pelo método de regressão em Cristas, Ullah *et al.* (1984) definiram a estatística F aperfeiçoada, denotada por $F_{(k)}$, que consiste em usar o estimador $\hat{\boldsymbol{\beta}}^*_{(k)}$ ao invés de $\hat{\boldsymbol{\beta}}$.

Dessa forma, $F_{(k)}$ é dada por

$$F_{(k)} = \frac{(\hat{\boldsymbol{\beta}}^*_{(k)} - \boldsymbol{\beta})^\top \mathbf{C} (\hat{\boldsymbol{\beta}}^*_{(k)} - \boldsymbol{\beta})}{p\hat{\sigma}^2}. \tag{4.6}$$

Os autores também obtiveram uma distribuição assintótica para $F_{(k)}$ para valores pequenos de σ^2, que no entanto, do ponto de vista inferencial, é de difícil aplicação.

Ohtani (1985) derivou os limites inferior e superior para $F_{(k)}$ sob a hipótese nula, ou seja, considerando que todos os coeficientes de regressão do modelo são zero. Esses limites foram derivados considerando o modelo de regressão linear em sua forma canônica discutido em (2.30), na qual $F_{(k)}$ pode ser escrita da seguinte forma

$$F_{(k)} = \frac{1}{p\hat{\sigma}^2} \sum_{i=1}^{p} (\hat{\alpha}_{i(k)} - \alpha_i)^2 \lambda_i, \tag{4.7}$$

em que $\hat{\sigma}^2$ é o mesmo definido em (2.40).

É sob a hipótese nula que há interesse em conhecer o limite inferior e o limite superior para a estatística. No modelo em forma canônica, temos que $\boldsymbol{\alpha} = \mathbf{V}^\top \boldsymbol{\beta}$, na qual a matriz $\mathbf{V}$, definida em (2.28), é uma matriz ortogonal, o que implica que as hipóteses $H_0 : \boldsymbol{\beta} = 0$ e $H_0^* : \alpha = 0$ são equivalentes.

Sob a hipótese H_0^*, o estimador $\hat{\alpha}_i$ tem distribuição normal com média 0 e variância σ^2/λ_i, com $i = 1, \ldots, p$. Além disso, para $i \neq j$, $\hat{\alpha}_i$ e $\hat{\alpha}_j$ são mutualmente independentes. Portanto, ainda sob a hipótese nula, a estatística $F_{(k)}$ pode ser definida como

$$\begin{aligned} F_{(k)} &= \frac{1}{p\hat{\sigma}^2} \sum_{i=1}^{p} \hat{\alpha}^2_{i(k)} \lambda_i \\ &= \frac{1}{p\hat{\sigma}^2} \sum_{i=1}^{p} \left(\frac{\lambda_i \hat{\alpha}_{i(0)}}{\lambda_i + k} \right)^2 \lambda_i \\ &= \frac{1}{p\hat{\sigma}^2} \sum_{i=1}^{p} \frac{\lambda_i^3 \hat{\alpha}^2_{i(0)}}{(\lambda_i + k)^2}, \end{aligned}$$

a obtenção dessa quantidade se dá pela substituição de $\hat{\alpha}_{i(k)}$ como descrito em (2.33). Além disso, a quantidade k será substituída pelo estimador proposto por Hoerl *et al.* (1975) e dado por $\hat{k} = p\hat{\sigma}^2 / \sum_{j=1}^{p} \hat{\alpha}^2_{j(0)}$, e então, substituindo na estatística, temos

$$F_{(\hat{k})} = \frac{1}{p\hat{\sigma}^2} \sum_{i=1}^{p} \frac{\lambda_i^3 \hat{\alpha}^2_{i(0)}}{\left(\lambda_i + \frac{p\hat{\sigma}^2}{\sum_{j=1}^{p} \hat{\alpha}^2_{j(0)}} \right)^2}. \tag{4.8}$$

Limite Superior

Para construir um limite superior para $F_{(\hat{k})}$ sob H_0^*, será definida a quantidade $\delta_i = \lambda_i/(\lambda_i + \frac{p\hat{\sigma}^2}{\sum_{j=1}^p \alpha_{j(0)}^2})$. Como λ_i corresponde aos autovalores da matriz $\mathbf{C}$, então, $\lambda_i > 0$ para todo $i = 1, \ldots, p$.

Portanto, é possível constatar que $0 < \delta_i < 1$ e por consequência $0 < \delta_i^2 < 1$, logo temos que

$$\begin{aligned} F_{(\hat{k})} &= \sum_{i=1}^{p} \frac{\delta_i \lambda_i \alpha_{i(0)}^2}{p\hat{\sigma}^2} \\ &< \sum_{i=1}^{p} \frac{\lambda_i \hat{\alpha}_{i(0)}^2}{p\hat{\sigma}^2}. \end{aligned} \tag{4.9}$$

Definido esse limite para $F_{(\hat{k})}$, é necessário então definir valores de referência para essa quantidade. Fazendo $\hat{\eta}_i = \lambda_i^{1/2}\hat{\alpha}_{i(0)}/\sigma$, implica que essa quantidade segue uma distribuição qui-quadrado com um grau de liberdade, ou seja, $\hat{\eta}_i^2 \sim \chi_1^2$e à vista disso, temos

$$\sum_{i=1}^{p} \lambda_i \hat{\alpha}_{i(0)}^2/\sigma^2 = \sum_{i=1}^{p} \hat{\eta}_i^2 \sim \chi_p^2 \tag{4.10}$$

$$(n-p-1)\hat{\sigma}^2/\sigma^2 \sim \chi_{(n-p-1)}^2. \tag{4.11}$$

As quantidades (4.10) e (4.11) são independentes e portanto, valores de referência para o limite superior de $F_{(\hat{k})}$ são encontrados através de

$$F_{(\hat{k})} < (n-p-1)\chi_p^2/p\chi_{(n-p-1)}^2 \sim F_{(p,n-p-1)}, \tag{4.12}$$

em que $F_{(p,n-p-1)}$ representa a distribuição F com graus de liberdade p e $(n-p-1)$.

Limite Inferior

Para definir um limite inferior para $F_{(\hat{k})}$, é necessário usar o fato de que a soma dos quadrados de todos os coeficientes $\hat{\alpha}_{j(0)}$ é maior que o quadrado de cada coeficiente um a um, para $j = 1, \ldots, p$, ou seja, $\sum_{j=1}^p \hat{\alpha}_{j(0)}^2 \geq \hat{\alpha}_{i(0)}^2$, para qualquer $i = 1, \ldots, p$.

Dessa forma, a seguinte inequação é válida

$$\frac{\lambda_i^3 \hat{\alpha}_{i(0)}^2}{\left(\lambda_i + \frac{p\hat{\sigma}^2}{\sum_{j=1}^p \hat{\alpha}_{j(0)}^2}\right)^2} \geq \frac{\lambda_i^3 \hat{\alpha}_{i(0)}^2}{\left(\lambda_i + \frac{p\hat{\sigma}^2}{\hat{\alpha}_{i(0)}^2}\right)^2}, \quad i = 1, \ldots, p. \tag{4.13}$$

Logo, utilizando a inequação (4.13), temos

$$F_{(\hat{k})} \geq \sum_{i=1}^{p} \frac{(\lambda_i \hat{\alpha}_{i(0)}^2)^3}{\left[p\hat{\sigma}^2(\lambda_i \hat{\alpha}_{i(0)}^2 + p\hat{\sigma}^2)^2\right]},$$

que é um limite inferior para $F_{(\hat{k})}$.

O passo seguinte é encontrar valores de referência para essa quantidade, para isso usaremos o

fato que as quantidades $\lambda_i\hat{\alpha}^2_{i(0)}/\sigma^2 \sim \chi^2_1$ e $(n-p-1)\hat{\sigma}^2/\sigma^2 \sim \chi^2_{n-p-1}$, então

$$\begin{aligned} F_{(\hat{k})} &\geq \sum_{i=1}^{p} \frac{(\chi^2_1)^3}{\left[\left(\chi^2_1 + \frac{p\chi^2_{n-p-1}}{(n-p-1)}\right)^2 \frac{p\chi^2_{n-p-1}}{(n-p-1)}\right]} \\ &= \sum_{i=1}^{p} \frac{\left[\frac{(n-p-1)\chi^2_1}{\chi^2_{n-p-1}}\right]^3}{p\left[p + \frac{(n-p-1)\chi^2_1}{\chi^2_{n-p-1}}\right]^2}. \end{aligned}$$

As distribuições χ^2_1 e χ^2_{n-p-1} são independentes, então $\frac{(n-p-1)\chi^2_1}{\chi^2_{n-p-1}} \sim F_{(1,n-p-1)}$ e valores de referência para o limite inferior para $F_{(\hat{k})}$ são obtidos através da seguinte relação

$$F_{(\hat{k})} \geq \sum_{i=1}^{p} \frac{(F_{(1,n-p-1)})^3}{p\left(p + F_{(1,n-p-1)}\right)^2}. \tag{4.14}$$

Denotando $\psi = (F_{(1,n-p-1)})^3 / \left[p\left(p + F_{(1,n-p-1)}\right)^2\right]$, então se $\psi \geq c$, c uma constante qualquer, portanto é válido que $\sum_{i=1}^{p} \psi \geq pc$, logo é possível fazer a seguinte dedução

$$P(F_{(\hat{k})} \geq pc) \geq P\left(\sum_{i=1}^{p} \psi \geq pc\right) \geq P\left(\psi \geq c\right). \tag{4.15}$$

Verifica-se que ψ é uma função monótoma crescente de $F_{1,n-p-1}$ e assim, é valido que

$$P\left(\psi \geq \frac{(F_{(\alpha;(1,n-p-1))})^3}{\left(p + F_{(\alpha;(1,n-p-1))}\right)^2}\right) = P\left(F_{(1,n-p-1)} \geq F_{(\alpha;(1,n-p-1))}\right), \tag{4.16}$$

em que $F_{(\alpha;(1,n-p-1))}$ é o valor crítico da distribuição F com 1 e $n-p-1$ graus de liberdades correspondendo ao nível de significância α. Portanto, o limite inferior para $F_{(\hat{k})}$ pode ser calculado pela distribuição F com graus de liberdade 1 e $(n-p-1)$ respectivamente.

Procedimento para teste

O procedimento para testar a hipótese $H_0^* : \boldsymbol{\alpha} = 0$ (ou equivalente $H_0 : \boldsymbol{\beta} = 0$) com um nível de significância α e usando a estatística $F_{(\hat{k})}$ é descrito da seguinte forma

$$\begin{cases} \text{Rejeitar } H_0^*, & \text{se } F_{(\hat{k})} \geq F^{\alpha}_{p,n-p-1}, \\ \text{Não rejeitar } H_0^*, & \text{se } F_{(\hat{k})} \leq (F^{\alpha}_{1,n-p-1})^3 / \left(p + F^{\alpha}_{1,n-p-1}\right)^2, \\ \text{Inconclusivo}, & \text{se } (F^{\alpha}_{1,n-p-1})^3/(p + F^{\alpha}_{1,n-p-1})^2 < F_{(\hat{k})} < F^{\alpha}_{p,n-p-1}. \end{cases} \tag{4.17}$$

Este teste apresenta algumas limitações. Há um região em que esse teste é inconclusivo além de não existir forma de calcular o poder do teste através desse método.

4.3 Análise de Variância

Em modelos de Regressão Linear, a decomposição da soma de quadrados total é um procedimento que permite testar a significância dos parâmetros modelo geral. Quando o estimador de $\boldsymbol{\beta}$ é obtido através do método de mínimos quadrados, a decomposição da soma de quadrados total é bem definida e discutida em Montgomery *et al.* (2012).

Como comentado anteriormente, em alguns casos há multicolinearidade na matriz $\mathbf{C}$ e é recomendado o uso dos estimadores pelo método de Regressão das cristas. Entretanto, segundo Hoerl e W. Kennard (1990), utilizar decomposição da soma de quadrados total para o estimador pelo MMQ e apenas substituir $\hat{\boldsymbol{\beta}}$ por $\hat{\boldsymbol{\beta}}^*_{(k)}$ gera componentes não ortogonais, o que pode acarretá em inferências errôneas. Os autores apresentaram uma decomposição ortogonal da soma de quadrados total do modelo de regressão em cristas em três componentes, a soma de quadrados dos resíduos, a soma de quadrados de regressão e o componente não-ortogonal.

A soma de quadrados dos resíduos é definida em (2.18). O valor esperado dessa quantidade pode ser dividido em duas partes. A primeira parte consiste no valor esperado de $\text{SQRes}(\hat{\boldsymbol{\beta}})$ dado por

$$E[\text{SQRes}(\hat{\boldsymbol{\beta}})] = (n-1-p)\sigma^2, \tag{4.18}$$

e a segunda consiste no valor esperando de uma forma quadrática que é calculado usando a relação $\text{E}(\mathbf{y}^\top\mathbf{A}\mathbf{y}) = \text{tr}(\mathbf{A}\text{Var}(\mathbf{y})) + \text{E}(\mathbf{y})^\top\mathbf{A}\text{E}(\mathbf{y})$. O valor esperado da segunda parte é dado por

$$E\left[k^2\hat{\boldsymbol{\beta}}^\top\mathbf{Z}^\top\mathbf{C}^{-1}\mathbf{Z}\hat{\boldsymbol{\beta}}\right] = k^2\sigma^2\sum_{i=1}^{p}1/(\lambda_i+k)^2 + k^2\boldsymbol{\beta}^\top\mathbf{Z}\mathbf{C}^{-1}\mathbf{Z}\boldsymbol{\beta}. \tag{4.19}$$

Portanto, o valor esperado de $\text{SQRes}(\hat{\boldsymbol{\beta}}^*_{(k)})$ é a soma das duas partes obtidas acima, resultando em

$$\begin{aligned} E\left[\text{SQRes}(\hat{\boldsymbol{\beta}}^*_{(k)})\right] &= (n-1-p)\sigma^2 + k^2\sigma^2\sum_{i=1}^{p}1/(\lambda_i+k)^2 + k^2\boldsymbol{\beta}^\top\mathbf{Z}\mathbf{C}^{-1}\mathbf{Z}\boldsymbol{\beta} \\ &= (n-1-m)\sigma^2 + k^2\boldsymbol{\beta}^\top\mathbf{Z}\mathbf{C}^{-1}\mathbf{Z}\boldsymbol{\beta}, \end{aligned} \tag{4.20}$$

em que $m = p - k^2\sum_{i=1}^{p}1/(\lambda_i+k)^2$. Note que quando $k=0$, então $m=p$, logo é obtido o mesmo valor esperado quando é utilizado o método dos mínimos quadrados. É possível observar em (4.20) que, por tomarmos valores não-negativos para k, a quantidade m diminui a medida que k aumenta. Além disso, a medida que $k\to\infty$, $m\to 0$ e $(n-1-m)\to(n-1)$. Os graus de liberdade associados com $\text{SQRes}(\hat{\boldsymbol{\beta}}^*_{(k)})$ será definido como $(n-1-m)$.

A soma de quadrados de regressão é definida por

$$\text{SQReg}(\hat{\boldsymbol{\beta}}^*_{(k)}) = \hat{\boldsymbol{\beta}}^\top_{(k)}\mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}}_{(k)}. \tag{4.21}$$

O valor esperado de $\text{SQReg}(\hat{\boldsymbol{\beta}}^*_{(k)})$ é calculado usando o valor esperado de uma forma quadrática e

é expressa por

$$\begin{aligned} E\left[\text{SQReg}(\hat{\boldsymbol{\beta}}^*_{(k)})\right] &= E(\hat{\boldsymbol{\beta}}^\top_{(k)}\mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}}_{(k)}) \\ &= E\left[\hat{\boldsymbol{\beta}}^\top\mathbf{Z}^\top\mathbf{C}\mathbf{Z}\hat{\boldsymbol{\beta}}\right] \\ &= \sigma^2\sum_{i=1}^{p}\lambda_i^2/(\lambda_i+k)^2 + \boldsymbol{\beta}^\top\mathbf{Z}^\top\mathbf{C}\mathbf{Z}\boldsymbol{\beta} \\ &= q\sigma^2 + \boldsymbol{\beta}^\top\mathbf{Z}^\top\mathbf{C}\mathbf{Z}\boldsymbol{\beta}, \end{aligned} \tag{4.22}$$

em que $q = \sum_{i=1}^{p}\lambda_i^2/(\lambda_i+k)^2$.

A quantidade q se comporta como o número de graus de liberdade do componente $\text{SQReg}(\hat{\boldsymbol{\beta}}^*_{(k)})$ pois $q = p$ quando $k = 0$, o mesmo valor obtido na decomposição das somas de quadrados pelo método de mínimos quadrados. Quando $k \to \infty$, a quantidade q tende a zero.

O componente não-ortogonal é definido como a diferença entre a soma de quadrados total menos a adição entre soma de quadrados de resíduos e a soma de quadrado de regressão. A expressão para o componente não-ortogonal é

$$\begin{aligned} \text{CNO}(\hat{\boldsymbol{\beta}}^*_{(k)}) &= \mathbf{y}^\top\mathbf{y} - (\text{SQRes}(\hat{\boldsymbol{\beta}}^*_{(k)}) + \text{SQReg}(\hat{\boldsymbol{\beta}}^*_{(k)})) \\ &= 2\left[\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}_{(k)}\right]^\top\mathbf{X}\hat{\boldsymbol{\beta}}_{(k)} \\ &= 2\left[\mathbf{y} - \mathbf{X}\mathbf{Z}\hat{\boldsymbol{\beta}}\right]^\top\mathbf{X}\mathbf{Z}\hat{\boldsymbol{\beta}} \\ &= 2\left[\hat{\boldsymbol{\beta}}^\top\mathbf{C}\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}^\top\mathbf{Z}^\top\mathbf{X}^\top\mathbf{X}\mathbf{Z}\hat{\boldsymbol{\beta}}\right] \\ &= 2k\hat{\boldsymbol{\beta}}^\top\mathbf{Z}^\top\mathbf{Z}\hat{\boldsymbol{\beta}}. \end{aligned} \tag{4.23}$$

O valor esperando para a quantidade $\text{CNO}(\hat{\boldsymbol{\beta}}^*_{(k)})$ é dado por

$$E[\text{CNO}(\hat{\boldsymbol{\beta}}^*_{(k)})] = 2kr\sigma^2 + 2k\boldsymbol{\beta}^\top\mathbf{Z}^\top\mathbf{Z}\boldsymbol{\beta}, \tag{4.24}$$

em que $r = \sum_{i=1}^{p}\lambda_i/(\lambda_i+k)$. os graus de liberdade do componente $\text{CNO}(\hat{\boldsymbol{\beta}}^*_{(k)})$ é $2kr$.

A decomposição da soma de quadrados totais pode ser descrita como a soma dos três componentes e é dada por

$$\begin{aligned} \mathbf{y}^\top\mathbf{y} &= ||(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}_{(k)}) + \mathbf{X}\hat{\boldsymbol{\beta}}_{(k)}||^2 \\ &= \left[\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}_{(k)}\right]^\top\left[\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}_{(k)}\right] + \hat{\boldsymbol{\beta}}^\top_{(k)}\mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}}_{(k)} + 2\left[\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}_{(k)}\right]^\top\mathbf{X}\hat{\boldsymbol{\beta}}_{(k)} \\ &= \text{SQRes}(\hat{\boldsymbol{\beta}}^*_{(k)}) + \text{SQReg}(\hat{\boldsymbol{\beta}}^*_{(k)}) + \text{CNO}(\hat{\boldsymbol{\beta}}^*_{(k)}), \end{aligned} \tag{4.25}$$

em que $||.||$ é o produto escalar e $||\mathbf{a}+\mathbf{b}||^2 = ||\mathbf{a}||^2 + ||\mathbf{b}||^2 + 2||\mathbf{a}||||\mathbf{b}||$.

Utilizando o valor esperado de cada um dos componentes da decomposição da soma de quadrados totais $\mathbf{y}^\top\mathbf{y}$, é possível construir a Tabela 4.1.

Tabela 4.1: *Decomposição de $\boldsymbol{y}^\top\boldsymbol{y}$ em soma de quadrados de resíduos, soma de quadrados de regressão e componente não-ortogonal para o modelo de Regressão em Cristas.*

Fonte de variação	Soma de quadrados	Graus de liberdade	Valor esperado
Regressão	$\hat{\boldsymbol{\beta}}_{(k)}^\top\mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}}_{(k)}$	q	$q\sigma^2+\boldsymbol{\beta}^\top\mathbf{Z}^\top\mathbf{C}\mathbf{Z}\boldsymbol{\beta}$
Resíduo	$(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}_{(k)}^*)^\top(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}_{(k)}^*)$	n-1-m	$(n-1-m)\sigma^2+k^2\boldsymbol{\beta}^\top\mathbf{Z}\mathbf{C}^{-1}\mathbf{Z}\boldsymbol{\beta}$
CNO	$2k\hat{\boldsymbol{\beta}}_{(k)}^\top\hat{\boldsymbol{\beta}}_{(k)}$	2kr	$2kr\sigma^2+2k\hat{\boldsymbol{\beta}}^\top\mathbf{Z}^\top\mathbf{Z}\hat{\boldsymbol{\beta}}$
Total	$\mathbf{y}^\top\mathbf{y}$	n-1	$(n-1)\sigma^2+\boldsymbol{\beta}^\top\mathbf{C}\boldsymbol{\beta}$

Os graus de liberdade da soma de quadrados total resulta em $(n-1)$, pois $-m+q+2kr=0$. Hoerl e W. Kennard (1990) enfatizam que, conforme pode ser observado na Tabela 4.1, sob a hipótese de que $\boldsymbol{\beta}=\mathbf{0}$, o quadrado médio de cada uma das três componentes é um estimador não viciado de σ^2.

No entanto, ao contrário do que ocorre na regressão via mínimos quadrados, não existe um estimador não viciado de σ^2, para $\boldsymbol{\beta}\neq\mathbf{0}$.

Os autores não definiram valores de comparação para a distribuição da estatística $F_{(k)}^{\text{ort}}$ definida por

$$F_{(k)}^{\text{ort}}=\frac{(n-1-m)}{q}\frac{\text{SQReg}(\hat{\boldsymbol{\beta}}_{(k)}^*)}{\text{SQRes}(\hat{\boldsymbol{\beta}}_{(k)}^*)}. \tag{4.26}$$

Contudo, é possível utilizar técnicas de re-amostragem para estimar o valor-p associado à estatística definida em (4.26).

Os resultados discutidos na Seções 4.2 e 4.3 foram definidos sob a suposição de que k é não-estocástico. Na prática, k é calculado usando os dados disponíveis e portanto os testes discutidos precisam ser validados utilizando métodos de simulação ou técnicas de re-amostragem.

4.4 Teste não-exato para β_i

Halawa e El Bassiouni (2000) abordaram o problema de testar a hipótese $H_0:\beta=0$ de uma forma diferente. Os autores consideraram uma estatística de teste não-exata. Um teste não-exato, ou aproximado, é quando o nível de significância do teste é aproximadamente a quantidade fixada, digamos α. A estatística proposta pelos autores é

$$t_{(k)}^{\text{ne}}=\frac{\hat{\beta}_{i(k)}^*}{S\left(\hat{\beta}_{i(k)}^*\right)}, \tag{4.27}$$

em que $\hat{\beta}_{i(k)}^*$ é o i-ésimo elemento do vetor $\hat{\boldsymbol{\beta}}_{(k)}^*$ e $S(\hat{\beta}_{i(k)}^*)$ é a raiz quadrada da estimativa da variância de $\hat{\beta}_{i(k)}^*$ dada pelo i-ésimo elemento da diagonal da matriz descrita em (2.11) e substituindo

σ^2 por sua estimativa dada em (2.40).

Como discutido no Capítulo 2, a adição de um viés e a redução da magnitude do estimador $\hat{\beta}^*_{i(k)}$ para β_i é compensada pela redução de $\text{Var}(\hat{\beta}^*_{i(k)})$ em comparação com $\text{Var}(\hat{\beta}^*_{i(0)})$. Assim, é esperado que $|t^{\text{ne}}_{(k)}|$ seja maior que $|t_{(0)}|$ e, por consequência, utilizar a estatística $t^{\text{ne}}_{(k)}$ resulta em um teste mais poderoso, dependendo do quão próximo o nível de significância de (4.27) for de α.

A quantidade $\hat{\sigma}^2_{(k)}$ definida por

$$\hat{\sigma}^2_{(k)} = \frac{\text{SQReg}(\hat{\boldsymbol{\beta}}^*_{(k)})}{(n-p-1)}, \tag{4.28}$$

é um estimador consistente de σ^2, e (4.27) tem, assintoticamente, distribuição normal. Embora, para amostras pequenas, $\hat{\sigma}^2_{(k)}$ superestime σ^2, com o seguinte viés

$$\text{Vies}(\hat{\sigma}^2_{(k)}) = \frac{k^2}{(n-p-1)}\left[\boldsymbol{\beta}^\top(\mathbf{C}+k\mathbf{I}_p)^{-2}\boldsymbol{\beta} + \sigma^2\text{tr}(\mathbf{C}+k\mathbf{I}_p)^{-2}\mathbf{C}^{-1}\right]. \tag{4.29}$$

É esperado que para valores de k mais próximos de 0, o nível de significância do teste (4.27) se aproxime de α.

Contudo, quando k é determinado utilizando os dados, esse argumento não é válido e métodos de simulação ou re-amostragem são necessários para determinar o nível de significância e o poder do teste.

No Capítulo 5, será abordado a técnica de re-amostragem *bootstrap* e como essa técnica pode ser útil para a inferência sobre $\boldsymbol{\beta}$ quando se utiliza o procedimento de Regressão em Cristas.

4.5 Inferência pelo método de *bootstrap*

É possível utilizar o método de *bootstrap*, descrito no Capítulo 5, para realizar inferência sobre os coeficientes no modelo quando se adota o procedimento de Regressão em Cristas. A inferência pode ser realizada construindo intervalos de confianças por *bootstrap* para um determinado nível de significância α, utilizando alguma metodologia de construção de intervalo de confiança como por exemplo os métodos básico, normal, percentil e BC_a.

Uma vantagem do método de *boostrap* para construir intervalo de confiança para os coeficientes de regressão neste caso é que esse método não faz nenhuma suposição sobre a forma da distribuição dos erros do modelo.

Além disso, o fato da quantidade $\hat{k}$ ser estocástica também não é um problema para esse método como é para outros métodos de inferência para o modelo de Regressão em Cristas.

Através do intervalo de confiança é possível realizar o teste bilateral

$$\begin{cases} H_0 : \beta_i = c \\ H_1 : \beta_i \neq c, \end{cases}$$

para isso é preciso construir o intervalo de $(1-\alpha)\%$ confiança para β_i e observar se a quantidade c está presente dentro do intervalo de confiança. Caso a quantidade c não esteja presente no intervalo,

podemos então rejeitar a hipótese H_0 com nível de significância α.

O método de *boostrap* e metodologias para criação de intervalos de confiança para esse método serão descritas no Capítulo 5. Um estudo de simulação será realizado no Capítulo 6 para observar o comportamento de intervalos de confiança construídos pelo método de *boostrap* quando há alterações em diversos parâmetros como n, p, σ, o número de condição, a metodologia de estimação para a quantidade k, a orientação do vetor de coeficientes $\boldsymbol{\beta}$ e também os diferentes métodos de construção de intervalos de confiança por *bootstrap*.

Capítulo 5

Método *Bootstrap*

5.1 Introdução

O *bootstrap* é um método computacional introduzido por Efron (1982) e utilizado para obter melhores estimativas e para mensurar a precisão de estimativas através da estimação de medidas como viés e erro padrão e na construção de intervalos de confiança, evitando inferências errôneas. A técnica é útil em situações em que há problemas nas suposições, número pequeno de observações, quando a distribuição dos dados não é normal, problemas com complexidade analítica, situações em que não há consenso teórico em uma abrangente série de aplicações.

Nessa capítulo será explicado a técnica de *bootstrap* sob o ponto de vista não condicional, como construir intervalos de confiança através desse procedimento de re-amostragem, estimar valores-p de teste de hipóteses, e também como aplicar esse método em modelos de regressão, o que possibilitará a utilização do *bootstrap* para realizar melhores inferências em modelos de Regressão em Cristas. Montgomery *et al.* (2012) cita que a técnica de *bootstrap* pode ser útil para construir inferências a partir do estimador $\boldsymbol{\beta}^*_{(k)}$.

5.2 Método

Suponha que temos uma amostra aleatória de n observações de uma variável $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ cuja distribuição de probabilidade depende de um parâmetro desconhecido θ. Desejamos encontrar uma estimativa para o parâmetro θ através da estatística S seguindo uma distribuição F com forma desconhecida totalmente ou parcialmente. Usualmente, encontramos a estimativa $\hat{\theta} = S(\mathbf{x})$,que é o valor estatística S calculado utilizando a amostra $\mathbf{X}$.

O método de *bootstrap* consiste em estimar a distribuição da estatística S gerando um número B de amostras independentes de tamanho n com reposição e calcular para cada uma delas o valor de S, obtendo dessa forma uma estimativa de F.

Em situações em que a distribuição F é conhecida a não ser por seus parâmetros, podemos gerar B amostras utilizando essa distribuição e o valor da estimativa $\hat{\theta}$ do parâmetro e então calcular para cada uma delas o valor de S. Por exemplo, considerando que S tem distribuição normal com média desconhecida e variância conhecida σ^2, então é possível replicar amostras de uma distribuição normal usando os parâmetros $\hat{\theta} = (\hat{\mu}, \sigma^2)$, em seguida a estatística S é calculada para todas as réplicas e é obtida uma estimativa para a distribuição da estatística, esse método é chamado de *bootstrap* paramétrico.

Quando nada é conhecido sobre a distribuição F, a distribuição empírica $\hat{F}$ é utilizada. A construção da distribuição empírica consiste em atribuir para cada valor amostrado a probabilidade $1/n$. Para construí-la, são geradas B amostras independentes de tamanho n com reposição de $\mathbf{X}$ e para cada amostra $\mathbf{X}_b^* = (X_1^*, X_2^*, \ldots, X_n^*)$, com $b = 1, \ldots, B$, é calculado o valor de $\hat{\theta}_b^* = S(\mathbf{x}_b^*)$.

Através da construção da distribuição empírica utilizando os valores estimados $S(\mathbf{x}_b^*)$, uma estimativa da distribuição de S é obtida com média $\bar{\hat{\theta}}^*$, a média amostral de todos os valores de $\hat{\theta}_b^*$. Esse é o método chamado de *bootstrap* não-paramétrico, comumente utilizado por não ser necessário fazer suposições sobre a distribuição F.

Quando não se conhece a distribuição F ou a forma analítica para o erro-padrão de $\hat{\theta}$ é de difícil obtenção, é possível obter uma boa aproximação de $\mathrm{EP}(\hat{\theta})$ calculando o desvio-padrão dos valores de $\hat{\theta}_b^*$, essa quantidade é denotada por estimativa de *bootstrap* para o erro-padrão de $\hat{\theta}$ e é obtida por:

$$\widehat{\mathrm{EP}}^*(\hat{\theta}) = \sqrt{\sum_{b=1}^{B} \frac{\left(\hat{\theta}_b^* - \bar{\hat{\theta}}^*\right)^2}{(B-1)}}, \tag{5.1}$$

em que $\bar{\hat{\theta}}^* = \sum_{b=1}^{B} \hat{\theta}_b^*/B$.

O limite de $\widehat{\mathrm{EP}}^*(\hat{\theta})$ quando B tende a infinito é $\mathrm{EP}(\hat{\theta})$, ou seja,

$$\lim_{B\to\infty} \widehat{\mathrm{EP}}^*(\hat{\theta}) = \mathrm{EP}(\hat{\theta}), \tag{5.2}$$

esse limite é explicado pelo fato que o desvio-padrão amostral dos $\hat{\theta}_b^*$ se aproxima do desvio-padrão populacional à medida que B aumenta, pois nesse caso a população em questão são todos os possíveis valores de $\hat{\theta}$.

Uma questão frequente é qual número de réplicas B utilizar para obter resultados satisfatórios. Efron e Tibshirani (1994) e Davison e Hinkley (1997) sugerem que para construir intervalos de confiança de 90% ou 95% de confiança para o parâmetro de interesse, B deve ser pelo menos maior que 1000. Devido aos avanços computacionais, é possível gerar um número satisfatório de réplicas para obter ótimas aproximações.

5.3 Intervalos de Confiança baseados no método de *Bootstrap*

Intervalos de confiança são frequentemente desejados em análises estatísticas por conter mais informações de um estimador do que uma estimativa pontual e portanto possibilitando melhores inferências. Suponha que queremos construir um intervalo de confiança para θ utilizando a estatística S e com coeficiente de confiança $1-\alpha$.

Se os quantis de $\hat{\theta}-\theta$ forem denotados por a_p, então $\mathrm{P}(\hat{\theta}-\theta \leq a_{\alpha/2}) = \mathrm{P}(\hat{\theta}-\theta \geq a_{1-\alpha/2}) = \alpha/2$, é possível rescrever os eventos $\hat{\theta}-\theta \leq a_{\alpha/2}$ e $\hat{\theta}-\theta \geq a_{1-\alpha/2}$ da seguinte forma

$$\begin{aligned} P(a_{\alpha/2} \leq \hat{\theta} - \theta \leq a_{1-\alpha/2}) &= (1-\alpha) \\ P(\hat{\theta} - a_{1-\alpha/2} \leq \theta \leq \hat{\theta} - a_{\alpha/2}) &= (1-\alpha), \end{aligned} \tag{5.3}$$

portanto, após isolar o parâmetro θ, é obtido o intervalo de confiança

$$\text{IC}(\theta;(1-\alpha)) = \left[\hat{\theta} - a_{1-\alpha/2}, \hat{\theta} - a_{\alpha/2}\right]. \tag{5.4}$$

Entretanto, a distribuição de $\hat{\theta}-\theta$ é geralmente desconhecida e então uma outra abordagem geralmente utilizada é através da aproximação pela distribuição normal padrão através da quantidade pivotal Z, definida por:

$$Z = \frac{\hat{\theta}-\theta}{\text{EP}(\hat{\theta})} \sim \mathcal{N}(0,1). \tag{5.5}$$

Dessa forma, o intervalo de confiança com coeficiente de confiança $1-\alpha$, é construído da seguinte forma

$$\begin{aligned} P\left(z_{\alpha/2} \leq \frac{\hat{\theta}-\theta}{\text{EP}(\hat{\theta})} \leq z_{1-\alpha/2}\right) &= (1-\alpha) \\ P\left(\hat{\theta} - \text{EP}(\hat{\theta}) z_{1-\alpha/2} \leq \theta \leq \hat{\theta} - \text{EP}(\hat{\theta}) z_{\alpha/2}\right) &= (1-\alpha), \end{aligned} \tag{5.6}$$

em que z_α é o quantil de ordem α da distribuição normal padrão. Então, utilizando o fato que $z_{\alpha/2} = -z_{1-\alpha/2}$, o intervalo de confiança é dado por

$$\text{IC}(\theta;(1-\alpha)) = \left[\hat{\theta} - \text{EP}(\hat{\theta}) z_{1-\alpha/2}; \hat{\theta} + \text{EP}(\hat{\theta}) z_{1-\alpha/2}\right], \tag{5.7}$$

em que $z_{1-\alpha/2} = \Phi^{-1}(1-\alpha/2)$ e $\Phi(.)$ é a distribuição acumulada da normal padrão. Quando não conhecemos $\text{EP}(\hat{\theta})$, é possível substituí-lo por sua estimativa e obtemos a quantidade pivotal t dada por

$$t = \frac{\hat{\theta}-\theta}{\widehat{\text{EP}}(\hat{\theta})} \sim t_{n-1}, \tag{5.8}$$

e podemos construir um intervalo de confiança substituindo $z_{1-\alpha/2}$ em (5.7) por $t_{(n-1;1-\alpha/2)}$ o quantil de ordem $(1-\alpha/2)$ da distribuição *t-student* com $n-1$ graus de liberdade.

Entretanto, como discutido anteriormente, nem sempre é possível obter uma estimativa para $\text{EP}(\hat{\theta})$ de forma teórica ou uma estimativa satisfatória. Uma solução é aplicar o método de *bootstrap* para encontrar intervalos de confiança para θ.

Existem alguns métodos de criar intervalos de confiança utilizando o procedimento de *bootstrap* como o método básico, método normal, método de percentil e o método de percentil aperfeiçoado. Todos esses são discutidos em Efron e Tibshirani (1994) e Davison e Hinkley (1997).

A seguir serão discutidos cada um dos métodos, suas vantagens e desvantagens, se o método respeita transformações em θ, ou seja, para qualquer transformação monótona de θ, $m(\theta)$, o intervalo de confiança é construído aplicando a mesma transformação nos limites do intervalo construído para

θ, ou seja, considerando θ_I e θ_S os limites inferior e superior do intervalo com nível de significância α para θ, então o intervalo de confiança para $m(\theta)$ é dado por

$$\text{IC}(m(\theta); (1-\alpha)) = [m(\theta_I); m(\theta_S)]\,. \tag{5.9}$$

Além disso, um breve comentário sobre o erro de cobertura dos métodos de construção de intervalos por *bootstrap* é feita com finalidade de compará-los.

A construção de alguns intervalos de confiança descritos a seguir exigem que as quantidades $(B+1)\alpha$ sejam números inteiros, pois essa quantidade é utilizada para se obter a $(B+1)\alpha$-ésima estatística de ordem. Se essas quantidades não forem número inteiros, então é preciso usar uma interpolação. Uma interpolação descrita em Davison e Hinkley (1997) é

$$x_{B\alpha} = x_k + \frac{\Phi^{-1}(\alpha) - \Phi^{-1}\left(\frac{k}{B+1}\right)}{\Phi^{-1}\left(\frac{k+1}{B+1}\right) - \Phi^{-1}\left(\frac{k}{B+1}\right)}\left(x_{k+1} - x_k\right), \tag{5.10}$$

com $k = [(B+1)\alpha]$, em que $[.]$ representa a parte inteira de um número real, e x_k é o k-ésimo valor ordenado de uma amostra.

5.3.1 Intervalo de confiança básico por *bootstrap*

O princípio do *bootstrap* implica que é possível obter informação sobre a relação entre θ e $\hat{\theta}$ assumindo que $\hat{\theta}$ é o valor verdadeiro de θ e então analisar a relação de $\hat{\theta}$ e os valores obtidos nas réplicas de *boostrap* $\hat{\theta}^*_b$.

O intervalo de confiança básico pelo método de *bootstrap* consiste em estimar os quantis em (5.4) através das estatísticas de ordem dos valores de $\hat{\theta}^*_b - \hat{\theta}$, com $b = 1, \ldots, B$. E então é possível construir o seguinte intervalo de confiança

$$\text{IC}(\theta; (1-\alpha)) = \left[2\hat{\theta} - a^*_{(B+1)(1-\alpha/2)}; 2\hat{\theta} - a^*_{(B+1)(\alpha/2)}\right], \tag{5.11}$$

em que a^*_p é a p-ésima estatística de ordem dos valores de $\hat{\theta}^*_b - \hat{\theta}$ usado para estimar os valores de a_p, o valor referente ao quantil de ordem p da distribuição de $\hat{\theta} - \theta$.

Esse é um método simples, entretanto o erro em substituir os quantis a_p por a^*_p pode ser grande, além de o método ter a possibilidade de fornecer valores para o intervalo fora do espaço paramétrico de θ.

5.3.2 Intervalo de confiança Normal

Uma forma de ajustar o possível erro em estimar os quantis da distribuição é utilizando o princípio de quantidade pivotal. O intervalo de confiança normal consiste em aproximar a quantidade pivotal Z em (5.5) utilizando o método de *bootstrap*.

Para aproximar essa quantidade pivotal, primeiramente é subtraído o viés estimado do estimador para o parâmetro θ . Esse viés é estimado através da diferença entre a média dos valores

encontrados para $\hat{\theta}_b$, para $b = 1, \ldots, B$ e o valor para $\hat{\theta}$ estimado utilizando a amostra original, que pode ser escrito como

$$\hat{\mathrm{V}}_b(\hat{\theta}) = \frac{\sum_{b=1}^{B} \hat{\theta}_b}{B} - \hat{\theta}. \tag{5.12}$$

Em seguida, uma estimativa para o erro-padrão do estimador $\hat{\theta}$ é obtida através do desvio padrão amostral dos valores de $\hat{\theta}_b$ também para $b = 1, \ldots, B$. Portanto, a aproximação para a quantidade pivotal se dá através da seguinte expressão

$$Z \approx \frac{\left[\hat{\theta} - \hat{V}_b(\hat{\theta})\right] - \theta}{\sqrt{\mathrm{Var}(\hat{\theta}_b)}}, \tag{5.13}$$

através dessa aproximação da quantidade pivotal Z, é possível construir o intervalo de confiança pelo método de *bootstrap* e utilizando os quantis da distribuição normal da seguinte maneira

$$\mathrm{IC}(\theta; (1-\alpha)) = \left[\left(\hat{\theta} - \hat{V}_b(\hat{\theta})\right) - z_{(1-\alpha/2)}\sqrt{\mathrm{Var}(\hat{\theta}_b)}; \left(\hat{\theta} - \hat{V}_b(\hat{\theta})\right) + z_{(1-\alpha/2)}\sqrt{\mathrm{Var}(\hat{\theta}_b)}\right]. \tag{5.14}$$

Esse método de construir intervalos de confiança em que a quantidade pivotal Z é aproximada utilizando os valores obtidos através das re-amostragens é um método mais fácil de ser obtido computacionalmente, entretanto necessita que a distribuição de $\left(\hat{\theta} - \hat{V}_b(\hat{\theta})\right)$ seja próxima da distribuição normal para obtermos valores mais próximos do esperado. Quando a distribuição dessa quantidade não se aproxima de uma distribuição normal, os intervalos construídos através desse método perdem eficiência.

5.3.3 Método Percentil

O método de construir intervalos por percentil é menos intuitivo que os métodos básico e normal discutidos anteriormente, mas tem a vantagem de não necessitar de uma estimativa do erro-padrão de $\hat{\theta}$. Considere a função monótona crescente $g(.)$ e então escrevendo $\phi = g(\theta)$, $\hat{\phi} = g(\hat{\theta})$ e $\hat{\phi}^* = g(\hat{\theta}^*)$ e escolhendo $g(.)$ de forma que $\hat{\phi}^* - \hat{\phi}$ tenha a mesma distribuição que $\hat{\phi} - \phi \sim \mathcal{N}(0, \sigma^2)$, e então o intervalo de confiança para θ com confiança $(1-\alpha)100\%$ é construído utilizando $g^{-1}(\hat{\phi} \pm \sigma z_{(1-\alpha/2)})$, em que $z_{1-\alpha/2}$ é o quantil de ordem $(1-\alpha/2)$ da distribuição normal padrão.

Além disso, é possível manipular a quantidade pivotal de forma a escrever a probabilidade em função de $\hat{\phi}^*$ da seguinte maneira

$$\begin{aligned} P(z_{\alpha/2} < \frac{\hat{\phi}^* - \hat{\phi}}{\sigma} < z_{1-\alpha/2}) = \\ P(\hat{\phi} + \sigma z_{\alpha/2} < \hat{\phi}^* < \hat{\phi} + \sigma z_{1-\alpha/2}) = (1-\alpha) \end{aligned} \tag{5.15}$$

isso implica que $\hat{\phi} + \sigma z_{\alpha/2} = F^{-1}_{\hat{\phi}^*}(\alpha/2)$ e $\hat{\phi} + \sigma z_{1-\alpha/2} = F^{-1}_{\hat{\phi}^*}(1-\alpha/2)$, e como $g(.)$ foi definida monótona crescente, então é válido que $F^{-1}_{\hat{\phi}^*}(\alpha/2) = g(F^{-1}_{\hat{\theta}^*}(\alpha/2))$. Substituindo essa quantidade

em $g^{-1}(\hat{\phi} \pm \sigma z_{(1-\alpha/2)})$, o intervalo de confiança é construído utilizando os quantis da distribuição empírica construída através das réplicas de *bootstrap*, o intervalo é dado por

$$\begin{aligned} \text{IC}(\theta; (1-\alpha)) &= \left[\hat{F}^{-1}_{\hat{\theta}^*}(\alpha/2); \hat{F}^{-1}_{\hat{\theta}^*}(1-\alpha/2)\right] \\ &= \left[\hat{\theta}^*_{(B+1)(\alpha/2)}; \hat{\theta}^*_{(B+1)(1-\alpha/2)}\right], \end{aligned} \tag{5.16}$$

em que $\hat{\theta}^*_{(B+1)(\alpha/2)}$ é o $100(\alpha/2)$-ésimo percentil empírico dos valores $\hat{\theta}^*_b$, com $b = 1, \ldots, B$, ou seja, o valor $(B+1)(\alpha/2)$ na lista ordenada das réplicas $\hat{\theta}^*_b$.

Esse método de construir intervalo de confiança é bem simples e diferentemente do método normal, não necessita da estimação da quantidade σ. Além disso, esse procedimento respeita transformações em θ, e não retorna valores fora do espaço paramétrico de θ. Existem outras situações em que essa técnica pode ser falha, como quando a distribuição de $\hat{\theta}$ não for próxima de uma distribuição simétrica e em situações em que $\hat{\theta}$ é viciado. Na próxima subseção será descrito uma extensão para o método de percentil que é efetivo para possíveis distribuições assimétricas e viés de $\hat{\theta}$.

5.3.4 Método de Percentil Aperfeiçoado: BC_a

Uma versão aperfeiçoada para o método de percentil para construir intervalos de confiança para θ é chamada de BC_a, que é uma abreviatura para *bias-corrected and accelerated*, que significa correção de viés e aceleração. Esse procedimento é considerado aperfeiçoado pois se aproxima de intervalos construídos em situação em que há conhecimento teórico, e também é eficaz em um maior número de situações, por corrigir possível deslocamento e assimetria da distribuição do estimador para θ.

Assim como o método de percentil, o intervalo através do método de BC_a também utiliza os percentis empíricos dos valores de *bootstrap*, entretanto eles não são os mesmos que os definidos em (5.16). Considerando novamente a função monótona crescente $g(.)$ e escrevendo $\phi = g(\theta)$, $\hat{\phi} = g(\hat{\phi})$ e $\hat{\phi}^* = g(\hat{\theta}^*)$, mas dessa vez escolhendo $g(.)$ de forma que $\hat{\phi} \sim \mathcal{N}(\phi - w\sigma(\phi), \sigma^2(\phi))$ e $\hat{\phi}^* \sim \mathcal{N}(\hat{\phi} - w\sigma(\hat{\phi}), \sigma^2(\hat{\phi}))$, em que $\sigma(x) = 1 + ax$. E então, analogamente ao que foi feito para o método de percentil, através de manipulação da quantidade pivotal é possível construir o seguinte intervalo de confiança para θ

$$\text{IC}(\theta; (1-\alpha)) = \left[\hat{\theta}^*_{(B+1)(\alpha_1)}, \hat{\theta}^*_{(B+1)(\alpha_2)}\right], \tag{5.17}$$

em que

$$\begin{aligned} \alpha_1 &= \Phi\left(\hat{w} + \frac{\hat{w} + z_{(\alpha/2)}}{1 - \hat{a}(\hat{w} + z_{(\alpha/2)})}\right) \\ \alpha_2 &= \Phi\left(\hat{w} + \frac{\hat{w} + z_{(1-\alpha/2)}}{1 - \hat{a}(\hat{w} + z_{(1-\alpha/2)})}\right), \end{aligned} \tag{5.18}$$

a função $\Phi(.)$ e z_α representam, respectivamente, a função de distribuição acumulada e o percentil

α de uma normal padrão.

Os percentis usados nesse método dependem de duas quantidades:$\hat{a}$ e $\hat{w}$. O valor de $\hat{w}$ é obtido utilizando a proporção de réplicas em que a estimativa de *bootstrap* é menor que a estimativa original de θ, e então $\hat{w}$ é calculado da seguinte forma:

$$\hat{w} = \Phi^{-1}\left(\frac{\#(\hat{\theta}_b^* < \hat{\theta})}{(B+1)}\right), \tag{5.19}$$

em que $\Phi^{-1}(.)$ representa o inverso da função de distribuição acumulada de uma normal padrão. Para valores de $\hat{w}$ e $\hat{a}$ iguais a zero, o intervalo é o mesmo que pelo método percentil. Grosseiramente, $\hat{w}$ mede a mediana do viés de $\hat{\theta}^*$ em unidades da distribuição normal padrão.

A quantidade $\hat{a}$ é chamado de fator de aceleração pois refere-se à taxa de variação do erro-padrão de $\hat{\theta}$ em relação ao valor de θ. Utilizar a aproximação de $\hat{\theta}$ através da distribuição normal de parâmetros θ e $EP(\hat{\theta})$ é assumir que o erro padrão de $\hat{\theta}$ é o mesmo para todos os valores de θ. Entretanto nem sempre isso acontece em situação práticas e a quantidade $\hat{a}$ é uma correção para esses casos. Existem diversas formas de calcular o valor de $\hat{a}$, uma delas é através do método de *jacknife*, em que $\hat{\theta}$ é estimado removendo a i-ésima observação, resultando no estimador $\hat{\theta}_{(i)}$, e tomando $\hat{\theta}_{(.)} = \sum_{i=1}^{n} \hat{\theta}_{(i)}/n$ então uma expressão para o valor de aceleração é

$$\hat{a} = \frac{\sum_{i=1}^{n}\left(\hat{\theta}_{(.)} - \hat{\theta}_{(i)}\right)^3}{6\left[\sum_{i=1}^{n}\left(\hat{\theta}_{(.)} - \hat{\theta}_{(i)}\right)^2\right]^{3/2}}. \tag{5.20}$$

A acurácia ou erro de cobertura de um intervalo é medida através do quão precisa é a probabilidade, $P(\hat{\theta}_{\text{inf}} < \theta < \hat{\theta}_{\text{sup}}) = 1-\alpha + \text{O(g(n))}$. Os intervalos BC_a podem ser classificados por precisão de segunda ordem, o que significa que o erro em aproximar a probabilidade vai para zero com a taxa $1/n$. Efron e Tibshirani (1994) dá mais detalhes sobre o erro de cobertura dos intervalos de confiança construídos utilizado o método de *bootstrap*.

Um potencial problema do método de BC_a é o fato que α_1 e α_2 podem ser bem próximos de 0 e 1, e portanto $B(\alpha_1)$ e $B(\alpha_2)$ podem ser maior que 1 ou menor que B, respectivamente. Portanto, neste caso, a interpolação para o quantil não pode ser calculada, a solução é aumentar o valor de replicações B.

5.4 Teste de hipóteses

Nessa seção, será discutido o uso da técnica de *bootstrap* em testes de hipóteses, mas antes será feita uma breve introdução a teste de hipóteses. Considerando a situação mais simples em que temos uma hipótese sobre a distribuição dos dados, chamada de hipótese nula ou H_0, e supondo que temos uma amostra aleatória $y_1, \ldots, y_n$ de uma população com distribuição de probabilidades F, então sob a hipótese nula algum aspecto da distribuição F é conhecido.

Testes de hipóteses são construídos através de uma estatística de teste S, que mensura o grau de concordância da amostra obtida e a distribuição sob H_0. Além disso, α e p são definidos respectivamente como nível de significância e o nível descritivo do teste.

É importante que S seja tomada de forma que a distribuição dessa variável aleatória não dependa de outros possíveis parâmetros que permaneçam desconhecidos sob H_0, ou seja, que S seja suficiente para θ.

Há equivalência entre testes de hipóteses e intervalos de confiança, essa é explicada considerando θ_0, o valor de θ sob H_0. Se o valor θ_0 estiver fora do intervalo de $(1-\alpha)100\%$ de confiança construído para θ, então a hipótese nula é rejeitada com valor-p menor ou igual a α.

Métodos de re-amostragem são úteis em situações em que as técnicas de calcular o valor-p não são eficazes ou são inapropriadas. Na prática, o valor-p exato pode ser difícil de calcular, e então o método de Monte Carlo é conveniente para aproximar esse valor. O método de Monte Carlo básico consiste em comparar o valor da estatística observado s com valores de S calculados sob a hipótese nula utilizando métodos de re-amostragem. Denotando esses valores por $s_1^*, \ldots, s_B^*$, então sob H_0 todos os $B+1$ valores, $s, s_1^*, \ldots, s_B^*$, são valores possíveis para S, logo

$$P(S \leq s_{(b)}^* | H_0) = \frac{b}{B+1}, \tag{5.21}$$

em que $s_{(b)}^*$ é o b-ésimo valor ordenado de S^*, então o valor-p pelo método de Monte Carlo é calculado por:

$$p_{mc} = \frac{1 + \#\left[s_b^* > s\right]}{B+1} \doteq P(S \geq s | H_0), \tag{5.22}$$

em que $\#[s_b^* > s]$ indica o número de valores de s_b^* maiores que s.

O método de Monte Carlo para encontrar uma aproximação do valor-p tem duas vantagens. A primeira é que é necessário apenas simular dados sob a hipótese nula, e a segunda é que os valores re-amostrados não precisam ser independentes um dos outros, somente é necessária a propriedade de permutabilidade, ou seja, a distribuição conjunta de $S, S_1^*, \ldots, S_B^*$ sob H_0 é invariante para permutações dos seus argumentos. Dessa forma é possível utilizar o método de *bootstrap* para encontrar vários valores de S, e então calcular a aproximação para o valor-p.

Calcular aproximação par ao valor-p pode diminuir o poder do teste por deslocar a região crítica de forma aleatória, entretanto Davison e Hinkley (1997) mostram que para $B \geq 999$, a perda no poder do teste é ínfima.

O teste de hipóteses utilizando o procedimento de *bootstrap* pode ser baseado em uma estatística de teste sob a hipótese nula. Por exemplo, considerando o teste-t para testar a hipótese nula de que $\theta = \theta_0$, então uma aproximação da distribuição da quantidade pivotal t pode ser obtida re-amostrando os valores $x_1, \ldots, x_n$ com reposição B vezes e calculando para cada uma das réplicas a quantidade

$$t^* = \frac{\hat{\theta}^* - \hat{\theta} + \theta_0}{\widehat{\mathrm{EP}}(\hat{\theta}^*)}. \tag{5.23}$$

Há uma equivalência entre esse método de teste de hipóteses e o intervalo de confiança construído através do método normal.

5.5 *Bootstrap* em Modelos de Regressão

Como discutido na seção anterior, frequentemente o objetivo de se aplicar o método de *bootstrap* não é apenas avaliar as estimativas dos parâmetros mas obter estimativas para o erro-padrão utilizando a distribuição gerada pelas réplicas. Isso pode ser bastante útil em situações que não se consegue obter formas analíticas fechadas para o erro-padrão ou situações em que o método de mínimos quadrados não é apropriado.

Há duas formas populares de aplicar métodos de *bootstrap* em modelos de regressão. A primeira é realizando re-amostragem dos pares $(\mathbf{x}_i^\top, y_i)$, em que y_i é o valor da variável resposta para o i-ésimo elemento amostral e $\mathbf{x}_i$ é a i-ésima linha da matriz de variáveis explicativas $\mathbf{X}$. Em cada uma das réplicas ou re-amostragens de *bootstrap*, é retirada uma amostra aleatória e com reposição de tamanho n dos valores de $(\mathbf{x}_i^\top, y_i)$, para $i = 1, \ldots, n$ e em seguida é ajustado um modelo de regressão obtendo as estimativas para os coeficientes.

Um segundo método seria ajustando o modelo de regressão utilizando os amostras originais de $\mathbf{X}$ e $\mathbf{y}$ e então, e gerar amostras com reposição de tamanho n dos resíduos, $\hat{\mathbf{e}} = \hat{\mathbf{y}} - \mathbf{y}$, denotadas por $\mathbf{e}^*$. Então, ajustar o modelo de regressão utilizando a matriz $\mathbf{X}$ e $\mathbf{y}^{**} = \hat{\mathbf{y}} + \mathbf{e}^*$.

O método a ser utilizado depende do quão válida são as suposições do modelo, por exemplo, suponha no modelo de regressão linear em que os erros aleatórios não dependem de $\mathbf{X}$, mas os dados apresentam heterocedasticidade, então nesse caso re-amostrar os pares de observações pode ser menos sensitivo do que re-amostrar os resíduos, pois a única suposição é que $(\mathbf{x}_i^\top, y_i)$ foram aleatoriamente amostrados de uma mesma distribuição. Exemplos e mais detalhes sobre aplicação do método de *bootstrap* em modelos de regressão podem ser encontrados em Efron e Tibshirani (1994).

Capítulo 6

Simulação

6.1 Introdução

Como descrito nos capítulos anteriores, o método de Regressão em Cristas pode ser útil em casos onde há presença de multicolinearidade entre as variáveis explicativas, entretanto é necessário encontrar um valor satisfatório para o parâmetro da crista k, que como visto no Capítulo 3, pode ser feito de diversas maneiras utilizando os dados disponíveis.

No entanto, quando k é estocástico, ou seja, estimado através dos dados, é introduzido no modelo outra fonte de variabilidade e dessa forma as técnicas de inferência derivadas do método de regressão utilizando mínimos quadrados podem trazer resultados menos eficientes.

Construir intervalos de confiança utilizando o procedimento de *bootstrap* é uma forma de inferir sobre os parâmetros de regressão método das cristas, uma vez que esse procedimento pode ser aplicado em qualquer amostra de dados e funciona bem quando há diversas fontes de variabilidade na estimação dos parâmetros.

Nesse capítulo será feito um estudo de simulação para determinar a eficiência dos intervalos de confiança para os coeficientes do modelo de Regressão em Cristas construídos pelo método de *bootstrap* discutidos no Capítulo 5, além disso será observada a performance das múltiplas formas de estimar o valor de k em situações em que há presença de multicolinearidade.

6.2 Metodologia

Para realizar esse estudo de simulação é preciso criar situações em que há presença de multicolinearidade na matriz de variáveis explicativas, para isso é preciso simular valores para a matriz de variáveis explicativas, para o vetor de coeficientes $\boldsymbol{\beta}$, o vetor aleatório de erros $\boldsymbol{\epsilon}$, e então utilizar esses valores para gerar o vetor resposta $\mathbf{y}$.

Para gerar a matriz $\mathbf{X}$ em que as colunas são correlacionadas, será usado o método descrito em McDonald e Galarneau (1975) e Gibbons (1981) em que é possível gerar vários graus de multicolinearidade. O método consiste em gerar x_{ij}, o elemento da i-ésima linha e j-ésima coluna da matriz $\mathbf{X}$, da seguinte forma

$$x_{ij} = (1 - \rho^2)^{(1/2)} z_{ij} + \rho z_{ip}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, p, \tag{6.1}$$

em que z_{ij} são número aleatórios gerados de uma distribuição normal padrão e ρ é escolhido de forma que a correlação entre duas variáveis explicativas seja dada por ρ^2. Além disso, todas as colunas da matriz $\mathbf{X}$ são padronizadas.

Os coeficientes do vetor $\boldsymbol{\beta}$ serão tomados de duas formas, utilizando os autovetores normalizados correspondentes ao maior e menor autovalores da matriz $\mathbf{C}$, dessa forma temos a melhor e pior orientação para $\boldsymbol{\beta}$, segundo Rao *et al.* (1973). Para ambos os vetores de $\boldsymbol{\beta}$, o i-ésimo elemento será substituído pelo valor de β_i original multiplicado por uma constante definida em cada simulação, dessa forma é possível obter diversos valores de β_i.

O valor de i é tomado de forma aleatória, entretanto esse valor é sempre reposicionado na primeira linha do vetor $\boldsymbol{\beta}$, ou seja, o valor de β_i é substituído por β_1 e esse substituído pelo o valor de β_i. Esse passo foi feito para tentar reduzir o viés de sempre pegar o mesmo valor para ser testado tendo em vista que os valores dos autovetores normalizados são sempre ordenados do maior para o menor.

Ademais, as quantidades ϵ_i serão simuladas através de valores gerados de uma distribuição normal com média 0 e variância σ^2.

Com os valores de $\mathbf{X}$, $\boldsymbol{\beta}$ e $\boldsymbol{\epsilon}$, serão simulados valores da variável resposta y utilizando o modelo de regressão descrito em (2.1). Com objetivo de abordar diferentes tipos situações, serão utilizados diversos valores de n, p, ρ, σ^2 e α descritos na Tabela 6.1.

Tabela 6.1: *Valores dos parâmetros para o estudo de simulação*

Parâmetro	Valores
n	50; 250
p	5; 10; 25
ρ	0,5; 0,7; 0,9
σ	0,1; 1,0; 5,0
α	0,01; 0,05; 0,10

Para cada uma das combinações distintas desses parâmetros, serão gerados uma matriz $\mathbf{X}$, dois vetores de $\boldsymbol{\beta}$ correspondendo à melhor e à pior orientação, e 500 réplicas do vetor aleatório de erros $\boldsymbol{\epsilon}$, sendo possível reproduzir 500 $\mathbf{y}$ distintos para ambas orientações do vetor $\boldsymbol{\beta}$, totalizando 1000 vetores de variável resposta.

Então, utilizando apenas a matriz $\mathbf{X}$ e o vetores $\mathbf{y}$, serão feitas as estimações dos coeficientes de regressão através do método de Regressão em Cristas, e para isso serão usados 9 métodos diferentes para estimar o valor de k, que são: $\hat{k}_{\text{HKB}}$, $\hat{k}_{\text{MG}}$, $\hat{k}_{\text{LW}}$, $\hat{k}_{\text{mg}}$, $\hat{k}_{\text{KS}}$, $\hat{k}^{\text{AKS}}_{\text{ma}}$, $\hat{k}^{\text{AKS}}_{\text{max}}$, $\hat{k}_{\text{DK}}$ descritos no Capítulo 3 e também $\hat{k}_{\text{MMQ}} = 0$.

Como será utilizado o procedimento de *bootstrap* para construir intervalos de confiança, cada par $(\mathbf{x}_i, y_i)$ serão re-amostrados $B = 5000$ vezes e, em cada re-amostragem, os coeficientes do vetor $\boldsymbol{\beta}$, de dimensão p, são estimados. Dessa forma é possível ter uma ideia da distribuição do estimador e construir os intervalos de confiança para β_1 utilizando cada um dos quatro métodos discutidos no Capítulo 5.

Para avaliar os resultados, serão utilizadas as três métricas. A primeira é a precisão ou acurácia estimada do intervalo, representada pela frequência de intervalos que contém o verdadeiro valor de β_i utilizado para gerar o vetor resposta, essa quantidade precisa se aproximar do nível de confiança $1 - \alpha$, e a forma de interpretar é quanto maior o valor, melhor é o resultado.

A segunda métrica é o comprimento estimado do intervalo de confiança, que é medido pela

diferença em valor absoluto entre o limite superior e o limite inferior do intervalo, para um valor fixado de α, o resultado satisfatório seria um intervalo de menor comprimento. Os resultados serão apresentados através da distribuição dos comprimento dos intervalos de confiança que será representado em cada *boxplot*.

Por fim, a terceira métrica usada é uma estimativa do poder do teste que mensura a probabilidade de rejeitar a hipótese nula, H_0, dado que a hipótese alternativa, H_1, é verdadeira. Para realizar essa estimativa, a frequência de intervalos de confiança em que o valor zero não estava presente no intervalo será calculada em intervalos de valores verdadeiro de β_i.

Os resultados serão apresentados através de gráficos, para cada uma das métricas será apresentado um gráfico com o resultado geral dividido por diferentes valores de α e pelo parâmetro selecionado e outro gráfico em que o resultado será mostrado por diferentes valores de σ, número de condição, α fixado em 0,05, para cada parâmetro. Como há um paralelismo entre o número de condição e o parâmetro ρ, esse último será então analisado através apenas do número de condição.

6.3 Resultados

6.3.1 Tipo de Intervalo de Confiança por *bootstrap*

Foram testados quatro tipos diferentes de intervalo de confiança pelo método de *bootstrap*, são eles: método básico, normal, percentil e BC_a. Todos esses foram discutidos no Capítulo 5.

Devido à escolha de simular os valores de ϵ_i através da distribuição normal, não é esperado grandes diferenças entre os resultados dos tipos de intervalo de confiança, pelo fato de que a distribuição normal é simétrica e com caudas leves.

Ao observar as Figuras 6.1 e 6.2, é possível constatar que os resultados da simulação mostram que, em relação à acurácia do intervalo de confiança, não há diferença entre os tipos de intervalo de confiança se o valor de α for o mesmo. Na Figura 6.2, o valor de α é fixado em 0,05 e são observados diferentes valores de σ e do número de condição.

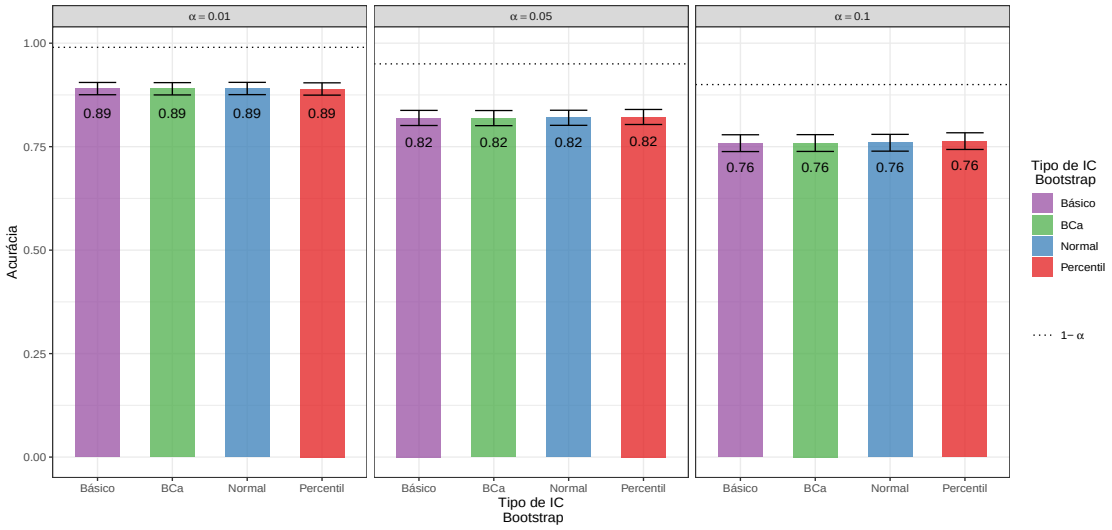

Figura 6.1: *Acurácia estimada por simulações dos intervalos de confiança para diferentes valores de α e do tipo de intervalo de confiança por bootstrap.*

Entretanto, é notado uma diminuição na acurácia estimada dos intervalos de confiança cons-

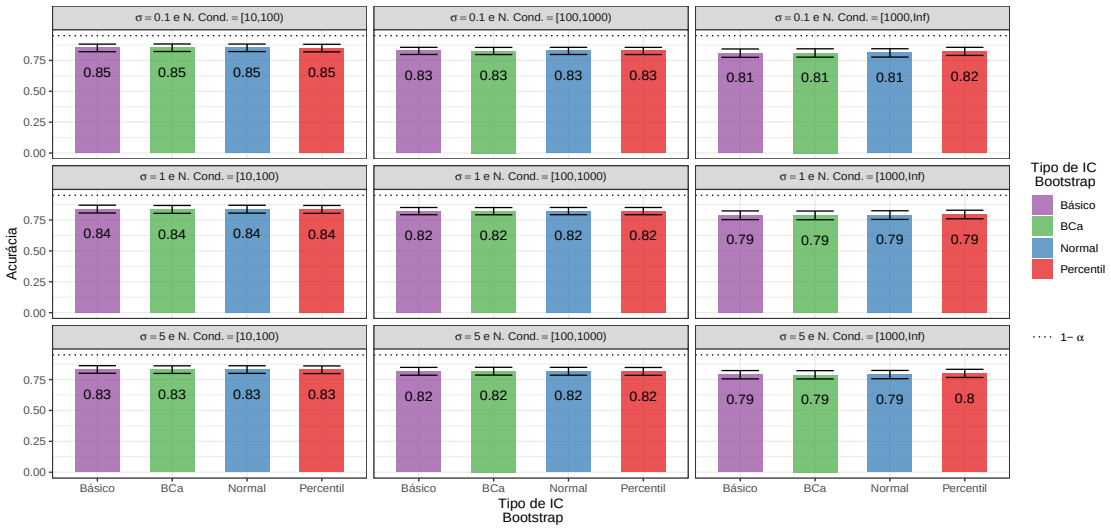

Figura 6.2: *Acurácia estimada por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e do tipo de intervalo de confiança por bootstrap.*

truídos com valores de σ e número de condição maiores. Em relação ao comprimento do intervalo de confiança, é possível constatar um comportamento similar à acurácia, em que não há diferença entre os tipos de intervalos de confiança, mas em geral, o comprimento do intervalo aumentou para valores maiores de σ e número de condição, e é possível analisar esse comportamento através das Figuras 6.3 e 6.4.

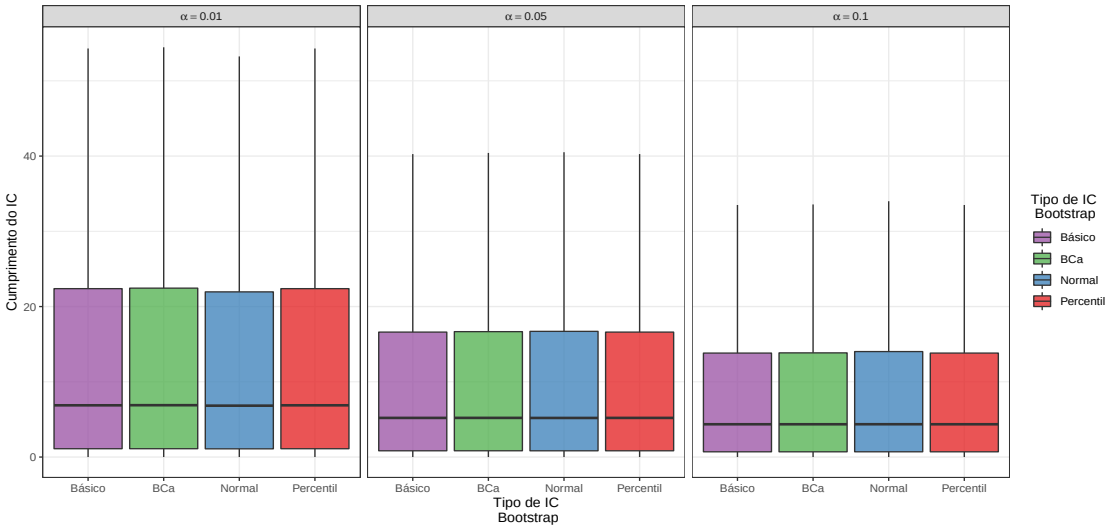

Figura 6.3: *Comprimento estimado por simulações dos intervalos de confiança para diferentes valores de α e do tipo de intervalo de confiança por bootstrap.*

Nas Figuras 6.5 e 6.6 estão representados os resultados para a estimativa do poder do teste, e através dessas figuras é possível ver que os intervalos de confiança construídos utilizando o método de percentil apresentaram estimativa para o poder do teste inferior aos outros tipos. Como esperado, diferentes valores para α e σ afetam o poder estimado do teste, além disso é notável também uma diminuição no poder estimado do teste à medida que o número de condição aumenta.

O tipo de intervalo de confiança não afeta os resultados, entretanto, vale ressaltar que isso pode ter ocorrido devido a escolha de usar valores simulados através da distribuição normal para os

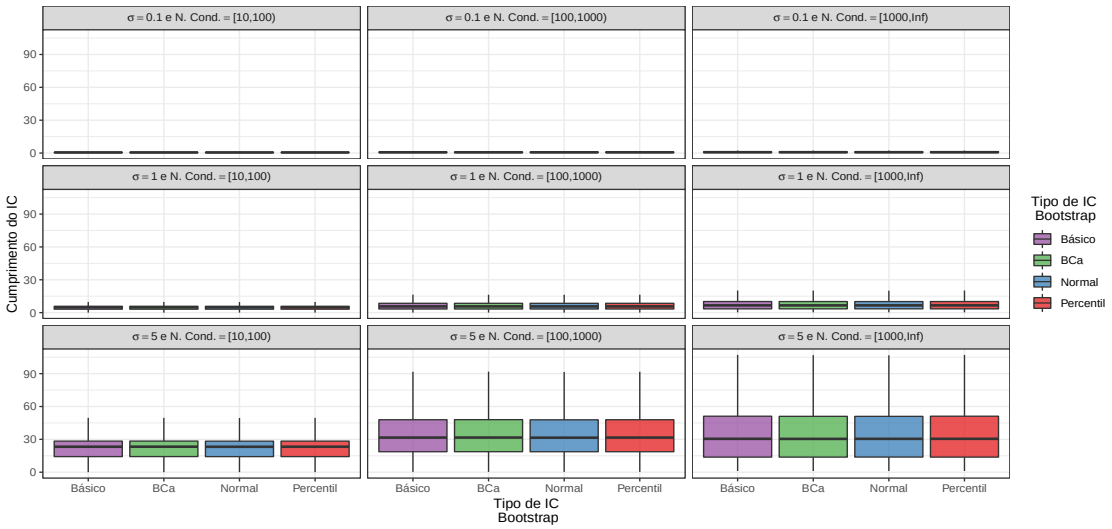

Figura 6.4: *Comprimento estimado por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e do tipo de intervalo de confiança por bootstrap.*

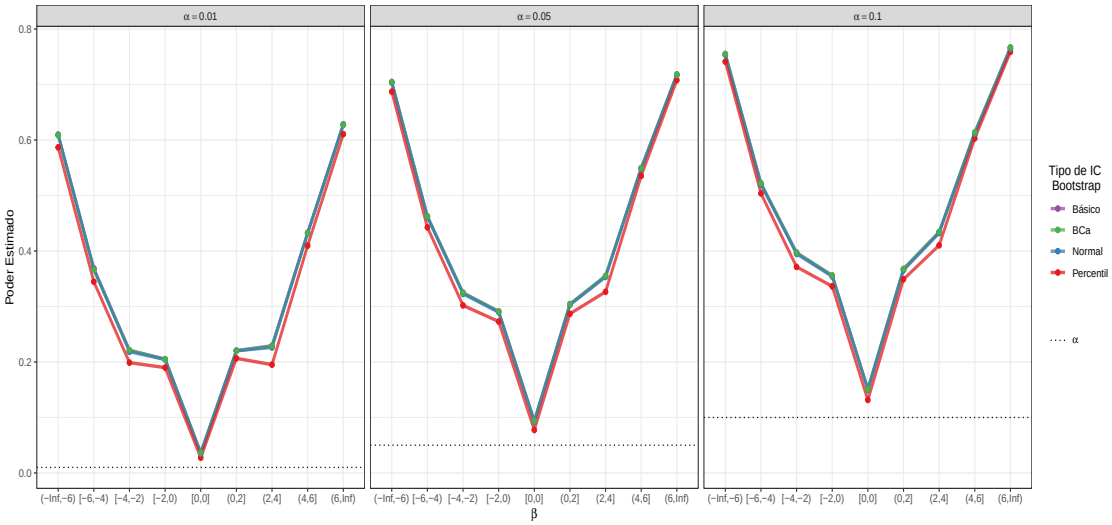

Figura 6.5: *Poder do teste estimado por simulações dos intervalos de confiança para diferentes valores de α e do tipo de intervalo de confiança por bootstrap.*

elementos do vetor de erros aleatórios. Nas próximas seções, as análise serão feitas utilizando todos os tipos de intervalos de confiança.

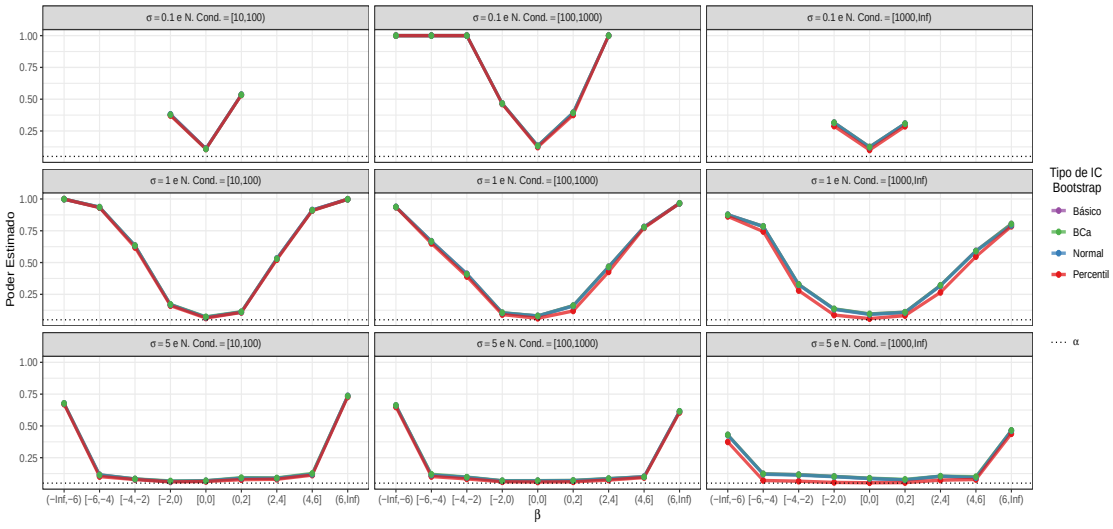

Figura 6.6: *Poder do teste estimado por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e do tipo de intervalo de confiança por bootstrap.*

6.3.2 Método de estimação do valor para k

Nesse estudo de simulação foram utilizados oito métodos de estimação para o valor de k, além do método de mínimos quadrados que representa $k = 0$. Todos esses métodos foram discutidos no Capítulo 3.

No Capítulo 3 também foi discutido que é difícil comprovar teoricamente qual o melhor método de estimação para k, entretanto, em situações em que há presença de multicolinearidade, o método de mínimos quadrados tem desempenho inferior em relação aos demais.

Através das Figuras 6.7 e 6.8, é possível constatar que os resultados das simulações mostram um maior desempenho, em geral, do método de mínimos quadrados em termos de acurácia, esse resultado é esperado pois o método de mínimos quadrados é não-viciado para estimar $\boldsymbol{\beta}$.

Dentre os métodos de estimação para o valor de k, os que se destacam positivamente em termos de acurácia estimada são os métodos propostos por Alkhamisi *et al.* (2006) ($\hat{k}_{\text{MA}}^{\text{AKS}}$) e Khalaf e Shukur (2005) ($\hat{k}_{\text{KS}}$). Os métodos propostos por Lawless e Wang (1976) ($\hat{k}_{\text{KW}}$) e Kibria (2003) ($\hat{k}_{\text{mg}}$) apresentaram acurácia estimada bem abaixo dos outros métodos estudados.

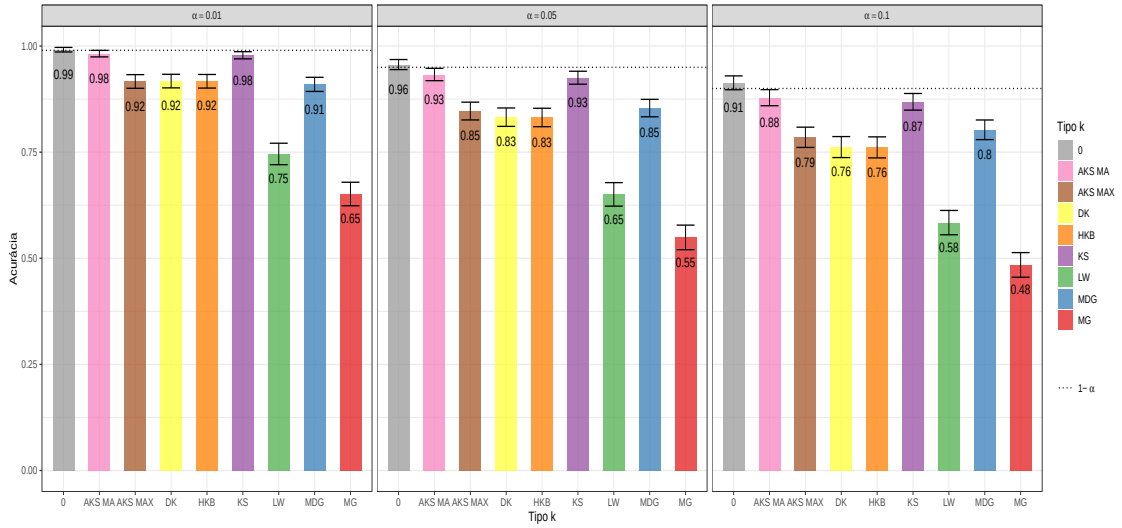

Figura 6.7: *Acurácia estimada por simulações dos intervalos de confiança para diferentes valores de α e métodos de estimação para o valor de k.*

Em termos de comprimento, os métodos de estimação para k que resultaram, em média, em intervalos de confiança por *bootstrap* com maior amplitude foram os métodos de mínimos quadrados, e os propostos por Alkhamisi *et al.* (2006) ($\hat{k}_{\text{MA}}^{\text{AKS}}$), Khalaf e Shukur (2005) ($\hat{k}_{\text{KS}}$) e McDonald e Galarneau (1975) ($\hat{k}_{\text{MDG}}$).

Em contrapartida, os métodos propostos por Lawless e Wang (1976) ($\hat{k}_{\text{KW}}$) e Kibria (2003) ($\hat{k}_{\text{mg}}$), apesar de terem resultado em baixa acurácia estimada, apresentaram pouca variabilidade na distribuição do comprimento dos intervalos de confiança além de valores medianos da amplitude dos intervalos menores em comparação com os outros métodos.

Além disso, é possível observar que o comprimento dos intervalos de confiança construídos através do método de regressão em cristas e pelo método de mínimos quadrados é influenciado por α (como esperado), σ e pelo número de condição. Esses resultados discutidos em relação ao comprimento dos intervalos de confiança podem ser constatados nas Figuras 6.9 e 6.10.

Considerando a hipótese H_0: $\beta_1 = 0$, os intervalos de confiança por *bootstrap* foram utilizados

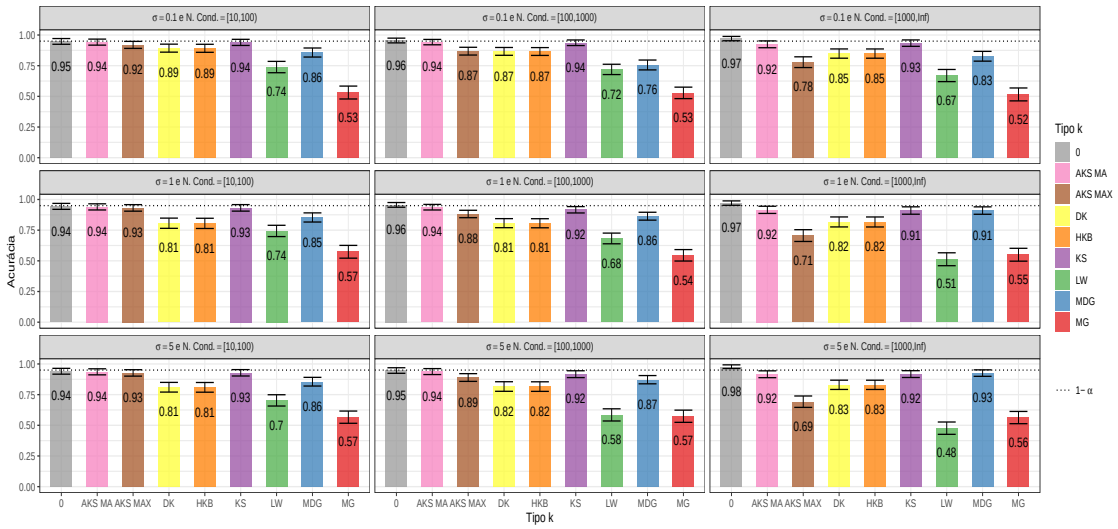

Figura 6.8: *Acurácia estimada por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e métodos de estimação para o valor de k.*

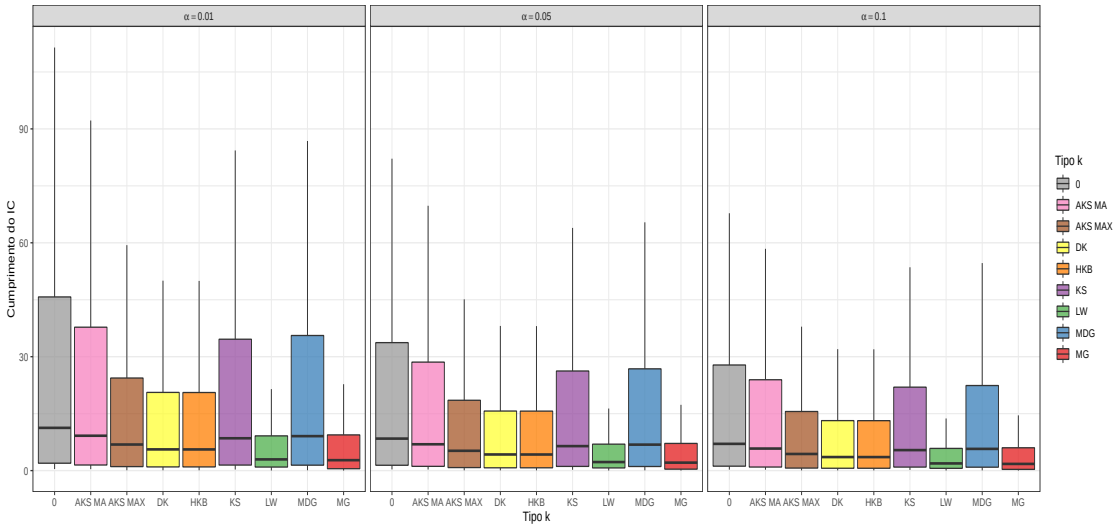

Figura 6.9: *Comprimento estimado por simulações dos intervalos de confiança para diferentes valores de α e métodos de estimação para o valor de k.*

para calcular uma estimativa para o poder do teste para $\alpha = 0{,}01$, 0,05 e 0,1 e avaliar qual método de estimação para k resulta em testes mais poderosos.

Os resultados sumarizados nas Figuras 6.11 e 6.12 mostram que o método proposto por Kibria (2003) ($\hat{k}_{\text{mg}}$) resultou em testes mais poderosos, para diversos valores de σ e número de condição. Outros métodos de estimação para k que também tiveram bons resultados em relação ao poder do teste foram os propostos por Dorugade e Kashid (2010) ($\hat{k}_{\text{DK}}$) e Hoerl *et al.* (1975) ($\hat{k}_{\text{HKB}}$).

O método proposto por McDonald e Galarneau (1975) apresentou bom poder do teste em situações em que o valor de σ é pequeno. O método de mínimos quadrados foi o que teve os piores resultados para o poder do teste, em alguns casos bem inferior aos demais métodos testados.

Aos analisar a acurácia, comprimento e a eficácia dos intervalos de confiança, os métodos propostos por Hoerl *et al.* (1975) (k_{HKB}), Dorugade e Kashid (2010) ($\hat{k}_{\text{DK}}$) e Alkhamisi *et al.* (2006) ($\hat{k}_{\text{max}}^{\text{AKS}}$) apresentaram resultados consistentes em relação aos demais, tendo bons resultados nos três quesitos testados.

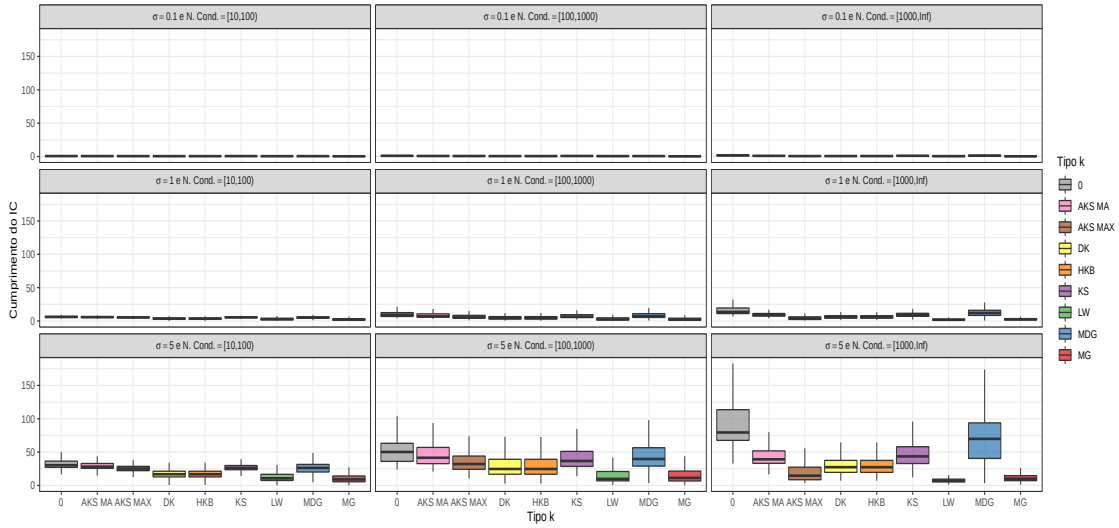

Figura 6.10: *Comprimento estimado por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e métodos de estimação para o valor de k.*

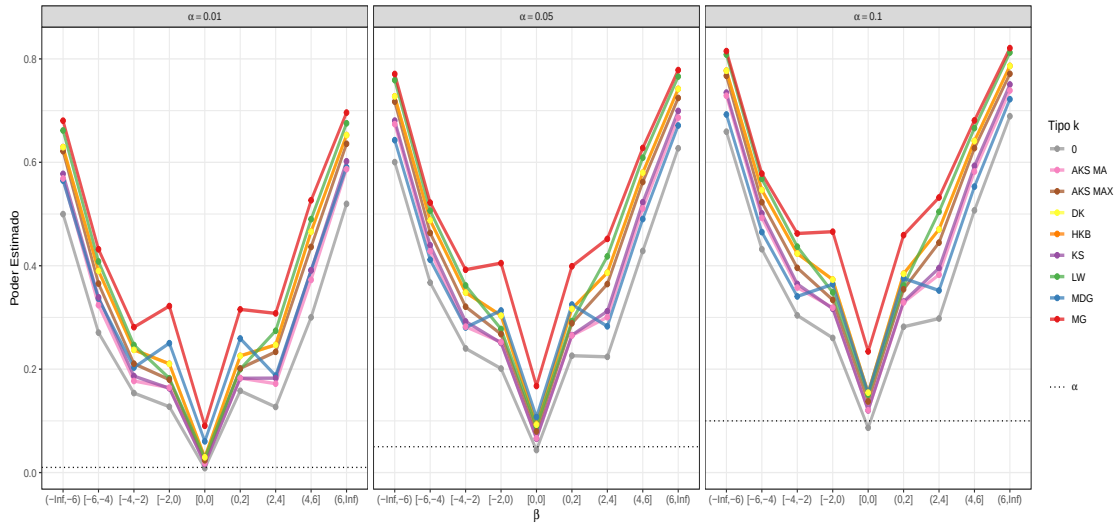

Figura 6.11: *Poder do teste estimado por simulações dos intervalos de confiança para diferentes valores de α e métodos de estimação para o valor de k.*

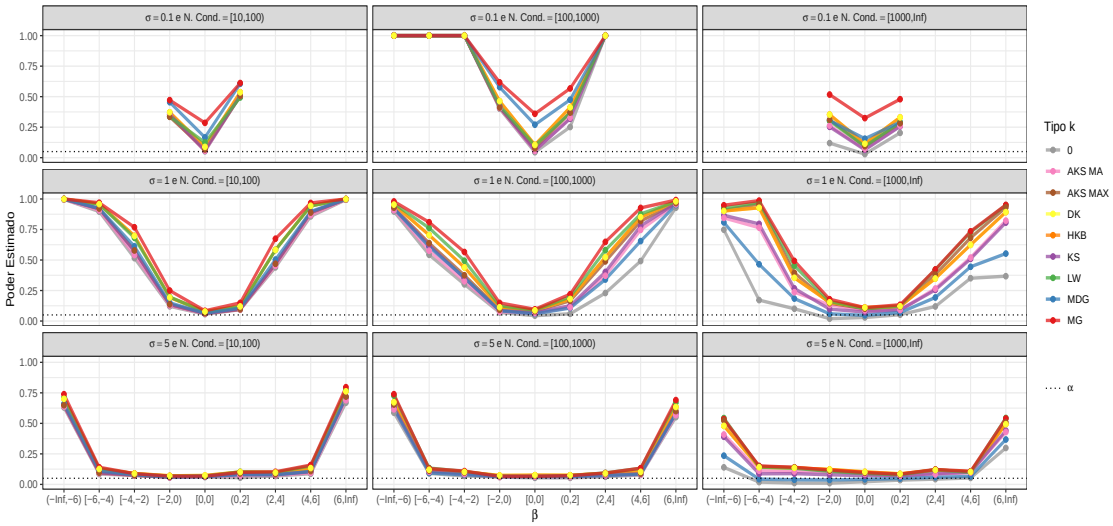

Figura 6.12: *Poder do teste estimado por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e métodos de estimação para o valor de k.*

6.3.3 Orientação do vetor de coeficientes

Para realizar as simulações, duas orientações para o vetor de coeficientes $\boldsymbol{\beta}$ foram testadas. A melhor orientação para $\boldsymbol{\beta}$ é quando esse é o autovetor correspondente ao maior autovalor da matriz $\mathbf{C}$, e a pior orientação é quando $\boldsymbol{\beta}$ é igual ao autovetor correspondente ao menor autovalor da matriz $\mathbf{C}$.

A justificativa para esse estudo foi baseada em Burr e Fry (2005), em que os autores comentam que a melhor orientação de $\boldsymbol{\beta}$ podem favorecer o método de regressão em cristas.

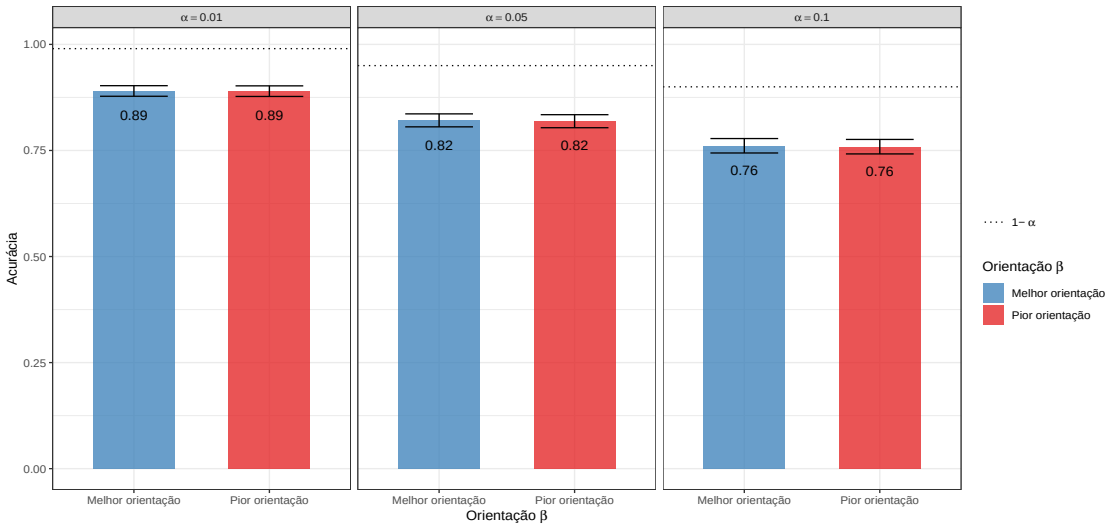

Figura 6.13: *Acurácia estimada por simulações dos intervalos de confiança para diferentes valores de α e orientações do vetor $\boldsymbol{\beta}$.*

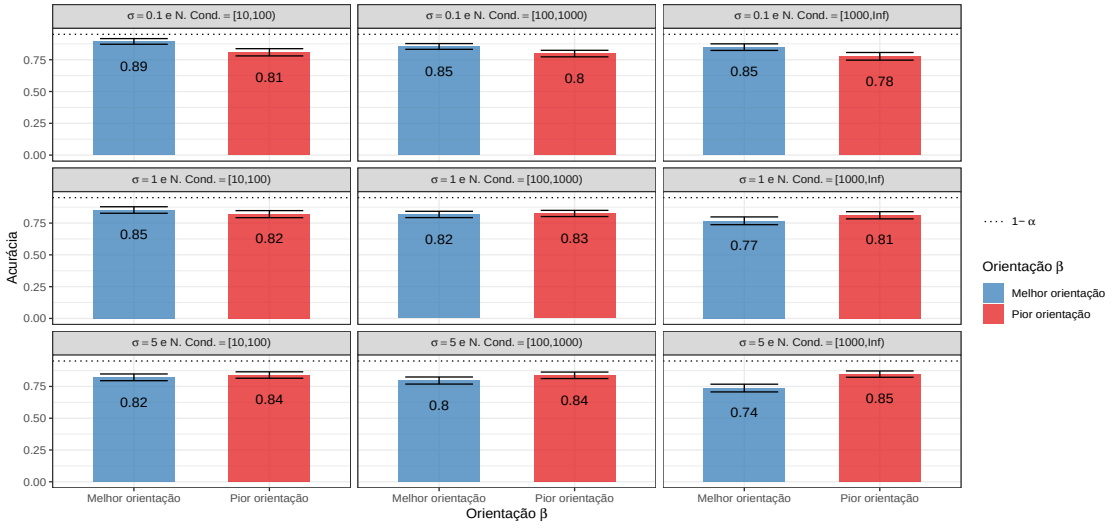

Figura 6.14: *Acurácia estimada por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e orientações do vetor $\boldsymbol{\beta}$.*

Ao observar a Figura 6.13, é possível ver que em geral não há muita diferença na acurácia dos intervalos de confiança para as duas diferentes orientações do vetor de coeficientes. Entretanto, ao analisar a Figura 6.14, é possível ver que são obtidos melhores resultados para a melhor orientação para valores menores de σ e melhores resultados para a pior orientação para valores maiores de σ.

O comprimento dos intervalos de confiança construídos sob a pior orientação de $\boldsymbol{\beta}$ são menores,

e uma justificativa pra isso seria o fato que o os coeficientes estão atrelados ao autovetor referente ao menor autovalor da matriz **C**.

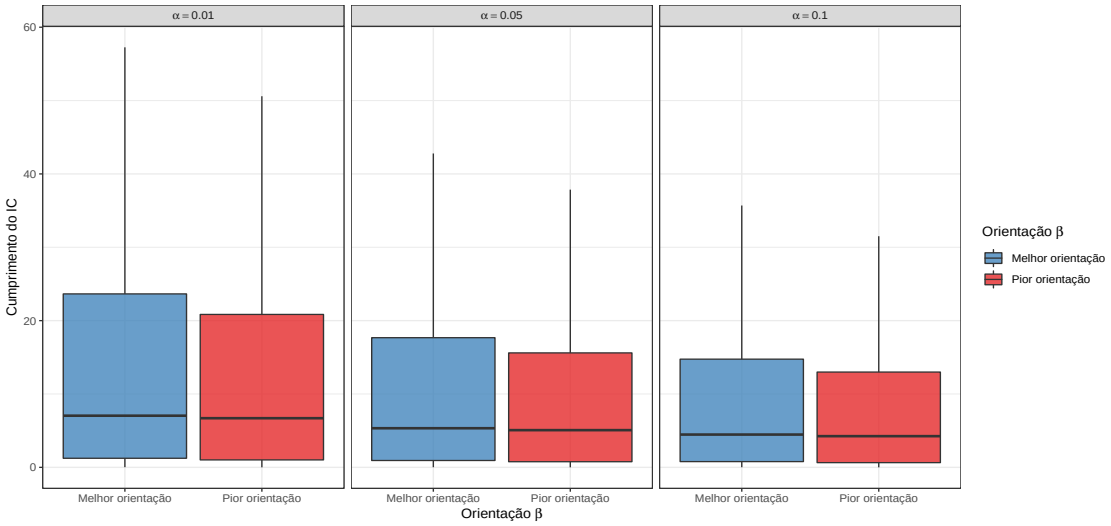

Figura 6.15: *Comprimento estimado por simulações dos intervalos de confiança para diferentes valores de α e do número de condição e orientações do vetor $\boldsymbol{\beta}$.*

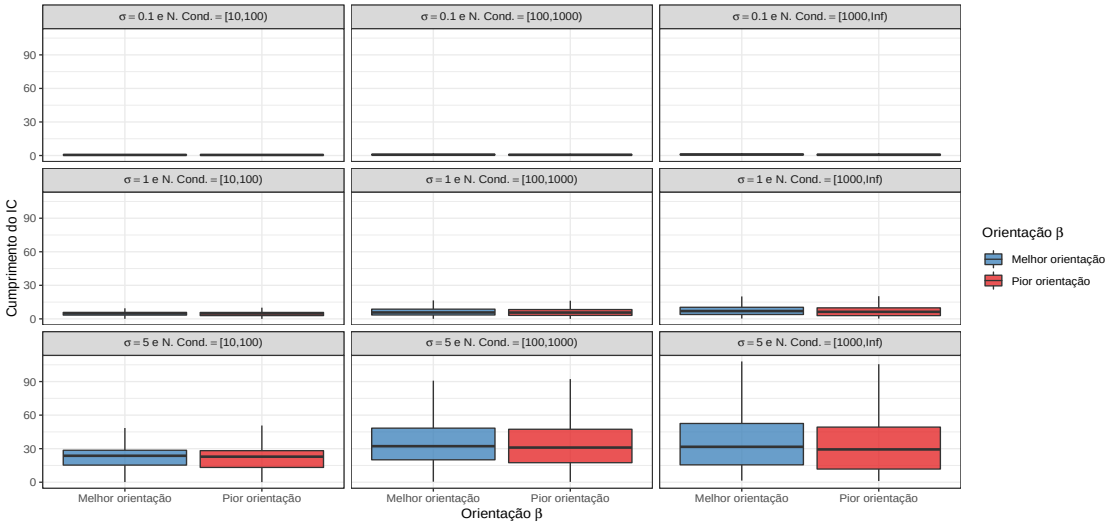

Figura 6.16: *Comprimento estimado por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e orientações do vetor $\boldsymbol{\beta}$.*

Entretanto, os intervalos construídos sob a melhor orientação são mais eficazes, ou seja, se o intervalo de confiança fosse usado para testar a hipótese nula de que o coeficiente é igual a zero, testes construídos sob a melhor orientação são mais poderosos, como pode ser visto nas Figuras 6.17 e 6.18.

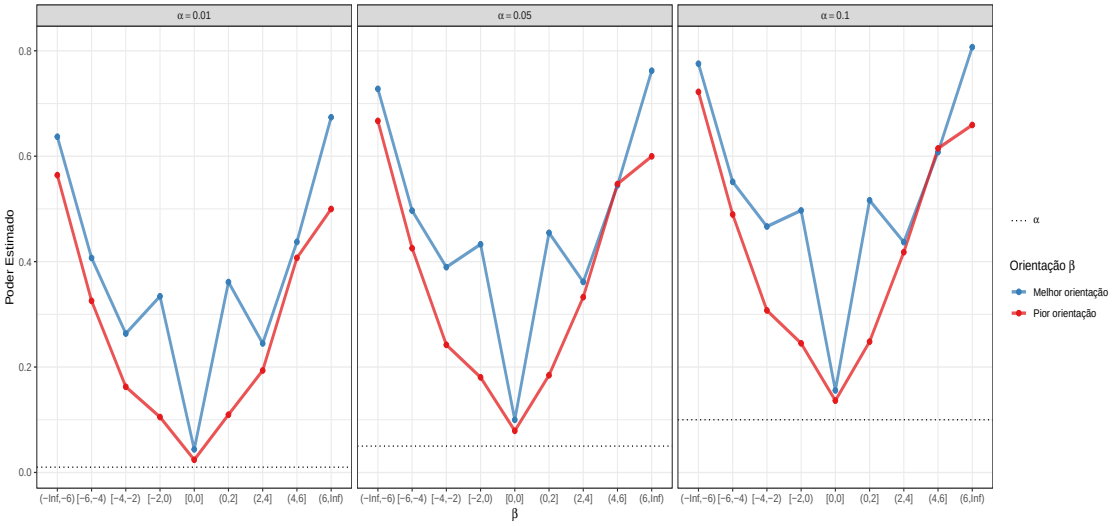

Figura 6.17: *Poder do teste estimado por simulações dos intervalos de confiança para diferentes valores de α e do número de condição e orientações do vetor $\boldsymbol{\beta}$.*

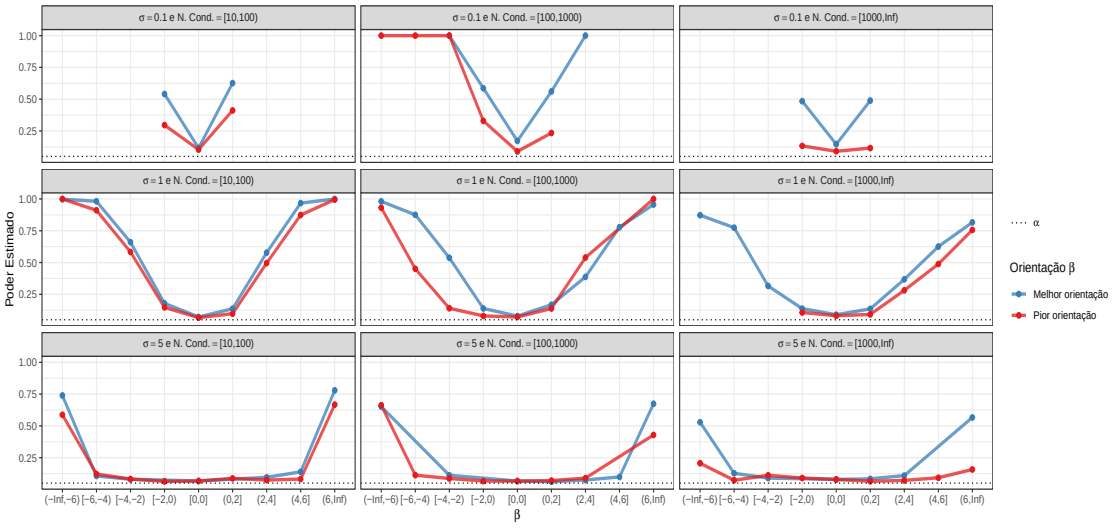

Figura 6.18: *Poder do teste estimado por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e orientações do vetor $\boldsymbol{\beta}$.*

6.3.4 Tamanho da amostra

Dois valores diferentes para o tamanho da amostra foram testados no estudo de simulações. Para construir os intervalos de confiança por *bootstrap* foram usadas amostras de tamanho 50 e 250.

Era esperado que o tamanho da amostra influenciasse na acurácia dos intervalos, e como pode ser constatado nas Figuras 6.19 e 6.20, tamanho maiores de amostra aumentam a acurácia dos intervalos de confiança construídos pelo método de *booostrap*.

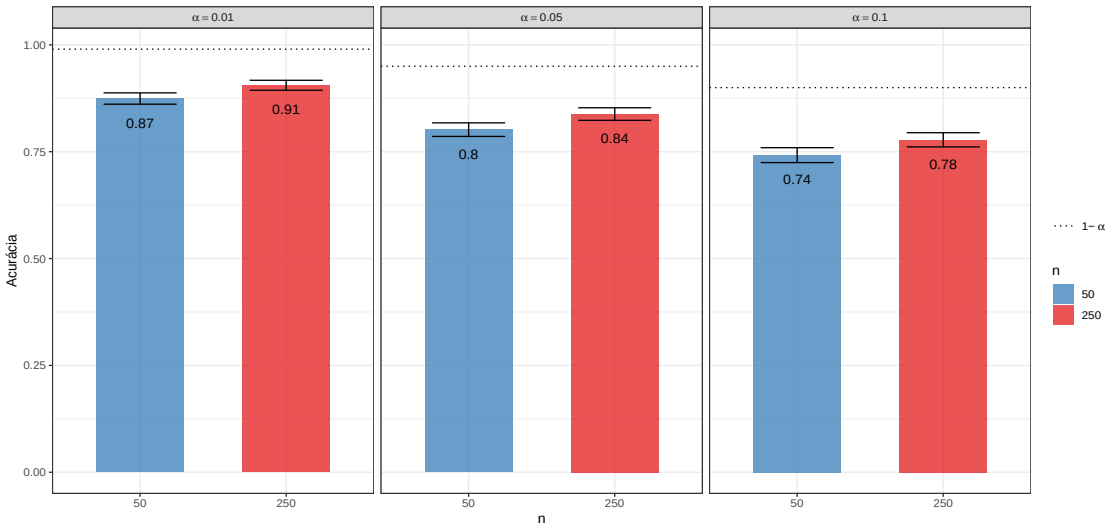

Figura 6.19: *Acurácia estimada por simulações dos intervalos de confiança para diferentes valores de α e tamanho de amostra.*

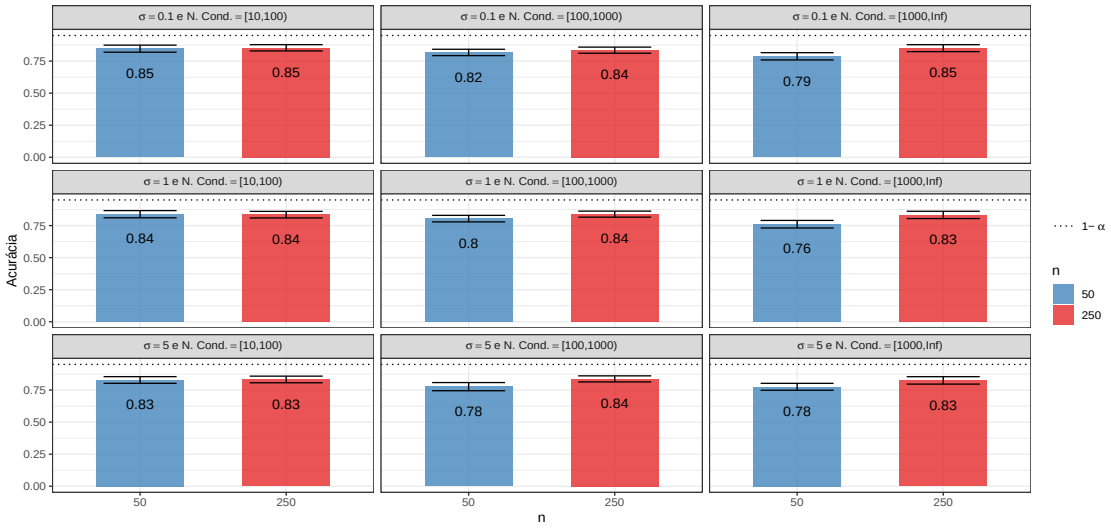

Figura 6.20: *Acurácia estimada por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e tamanho de amostra.*

É possível notar também, através das Figuras 6.21 e 6.22, que não há tanta diferença entre os dois tamanhos de amostra testado em relação ao comprimento do intervalo. E também pelas Figuras 6.23 e 6.24 que também não há muita diferença em relação ao poder do teste.

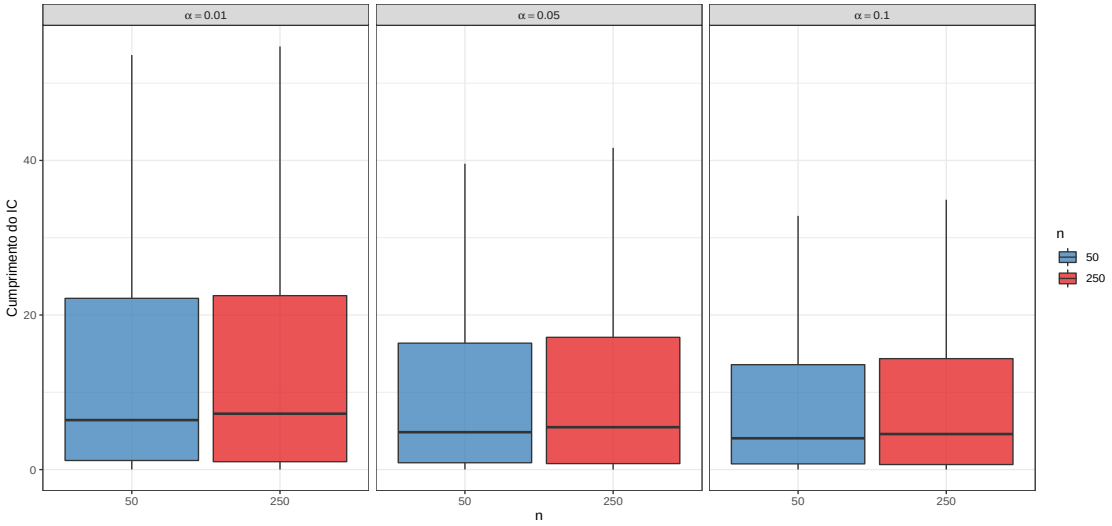

Figura 6.21: *Comprimento estimado por simulações dos intervalos de confiança para diferentes valores de α e do número de condição e tamanho de amostra.*

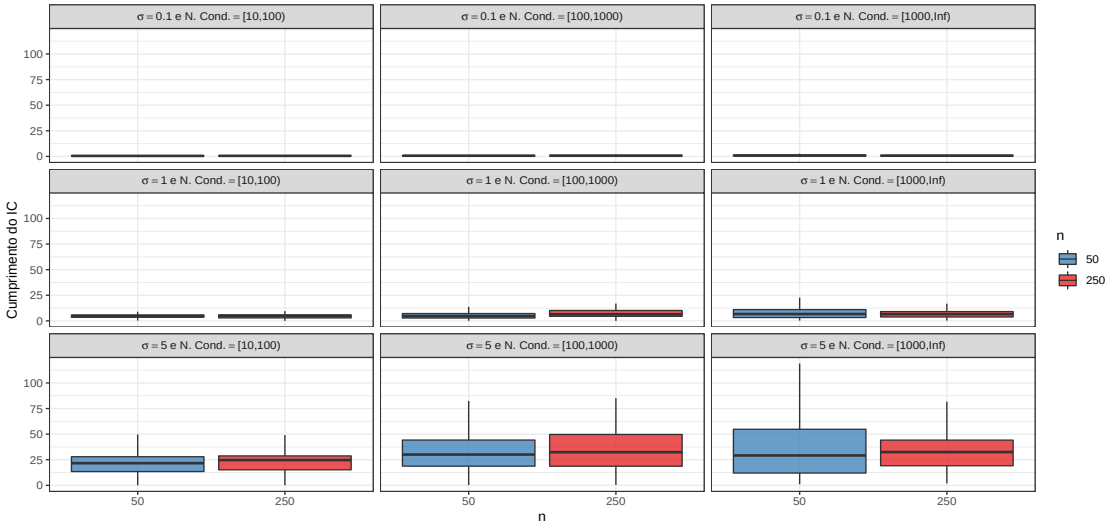

Figura 6.22: *Comprimento estimado por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e tamanho de amostra.*

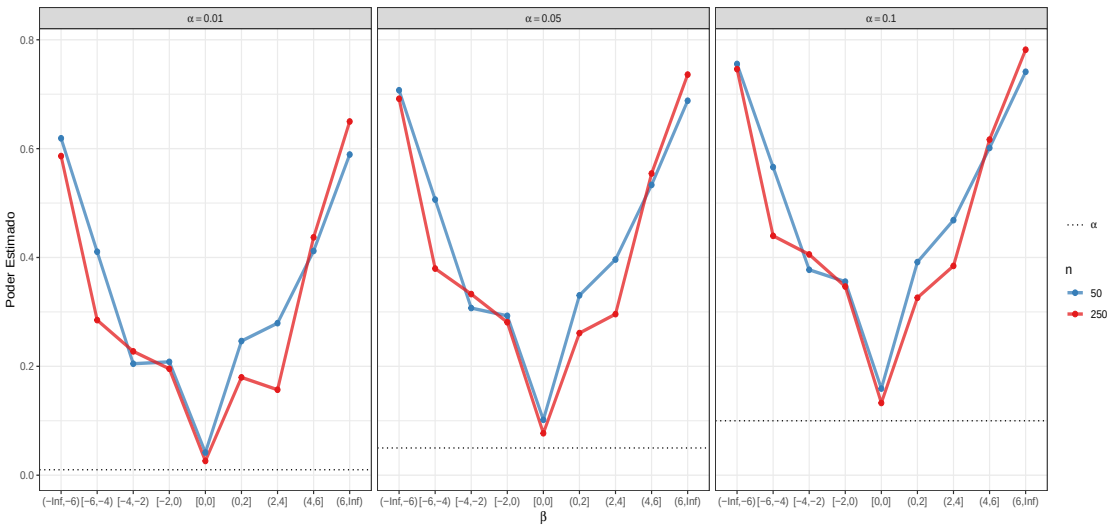

Figura 6.23: *Poder do teste estimado por simulações dos intervalos de confiança para diferentes valores de α e do número de condição e tamanho de amostra.*

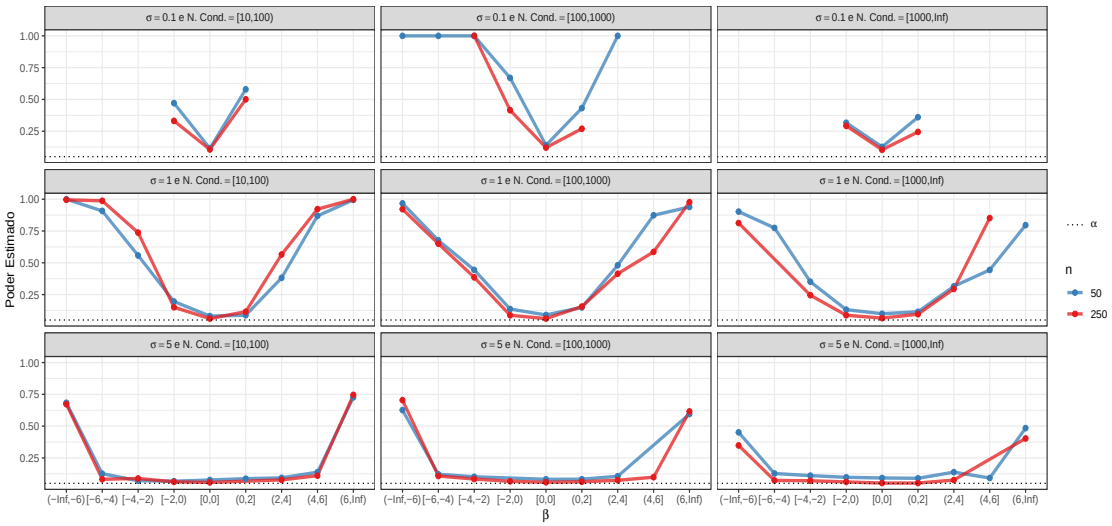

Figura 6.24: *Poder do teste estimado por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e tamanho de amostra.*

6.3.5 Número de variáveis explicativas

No estudo de simulações, três diferentes números de variáveis explicativas foram testados para avaliar a acurácia, comprimento e também o poder do teste. Os valores testados foram 5, 10 e 25 variáveis explicativas.

A Figura 6.25 mostra que em geral há uma leve piora na acurácia dos intervalos de confiança à medida que aumenta o número de variáveis explicativas. Entretanto, ao observar a Figura 6.26, é possível ver que essa diferença na verdade é devido ao fato que número maior de variáveis explicativas está atrelado a maiores números de condição, e esse interfere na acurácia do intervalo de confiança.

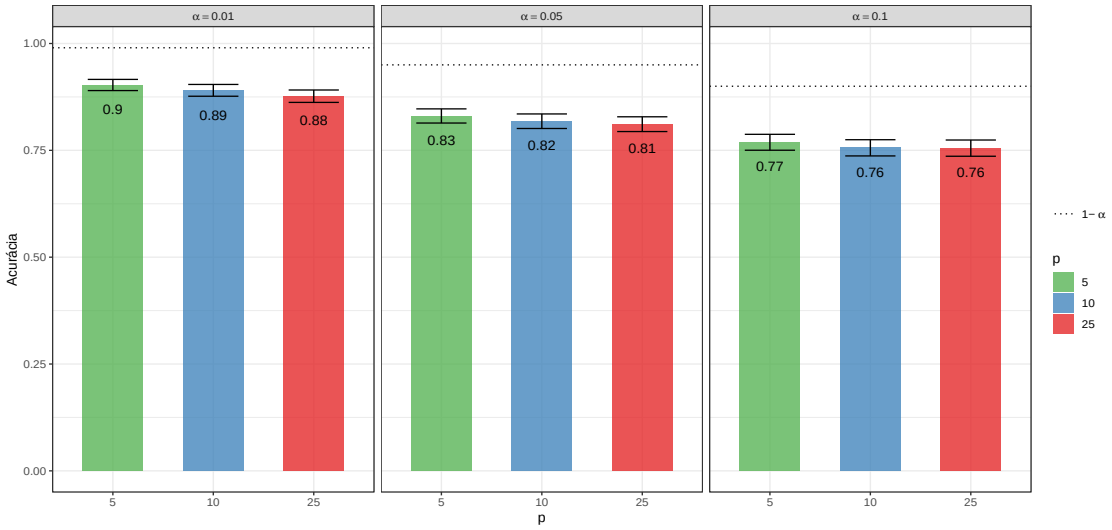

Figura 6.25: *Acurácia estimada por simulações dos intervalos de confiança para diferentes valores de α e número de variáveis explicativas.*

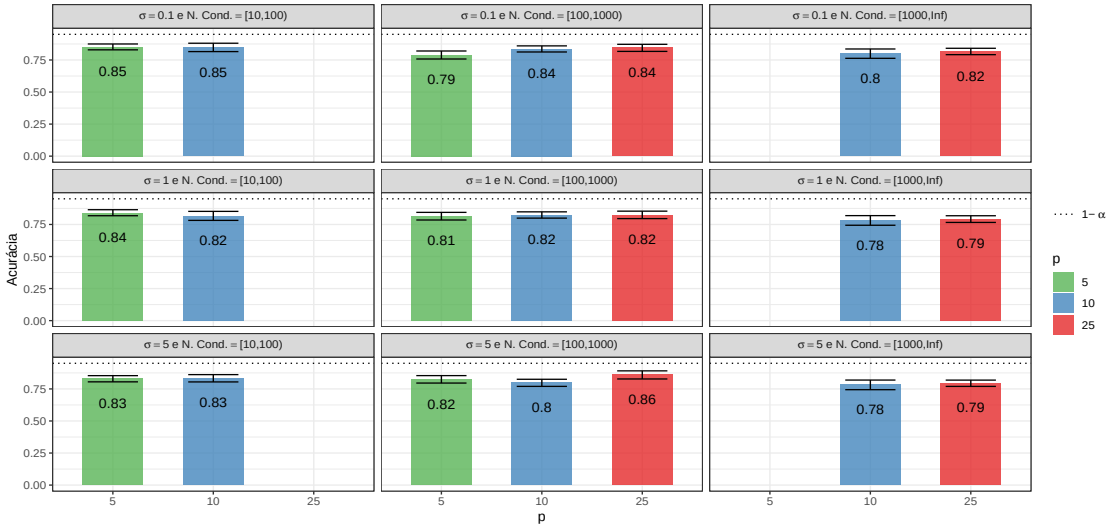

Figura 6.26: *Acurácia estimada por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e número de variáveis explicativas.*

Não há também diferença entre o comprimento do intervalo de confiança para esses valores testado do número de variáveis explicativas. É possível apenas notar uma maior variabilidade da distribuição dos comprimento dos intervalos construídos com número menor de variáveis explica-

tivas.

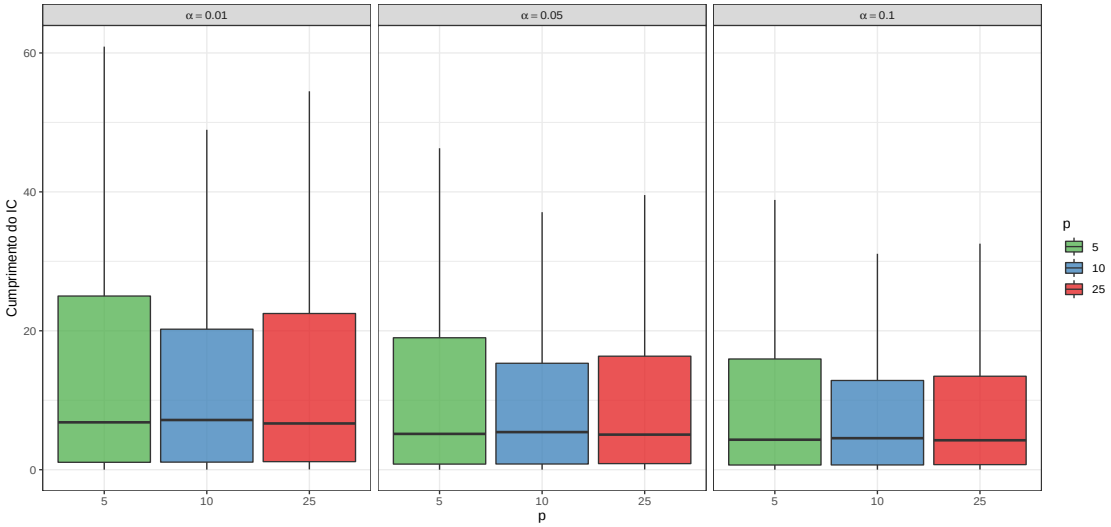

Figura 6.27: *Comprimento estimado por simulações dos intervalos de confiança para diferentes valores de α, e número de variáveis explicativas.*

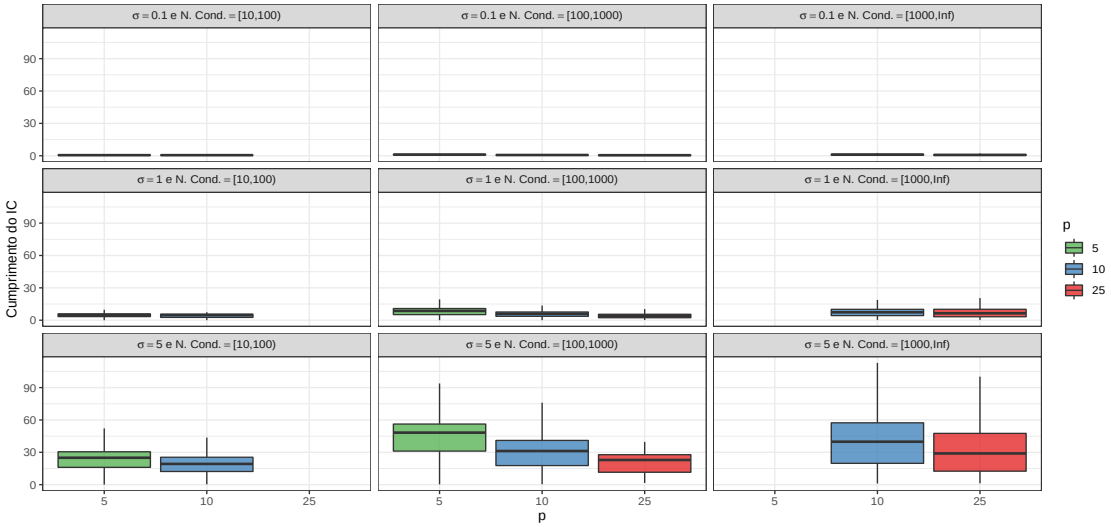

Figura 6.28: *Comprimento estimado por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e número de variáveis explicativas.*

O poder do teste parece ser, em geral, maior para as simulações realizadas com número menor de variáveis explicativas ser analisada através da Figura 6.29. Todavia, ao observar para diferentes valores de σ e número de condição e um nível de significância fixo na Figura 6.30, essa relação não parece ser bem estabelecida.

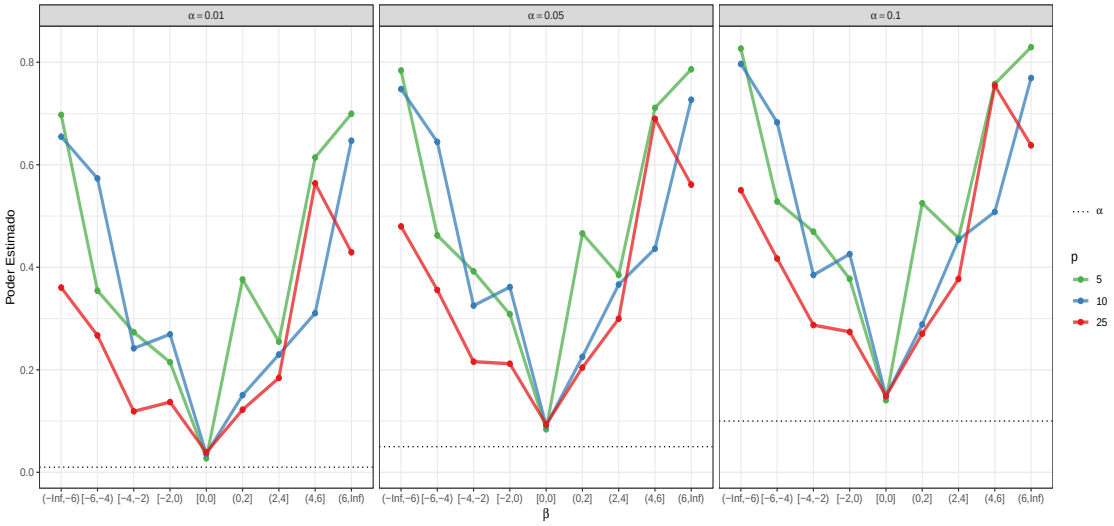

Figura 6.29: *Poder do teste estimado por simulações dos intervalos de confiança para diferentes valores de α, do número de condição e número de variáveis explicativas.*

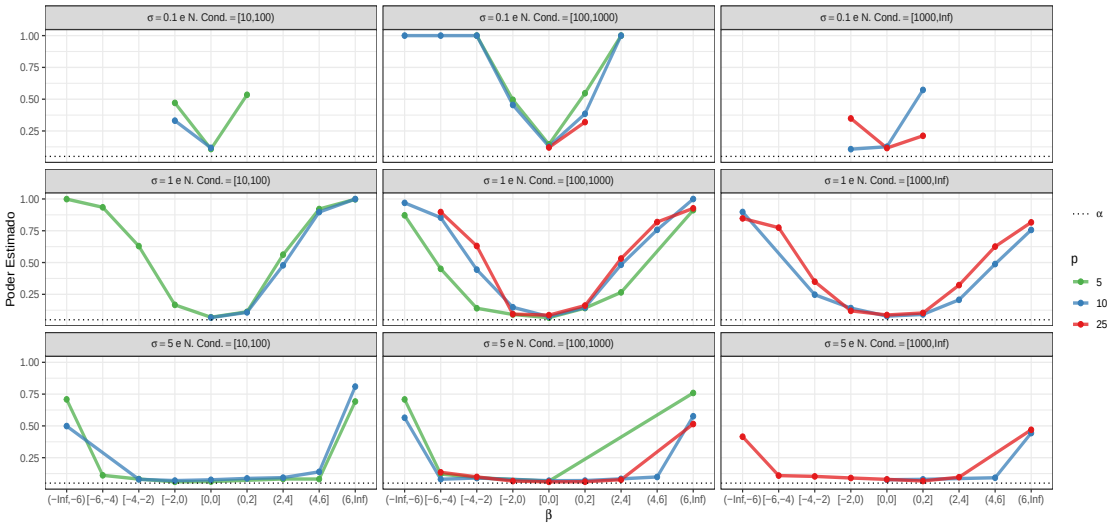

Figura 6.30: *Poder do teste estimado por simulações dos intervalos de confiança para $\alpha = 0{,}05$ e diferentes valores de σ, do número de condição e número de variáveis explicativas.*

Capítulo 7

Aplicação

7.1 Introdução

Nesse capítulo serão feitas aplicações dos métodos de regressão em cristas discutidos anteriormente ao conjunto de dados utilizado em um projeto da área de farmacologia que investiga o efeito de diversos tipos de anestésicos locais sobre o coração de ratos. O interesse deste estudo consiste em verificar quais características físico-químicas da molécula de determinada droga influenciam mais em sua potência tóxica, essa é definida como a dose de droga necessária para se obter uma redução de 30% na frequência do átrio.

Para realizar o estudo, foram utilizados setenta e dois ratos, homogêneos entre si, divididos em quatorze grupos, contendo de três a oito ratos. Cada grupo foi submetido a uma droga de interesse e a potência tóxica foi calculada após a aplicação. O estudo pode ser encontrado por: **Relação Estrutura-Atividade de Anestésicos Locais N.N [Dimetilamina] Etil Benzoatos Para-Substituídos**, (RAE-CEA-9710, André, Elian e Bruscato, 1997).

7.2 Descrição das Variáveis

Foram consideradas dez variáveis independentes, essa são características físico-químicas das próprias drogas, dessa forma, para cada droga seus valores são determinadas sem erro. As variáveis estão divididas em quatro grupos, de acordo com o efeito ao qual se referem.

As variáveis do grupo *efeito estérico* mediam um comprimento relacionado ao grupamento substituinte. São elas:

- **L**: comprimento do grupo substituinte ao longo do eixo da ligação com o esqueleto da molécula (medido em Ângstron);
- **B1** e **B4**: larguras do comprimento substituinte a partir do eixo da ligação, perpendiculares a ele (medidas em ângstron).

As variáveis do grupo *efeito eletrônico* mediam o efeito de dispersão de carga em torno do anel benzênico, essas são descritas como:

- **F**: componente de campo (adimensional);
- **R**: componente de ressonância (adimensional);

- **SIGMA**: constante de Hammet - combinação linear de **F** e **R** (adimensional);
- **C (CARBONILA)**: frequência de estiramento da carbonila medida por espectroscopia no infra-vermelho (em cm^{-1}).

As variáveis do grupo do *efeito hidrofóbico* mediam quanto as moléculas da droga misturam-se na água definidas como:

- **logPAPP**: logaritmo do coeficiente de partição óleo-água medido (adimensional);
- **PI**: coeficiente de partição óleo-água calculado (adimensional).

A variável do grupo *outros efeitos* é:

- **MR4**: refratividade molar (adimensional),

e a variável resposta é dada por:

- **POTÊNCIA**: $-\log(DE_{30})$, onde DE_{30} é a dose de droga necessária para ocorrer uma redução de 30% na frequência do átrio em relação ao controle (adimensional).

7.3 Modelagem através do método de Regressão em Cristas

Nessa seção será feito um ajuste no conjunto de dados descritos em 7.2. Inicialmente, será feito uma padronização das variáveis explicativas de forma que C seja a matriz de correlação entre elas.

A matriz de correlação das variáveis explicativas do conjunto de dados é apresentada na Tabela 7.1, nela é possível observar que os pares de variáveis PI e logPAPP, MR4 e L, MR4 e B4, L e B4, C e R, C e SIGMA têm coeficiente de correlação linear acima de 0,90 em valor absoluto. Além disso, é possível observar que outros pares de variáveis tem dependência linear forte.

Tabela 7.1: *Matriz de correlação linear entre as variáveis explicativas do conjunto de dados dos ratos.*

	B1	B4	L	SIGMA	F	R	C	logPAPP	PI	MR4
B1	1,00	-0,26	-0,32	0,81	0,40	0,70	0,78	-0,41	-0,36	-0,28
B4	-0,26	1,00	0,97	-0,53	-0,15	-0,56	-0,63	0,80	0,77	0,98
L	-0,32	0,97	1,00	-0,50	-0,04	-0,57	-0,61	0,79	0,76	0,99
SIGMA	0,81	-0,53	-0,50	1,00	0,55	0,86	0,93	-0,67	-0,61	-0,50
F	0,40	-0,15	-0,04	0,55	1,00	0,07	0,29	-0,49	-0,53	-0,11
R	0,70	-0,56	-0,57	0,86	0,07	1,00	0,94	-0,54	-0,45	-0,52
C	0,78	-0,63	-0,61	0,93	0,29	0,94	1,00	-0,60	-0,52	-0,59
log_PAPP	-0,41	0,80	0,79	-0,67	-0,49	-0,54	-0,60	1,00	0,99	0,78
PI	-0,36	0,77	0,76	-0,61	-0,53	-0,45	-0,52	0,99	1,00	0,76
MR4	-0,28	0,98	0,99	-0,50	-0,11	-0,52	-0,59	0,78	0,76	1,00

O número de condição da matriz de variáveis explicativas **C** é dado utilizando a formula em (2.4). Para esse conjunto de dados o número de condição é 12223,62, que é um número bastante elevado, indicando uma forte multicolinearidade. O FIV também foi calculado para cada uma das

variáveis explicativas e são descritos na Tabela (7.2), e podemos observar que todas as variáveis explicativas têm alto fator de inflação da variância.

Tabela 7.2: *Fator de Inflação da Variância para o conjunto de dados dos ratos.*

B1	B4	L	SIGMA	F	R	C	logPAPP	PI	MR4
10,83	292,46	286,18	658,58	216,51	471,79	61,63	258,55	290,16	448,55

O ajuste será realizado considerando o seguinte modelo

$$\begin{aligned} y_i = \beta_0 + \beta_1 \cdot \mathrm{B1}_i + \beta_2 \cdot \mathrm{B4}_i + \beta_3 \cdot \mathrm{L}_i + \beta_4 \cdot \mathrm{SIGMA}_i + \\ \beta_5 \cdot \mathrm{F}_i + \beta_6 \cdot \mathrm{R}_i + \beta_7 \cdot \mathrm{C}_i + \beta_8 \cdot \mathrm{logPAPP}_i + \beta_9 \cdot \mathrm{PI}_i + \beta_{10} \cdot \mathrm{MR4} + \epsilon_i, \end{aligned} \tag{7.1}$$

em que y_i denota a i-ésima observação da variável resposta POTÊNCIA, com $i = 1, \ldots, 72$. Devido à alta multicolinearidade presente na matriz $\mathbf{C}$, os coeficientes desse modelos serão estimados através do método de Regressão em Cristas.

O procedimento de modelagem é descrito nas seguintes etapas:

1. Determinar uma estimativa para k utilizando o método que resulte na menor estimativa do erro quadrático médio de $\hat{\boldsymbol{\beta}}^*_{(\hat{k})}$ dentre os métodos descritos no Capítulo 3.
2. Estimar os coeficientes do modelo descrito em (7.1) pelo método de Regressão em Cristas utilizando k definido no Passo 1.
3. Construir intervalos de 95% confiança através da técnica de *bootstrap* pelo método de BC_a para os coeficientes de regressão.
4. Eliminar uma variável explicativa do modelo em que o intervalo de confiança construído para o coeficiente no Passo 4 contenha o valor 0, priorizando a variável em que o intervalo de confiança seja de maior comprimento.
5. Repetir os Passos 1 a 4 até que todos os intervalos construídos no Passo 3 não contenham o valor 0.

Os resultados do ajuste do modelo de regressão descrito em (7.1) utilizando o método da Regressão em Cristas e considerando o procedimento de modelagem proposto serão apresentados em duas partes. A primeira parte serão mostrados os resultados para o modelo com todas as variáveis explicativas disponíveis e a segunda parte serão mostrados os resultados do modelo com as variáveis em que o intervalo de 95% de confiança não contenha o valor 0.

O modelo inicial com todas as variáveis explicativas foi ajustado utilizando a estimativa $\hat{k}^{\mathrm{AKS}}_{\mathrm{ma}}$, pois como pode ser visto na Tabela (7.3), apresentou o menor erro quadrático médio para $\hat{\boldsymbol{\beta}}^*_{(k)}$.

Tabela 7.3: *Estimativas para o valor de k obtidas por diversos métodos e a estimativa do erro quadrático médio correspondente para o modelo com todas as variáveis explicativas presentes.*

Método	$\hat{k}_{\mathrm{MMQ}}$	$\hat{k}_{\mathrm{HK}}$	$\hat{k}_{\mathrm{MDG}}$	$\hat{k}_{\mathrm{LW}}$	$\hat{k}_{\mathrm{MG}}$	$\hat{k}_{\mathrm{KS}}$	$\hat{k}^{\mathrm{AKS}}_{\mathrm{ma}}$	$\hat{k}^{\mathrm{AKS}}_{\mathrm{max}}$	$\hat{k}_{\mathrm{DK}}$
Estimativa	0,000	0,011	0,000	0,000	0,028	0,004	0,006	0,025	0,011
$\widehat{\mathrm{EQM}}(\hat{\boldsymbol{\beta}}^*_{(\hat{k})})$	142006,220	4,789	142006,220	142006,220	6,097	4,942	4,554	5,907	4,789

Os Passos 2 e 3 para o modelo inicial são descritos na Tabela 7.4. A primeira variável a ser eliminada foi B4, em seguida foram eliminadas as variáveis logPAPP, SIGMA, L e C.

Tabela 7.4: *Estimativas para os coeficientes do modelo com todas as variáveis explicativas presentes e o intervalo de 95% de confiança obtido pelo método de BC_a para os coeficientes.*

Variável	Estimativa	IC(95%)
Intercepto	3,651	[3,608; 3,692]
B1	0,652	[0,17; 1,156]
B4	0,057	[-1,35; 1,583]
L	-0,746	[-2,168; 0,333]
SIGMA	-1,349	[-2,572; -0,092]
F	-0,296	[-1,296; 0,469]
R	-2,348	[-3,408; -1,273]
C	1,025	[-0,184; 2,294]
logPAPP	1,694	[0,037; 3,034]
PI	1,493	[0,124; 3,075]
MR4	-0,368	[-1,607; 1,013]

O modelo com as variáveis explicativas que não apresentam o valor 0 no intervalo de 95% de confiança será composto pela variáveis explicativas B1, F, R, PI e MR4. O valor de k utilizado para estimar os coeficientes do modelo será calculado usando $\hat{k}_{MG}$, por apresentar o menor erro quadrático médio para $\hat{\boldsymbol{\beta}}^*_{(k)}$ comparado aos demais métodos, essa comparação pode ser vista na Tabela 7.5.

Tabela 7.5: *Estimativas para o valor de k obtidas por diversos métodos e a estimativa do erro quadrático médio correspondente para o modelo com as variáveis explicativas selecionadas.*

Método	$\hat{k}_{\text{MMQ}}$	$\hat{k}_{\text{HK}}$	$\hat{k}_{\text{MDG}}$	$\hat{k}_{\text{LW}}$	$\hat{k}_{\text{MG}}$	$\hat{k}_{\text{KS}}$	$\hat{k}^{\text{AKS}}_{\text{ma}}$	$\hat{k}^{\text{AKS}}_{\text{max}}$	$\hat{k}_{\text{DK}}$
Estimativa	0,000	0,006	0,003	0,000	0,025	0,003	0,007	0,015	0,003
$\widehat{\text{EQM}}(\hat{\boldsymbol{\beta}}^*_{(\hat{k})})$	3,833	1,999	2,654	3,833	0,905	2,654	1,845	1,163	2,654

Portanto, as estimativas dos coeficientes de regressão são calculadas através do método de Regressão em Cristas utilizando $k = 0,025$. O número de condição calculado utilizando (2.9) é 20,665, indicando fraca dependência linear entre as variáveis explicativas.

Tabela 7.6: *Estimativas para os coeficientes do modelo com as variáveis explicativas selecionadas e o intervalo de 95% de confiança feito pelo método de BC_a para os coeficientes.*

Variável	Estimativa	IC(95%)
Intercepto	3,650	[3,609; 3,692]
B1	0,617	[0,275; 1,088]
F	-0,839	[-1,376; -0,293]
R	-2,617	[-3,141; -2,161]
PI	2,827	[2,153; 3,488]
MR4	-0,852	[-1,465; -0,24]

A equação do modelo final é dada por

$$\hat{y}_i = 3,650 + 0,617 \cdot \text{B1} - 0,839 \cdot \text{F} - 2,617 \cdot \text{R} + 2,827 \cdot \text{PI} - 0,852 \cdot \text{MR4}. \tag{7.2}$$

É possível concluir que o método de Regressão em Cristas foi capaz de contornar o problema de multicolinearidade presente entre as variáveis explicativas do conjunto de dados. Além disso, observou-se a grande utilidade da construção de intervalos de confiança pelo método de *bootstrap* na seleção de variáveis.

Capítulo 8

Conclusões

8.1 Considerações Finais

A abordagem descrita nesse trabalho revelou que o método de mínimos quadrados é inadequado quando há presença de multicolinearidade e que o método de Regressão em Cristas é uma ótima alternativa. O trabalho realizado por Oishi (1983) reuniu boas referências sobre esse assunto, outros artigos relevantes foram desenvolvidos após a publicação desse trabalho.

Essa dissertação abordou artigos sobre propriedades dos estimadores obtidos pelo método de Regressão em Cristas, como sua relação com o estimador de mínimos quadrados e a decomposição do erro quadrático médio em duas partes, uma representando a soma de variâncias total dos estimadores e a outra parte é o quadrado do viés do estimador. A partir dessa decomposição, foi possível concluir que existe um valor de k em que o estimador em cristas tem erro quadrático médio menor do que o estimador pelo MMQ. Entretanto, k é uma quantidade desconhecida e precisa ser estimado.

Diversas formas de estimação para o parâmetro das cristas k foram abordadas nesse trabalho. Com base em estudos de simulação realizados por diversos autores para testar a eficiência em termos de erro quadrático médio, as formas de estimação apresentam bons resultados quando há uma forte multicolinearidade entre as variáveis explicativas, mas não há um critério que seja uniformemente melhor que os demais.

No presente estudo foram feitas discussões sobre técnicas de inferência para os coeficientes utilizando os estimadores da Regressão em Cristas, foram apresentados os testes $t_{(k)}$, $F_{(k)}$, a decomposição da soma de quadrados totais e o teste não-exato $t^{\text{ne}}_{(ke)}$. Entretanto, em aplicações práticas do método de Regressão em Cristas, k é estimado através dos dados, e portanto sua natureza é estocástica, e como consequência, os resultados dos testes não apresentam uma distribuição com forma fechada para comparação.

O método de *bootstrap* foi apresentado nesse trabalho como uma solução para realizar inferência para os coeficientes de regressão utilizando o estimador em cristas. O estudo de simulação realizado mostrou que intervalos de confiança construídos utilizando o método de *bootstrap* podem ser bem precisos em diversas situações. Além disso, devido ao grande número de cenários testados e a falta de uma rotina que calculasse um grande número de réplicas e re-amostragens para o método de *bootstrap* de forma eficiente, uma nova rotina de programação para calcular as estimativas e os intervalos de confiança por *bootstrap* utilizando os métodos das cristas foi desenvolvida com o uso das linguagens R Core Team (2019) e C++ (especificamente a biblioteca desenvolvida por

Eddelbuettel e Sanderson (2014)), uma versão inicial pode ser encontrada no apêndice.

Possíveis extensões desse trabalho seria ampliar a quantidade de níveis de correlação e tamanhos de amostras testadas no estudo de simulação, desenvolver uma seção da análise de influência no modelo de Regressão em Cristas, além de desenvolver mais sobre a visão bayesiana desse método. Uma outra possível extensão é estudar através de simulação os testes de hipóteses e a análise de variância discutidos no Capítulo 4. Além disso, calcular o erro quadrático médio junto com a acurácia dos intervalos de confiança por *bootstrap* seria uma forma de analisar de forma eficiente as vantagens de utilizar o método das cristas ao invés do método de mínimos quadrados em termos de nível de significância.

Apêndice A

Rotinas computacionais

Serão apresentados as rotinas computacionais para a obtenção dos intervalos de confiança pelo método das cristas, as estimativas dos coeficientes da regressão em Cristas e aplicação dos métodos no conjunto de dados.

A rotina em C++ precisa ser salva em dois arquivos distintos, a primeira parte com o nome *cpp_bootstrap_confidence_interval.h* e a segunda parte como *cpp_ridge_regression.cpp*.

A primeira parte da rotina é dada por:

```
1
2
3 // [[Rcpp::depends(RcppArmadillo)]]
4
5 #include <RcppArmadillo.h>
6
7
8
9 using namespace Rcpp;
10 using namespace arma;
11
12
13 Rcpp::NumericVector arma2vec(arma::vec x) {
14   return Rcpp::NumericVector(x.begin(), x.end());
15 }
16
17 // [[Rcpp::export]]
18 arma::mat cpp_normal_interpolation(arma::vec t, arma::vec alpha){
19   // Basic Boostrap Confidence Method
20
21   // if alpha equal to 1, then substitute to 0.999
22   alpha.replace(1, .999);
23
24
25   arma::vec finite_t = t.elem(find_finite(t));
26   int R = finite_t.n_elem;
27   arma::vec rk = (R+1) * alpha;
28   arma::vec k = trunc(rk);
29
30
31   // values of k that are not 0 or R
```

```
  arma::uvec ind_kvs = find(k > 0 and k < R);
  arma::vec kvs = k.elem(ind_kvs);

  // sorted t
  arma::vec sorted_t = sort(t);

  // output
  arma::vec out(alpha.n_elem);

  // find integers
  arma::uvec int_rk = find(k == rk);
  if(int_rk.n_elem > 0){
    arma::uvec ind_sorted_t = conv_to<uvec>::from(k.elem(int_rk));
    out.elem(int_rk) = sorted_t.elem(ind_sorted_t);
  }

  // Find k==0 or k == R
  arma::uword Ri = R;
  out.elem(find(k == 0)).fill(sorted_t(0));
  out.elem(find(k == R)).fill(sorted_t(Ri-1));

  // find non-interger, non-0 and non-R
  arma::uvec ind_not = find(k != rk and k > 0 and k < R);

  // interpolate
  arma::vec temp1 = Rcpp::qnorm(arma2vec(alpha.elem(ind_not)));
  arma::vec temp2 = Rcpp::qnorm(arma2vec(k.elem(ind_not)/(R+1)));
  arma::vec temp3 = Rcpp::qnorm(arma2vec((k.elem(ind_not) + 1)/(R+1)));

  // tks
  arma::uvec ind_tk = conv_to<uvec>::from(k.elem(ind_not) - 1);
  arma::uvec ind_tk1 = conv_to<uvec>::from(k.elem(ind_not));
  arma::vec tk = sorted_t(ind_tk);
  arma::vec tk1 = sorted_t(ind_tk1);

  // complete output
  out.elem(ind_not) = tk + (temp1-temp2)/(temp3-temp2)%(tk1 - tk);

  // output matrix
  arma::mat out_mat = join_rows(rk, out);

  return out_mat;
}

// [[Rcpp::export]]
arma::mat cpp_mat_normal_interpolation(arma::mat t, arma::mat alpha){
  mat out_mat(2*alpha.n_rows, alpha.n_cols);

  for(uword i = 0; i < t.n_cols; ++i){
    mat aux_i = cpp_normal_interpolation(t.col(i), alpha.col(i));
    out_mat.col(i) = vectorise(aux_i);
  }

```

```
85
86   return out_mat;
87 }
88
89
90 // [[Rcpp::export]]
91 arma::mat cpp_boot_basic_ci(arma::mat t0, arma::cube t, arma::vec conf){
92   // output matrix
93   arma::mat mat_ci_output(t.n_cols * t.n_slices * conf.n_elem, 7);
94
95   // creating alpha
96   arma::mat alpha(conf.n_rows, 2);
97   arma::vec uns(conf.n_elem, fill::ones);
98   alpha.col(0) = (uns + conf)/2;
99   alpha.col(1) = (uns - conf)/2;
100  arma::vec alpha_vec = vectorise(alpha);
101
102  // calcualting interpolation
103  uword r = 0;
104  for(uword s = 0; s < t.n_slices; ++s){
105    for(uword j = 0; j < t.n_cols; ++j){
106      arma::mat qq = cpp_normal_interpolation(t.slice(s).col(j), alpha_vec);
107      for(uword i = 0; i < conf.n_elem; ++ i){
108        mat_ci_output(r, 0) = s + 1;
109        mat_ci_output(r, 1) = j + 1;
110        mat_ci_output(r, 2) = conf(i);
111        arma::uvec qq_row_idx = {i, i + conf.n_elem};
112        arma::uvec qq_col_idx(1, fill::zeros);
113        mat_ci_output.row(r).cols(3, 4) = qq(qq_row_idx, qq_col_idx).t();
114        mat_ci_output.row(r).cols(5, 6) = 2*t0(j,s) - qq(qq_row_idx, qq_col_idx + 1
               ).t();
115        r++;
116      }
117    }
118  }
119
120  return mat_ci_output;
121 }
122
123 // [[Rcpp::export]]
124 arma::mat cpp_boot_perc_ci(arma::mat t0, arma::cube t, arma::vec conf){
125   // output matrix
126   arma::mat mat_ci_output(t.n_cols * t.n_slices * conf.n_elem, 7);
127
128   // creating alpha
129   arma::mat alpha(conf.n_rows, 2);
130   arma::vec uns(conf.n_elem, fill::ones);
131   alpha.col(0) = (uns - conf)/2;
132   alpha.col(1) = (uns + conf)/2;
133   arma::vec alpha_vec = vectorise(alpha);
134
135   // calcualting interpolation
136   uword r = 0;
```

```
  for(uword s = 0; s < t.n_slices; ++s){
    for(uword j = 0; j < t.n_cols; ++j){
      arma::mat qq = cpp_normal_interpolation(t.slice(s).col(j), alpha_vec);
      for(uword i = 0; i < conf.n_elem; ++ i){
        mat_ci_output(r, 0) = s + 1;
        mat_ci_output(r, 1) = j + 1;
        mat_ci_output(r, 2) = conf(i);
        arma::uvec qq_row_idx = {i, i + conf.n_elem};
        arma::uvec qq_col_idx(1, fill::zeros);
        mat_ci_output.row(r).cols(3, 4) = qq(qq_row_idx, qq_col_idx).t();
        mat_ci_output.row(r).cols(5, 6) = qq(qq_row_idx, qq_col_idx + 1).t();
        r++;
      }
    }
  }

  return mat_ci_output;
}

// [[Rcpp::export]]
arma::vec cpp_empirical_influence_reg(arma::vec t, arma::mat boot_array, int n){
  /* Function to estimate empirical influence values using regression.
   This method regresses the observed bootstrap values on the bootstrap
   frequencies to estimate the empirical influence values
   */

  arma::uvec fins = find_finite(t);
  arma::vec finite_t = t.elem(fins);
  int R = finite_t.n_elem;

  // if strata is null
  arma::vec strata(n, fill::ones);


  // boot array
  arma::mat finite_boot_array = boot_array.rows(fins);

  arma::mat X(R, n);
  X.fill(strata.n_elem);
  X = finite_boot_array / X;


  // output
  uword out = 0;
  arma::uvec inc = conv_to<uvec>::from(linspace(0, n-1, n));
  inc.shed_row(out);
  X = X.cols(inc);
  arma::vec uns(X.n_rows, fill::ones);
  X = join_rows(uns, X);
  arma::vec beta = inv(X.t()*X) * X.t() * t;
  beta.shed_row(0);
  arma::vec l(n, fill::zeros);
  l.elem(inc) = beta;
```

```
190   l = l - mean(l);
191
192
193   return l;
194
195 }
196
197 // [[Rcpp::export]]
198 arma::vec cpp_empirical_influence(arma::vec t, arma::mat boot_array, int n){
199   /* Calculation of empirical influence values.  Possible types are
200     0: "inf" = infinitesimal jackknife (numerical differentiation)
201     1: "reg" = regression based estimation
202     2: "jack" = usual jackknife estimates
203     3: "pos" = positive jackknife estimates */
204
205   arma::vec L = cpp_empirical_influence_reg(t, boot_array, n);
206
207   return L;
208 }
209
210 // [[Rcpp::export]]
211 arma::mat cpp_mat_empirical_influence_reg(arma::mat t, arma::mat boot_array, int n)
      {
212   /* Function to estimate empirical influence values using regression.
213    This method regresses the observed bootstrap values on the bootstrap
214    frequencies to estimate the empirical influence values
215    */
216
217   // original t and finit_t
218   arma::uvec fins    = find_finite(t);
219   arma::mat finite_t = t.elem(fins);
220   finite_t           = reshape(finite_t, t.n_rows, t.n_cols);
221   int R              = finite_t.n_rows;
222
223   // if strata is null
224   arma::vec strata(n, fill::ones);
225
226
227   // boot array
228   arma::mat X(R, n);
229   X.fill(strata.n_elem);
230   X = boot_array / X;
231
232   // output
233   uword out = 0;
234   X.col(out).fill(1.0);
235   arma::mat beta = inv(X.t()*X) * X.t() * t;
236   beta.row(out).fill(0.0);
237   arma::mat mean_beta = mean(beta);
238   for(uword c = 0; c < beta.n_cols; ++c){
239     beta.col(c) = beta.col(c) - as_scalar(mean_beta(c));
240   }
241
```

```
242
243   return beta;
244
245 }
246
247
248
249 // [[Rcpp::export]]
250 arma::mat cpp_boot_bca_ci(arma::mat t0, arma::cube t, arma::vec conf, arma::mat
      boot_array, int n){
251   // output matrix
252   arma::mat mat_ci_output(t.n_cols * t.n_slices * conf.n_elem, 7);
253
254   // creating alpha
255   arma::mat alpha(conf.n_rows, 2);
256   arma::vec uns(conf.n_elem, fill::ones);
257   alpha.col(0) = (uns - conf)/2;
258   alpha.col(1) = (uns + conf)/2;
259
260
261   // creating zalpha
262   arma::mat zalpha = alpha;
263   arma::vec temp0 = Rcpp::qnorm(arma2vec(zalpha.col(0)));
264   arma::vec temp1 = Rcpp::qnorm(arma2vec(zalpha.col(1)));
265   zalpha.col(0) = temp0;
266   zalpha.col(1) = temp1;
267
268   // calcualting interpolation
269   arma::umat t_menor_que_t0;
270   arma::mat t_o;
271   arma::mat finite_t;
272   arma::mat new_t0;
273   arma::mat w;
274   arma::mat a;
275   arma::mat adj_alpha;
276   arma::mat qq;
277
278   // to help adjust indices in output matrix
279   uword first_row = 0;
280   uword pos       = 0;
281
282   for(uword s = 0; s < t.n_slices; ++s){
283
284     // original t and finit_t
285     t_o      = t.slice(s);
286     arma::uvec finite_t_indices = find_finite(t_o);
287     finite_t = t_o(finite_t_indices);
288     finite_t = reshape(finite_t, t.slice(s).n_rows, t.slice(s).n_cols);
289
290     // Calculate W
291     new_t0         = trans(repmat(t0.col(s), 1,  t.slice(s).n_rows));
292     t_menor_que_t0 = finite_t < new_t0;
293     w              = trans(mean(conv_to<mat>::from(t_menor_que_t0)));
```

```
294     w              = Rcpp::qnorm(arma2vec(w));
295
296     // Calculate a
297     arma::mat L = cpp_mat_empirical_influence_reg(t_o, boot_array, n);
298     a           = trans(sum(pow(L, 3)) / (6*pow(sum(pow(L, 2)), 1.5)));
299
300     // adjusted alpha
301     adj_alpha = trans(vectorise(zalpha));
302     adj_alpha = repmat(adj_alpha, w.n_rows, 1);
303     adj_alpha.each_col() += w;
304     a         = repmat(a, 1, adj_alpha.n_cols);
305     adj_alpha = adj_alpha / (1-a%adj_alpha);
306     adj_alpha.each_col() += w;
307     adj_alpha = trans(normcdf(adj_alpha));
308
309     // interpolation
310     qq = trans(cpp_mat_normal_interpolation(t_o, adj_alpha));
311
312     // output matrix
313     vec t_idx = linspace(1, qq.n_rows, qq.n_rows);
314     uword total_conf_elem = conf.n_elem;
315     uword total_qq_rows   = qq.n_rows;
316
317     for(uword i = 0; i < conf.n_elem; ++ i){
318       uword last_row  = ((i+1)*total_qq_rows) + (pos - 1);
319
320       vec s_vec(total_qq_rows); s_vec.fill(s + 1);
321       vec conf_vec(total_qq_rows); conf_vec.fill(conf(i));
322
323       mat_ci_output.submat(first_row, 0, last_row, 0) = s_vec;
324       mat_ci_output.submat(first_row, 1, last_row, 1) = t_idx;
325       mat_ci_output.submat(first_row, 2, last_row, 2) = conf_vec;
326       mat_ci_output.submat(first_row, 3, last_row, 3) = qq.col(i);
327       mat_ci_output.submat(first_row, 4, last_row, 4) = qq.col(i + total_conf_elem)
              ;
328       mat_ci_output.submat(first_row, 5, last_row, 5) = qq.col(i + (2*total_conf_
              elem));
329       mat_ci_output.submat(first_row, 6, last_row, 6) = qq.col(i + (3*total_conf_
              elem));;
330
331       first_row += total_qq_rows;
332     }
333     pos += total_conf_elem * total_qq_rows;
334
335     // }
336   }
337
338
339   return mat_ci_output;
340 }
341
342
343 // [[Rcpp::export]]
```

```
344 arma::mat cpp_boot_norm_ci(arma::mat t0, arma::cube t, arma::vec conf){
345   // output matrix
346   arma::mat mat_ci_output(t.n_cols * t.n_slices * conf.n_elem, 7);
347
348
349   // calcualting interpolation
350   uword r = 0;
351   for(uword s = 0; s < t.n_slices; ++s){
352     for(uword j = 0; j < t.n_cols; ++j){
353
354       // original t and finite_t
355       arma::vec t_o = t.slice(s).col(j);
356       arma::uvec finite_t_indices = find_finite(t_o);
357       arma::vec finite_t = t_o.elem(finite_t_indices);
358
359       // variance of t0
360       double var_t0 = var(finite_t);
361
362       // bias of t
363       double bias = mean(finite_t) - t0(j, s);
364
365       // erro padrao
366       arma::vec errop = sqrt(var_t0) * Rcpp::qnorm(arma2vec((1 + conf)/2));
367
368       for(uword i = 0; i < conf.n_elem; ++ i){
369         mat_ci_output(r, 0) = s + 1;
370         mat_ci_output(r, 1) = j + 1;
371         mat_ci_output(r, 2) = conf(i);
372         mat_ci_output(r, 3) = datum::nan;
373         mat_ci_output(r, 4) = datum::nan;
374         mat_ci_output(r, 5) = t0(j, s) - bias - errop(i);
375         mat_ci_output(r, 6) = t0(j, s) - bias + errop(i);
376         r++;
377       }
378     }
379   }
380
381   return mat_ci_output;
382 }
383
384
385
386 //arma::mat cpp_boot_stud_ci(arma::mat t0, arma::cube t, arma::vec conf, arma::mat
        var_t0){
387 //   // output matrix
388 //   arma::mat mat_ci_output(t.n_cols * t.n_slices * conf.n_elem, 7);
389 //
390 //   // creating alpha
391 //   arma::mat alpha(conf.n_rows, 2);
392 //   arma::vec uns(conf.n_elem, fill::ones);
393 //   alpha.col(0) = (uns + conf)/2;
394 //   alpha.col(1) = (uns - conf)/2;
395 //   arma::vec alpha_vec = vectorise(alpha);
```

```
//
//  // calcualting interpolation
//  uword r = 0;
//  for(uword s = 0; s < t.n_slices; ++s){
//    for(uword j = 0; j < t.n_cols; ++j){
//
//      arma::vec z = (t.slice(s).col(j) - t0)/
//
//      arma::mat qq = cpp_normal_interpolation(t.slice(s).col(j), alpha_vec);
//      for(uword i = 0; i < conf.n_elem; ++ i){
//        mat_ci_output(r, 0) = s + 1;
//        mat_ci_output(r, 1) = j + 1;
//        mat_ci_output(r, 2) = conf(i);
//        arma::uvec qq_row_idx = {i, i + conf.n_elem};
//        arma::uvec qq_col_idx(1, fill::zeros);
//        mat_ci_output.row(r).cols(3, 4) = qq(qq_row_idx, qq_col_idx).t();
//        mat_ci_output.row(r).cols(5, 6) = 2*t0(j,s) - qq(qq_row_idx, qq_col_idx +
      1).t();
//        r++;
//      }
//    }
//  }
//
//   return mat_ci_output;
// }
```

Funções em C++ para fazer cálculos da simulação (Segunda rotina em C++)

```
1  // [[Rcpp::depends(RcppArmadillo)]]
2
3  #include <RcppArmadillo.h>
4  #include <cpp_boot_ci.h>
5
6
7  using namespace Rcpp;
8  using namespace arma;
9
10 // [[Rcpp::export]]
11 List cpp_svd(arma::mat X){
12
13   // Matriz X sizes
14   int p = X.n_cols;
15
16   // Singular value Decomposition of X
17   arma::mat P; //U
18   arma::vec s; //s
19   arma::mat V; //V
20   arma::svd(P,s,V,X);
21
22   // Reduce U;
23   arma::mat U = P.cols(0, p - 1);
24
25   // Lambda
26   s = pow(s, 2);
```

```
27   arma::mat L = arma::diagmat(s);
28
29   return List::create(Named("U") = U,
30                       Named("s") = s,
31                       Named("V") = V,
32                       Named("L") = L);
33
34 }
35
36 // [[Rcpp::export]]
37 arma::vec cpp_residuals(arma::mat X, arma::vec y, arma::vec beta){
38   // fit
39   arma::vec ols_fit = X * beta;
40
41   // residuals
42   arma::vec resid = y - ols_fit;
43
44   return resid;
45 }
46
47 // [[Rcpp::export]]
48 arma::vec cpp_hat_beta_ols(arma::mat X, arma::vec y){
49
50   // OLS --------------
51   return inv(X.t() *  X) * X.t() * y;
52
53 }
54
55
56 // [[Rcpp::export]]
57 arma::vec cpp_hat_beta_ridge(arma::mat X, arma::vec y, double k){
58   // Matriz X sizes
59   //int n = X.n_rows;
60   int p = X.n_cols;
61
62   // matriz diagonal preenchida com k
63   arma::mat K(p, p, fill::zeros);
64   K.diag() += k;
65
66   // vetor beta ridge
67   arma::vec hat_beta_ridge = inv((X.t()*X)+ K) * X.t() * y;
68
69   return hat_beta_ridge;
70 }
71
72 // [[Rcpp::export]]
73 arma::vec cpp_hat_beta_ridge_from_beta_ols(arma::mat X, arma::vec beta_ols, double
       k){
74   int p = X.n_cols;
75   arma::mat Ip = eye(p, p);
76   arma::mat W = inv(X.t()*X + k*Ip);
77   arma::mat Z = Ip - k*W;
78   arma::vec hat_beta_ridge = Z*beta_ols;
```

```
79   return hat_beta_ridge;
80 }
81
82
83 // [[Rcpp::export]]
84 double cpp_hat_sigma(arma::mat X, arma::vec y, arma::vec beta){
85   arma::vec resid = cpp_residuals(X, y, beta);
86   double n = X.n_rows;
87   double p = X.n_cols;
88   double sigma = arma::as_scalar(resid.t() * resid) / (n - p - 1);
89
90   return pow(sigma, 0.5);
91 }
92
93 // [[Rcpp::export]]
94 double cpp_mse_k(mat X, vec y, double k){
95   //int n = X.n_rows;
96   int p = X.n_cols;
97
98   // matriz diagonal preenchida com k
99   arma::mat K(p, p, fill::zeros);
100  K.diag() += k;
101
102  // beta ols
103  arma::vec hat_beta_ols = inv(X.t()*X) * X.t() * y;
104
105  // calcualte sigma
106  double sigma = cpp_hat_sigma(X, y, hat_beta_ols);
107
108  // svd
109  List svd_x = cpp_svd(X);
110
111
112  // lambdas
113  vec lambda = svd_x["s"];
114  lambda = pow(lambda, 2);
115
116  // calcualte mse parts
117  mat W = inv((X.t()*X)+ K);
118  double part1 = as_scalar(accu(lambda / pow(lambda + k, 2)));
119  double part2 = pow(k,2) * as_scalar(accu(hat_beta_ols.t() * W.t() * W * hat_beta_
       ols));
120
121  // MSE
122  double mse = pow(sigma,2) *  part1 + part2;
123
124  return mse;
125 }
126
127 // [[Rcpp::export]]
128 double k_hkb(double hat_sigma, arma::vec beta_ols){
129   double p = beta_ols.n_elem;
130   double den = arma::as_scalar(beta_ols.t() * beta_ols);
```

```
131   return p * pow(hat_sigma, 2) / den;
132 }
133
134
135 // [[Rcpp::export]]
136 double k_lw(double hat_sigma, arma::mat V, arma::vec l, arma::vec beta_ols){
137   double p = beta_ols.n_elem;
138   arma::vec alpha = V.t() * beta_ols;
139   double den = arma::as_scalar(l.t() * pow(alpha, 2));
140   return p * pow(hat_sigma, 2) / den;
141 }
142
143 // [[Rcpp::export]]
144 double k_mg(double hat_sigma, arma::mat V, arma::vec l, arma::vec beta_ols){
145   //double p = beta_ols.n_elem;
146   arma::vec alpha = V.t() * beta_ols;
147   double den = exp(as_scalar(mean(log(pow(alpha, 2)))));
148   return pow(hat_sigma, 2) / den;
149 }
150
151 // [[Rcpp::export]]
152 double k_ks(double hat_sigma, arma::mat V, arma::vec l, arma::vec beta_ols, double
      n){
153   double p = beta_ols.n_elem;
154   arma::vec alpha = V.t() * beta_ols;
155   double max_alpha = alpha.max();
156   double max_l = l.max();
157   double num = (max_l* pow(hat_sigma, 2));
158   double den = ((n-p)*pow(hat_sigma, 2) + (max_l*pow(max_alpha, 2)));
159   double k_ks = num / den;
160   return k_ks;
161 }
162
163 // [[Rcpp::export]]
164 arma::vec k_ma_ks(double hat_sigma, arma::mat V, arma::vec l, arma::vec beta_ols,
      double n){
165   double p = beta_ols.n_elem;
166   arma::vec alpha = V.t() * beta_ols;
167   arma::vec num = (pow(hat_sigma, 2) * l);
168   arma::vec den = (n - p)*pow(hat_sigma, 2) + l % pow(alpha,2);
169
170   arma::vec nuc =  num/ den;
171   return nuc;
172 }
173
174 // [[Rcpp::export]]
175 arma::vec cpp_fiv(arma::mat X){
176   arma::vec c_inv = diagvec(inv(X.t()*X));
177   return c_inv;
178 }
179
180 // [[Rcpp::export]]
181 double k_dk(double hat_sigma, arma::vec beta_ols, arma::mat X){
```

```
182   double n = X.n_rows;
183   arma::vec fiv = cpp_fiv(X);
184   double est_k_hkb = k_hkb(hat_sigma, beta_ols);
185   double est_k = est_k_hkb - (1/(n*fiv.max()));
186   if(est_k > 0){
187     return est_k;
188   } else {
189     return 0;
190   }
191
192 }
193
194
195
196 // [[Rcpp::export]]
197 double k_mdg(double hat_sigma, arma::vec beta_ols, arma::vec l, arma::mat X){
198   double inv_l = as_scalar(accu(1 / l));
199   double Q = as_scalar(beta_ols.t() * beta_ols) - (pow(hat_sigma, 2) * inv_l);
200   if(Q <= 0){
201     return 0;
202   } else {
203     arma::vec poss_k = linspace(0, 1, 1000);
204     arma::vec aux(poss_k.n_elem);
205     for(uword i=0; i < poss_k.n_elem; ++i){
206       arma::vec hat_beta_ridge = cpp_hat_beta_ridge_from_beta_ols(X, beta_ols, poss
              _k(i));
207       aux(i) = as_scalar(hat_beta_ridge.t()*hat_beta_ridge);
208     }
209     uword which_nearest = index_min(abs(aux - Q));
210     return poss_k(which_nearest);
211   }
212 }
213
214
215
216 // [[Rcpp::export]]
217 arma::vec cpp_hat_var_beta_ridge(arma::mat X, arma::vec y, double k){
218   double p = X.n_cols;
219
220   // beta ols
221   arma::vec hat_beta_ols = inv(X.t()*X) * X.t() * y;
222
223   //calcualte sigma_k
224   double sigma = cpp_hat_sigma(X, y, hat_beta_ols);
225
226
227   arma::mat I_p(p, p, fill::eye);
228   arma::mat Cinv = inv(X.t() * X);
229   arma::mat Z = inv(I_p + k*Cinv);
230   arma::vec diag_mat_var = diagvec(Z*Cinv*Z.t());
231
232   return pow(sigma, 2) * diag_mat_var;
233 }
```

```
234
235 // [[Rcpp::export]]
236 arma::vec cpp_hat_var_beta_ridge_sigma_k(arma::mat X, arma::vec y, double k){
237   double p = X.n_cols;
238
239   // calculate hat_beta
240   arma::vec hat_beta_0 = cpp_hat_beta_ridge(X, y, k);
241
242   //calcualte sigma_k
243   double sigma = cpp_hat_sigma(X, y, hat_beta_0);
244
245
246   arma::mat I_p(p, p, fill::eye);
247   arma::mat Cinv = inv(X.t() * X);
248   arma::mat Z = inv(I_p + k*Cinv);
249   arma::vec diag_mat_var = diagvec(Z*Cinv*Z.t());
250
251   return pow(sigma, 2) * diag_mat_var;
252 }
253
254
255
256
257 // [[Rcpp::export]]
258 arma::vec cpp_calculate_all_k(arma::mat X, arma::vec y){
259   // Calcualte svd
260   List X_svd = cpp_svd(X);
261
262
263   // calculate beta ols
264   arma::vec hat_beta_ols = cpp_hat_beta_ols(X, y);
265
266
267   // calculate estimated sigma
268   double hat_sigma = cpp_hat_sigma(X, y, hat_beta_ols);
269
270
271   // calculate k
272   double hat_k_hkb = k_hkb(hat_sigma, hat_beta_ols);
273   vec lambda = X_svd["s"];
274   double hat_k_mdg = k_mdg(hat_sigma, hat_beta_ols, lambda, X);
275   double hat_k_lw = k_lw(hat_sigma, X_svd["V"], lambda, hat_beta_ols);
276   double hat_k_mg = k_mg(hat_sigma, X_svd["V"], lambda, hat_beta_ols);
277   double hat_k_ks = k_ks(hat_sigma, X_svd["V"], lambda, hat_beta_ols, X.n_rows);
278   arma::vec hat_k_aks = k_ma_ks(hat_sigma, X_svd["V"], lambda, hat_beta_ols, X.n_
        rows);
279   double hat_k_aks_ma = mean(hat_k_aks);
280   double hat_k_aks_max = hat_k_aks.max();
281   double hat_k_dk = k_dk(hat_sigma, hat_beta_ols, X);
282
283
284
285   // matrix of k
```

```
286   arma::vec vec_k(9);
287
288   vec_k.row(0) = 0;
289   vec_k.row(1) = hat_k_hkb;
290   vec_k.row(2) = hat_k_mdg;
291   vec_k.row(3) = hat_k_lw;
292   vec_k.row(4) = hat_k_mg;
293   vec_k.row(5) = hat_k_ks;
294   vec_k.row(6) = hat_k_aks_ma;
295   vec_k.row(7) = hat_k_aks_max;
296   vec_k.row(8) = hat_k_dk;
297
298
299
300   return vec_k;
301 }
302
303
304 // [[Rcpp::export]]
305 arma::cube cpp_calculate_all_k_mult_y(arma::mat X, arma::cube y){
306   arma::cube cube_k(9, y.n_cols, y.n_slices);
307
308   for(uword j = 0; j < y.n_slices; ++j){
309     for(uword i= 0; i< y.slice(j).n_cols; ++i){
310       cube_k.slice(j).col(i) = cpp_calculate_all_k(X, y.slice(j).col(i));
311     }
312   }
313
314   return(cube_k);
315 }
316
317
318
319
320
321 // [[Rcpp::export]]
322 arma::vec cpp_boot_t_test_padrao(arma::mat X, arma::vec y){
323
324   // calculate hat_beta
325   arma::vec hat_beta_0 = cpp_hat_beta_ridge(X, y, 0);
326
327   // Calculate var_hat_beta
328   arma::vec var_hat_beta_0 = cpp_hat_var_beta_ridge(X, y, 0);
329
330   // calculate t test
331   arma::vec t_0 = (hat_beta_0) / pow(var_hat_beta_0, 0.5);
332
333   return t_0;
334 }
335
336
337
338 // [[Rcpp::export]]
```

```
arma::vec cpp_boot_t_test_hb(arma::mat X, arma::vec y, double k){

  // calculate hat_beta
  arma::vec hat_beta_k = cpp_hat_beta_ridge(X, y, k);

  // Calculate var_hat_beta
  arma::vec var_hat_beta_k = cpp_hat_var_beta_ridge_sigma_k(X, y, k);

  // calculate t test
  arma::vec t_k = (hat_beta_k) / pow(var_hat_beta_k, 0.5);

  return t_k;
}



// [[Rcpp::export]]
arma::vec cpp_generalized_alpha(arma::mat X, arma::vec y){

  // Matriz X sizes
  int p = X.n_cols;

  // Singular value Decomposition of X
  arma::mat P; //U
  arma::vec s; //s
  arma::mat V; //V
  arma::svd(P,s,V,X);

  // Reduce U;
  arma::mat U = P.cols(0, p - 1);

  // Lambda
  s = pow(s, 2);
  arma::mat L = arma::diagmat(s);
  arma::mat inv_L = inv(L);

  // OLS --------------
  // coefficients
  arma::colvec beta_ols = cpp_hat_beta_ols(X, y);

  // sigma2
  double sigma_2 = pow(cpp_hat_sigma(X, y, beta_ols), 2);

  // Canonical ------------
  arma::vec r = U.t() * y;
  arma::mat D = U.t() * X * V;
  arma::vec alpha = r / s;

  // generalized Cp
  // finding values of k_i
  arma::vec k_i(p);
  k_i(0) = 0;
  for(uword i=1; i<k_i.n_elem; ++i){
```

```
392     double r_i = r(i);
393     double l_i = s(i);
394     if(pow(r_i, 2) > sigma_2){
395       k_i(i) = (pow(l_i, 2) * sigma_2) / (pow(r_i,2)  - sigma_2);
396     } else {
397       k_i(i) = datum::inf;
398     }
399   }
400
401   // ridge estimates
402   arma::vec g_alpha = (s / (s + k_i)) % alpha;
403
404   // MSE ridge alpha
405   //double mse_g_alpha = arma::accu(s * sigma_2 + pow(k_i, 2) % pow(alpha, 2) / pow
          (s + k_i, 2));
406
407
408    return g_alpha;
409 }
410
411
412
413 /* BOOTSTRAP FUNCTION */
414
415 // [[Rcpp::export]]
416 arma::umat cpp_boot_indices(int n, int B, int seed){
417   arma_rng::set_seed(seed);
418
419   // Create index values
420   imat A = randi<imat>(B, n, distr_param(0, n-1));
421   umat U = conv_to<umat>::from(A);
422
423   return U;
424
425 }
426
427 // [[Rcpp::export]]
428 arma::umat cpp_boot_array(arma::umat ind){
429   arma::mat ind_mat = conv_to<mat>::from(ind);
430   int n = ind_mat.n_cols;
431   vec cent = linspace(0, n-1, n);
432   arma::umat out = hist(ind_mat, cent, 1);
433   return out;
434 }
435
436 // [[Rcpp::export]]
437 Rcpp::List cpp_boot_func(arma::mat X, arma::cube y, arma::cube k, arma::uword idx){
438
439   // adjust idx
440   //idx = idx - 1;
441
442   // matriz para alocar betas
443   arma::mat hat_beta_ridge_mo(y.n_cols, k.n_rows);
```

```
444   arma::mat hat_beta_ridge_po(y.n_cols, k.n_rows);
445
446   // Alocar variancia de betas
447   arma::mat var_hat_beta_ridge_mo(y.n_cols, k.n_rows);
448   arma::mat var_hat_beta_ridge_po(y.n_cols, k.n_rows);
449
450   // vector to allocate t tests
451   arma::mat t_test_hb_mo(y.n_cols, k.n_rows);
452   arma::mat t_test_hb_po(y.n_cols, k.n_rows);
453   arma::vec t_test_padrao_mo(y.n_cols);
454   arma::vec t_test_padrao_po(y.n_cols);
455
456   // calculate hat beta ridge i
457   for(uword j = 0; j < k.slice(0).n_rows; ++j){
458     for(uword i= 0; i< y.slice(0).n_cols; ++i){
459
460       // coefs for the two slices: Most-favorable and Least-favorable orientations
461       arma::vec coefs1 = cpp_hat_beta_ridge(X, y.slice(0).col(i), k.slice(0)(j, i))
              ;
462       arma::vec coefs2 = cpp_hat_beta_ridge(X, y.slice(1).col(i), k.slice(1)(j, i))
              ;
463
464       // Calculate beta ridge variance
465       arma::vec var_coefs1 = cpp_hat_var_beta_ridge(X, y.slice(0).col(i), k.slice(0
              )(j, i));
466       arma::vec var_coefs2 = cpp_hat_var_beta_ridge(X, y.slice(1).col(i), k.slice(1
              )(j, i));
467
468
469       // Select only the desirable beta index
470       hat_beta_ridge_mo(i, j) = as_scalar(coefs1.row(idx));
471       hat_beta_ridge_po(i, j) = as_scalar(coefs2.row(idx));
472       var_hat_beta_ridge_mo(i, j) = as_scalar(var_coefs1.row(idx));
473       var_hat_beta_ridge_po(i, j) = as_scalar(var_coefs2.row(idx));
474
475       // t-tests
476       t_test_hb_mo(i, j) = as_scalar(cpp_boot_t_test_hb(X, y.slice(0).col(i), k.
              slice(0)(j, i)).row(idx));
477       t_test_hb_po(i, j) = as_scalar(cpp_boot_t_test_hb(X, y.slice(1).col(i), k.
              slice(1)(j, i)).row(idx));
478
479       if(j == 0){
480         t_test_padrao_mo(i) = as_scalar(cpp_boot_t_test_padrao(X, y.slice(0).col(i)
                ).row(idx));
481         t_test_padrao_po(i) = as_scalar(cpp_boot_t_test_padrao(X, y.slice(1).col(i)
                ).row(idx));
482       }
483
484     }
485   }
486
487
488   //vectorise and concatenate
```

```
489   //arma::vec out_1 = join_cols(vectorise(hat_beta_ridge_mo),
490   //                            vectorise(t_test_hb_mo),
491   //                            t_test_padrao_mo);
492   //
493   //arma::vec out_2 = join_cols(vectorise(hat_beta_ridge_po),
494   //                            vectorise(t_test_hb_po),
495   //                            t_test_padrao_po);
496
497   return List::create(Named("hat_beta_ridge_mo") = hat_beta_ridge_mo,
498                       Named("hat_beta_ridge_po") = hat_beta_ridge_po,
499                       Named("var_hat_beta_ridge_mo") = var_hat_beta_ridge_mo,
500                       Named("var_hat_beta_ridge_po") = var_hat_beta_ridge_mo,
501                       Named("t_test_hb_mo") = t_test_hb_mo,
502                       Named("t_test_hb_po") = t_test_hb_po,
503                       Named("t_test_padrao_mo") = t_test_padrao_mo,
504                       Named("t_test_padrao_po") = t_test_padrao_po);
505
506 }
507
508
509 // [[Rcpp::export]]
510 arma::cube cpp_boot_func_2(arma::mat X, arma::cube y, arma::cube k, arma::uword idx
        ){
511
512   // adjust idx
513   //idx = idx - 1;
514
515   // matriz para alocar betas
516   arma::mat hat_beta_ridge_mo(y.n_cols, k.n_rows);
517   arma::mat hat_beta_ridge_po(y.n_cols, k.n_rows);
518
519   // calculate hat beta ridge i
520   for(uword j = 0; j < k.slice(0).n_rows; ++j){
521     for(uword i= 0; i< y.slice(0).n_cols; ++i){
522
523       // coefs for the two slices: Most-favorable and Least-favorable orientations
524       arma::vec coefs1 = cpp_hat_beta_ridge(X, y.slice(0).col(i), k.slice(0)(j, i))
              ;
525       arma::vec coefs2 = cpp_hat_beta_ridge(X, y.slice(1).col(i), k.slice(1)(j, i))
              ;
526
527
528       // Select only the desirable beta index
529       hat_beta_ridge_mo(i, j) = as_scalar(coefs1.row(idx));
530       hat_beta_ridge_po(i, j) = as_scalar(coefs2.row(idx));
531     }
532   }
533   arma::cube output(y.n_cols, k.n_rows, 2);
534   output.slice(0) = hat_beta_ridge_mo;
535   output.slice(1) = hat_beta_ridge_po;
536
537
538   return output;
```

```
539 }
540
541 // [[Rcpp::export]]
542 arma::cube cpp_reorder_cube(cube y, uvec idx){
543
544   for(uword i = 0; i < y.n_slices; ++i){
545     arma::mat a = y.slice(i);
546     y.slice(i) = a.rows(idx);
547   }
548   return y;
549 }
550
551
552 // [[Rcpp::export]]
553 Rcpp::List cpp_boot_ridge_simulation(arma::mat X, arma::cube y, arma::cube k, arma
      ::uword idx, arma::vec conf, int B, int seed){
554   wall_clock timer;
555
556   // Calculate matrix X properties
557   int n = X.n_rows;
558
559   // Objects
560   arma::cube cube_hat_beta_ridge_mo(B, y.n_cols, 9);
561   arma::cube cube_hat_beta_ridge_po(B, y.n_cols, 9);
562
563   // generate Bootstrap indices
564   arma::umat boot_idx = cpp_boot_indices(n, B, seed);
565
566   // Calculate cpp_boot_func for all bootstrap indices
567   for(uword i = 0; i < boot_idx.n_rows; ++i){
568     Rcout << i << "\n";
569     arma::uvec bi = conv_to<uvec>::from(boot_idx.row(i));
570     //arma::cube new_y = cpp_reorder_cube(y, bi);
571     arma::cube out_i = cpp_boot_func_2(X.rows(boot_idx.row(i)), cpp_reorder_cube(y,
          bi), k, idx);
572
573     // Extract elements
574     arma::mat aux_1 = out_i.slice(0);
575     arma::mat aux_2 = out_i.slice(1);
576
577     // Assign element into t0
578     cube_hat_beta_ridge_mo.row(i) = aux_1;
579     cube_hat_beta_ridge_po.row(i) = aux_2;
580   }
581
582   // Calculate t0
583   arma::cube out_t0 = cpp_boot_func_2(X, y, k, idx);
584   arma::mat t0_hat_beta_ridge_mo = out_t0.slice(0);
585   arma::mat t0_hat_beta_ridge_po = out_t0.slice(1);
586
587   // Calculate boot_array
588   arma::mat boot_array = conv_to<mat>::from(cpp_boot_array(boot_idx));
589
```

```
590   // // // Calculate confidence interval
591   timer.tic();
592   arma::mat output1 = cpp_boot_basic_ci(t0_hat_beta_ridge_mo, cube_hat_beta_ridge_
          mo, conf);
593   double tt = timer.toc();
594   Rcout << "number of seconds: " << tt << endl;
595   timer.tic();
596   arma::mat output2 = cpp_boot_perc_ci(t0_hat_beta_ridge_mo, cube_hat_beta_ridge_mo
          , conf);
597   tt = timer.toc();
598   Rcout << "number of seconds: " << tt << endl;
599   timer.tic();
600   arma::mat output3 = cpp_boot_norm_ci(t0_hat_beta_ridge_mo, cube_hat_beta_ridge_mo
          , conf);
601   tt = timer.toc();
602   Rcout << "number of seconds: " << tt << endl;
603   timer.tic();
604   arma::mat output4 = cpp_boot_bca_ci(t0_hat_beta_ridge_mo, cube_hat_beta_ridge_mo,
           conf, boot_array, n);
605   tt = timer.toc();
606   Rcout << "number of seconds: " << tt << endl;
607
608   // Rcout << "-- Stop 7\n";
609   arma::mat output5 = cpp_boot_basic_ci(t0_hat_beta_ridge_po, cube_hat_beta_ridge_
          po, conf);
610   arma::mat output6 = cpp_boot_perc_ci(t0_hat_beta_ridge_po, cube_hat_beta_ridge_po
          , conf);
611   arma::mat output7 = cpp_boot_norm_ci(t0_hat_beta_ridge_po, cube_hat_beta_ridge_po
          , conf);
612   arma::mat output8 = cpp_boot_bca_ci(t0_hat_beta_ridge_po, cube_hat_beta_ridge_po,
           conf, boot_array, n);
613
614   //vectorise and concatenate
615   arma::mat mat_1(output1.n_rows, 1);
616   arma::mat mat_2(output2.n_rows, 1);
617   arma::mat mat_3(output3.n_rows, 1);
618   arma::mat mat_4(output4.n_rows, 1);
619   mat_1.fill(1);
620   mat_2.fill(2);
621   mat_3.fill(3);
622   mat_4.fill(4);
623   arma::mat output_1 = join_cols(join_rows(mat_1, output1),
624                                  join_rows(mat_2, output2),
625                                  join_rows(mat_3, output3),
626                                  join_rows(mat_4, output4));
627
628   arma::mat output_2 = join_cols(join_rows(mat_1, output5),
629                                  join_rows(mat_2, output6),
630                                  join_rows(mat_3, output7),
631                                  join_rows(mat_4, output8));
632
633   return List::create(Named("ci_beta_ridge_mo") = output_1,
634                       Named("ci_beta_ridge_po") = output_2);
```

```
635 }
636
637
638 // [[Rcpp::export]]
639 Rcpp::List debugger(arma::mat X, arma::cube y, arma::cube k, arma::uword idx, arma
      ::vec conf, int B, int seed){
640   // Calculate matrix X properties
641   int n = X.n_rows;
642
643   // Objects
644   arma::cube cube_hat_beta_ridge_mo(B, y.n_cols, 9);
645   arma::cube cube_hat_beta_ridge_po(B, y.n_cols, 9);
646
647   // generate Bootstrap indices
648   arma::umat boot_idx = cpp_boot_indices(n, B, seed);
649
650   // Calculate cpp_boot_func for all bootstrap indices
651   for(uword i = 0; i < boot_idx.n_rows; ++i){
652     Rcout << i << "\n";
653     arma::uvec bi = conv_to<uvec>::from(boot_idx.row(i));
654     //arma::cube new_y = cpp_reorder_cube(y, bi);
655     arma::cube out_i = cpp_boot_func_2(X.rows(boot_idx.row(i)), cpp_reorder_cube(y,
          bi), k, idx);
656
657     // Extract elements
658     arma::mat aux_1 = out_i.slice(0);
659     arma::mat aux_2 = out_i.slice(1);
660
661     // Assign element into t0
662     cube_hat_beta_ridge_mo.row(i) = aux_1;
663     cube_hat_beta_ridge_mo.row(i) = aux_2;
664   }
665
666   Rcout << "-- Stop 1\n";
667   // Calculate t0
668   arma::cube out_t0 = cpp_boot_func_2(X, y, k, idx);
669   arma::mat t0_hat_beta_ridge_mo = out_t0.slice(0);
670   arma::mat t0_hat_beta_ridge_po = out_t0.slice(1);
671   Rcout << "-- Stop 2\n";
672   // Calculate boot_array
673   arma::mat boot_array = conv_to<mat>::from(cpp_boot_array(boot_idx));
674   // Rcout << "-- Stop 3\n";
675   // // Calculate confidence interval
676   // arma::mat output1 = cpp_boot_basic_ci(t0_hat_beta_ridge_mo, cube_hat_beta_
        ridge_mo, conf);
677   // Rcout << "-- Stop 4\n";
678   // arma::mat output2 = cpp_boot_perc_ci(t0_hat_beta_ridge_mo, cube_hat_beta_ridge
        _mo, conf);
679   // Rcout << "-- Stop 5\n";
680   // arma::mat output3 = cpp_boot_norm_ci(t0_hat_beta_ridge_mo, cube_hat_beta_ridge
        _mo, conf);
681   // Rcout << "-- Stop 6\n";
682   // arma::mat output4 = cpp_boot_bca_ci(t0_hat_beta_ridge_mo, cube_hat_beta_ridge_
```

```
        mo, conf, boot_array, n);
683   // Rcout << "-- Stop 7\n";
684   // arma::mat output5 = cpp_boot_basic_ci(t0_hat_beta_ridge_po, cube_hat_beta_
        ridge_mo, conf);
685   // arma::mat output6 = cpp_boot_perc_ci(t0_hat_beta_ridge_po, cube_hat_beta_ridge
        _mo, conf);
686   // arma::mat output7 = cpp_boot_norm_ci(t0_hat_beta_ridge_po, cube_hat_beta_ridge
        _mo, conf);
687   // arma::mat output8 = cpp_boot_bca_ci(t0_hat_beta_ridge_po, cube_hat_beta_ridge_
        mo, conf, boot_array, n);
688   //
689   // //vectorise and concatenate
690   // arma::mat mat_1(output1.n_rows, 1);
691   // arma::mat mat_2(output2.n_rows, 1);
692   // arma::mat mat_3(output3.n_rows, 1);
693   // arma::mat mat_4(output4.n_rows, 1);
694   // mat_1.fill(1);
695   // mat_2.fill(2);
696   // mat_3.fill(3);
697   // mat_4.fill(4);
698   // arma::mat output_1 = join_cols(join_rows(mat_1, output1),
699   //                                join_rows(mat_2, output2),
700   //                                join_rows(mat_3, output3),
701   //                                join_rows(mat_4, output4));
702   //
703   // arma::mat output_2 = join_cols(join_rows(mat_1, output5),
704   //                                join_rows(mat_2, output6),
705   //                                join_rows(mat_3, output7),
706   //                                join_rows(mat_4, output8));
707
708   return List::create(Named("cube_hat_beta_ridge_mo") = cube_hat_beta_ridge_mo,
709                       Named("t0_hat_beta_ridge_mo") = t0_hat_beta_ridge_mo,
710                       Named("boot_array") = boot_array);
711 }
712
713
714 // [[Rcpp::export]]
715 Rcpp::List cpp_boot_ridge_aplication(arma::mat X, arma::vec y, double k, arma::vec
      conf, int B, int seed){
716   // Calculate matrix X properties
717   int n = X.n_rows;
718
719   // Objects
720   arma::cube cube_hat_beta_ridge(B, y.n_cols, X.n_cols);
721
722   // generate Bootstrap indices
723   arma::umat boot_idx = cpp_boot_indices(n, B, seed);
724
725   // Calculate cpp_boot_func for all bootstrap indices
726   for(uword i = 0; i < boot_idx.n_rows; ++i){
727
728     // Estimated Coefficients
729     arma::vec coefs = cpp_hat_beta_ridge(X.rows(boot_idx.row(i)), y.elem(boot_idx.
```

```
          row(i)), k);
730     arma::mat aux1 = conv_to<mat>::from(coefs.t());
731     // Assign element into t0
732     cube_hat_beta_ridge.row(i) = aux1;
733   }
734
735   // Calculate t0
736   arma::mat t0_hat_beta_ridge = cpp_hat_beta_ridge(X, y, k);
737
738   // Calculate boot_array
739   arma::mat boot_array = conv_to<mat>::from(cpp_boot_array(boot_idx));
740
741   // Calculate confidence interval
742   arma::mat output = cpp_boot_bca_ci(t0_hat_beta_ridge.t(), cube_hat_beta_ridge,
          conf, boot_array, n);
743
744
745   return List::create(Named("ci_beta_ridge") = output);
746 }
```

As rotinas em R são fornecidas a seguir:

```
1  ## pacotes
2  library(data.table)
3  Rcpp::sourceCpp('cpp_ridge_regression.cpp')
4
5
6  ## carregar dados
7  indat <- data.table::fread("data/RATOS.csv")
8
9  ## variaveis
10 attrs <- c("B1", "B4", "L", "SIGMA", "F", "R", "C", "log_PAPP", "PI", "MR4")
11 resp <- "POTENCIA"
12
13 ## Padronizando as variaveis
14 scl_valores <- vector("list", length(attrs))
15 names(scl_valores) <- attrs
16
17 for(i in attrs){
18   valor <- scale(indat[[i]])
19   raizn <-sqrt(length(indat[[i]]) - 1)
20   scl_valores[[i]][c(1,2)] = c(attr(valor, "scaled:center"), attr(valor, "scaled:
         scale") * raizn)
21   data.table::set(indat, j = i, value = valor / raizn)
22 }
23
24 ##  Matriz e vetor X e y para o ajuste
25 X <- cbind(1, as.matrix(indat[, .SD, .SDcols = attrs]))
26 y <- as.matrix(indat[, .SD, .SDcols = resp])
27 p <- ncol(X)
28
29 ## Matriz de correlacao
30 cor_x <- cor(X[, -1])
31
```

```
## copia dos dados e lista para salvar os passos dos ajustes
cindat <- data.table::copy(indat)
attrs_ajuste <- c("Intercepto", attrs)
lista_ajustes <- list()

set.seed(123)

## ajustar procedimento de modelagem

i <- 1
is_all_significant <- FALSE
p <- length(attrs_ajuste)
while(!is_all_significant | p == 1){
  print(i)
  ## Model argments
  if("Intercepto" %in% attrs_ajuste){
    X <- cbind(1, as.matrix(cindat[, .SD, .SDcols = attrs_ajuste[which(attrs_ajuste
        != "Intercepto")]]))
  } else {
    X <- as.matrix(cindat[, .SD, .SDcols = attrs_ajuste])
  }

  y <- as.matrix(cindat[, .SD, .SDcols = resp])
  p <- ncol(X)

  ## Decomposicao em valores singulares
  svd_x <- cpp_svd(X)
  if("Intercepto" %in% attrs_ajuste){
    svd_x_no_intercept <- cpp_svd(X[, -1])
    fiv_variaveis <- cpp_fiv(X[, -1])
    rownames(fiv_variaveis) <- attrs_ajuste[-1]
  } else {
    svd_x_no_intercept <- svd_x
    fiv_variaveis <- cpp_fiv(X)
    rownames(fiv_variaveis) <- attrs_ajuste
  }

  ## numero condicao
  max_lambda_i <- max(c(svd_x_no_intercept$s))
  min_lambda_i <- min(c(svd_x_no_intercept$s))
  num_condicao <- max_lambda_i / min_lambda_i


  ## beta estimado pelo metodo de minimos quadrados
  hat_beta_ols <- cpp_hat_beta_ridge(X, y, 0)

  ## estimativa de sigma
  hat_sigma <- cpp_hat_sigma(X, y, hat_beta_ols)

  ## encontrar valor para k
  all_ks <- cpp_calculate_all_k(X, y)
  all_ks <- data.table("k" = c("OLS", "HK", "MDG", "LW", "MG", "KS", "AKS_MA", "AKS
      _MAX", "DK"), "value" = round(c(all_ks), 3))
```

```
83  mse_k <- sapply(seq_len(nrow(all_ks)), function(i) round(cpp_mse_k(X, y, all_ks[[
        "value"]][i]), 3))
84  data.table::set(all_ks, j = "mse", value = mse_k)
85  k <- all_ks[which.min(mse)][["value"]]
86  which_k_type <- all_ks[which.min(mse)][["k"]]
87
88  ## Encontrar novo numero de condicao do novo sistema
89  num_condicao_ridge <- (max_lambda_i + k) / (min_lambda_i + k)
90
91  ## Estimativas coeficientes atraves do metodo de regressao em cristas
92  hat_beta_k <- cpp_hat_beta_ridge(X, y, k)
93
94  ## estimativa intervalo de confianca - metodo BCa
95  boot_obj <- cpp_boot_ridge_aplication(X, y, k, c(0.95), 5000, 123)
96  boot_obj <- boot_obj[["ci_beta_ridge"]]
97
98
99  ## data table com os coeficientes
100 coefs_dt <- data.table::data.table("variaveis" = attrs_ajuste,
101                                    "coeficientes_ols" = round(c(hat_beta_ols), 3)
                                           ,
102                                    "coeficientes_ridge" = round(c(hat_beta_k), 3)
                                           ,
103                                    "ci_limite_inferior" = round(c(boot_obj[,6]),
                                           3),
104                                    "ci_limite_superior" = round(c(boot_obj[,7]),
                                           3))
105 data.table::set(coefs_dt, j = "0_no_intervalo", value = data.table::between(0,
        coefs_dt$ci_limite_inferior, coefs_dt$ci_limite_superior))
106 data.table::set(coefs_dt, j = "compr_intervalo", value = round(abs(coefs_dt$ci_
        limite_superior-coefs_dt$ci_limite_inferior), 3))
107
108
109
110
111
112 ## Encontrar o maximo valor p maior que alpha=0.05
113 is_all_significant <- all(!coefs_dt[["0_no_intervalo"]])
114
115 which_not_significant_max_interval <- coefs_dt[which(`0_no_intervalo`),
116                                                .SD[which.max(compr_intervalo)]][[
                                                       "variaveis"]]
117
118
119 if(!is_all_significant & length(which_not_significant_max_interval) > 0){
120   attrs_ajuste <- attrs_ajuste[!attrs_ajuste %in% which_not_significant_max_
          interval]
121 }
122
123 data.table::set(coefs_dt, j = "IC(95%)", value = paste0("[",
124                                                          coefs_dt$ci_limite_
                                                                 inferior,
125                                                          ", ",
```

```
126                                                                  coefs_dt$ci_limite_
                                                                         superior,
127                                                                  "]"))
128   coefs_dt <- coefs_dt[, .SD, .SDcols = c("variaveis", "coeficientes_ridge", "IC(95
          %)")]
129
130
131   lista_ajustes[[i]] <- list("type_k" = which_k_type,
132                              "max_lambda_i" = max_lambda_i,
133                              "min_lambda_i" = min_lambda_i,
134                              "fiv" = fiv_variaveis,
135                              "numero_condicao_ols" = num_condicao,
136                              "numero_condicao_ridge" = num_condicao_ridge,
137                              "sigma_ols" = hat_sigma,
138                              "all_k_mse" = all_ks,
139                              "coeficientes" = coefs_dt,
140                              "variavel_eliminada" = which_not_significant_max_
                                    interval)
141
142   i = i + 1
143
144 }
145
146 run_simulation <- function(row_id, df_simulation){
147   starttime_all <- Sys.time()
148   ## semente
149   set.seed(row_id + 1234)
150
151   ## Extraindo os parametros da simulacao
152   n     <- df_simulation[row_id][["n"]]
153   p     <- df_simulation[row_id][["p"]]
154   rho   <- df_simulation[row_id][["rho"]]
155   sigma <- df_simulation[row_id][["sigma"]]
156   n_rep <- df_simulation[row_id][["n_rep"]]
157   B     <- df_simulation[row_id][["B"]]
158
159   ## Gerando a matriz X e pradronizando-a
160   Z <- matrix(rnorm(n * p), ncol = p, nrow = n)
161   X <- sqrt(1-rho^2) * Z+ rho * Z[, p]
162   X <- scale(X) / sqrt(n - 1)
163
164   ## Autovetor e autovalor
165   eigen_object <- cpp_svd(X)
166
167   ## numero condicao
168   num_cond <- c(sqrt(head(eigen_object[["s"]], 1) / tail(eigen_object[["s"]], 1)))
169
170   ## Criando os parametros - Betas
171   ind_zero <- 1 #indice que sera igualado a zero
172   beta_po <- sample(eigen_object[["V"]][, which.min(eigen_object[["s"]])])
173   beta_mo <- sample(eigen_object[["V"]][, which.max(eigen_object[["s"]])])
174   beta_po[ind_zero] <- beta_mo[ind_zero] <- 0
175
```

```
176 ## gerando epsilon
177 epsilon <- matrix(rnorm(n*n_rep, 0, sigma), ncol = n_rep)
178
179 ## Gerando y
180 y_mo <- apply(epsilon, 2, function(x) x + X %*% beta_mo)
181 y_po <- apply(epsilon, 2, function(x) x + X %*% beta_po)
182 cube_y <- array(c(y_mo, y_po), c(nrow(y_mo), ncol(y_mo), 2))
183 rm(list = c("epsilon", "y_mo", "y_po"))
184
185 ## calculating k for each y
186 cube_k <- cpp_calculate_all_k_mult_y(X, cube_y)
187
188 ### BOOSTRAP
189 starttime_boot <- Sys.time()
190 simulation <- cpp_boot_ridge_simulation(X, cube_y, cube_k, 0, c(0.90, 0.95, 0.99)
        , B, 123)
191 endtime_boot <- Sys.time()
192
193
194 ## Transform into a single data.table
195 simulation <- lapply(simulation, function(x){
196   aux <- data.table::as.data.table(x)
197   data.table::setnames(aux, names(aux),
198                        c("ci_type", "k_type", "rep", "alpha", "q1", "q2", "ci_
                             lower", "ci_upper"))
199 })
200 simulation <- data.table::rbindlist(simulation, idcol = "beta_orientation")
201
202 data.table::set(simulation, j = "sigma", value = sigma)
203 data.table::set(simulation, j = "n", value = n)
204 data.table::set(simulation, j = "p", value = p)
205 data.table::set(simulation, j = "rho", value = rho^2)
206 data.table::set(simulation, j = "n_rep", value = n_rep)
207 data.table::set(simulation, j = "B", value = B)
208 data.table::set(simulation, j = "num_cond", value = num_cond^2)
209
210 endtime_all <- Sys.time()
211
212
213
214 ## save object
215 list_output <- list("simulation_dt" = simulation,
216                     "time_bootstrap" =  difftime(endtime_boot, starttime_boot,
                          units = "s"),
217                     "time_total" = difftime(endtime_all, starttime_all, units = "
                          s"))
218
219 saveRDS(list_output, paste0("C:/Users/rober/OneDrive/Documentos/scripts_
        dissertacao/simulation_results/simulations_results_",
220                             row_id, ".rds"))
221
222 return(NULL)
223
```

```
}


run_simulation_parte_2 <- function(row_id, df_simulation, mult_fator){
  starttime_all <- Sys.time()
  ## semente
  set.seed(row_id + 1234)

  ## Extraindo os parametros da simulacao
  n     <- df_simulation[row_id][["n"]]
  p     <- df_simulation[row_id][["p"]]
  rho   <- df_simulation[row_id][["rho"]]
  sigma <- df_simulation[row_id][["sigma"]]
  n_rep <- df_simulation[row_id][["n_rep"]]
  B     <- df_simulation[row_id][["B"]]

  ## Gerando a matriz X e pradronizando-a
  Z <- matrix(rnorm(n * p), ncol = p, nrow = n)
  X <- sqrt(1-rho^2) * Z+ rho * Z[, p]
  X <- scale(X) / sqrt(n - 1)

  ## Autovetor e autovalor
  eigen_object <- cpp_svd(X)

  ## numero condicao
  num_cond <- c(sqrt(head(eigen_object[["s"]], 1) / tail(eigen_object[["s"]], 1)))

  ## Criando os parametros - Betas
  ind_zero          <- 1 #indice que sera igualado a zero
  idx               <- sample(seq_len(p), 1)
  beta_po           <- eigen_object[["V"]][, which.min(eigen_object[["s"]])]
  beta_mo           <- eigen_object[["V"]][, which.max(eigen_object[["s"]])]
  valor_beta_i      <- sqrt(diag(solve(t(X)%*%X)))*sigma * mult_fator
  if(idx == 1){
    beta_po[ind_zero] <- valor_beta_i[idx] * beta_po[idx]
    beta_mo[ind_zero] <- valor_beta_i[idx] * beta_mo[idx]
  } else {
    aux_po            <- beta_po[ind_zero]
    aux_mo            <- beta_mo[ind_zero]
    beta_po[ind_zero] <- valor_beta_i[idx] * beta_po[idx]
    beta_mo[ind_zero] <- valor_beta_i[idx] * beta_mo[idx]
    beta_po[idx]      <- aux_po
    beta_mo[idx]      <- aux_mo
  }



  ## gerando epsilon
  epsilon <- matrix(rnorm(n*n_rep, 0, sigma), ncol = n_rep)

  ## Gerando y
  y_mo <- apply(epsilon, 2, function(x) x + X %*% beta_mo)
  y_po <- apply(epsilon, 2, function(x) x + X %*% beta_po)
```

```
277 cube_y <- array(c(y_mo, y_po), c(nrow(y_mo), ncol(y_mo), 2))
278 rm(list = c("epsilon", "y_mo", "y_po"))
279
280 ## calculating k for each y
281 cube_k <- cpp_calculate_all_k_mult_y(X, cube_y)
282
283 ### BOOSTRAP
284 starttime_boot <- Sys.time()
285 simulation <- cpp_boot_ridge_simulation(X, cube_y, cube_k, 0, c(0.90, 0.95, 0.99)
        , B, 123)
286 endtime_boot <- Sys.time()
287
288
289 ## Transform into a single data.table
290 simulation <- lapply(simulation, function(x){
291   aux <- data.table::as.data.table(x)
292   data.table::setnames(aux, names(aux),
293                        c("ci_type", "k_type", "rep", "alpha", "q1", "q2", "ci_
                             lower", "ci_upper"))
294 })
295 simulation <- data.table::rbindlist(simulation, idcol = "beta_orientation")
296
297 data.table::set(simulation, j = "sigma", value = sigma)
298 data.table::set(simulation, j = "n", value = n)
299 data.table::set(simulation, j = "p", value = p)
300 data.table::set(simulation, j = "rho", value = rho^2)
301 data.table::set(simulation, j = "n_rep", value = n_rep)
302 data.table::set(simulation, j = "B", value = B)
303 data.table::set(simulation, j = "num_cond", value = num_cond^2)
304 data.table::set(simulation, j = "beta_po", value = beta_po[ind_zero])
305 data.table::set(simulation, j = "beta_mo", value = beta_mo[ind_zero])
306
307 endtime_all <- Sys.time()
308
309
310
311 ## save object
312 list_output <- list("simulation_dt" = simulation,
313                     "time_bootstrap" =  difftime(endtime_boot, starttime_boot,
                          units = "s"),
314                     "time_total" = difftime(endtime_all, starttime_all, units = "
                          s"))
315
316 ## create directory
317 output_basename <- paste0("C:/Users/rober/OneDrive/Documentos/scripts_dissertacao
        /simulation_results/mult_fator_", mult_fator)
318 if(!dir.exists(output_basename)){
319   dir.create(output_basename)
320 }
321 output_filename <- paste0(output_basename, "/simulations_results_", row_id, ".rds
        ")
322
323 saveRDS(list_output, output_filename)
```

```
324
325   return(NULL)
326
327 }
328
329 ## Simulacao
330 n      <- c(50, 250)
331 p      <- c(5, 10, 25)
332 rho    <- sqrt(c(0.5, 0.7, 0.9))
333 sigma <- c(0.1, 1, 5)
334 n_rep <- c(500)
335 B      <- c(5000)
336
337
338 ## df of simulation
339 df_simulation <- as.data.table(expand.grid(list(n = n,
340                                                 p = p,
341                                                 rho = rho,
342                                                 sigma = sigma,
343                                                 n_rep = n_rep,
344                                                 B = B)))
345
346 df_simulation <- df_simulation[n > p ]
347
348
349
350
351 ## Rodar Estudo
352 ids <- 1:54
353 for (i in ids) {
354   tryCatch({
355     print(i)
356     run_simulation_parte_2(i, df_simulation, 2)
357   }, error=function(e){})
358 }
359
360 ## Rodar resultados
361 ## pacotes
362 library(data.table)
363 library(ggplot2)
364
365 ## funcoes externas
366 source("008_funcoes_simulacao_sumarizados.R", encoding = "utf-8")
367
368 ## inputs
369 simulation_results_path   <- "simulation_results"
370 list_multipliers          <- c(0, 1, 2, 5, 10, 15)
371 outpath_data_by_parameter <- "simulation_results_by_parameters_sigma_num_cond"
372
373 inpath_data_by_parameter  <- paste0(outpath_data_by_parameter, "/", list.files(
      outpath_data_by_parameter))
374 breakpoints_beta          <- seq(-6, 6, 2)
375 ## run functions
```

```
extract_full_data_by_parameter(simulation_results_path, list_multipliers,
                               outpath_data_by_parameter)

tables_summarizations <- read_and_summarize_data_by_parameter(inpath_data_by_
    parameter, breakpoints_beta)
graphs                <- create_graph_acuracia(tables_summarizations)


## funcoes simulacoes (008_funcoes_simulacao_sumarizados.R)

extract_full_data_by_parameter <- function(simulation_results_path,
                                           list_multipliers,
                                           outpath_data_by_parameter){

  ## all parameters tested
  params <- c("ci_type", "k_type", "beta_orientation",
              "sigma", "rho", "n", "p", "num_cond")
  lapply(params, function(x){
    data_by_parameter <- lapply(list_multipliers, function(y){
      ## read files
      inpath_simulation_results_files <- paste0(simulation_results_path, "/mult_
          fator_", y)
      simulation_results_files        <- list.files(inpath_simulation_results_files
          )

      ## read all files and put them all in a single dataset
      input_data        <- lapply(simulation_results_files, function(y){
        readRDS(paste0(inpath_simulation_results_files, "/", y))[["simulation_dt"]]
      })
      names(input_data) <- simulation_results_files
      input_data        <- data.table::rbindlist(input_data, idcol = "simulation")
      ## TRUE beta for each orientation
      data.table::set(input_data, j = "true_beta",
                      value = with(input_data,
                                   dplyr::case_when(beta_orientation == "ci_beta_
                                       ridge_mo" ~ beta_mo,
                                                    beta_orientation == "ci_beta_
                                                        ridge_po" ~ beta_po)))
      ## subset by params
      input_data <- input_data[, .SD, .SDcols = unique(c("simulation", "alpha", "
          sigma", "num_cond",  x, "ci_lower", "ci_upper", "true_beta"))]

      ## create id
      data.table::set(input_data, j = "simulation", value = gsub("\\D", "", input_
          data[["simulation"]]))
      data.table::set(input_data, j = "simulation", value = paste0(y, input_data[["
          simulation"]], 1:nrow(input_data)))
      return(input_data)})
    data_by_parameter <- data.table::rbindlist(data_by_parameter)
    ## create directory
    if(!dir.exists(outpath_data_by_parameter)){
      dir.create(outpath_data_by_parameter)
    }
```

```
421     data_by_parameter <- split(data_by_parameter, by = "alpha")
422     alphas            <- gsub("\\.", "_", names(data_by_parameter))
423     path_to_save_file <- paste0(outpath_data_by_parameter, "/", "simulation_results
            _by_", x, "_alpha_", alphas, ".csv")
424     lapply(seq_len(length(data_by_parameter)), function(d){
425       data.table::fwrite(data_by_parameter[[d]], path_to_save_file[d])})
426     rm(data_by_parameter);gc()})
427   return(NULL)
428 }
429
430
431 read_and_summarize_data_by_parameter <- function(inpath_data_by_parameter,
        breakpoints_beta){
432   params         <- c("ci_type", "k_type", "beta_orientation", "sigma", "rho", "n",
          "p", "num_cond")
433
434   summary_tables <- lapply(inpath_data_by_parameter, function(x){
435     print(x)
436     ## read data
437     input_data <- data.table::fread(x, colClasses = list("character" = "simulation"
          ))
438
439     ## creating necessary columns
440     data.table::set(input_data, j = "poder_estimado", value = with(input_data, !
          between(0, ci_lower, ci_upper)))
441     data.table::set(input_data, j = "acuracia_estimada", value = with(input_data,
          between(true_beta, ci_lower, ci_upper)))
442     data.table::set(input_data, j = "amplitude_intervalo", value = with(input_data,
           abs(ci_upper - ci_lower)))
443
444     ## parameters
445     current_param <- gsub(paste0("(", paste(params, collapse = "|"), ").*"), "\\1",
446                           gsub(".*\\/simulation_results_by_", "", x))
447
448     ## classify num_cond
449     data.table::set(input_data, j = "num_cond",
450                     value = cut(input_data[["num_cond"]], c(0, 10, 100, 1000, Inf),
                              right = F, dig.lab = 4))
451
452
453     ## accuracy table ------
454     key_cols                <- c("alpha", current_param)
455     key_cols_sigma          <- unique(c("alpha", "sigma", current_param))
456     key_cols_num_cond       <- unique(c("alpha", "num_cond", current_param))
457     key_cols_sigma_num_cond <- unique(c("alpha", "sigma", "num_cond", current_param
          ))
458     list_key_cols           <- list(key_cols,
459                                     key_cols_sigma,
460                                     key_cols_num_cond,
461                                     key_cols_sigma_num_cond)
462     names_key_cols          <- c("param", "param_sigma", "param_num_cond", "param_
          sigma_num_cond")
463
```

```
464   accuracy_tables         <- lapply(list_key_cols, function(y){
465     accuracy_table <- input_data[, .("acuracia_estimada" = mean(acuracia_estimada
          ),
466                                      "acuracia_ep"       = sqrt(mean(acuracia_
                                             estimada)*(1-mean(acuracia_estimada))/
                                             sqrt(.N))),
467                                    keyby = y]
468   })
469   names(accuracy_tables)  <- names_key_cols
470
471
472   ## width table -------
473   width_tables <- lapply(list_key_cols, function(y){
474     input_data[, .("min_width" = min(amplitude_intervalo),
475                    "qt1_width" = quantile(amplitude_intervalo, 0.25),
476                    "med_width" = median(amplitude_intervalo),
477                    "qt3_width" = quantile(amplitude_intervalo, 0.75),
478                    "max_width" = max(amplitude_intervalo),
479                    "N"         = .N),
480                keyby = y]
481   })
482   names(width_tables) <- names_key_cols
483
484
485   ## power table ------
486   ### separate by less and greater than 0
487   which_less_than_0    <- which(input_data[["true_beta"]] < 0)
488   which_greater_than_0 <- which(input_data[["true_beta"]] > 0)
489
490   ### separate  breakpoits less, equal and greater than 0
491   negative_breakpoints <- breakpoints_beta[which(breakpoints_beta < 0)]
492   positive_breakpoints <- breakpoints_beta[which(breakpoints_beta > 0)]
493
494
495   ### creating true beta class
496   negative_classes        <- cut(input_data[["true_beta"]][which_less_than_0],
497                                  breaks = c(-Inf, negative_breakpoints, 0),
                                       include.lowest = F,
498                                  right = F)
499   positive_classes        <- cut(input_data[["true_beta"]][which_greater_than_0],
500                                  breaks = c(0, positive_breakpoints, Inf),
                                       include.lowest = F,
501                                  right = T)
502
503   levels_negative_classes <- levels(negative_classes)
504   levels_positive_classes <- levels(positive_classes)
505   levels_all_classes      <- c(levels_negative_classes, "[0,0]" ,levels_positive_
        classes)
506
507
508   ### fix infinite border
509   levels_all_classes      <- gsub("[-Inf", "(-Inf", levels_all_classes, fixed = T
        )
```

```
510     levels_all_classes      <- gsub("Inf]", "Inf)", levels_all_classes, fixed = T)
511     negative_classes        <- as.character(negative_classes)
512     negative_classes        <- gsub("[-Inf", "(-Inf", negative_classes, fixed = T)
513     positive_classes        <- as.character(positive_classes)
514     positive_classes        <- gsub("Inf]", "Inf)", positive_classes, fixed = T)
515
516     ### create column
517     data.table::set(input_data, j = "true_beta_class", value = "[0,0]")
518     data.table::set(input_data, i = which_less_than_0,    j = "true_beta_class",
            value = negative_classes)
519     data.table::set(input_data, i = which_greater_than_0, j = "true_beta_class",
            value = positive_classes)
520     data.table::set(input_data, j = "true_beta_class",
521                     value = factor(input_data[["true_beta_class"]],
522                                    levels = levels_all_classes))
523     power_key_cols                <- c("alpha", current_param, "true_beta_class")
524     power_key_cols_sigma          <- unique(c("alpha", "sigma", current_param, "
            true_beta_class"))
525     power_key_cols_num_cond       <- unique(c("alpha", "num_cond", current_param, "
            true_beta_class"))
526     power_key_cols_sigma_num_cond <- unique(c("alpha", "sigma", "num_cond", current
            _param, "true_beta_class"))
527     power_list_key_cols           <- list(power_key_cols,
528                                           power_key_cols_sigma,
529                                           power_key_cols_num_cond,
530                                           power_key_cols_sigma_num_cond)
531
532     power_tables <- lapply(power_list_key_cols, function(y){
533       input_data[, .("poder_estimado"    = mean(poder_estimado),
534                      "poder_estimado_sd" = sqrt(mean(poder_estimado)*(1-mean(poder_
                           estimado))/sqrt(.N)),
535                      "N"                 = .N),
536                  keyby = y]
537     })
538     names(power_tables) <- names_key_cols
539
540
541     rm(input_data);gc()
542     return(list("accuracy_table"    = accuracy_tables,
543                 "width_table"       = width_tables,
544                 "power_table"       = power_tables))
545
546   })
547
548   ## names of parameters
549   nms                   <- gsub("simulation_results_by_parameters_sigma_num_cond/
          simulation_results_by_", "", inpath_data_by_parameter)
550   nms                   <- gsub("\\.csv", "", nms)
551   names(summary_tables) <- nms
552   tables_in_list        <- c("param", "param_sigma", "param_num_cond", "param_sigma
          _num_cond")
553   output_table <- lapply(params, function(x){
554     which_nms <- grep(paste0("^", x, "_"), names(summary_tables))
```

```
555     ## param
556     accuracy_table <- lapply(tables_in_list, function(l){
557       data.table::rbindlist(lapply(summary_tables[which_nms],
558                                    function(y){
559                                      y[["accuracy_table"]][[l]]
560                                    }))
561     })
562     names(accuracy_table) <- tables_in_list
563
564
565     width_table <- lapply(tables_in_list, function(l){
566       data.table::rbindlist(lapply(summary_tables[which_nms],
567                                    function(y){
568                                      y[["width_table"]][[l]]
569                                    }))
570     })
571     names(width_table) <- tables_in_list
572
573     power_table <- lapply(tables_in_list,function(l){
574       data.table::rbindlist(lapply(summary_tables[which_nms],
575                                    function(y){
576                                      y[["power_table"]][[l]]
577                                    }))
578     })
579     names(power_table) <- tables_in_list
580
581
582     return(list("accuracy_table"    = accuracy_table,
583                 "width_table"       = width_table,
584                 "power_table"       = power_table))
585
586
587   })
588   names(output_table) <- params
589   return(output_table)
590 }
591
592 create_accuracy_graph_settings_by_parameter <- function(tab, param){
593
594   ## Classes
595   if(param == "k_type"){
596     levels <- with(tab, dplyr::case_when(k_type == "1" ~ "0",
597                                          k_type == "2" ~ "HKB",
598                                          k_type == "3" ~ "MDG",
599                                          k_type == "4" ~ "LW",
600                                          k_type == "5" ~ "MG",
601                                          k_type == "6" ~ "KS",
602                                          k_type == "7" ~ "AKS MA",
603                                          k_type == "8" ~ "AKS MAX",
604                                          k_type == "9" ~ "DK"))
605   } else {
606     if(param == "ci_type"){
607       levels <-  with(tab, dplyr::case_when(ci_type == "1" ~ "Basico",
```

```
608                                                ci_type == "2" ~ "Percentil",
609                                                ci_type == "3" ~ "Normal",
610                                                ci_type == "4" ~ "BCa"))
611     } else {
612       if(param == "beta_orientation"){
613         levels <- with(tab, dplyr::case_when(beta_orientation == "ci_beta_ridge_mo"
                ~ "Melhor orientacao",
614                                              beta_orientation == "ci_beta_ridge_po"
                     ~ "Pior orientacao"))
615       } else {
616         levels <- as.character((tab[[param]]))
617       }
618     }
619   }
620
621   title <- switch(param,
622                   "k_type"           = "Tipo k",
623                   "ci_type"          = "Tipo de IC\n Bootstrap",
624                   "beta_orientation" = bquote(.("Orientacao") ~ beta),
625                   "sigma"            = bquote(sigma),
626                   "rho"              = bquote(rho),
627                   "n"                = "n",
628                   "p"                = "p",
629                   "num_cond"         = "Numero Condicao")
630
631
632   return(list("title_guides" = title,
633               "param_levels" = levels))
634 }
635
636 create_graph_acuracia <- function(tables_summarizations){
637   params         <- c("ci_type", "k_type", "beta_orientation", "sigma", "rho", "n",
         "p", "num_cond")
638   graphs_acuracia <- lapply(names(tables_summarizations), function(x){
639     list_tabs <- tables_summarizations[[x]]
640     tabs      <- data.table::copy(list_tabs[["accuracy_table"]])
641     graphs <- lapply(names(tabs), function(y){
642       tab           <- data.table::copy(tabs[[y]])
643       current_param <- x
644       g_settings    <- create_accuracy_graph_settings_by_parameter(tab, current_
            param)
645
646       ## 1- alpha
647       data.table::set(tab, j = "alpha", value = 1- tab[["alpha"]])
648
649       ## bar labels
650       data.table::set(tab, j = "bar_label", value = round(tab[["acuracia_estimada"
            ]], 2))
651
652       ## change params levels
653       data.table::set(tab, j = current_param, value = g_settings[["param_levels"]])
654
655       ## labels colors
```

```
656       labels_fill_graph        <- unique(g_settings[["param_levels"]])
657       names(labels_fill_graph) <- labels_fill_graph
658
659       if(y == "param_sigma_num_cond"){
660         data.table::set(tab, j = "alpha", value = round(tab[["alpha"]], 2))
661         tab <- tab[alpha == 0.05]
662       }
663
664
665
666       plot1 <- ggplot(data = tab, aes(x = get(current_param), y = acuracia_estimada
              )) +
667         geom_bar(aes(group = get(current_param), fill = get(current_param)),
668                  stat = "identity", width = 0.6, alpha = 0.75) +
669         geom_text(aes(label = bar_label), vjust = 3) +
670         geom_errorbar(aes(ymin = acuracia_estimada - 1.96*acuracia_ep,
671                           ymax = acuracia_estimada + 1.96*acuracia_ep),
672                       width = 0.5)
673
674       if(y == "param"){
675         plot1 <- plot1 +
676           facet_wrap(vars(alpha),
677                      ncol = 3,
678                      labeller = label_bquote(cols = alpha == .(alpha)))
679       } else {
680         if( y == "param_sigma"){
681           plot1 <- plot1 +
682             facet_wrap(vars(alpha, sigma),
683                        ncol = 3,
684                        labeller = label_bquote(cols = alpha == .(alpha) ~ "e" ~
                                     sigma == .(sigma)))
685         } else {
686           if(y == "param_num_cond"){
687             data.table::set(tab, j = "num_cond", value = as.character(tab[["num_
                    cond"]]))
688             plot1 <- plot1 +
689               facet_wrap(vars(alpha, num_cond),
690                          ncol = 3,
691                          labeller = label_bquote(cols = alpha == .(alpha) ~ "e N.
                                       Cond." == .(num_cond)))
692           } else {
693             data.table::set(tab, j = "num_cond", value = as.character(tab[["num_
                    cond"]]))
694             plot1 <- plot1 +
695               facet_wrap(vars(sigma, num_cond),
696                          ncol = 3,
697                          labeller = label_bquote(cols = sigma == .(sigma) ~ "e N.
                                       Cond." == .(num_cond)))
698           }
699         }
700       }
701
702       plot1 <- plot1 +
```

```
703         theme_bw() +
704         ylab("Acuracia") +
705         scale_x_discrete(g_settings[["title_guides"]],
706                          labels = labels_fill_graph) +
707         scale_fill_brewer(type = "qual", palette = "Set1", direction = -1,
708                           labels = labels_fill_graph) +
709         geom_hline(aes(yintercept = 1- alpha, linetype  = "1-alpha")) +
710         scale_linetype_manual(values = c("1-alpha" = "dotted"),
711                               labels = c("1-alpha" = bquote("1-" ~ alpha))) +
712         labs(fill = g_settings[["title_guides"]],
713              linetype = "")
714       return(plot1)
715     })
716     names(graphs) <- names(tabs)
717     return(graphs)
718   })
719   names(graphs_acuracia) <- names(tables_summarizations)
720
721   graphs_width <- lapply(names(tables_summarizations), function(x){
722     list_tabs <- tables_summarizations[[x]]
723     tabs      <- data.table::copy(list_tabs[["width_table"]])
724     graphs <- lapply(names(tabs), function(y){
725       tab           <- data.table::copy(tabs[[y]])
726       current_param <- x
727       g_settings    <- create_accuracy_graph_settings_by_parameter(tab, current_
              param)
728
729       ## 1- alpha
730       data.table::set(tab, j = "alpha", value = 1- tab[["alpha"]])
731
732       ## change params levels
733       data.table::set(tab, j = current_param, value = g_settings[["param_levels"]])
734
735
736       ## labels colors
737       labels_fill_graph        <- unique(g_settings[["param_levels"]])
738       names(labels_fill_graph) <- labels_fill_graph
739
740       data.table::set(tab, j = "new_min", value = with(tab, pmax(min_width, qt1_
              width - 1.5*(qt3_width - qt1_width))))
741       data.table::set(tab, j = "new_max", value = with(tab, pmin(max_width, qt3_
              width + 1.5*(qt3_width - qt1_width))))
742
743       if(y == "param_sigma_num_cond"){
744         data.table::set(tab, j = "alpha", value = round(tab[["alpha"]], 2))
745         tab <- tab[alpha == 0.05]
746       }
747
748       plot1 <- ggplot(data = tab, aes(x = get(current_param),
749                                       group = get(current_param))) +
750         geom_boxplot(aes(ymin   = new_min,
751                          lower  = qt1_width,
752                          middle = med_width,
```

```
753                         upper  = qt3_width,
754                         ymax   = new_max,
755                         fill   = get(current_param)),
756                     stat = "identity", alpha = 0.75)
757
758       if(y == "param"){
759         plot1 <- plot1 +
760           facet_wrap(vars(alpha),
761                      ncol = 3,
762                      labeller = label_bquote(cols = alpha == .(alpha)))
763       } else {
764         if( y == "param_sigma"){
765           plot1 <- plot1 +
766             facet_wrap(vars(alpha, sigma),
767                        ncol = 3,
768                        labeller = label_bquote(cols = alpha == .(alpha) ~ "e" ~
                               sigma == .(sigma)))
769         } else {
770           if(y == "param_num_cond"){
771             data.table::set(tab, j = "num_cond", value = as.character(tab[["num_
                    cond"]]))
772             plot1 <- plot1 +
773               facet_wrap(vars(alpha, num_cond),
774                          ncol = 3,
775                          labeller = label_bquote(cols = alpha == .(alpha) ~ "e N.
                                Cond." == .(num_cond)))
776           } else {
777             data.table::set(tab, j = "num_cond", value = as.character(tab[["num_
                    cond"]]))
778             plot1 <- plot1 +
779               facet_wrap(vars(sigma, num_cond),
780                          ncol = 3,
781                          labeller = label_bquote(cols = sigma == .(sigma) ~ "e N.
                                Cond." == .(num_cond)))
782           }
783         }
784       }
785
786
787       plot1 <- plot1+
788         theme_bw() +
789         ylab("Cumprimento do IC") +
790         scale_x_discrete(g_settings[["title_guides"]],
791                          labels = labels_fill_graph) +
792         scale_fill_brewer(type = "qual", palette = "Set1", direction = -1,
793                           labels = labels_fill_graph) +
794         scale_shape_manual(values = c("max" = 8),
795                            labels = c("max" = "maximo")) +
796         labs(fill  = g_settings[["title_guides"]],
797              shape = "")
798       return(plot1)
799
800     })
```

```
801     names(graphs) <- names(tabs)
802     return(graphs)
803   })
804   names(graphs_width) <- names(tables_summarizations)
805
806   graphs_poder <- lapply(names(tables_summarizations), function(x){
807     list_tabs <- tables_summarizations[[x]]
808     tabs      <- data.table::copy(list_tabs[["power_table"]])
809     graphs <- lapply(names(tabs), function(y){
810       tab           <- data.table::copy(tabs[[y]])
811       current_param <- x
812       g_settings    <- create_accuracy_graph_settings_by_parameter(tab, current_
              param)
813
814       ## 1- alpha
815       data.table::set(tab, j = "alpha", value = 1- tab[["alpha"]])
816
817       ## change params levels
818       data.table::set(tab, j = current_param, value = g_settings[["param_levels"]])
819
820
821       ## labels colors
822       labels_fill_graph        <- unique(g_settings[["param_levels"]])
823       names(labels_fill_graph) <- labels_fill_graph
824
825
826       if(y == "param_sigma_num_cond"){
827         data.table::set(tab, j = "alpha", value = round(tab[["alpha"]], 2))
828         tab <- tab[alpha == 0.05]
829       }
830
831
832       plot1 <- ggplot(data = tab, aes(x = true_beta_class, y = poder_estimado )) +
833         geom_line(aes(group = get(current_param), colour = get(current_param)),
              size = 1.3, alpha = 0.75) +
834         geom_point(aes(group = get(current_param), colour = get(current_param)),
              size = 1.8)
835
836       if(y == "param"){
837         plot1 <- plot1 +
838           facet_wrap(vars(alpha),
839                      ncol = 3,
840                      labeller = label_bquote(cols = alpha == .(alpha)))
841       } else {
842         if( y == "param_sigma"){
843           plot1 <- plot1 +
844             facet_wrap(vars(alpha, sigma),
845                        ncol = 3,
846                        labeller = label_bquote(cols = alpha == .(alpha) ~ "e" ~
                           sigma == .(sigma)))
847         } else {
848           if(y == "param_num_cond"){
849             data.table::set(tab, j = "num_cond", value = as.character(tab[["num_
```

```
                cond"]]))
850             plot1 <- plot1 +
851               facet_wrap(vars(alpha, num_cond),
852                          ncol = 3,
853                          labeller = label_bquote(cols = alpha == .(alpha) ~ "e N.
                               Cond." == .(num_cond)))
854           } else {
855             data.table::set(tab, j = "num_cond", value = as.character(tab[["num_
                  cond"]]))
856             plot1 <- plot1 +
857               facet_wrap(vars(sigma, num_cond),
858                          ncol = 3,
859                          labeller = label_bquote(cols = sigma == .(sigma) ~ "e N.
                               Cond." == .(num_cond)))
860           }
861         }
862       }
863
864
865       plot1 <- plot1 +
866         theme_bw() +
867         ylab("Poder Estimado") +
868         xlab(expression(beta)) +
869         scale_colour_brewer(type = "qual", palette = "Set1", direction = -1,
870                             labels = labels_fill_graph) +
871         geom_hline(aes(yintercept = alpha, linetype  = "alpha")) +
872         scale_linetype_manual(values = c("alpha" = "dotted"),
873                               labels = c("alpha" = bquote(alpha))) +
874         labs(colour = g_settings[["title_guides"]],
875              linetype = "")
876       return(plot1)
877
878     })
879     names(graphs) <- names(tabs)
880     return(graphs)
881   })
882   names(graphs_poder) <- names(tables_summarizations)
883
884
885   return(list("graphs_acuracia"    = graphs_acuracia,
886               "graphs_width"       = graphs_width,
887               "graphs_poder"       = graphs_poder))
888 }
889
890 export_graphs_as_pdf <- function(graphs_list, width = 12, height = 6){
891   graphs_list  <- unlist(unlist(graphs_list, recursive = F), recursive = F)
892   graphs_names <- paste0(gsub("\\.", "_", names(graphs_list)), ".pdf")
893
894   if(!dir.exists("graph_results_2")){
895     dir.create("graph_results_2")
896   }
897
898   lapply(seq_len(length(graphs_list)), function(x){
```

```
899     ggplot2::ggsave(graphs_names[x],
900                     graphs_list[[x]],
901                     path = "graph_results_2",
902                     width = width,
903                     height = height)
904   })
905 }
906
907 export_graphs_selected_as_pdf <- function(graphs_list, width = 12, height = 6){
908
909   if(!dir.exists("graph_results_selected")){
910     dir.create("graph_results_selected")
911   }
912
913   ## names graphs
914   acuracia_param_names    <- names(graphs_list[[1]])
915   acuracia_names_seq      <- seq(1, length.out = length(acuracia_param_names), by =
          6)
916   acuracia_graphs_1_names <- paste0(acuracia_names_seq, "_acuracia_", acuracia_
          param_names, ".pdf")
917   acuracia_graphs_2_names <- paste0(acuracia_names_seq + 1, "_acuracia_", acuracia_
          param_names, ".pdf")
918
919   width_param_names       <- names(graphs_list[[2]])
920   width_names_seq         <- seq(3, length.out = length(width_param_names), by = 6)
921   width_graphs_1_names    <- paste0(width_names_seq, "_width_", width_param_names,
          ".pdf")
922   width_graphs_2_names    <- paste0(width_names_seq + 1, "_width_", width_param_
          names, ".pdf")
923
924   power_param_names       <- names(graphs_list[[3]])
925   power_names_seq         <- seq(5, length.out = length(power_param_names), by = 6)
926   power_graphs_1_names    <- paste0(power_names_seq, "_power_", power_param_names,
          ".pdf")
927   power_graphs_2_names    <- paste0(power_names_seq + 1, "_power_", power_param_
          names, ".pdf")
928
929
930
931   ## grafico acuracia
932
933   lapply(seq_len(length(acuracia_param_names)), function(x){
934     ## acuracia
935
936     ggplot2::ggsave(acuracia_graphs_1_names[x],
937                     graphs_list[[1]][[x]][[1]],
938                     path = "graph_results_selected",
939                     width = width,
940                     height = height)
941
942     ggplot2::ggsave(acuracia_graphs_2_names[x],
943                     graphs_list[[1]][[x]][[4]],
944                     path = "graph_results_selected",
```

```
                    width = width,
                    height = height)


    ## width
    ggplot2::ggsave(width_graphs_1_names[x],
                    graphs_list[[2]][[x]][[1]],
                    path = "graph_results_selected",
                    width = width,
                    height = height)

    ggplot2::ggsave(width_graphs_2_names[x],
                    graphs_list[[2]][[x]][[4]],
                    path = "graph_results_selected",
                    width = width,
                    height = height)

    ## power
    ggplot2::ggsave(power_graphs_1_names[x],
                    graphs_list[[3]][[x]][[1]],
                    path = "graph_results_selected",
                    width = width,
                    height = height)

    ggplot2::ggsave(power_graphs_2_names[x],
                    graphs_list[[3]][[x]][[4]],
                    path = "graph_results_selected",
                    width = width,
                    height = height)


  })

  return(NULL)
}
```

Referências Bibliográficas

Alkhamisi *et al.***(2006)** Mahdi Alkhamisi, Ghadban Khalaf e Ghazi Shukur. Some modifications for choosing ridge parameters. *Communications in Statistics-Theory and Methods*, 35(11):2005–2020. Citado na pág. 19, 47, 48

Billor e Loynes(1993) N Billor e RM Loynes. Local influence: a new approach. *Communications in Statistics-Theory and Methods*, 22(6):1595–1611. Citado na pág. 16

Burr e Fry(2005) Tom L Burr e Herbert A Fry. Biased regression: The case for cautious application. *Technometrics*, 47(3):284–296. Citado na pág. 51

Coutsourides *et al.***(1979)** EG Coutsourides *et al.* F and t tests for a general class of estimators. *South African Statistical Journal*, 13(2):113–119. Citado na pág. 22

Davison e Hinkley(1997) Anthony Christopher Davison e David Victor Hinkley. *Bootstrap methods and their application*, volume 1. Cambridge university press. Citado na pág. 32, 33, 34, 38

Dorugade e Kashid(2010) AV Dorugade e DN Kashid. Alternative method for choosing ridge parameter for regression. *Applied Mathematical Sciences*, 4(9):447–456. Citado na pág. 1, 20, 48

Draper e Smith(1998) Norman R Draper e Harry Smith. *Applied regression analysis*, volume 326. John Wiley & Sons. Citado na pág. 1

Eddelbuettel e Sanderson(2014) Dirk Eddelbuettel e Conrad Sanderson. Rcpparmadillo: Accelerating r with high-performance c++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063. URL http://dx.doi.org/10.1016/j.csda.2013.02.005. Citado na pág. 68

Efron(1982) Bradley Efron. *The jackknife, the bootstrap, and other resampling plans*, volume 38. Siam. Citado na pág. 31

Efron e Tibshirani(1994) Bradley Efron e Robert J Tibshirani. *An introduction to the bootstrap*. CRC press. Citado na pág. 32, 33, 37, 39

Eicker *et al.***(1963)** Friedhelm Eicker *et al.* Asymptotic normality and consistency of the least squares estimators for families of linear regressions. *The Annals of Mathematical Statistics*, 34 (2):447–456. Citado na pág. 4

Firinguetti-Limone e Pereira-Barahona(2019) Luis Firinguetti-Limone e Manuel Pereira-Barahona. Bayesian estimation of the shrinkage parameter in ridge regression. *Communications in Statistics-Simulation and Computation*, páginas 1–14. Citado na pág. 20

Gibbons(1981) Diane Galarneau Gibbons. A simulation study of some ridge estimators. *Journal of the American Statistical Association*, 76(373):131–139. Citado na pág. 41

Goldstein e Smith(1974) M Goldstein e Adrian FM Smith. Ridge-type estimators for regression analysis. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):284–291. Citado na pág. 10, 11

Halawa e El Bassiouni(2000) AM Halawa e MY El Bassiouni. Tests of regression coefficients under ridge regression models. *Journal of Statistical Computation and Simulation*, 65(1-4):341–356. Citado na pág. 1, 22, 28

Hocking *et al.***(1976)** Ronald R Hocking, FM Speed e MJ Lynn. A class of biased estimators in linear regression. *Technometrics*, 18(4):425–437. Citado na pág. 20

Hoerl(1959) Arthur E Hoerl. Optimum solution of many variables equations. *Chemical Engineering Progress*, 55(11):69–78. Citado na pág. 5

Hoerl e Kennard(1970a) Arthur E Hoerl e Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67. Citado na pág. 1, 3, 9, 11, 12, 17

Hoerl e Kennard(1970b) Arthur E Hoerl e Robert W Kennard. Ridge regression: applications to nonorthogonal problems. *Technometrics*, 12(1):69–82. Citado na pág. 1, 12, 17

Hoerl e W. Kennard(1990) Arthur E Hoerl e Robert W. Kennard. Ridge regression: degrees of freedom in the analysis of variance. *Communications in Statistics-Simulation and Computation*, 19(4):1485–1495. Citado na pág. 1, 22, 26, 28

Hoerl *et al.***(1975)** Arthur E Hoerl, Robert W Kannard e Kent F Baldwin. Ridge regression: some simulations. *Communications in Statistics-Theory and Methods*, 4(2):105–123. Citado na pág. 1, 17, 19, 23, 48

Khalaf e Shukur(2005) Ghadban Khalaf e Ghazi Shukur. Choosing ridge parameter for regression problems. Citado na pág. 1, 19, 47

Kibria(2003) BM Golam Kibria. Performance of some new ridge regression estimators. *Communications in Statistics-Simulation and Computation*, 32(2):419–435. Citado na pág. 1, 17, 19, 20, 47, 48

Lawless e Wang(1976) Jerald F Lawless e P Wang. Simulation study of ridge and other regression estimators. *Communications in Statistics Part A-Theory and Methods*, (4):307–323. Citado na pág. 1, 17, 18, 47

Lawrence e Marsh(1984) Kenneth D Lawrence e Lawrence C Marsh. Robust ridge estimation methods for predicting us coal mining fatalities. *Communications in Statistics-Theory and Methods*, 13(2):139–149. Citado na pág. 16

Lindley e Smith(1972) Dennis V Lindley e Adrian FM Smith. Bayes estimates for the linear model. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(1):1–18. Citado na pág. 1, 14

Mallows(1973) Colin L Mallows. Some comments on c p. *Technometrics*, 15(4):661–675. Citado na pág. 13

Mason e Gunst(1985) Robert L Mason e Richard F Gunst. Outlier-induced collinearities. *Technometrics*, 27(4):401–407. Citado na pág. 15

McDonald e Galarneau(1975) Gary C McDonald e Diane I Galarneau. A monte carlo evaluation of some ridge-type estimators. *Journal of the American Statistical Association*, 70(350):407–416. Citado na pág. 1, 17, 18, 41, 47, 48

Montgomery *et al.***(2012)** Douglas C Montgomery, Elizabeth A Peck e G Geoffrey Vining. *Introduction to linear regression analysis*, volume 821. John Wiley & Sons. Citado na pág. 1, 7, 26, 31

Nomura(1988) Masuo Nomura. On the almost unbiased ridge regression estimator. *Communications in Statistics-Simulation and Computation*, 17(3):729–743. Citado na pág. 20

Obenchain(1977) RL Obenchain. Classical f-tests and confidence regions for ridge regression. *Technometrics*, 19(4):429–439. Citado na pág. 1, 22

Ohtani(1985) Kazuhiro Ohtani. Bounds of the f-ratio incorporating the ordinary ridge regression estimator. *Economics Letters*, 18(2-3):161–164. Citado na pág. 22, 23

Oikawa(2008) Koki Fernando Oikawa. *Análise de Influência na Regressão em Cristas*. Tese de Doutorado, Instituto de Matemática e Estatística - USP. Citado na pág. 15

Oikawa e Elian(2012) Koki Fernando Oikawa e Silvia Nagib Elian. Análise de influência na regressão em cristas. Citado na pág. 15

Oishi(1983) Jorge Oishi. *Regressao sobre cristas*. Dissertação de Mestrado, Instituto de Matemática e Estatística - USP. Citado na pág. iii, v, 1, 67

R Core Team(2019) R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019. URL https://www.R-project.org/. Citado na pág. 67

Rao *et al.***(1973)** Calyampudi Radhakrishna Rao, Calyampudi Radhakrishna Rao, Mathematischer Statistiker, Calyampudi Radhakrishna Rao e Calyampudi Radhakrishna Rao. *Linear statistical inference and its applications*, volume 2. Wiley New York. Citado na pág. 42

Riley(1955) James D. Riley. Solving systems of linear equations with a positive definite, symmetric, but possibly ill-conditioned matrix. *Mathematical Tables and Other Aids to Computation*, 9(51):96–101. ISSN 08916837. URL http://www.jstor.org/stable/2002065. Citado na pág. 7, 8

Searle e Khuri(2017) Shayle R Searle e Andre I Khuri. *Matrix algebra useful for statistics*. John Wiley & Sons. Citado na pág. 5

Ullah *et al.***(1984)** Aman Ullah, Richard AL Carter e Virendra K Srivastava. The sampling distribution of shrinkage estimators and theirf-ratios in the regression model. *Journal of Econometrics*, 25(1-2):109–122. Citado na pág. 1, 22

Walker e Birch(1988) Esteban Walker e Jeffrey B Birch. Influence measures in ridge regression. *Technometrics*, 30(2):221–227. Citado na pág. 15, 16

Walker e Page(2001) Stephen G Walker e Christopher J Page. Generalized ridge regression and a generalization of the cp statistic. *Journal of applied statistics*, 28(7):911–922. Citado na pág. 1, 12, 13, 14

Wichern e Churchill(1978) Dean W Wichern e Gilbert A Churchill. A comparison of ridge estimators. *Technometrics*, 20(3):301–311. Citado na pág. 1, 18