



Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Modelagem Matemática e Computacional

ANÁLISE DE FORMAS E ROBUSTEZ: MÉTODOS EXISTENTES E
ALGUMAS PROPOSIÇÕES.

Rafael Oliveira do Nascimento

Dissertação de Mestrado apresentada ao
Programa de Pós-Graduação em Modelagem
Matemática e Computacional, da Universidade
Federal da Paraíba, como parte dos requisitos
necessários à obtenção do título de Mestre em
Modelagem Matemática e Computacional.

Orientadores: Eufrásio de Andrade Lima Neto
Marcelo R. Portela Ferreira

João Pessoa
Agosto de 2023

Ata da Sessão Pública de Defesa de Dissertação de
Mestrado de **RAFAEL OLIVEIRA DO
NASCIMENTO**, candidato ao título de Mestre em
Modelagem Matemática e Computacional, realizada no
dia 29 de agosto de 2023.

1 Aos vinte e nove dias do mês de agosto do ano de dois mil e vinte e três, às 10h30, via
2 videoconferência, reuniram-se os membros da Banca Examinadora constituída para julgar o
3 Trabalho Final do discente RAFAEL OLIVEIRA DO NASCIMENTO, vinculado a
4 Universidade Federal da Paraíba sob a matrícula nº 20211000786, candidato ao grau de
5 Mestre em “*Modelagem Matemática e Computacional*”, na linha de pesquisa “*Modelagem*
6 *Probabilística*”, do Programa de Pós-Graduação em Modelagem Matemática e
7 Computacional. A comissão examinadora foi composta pelos professores Eufrásio de Andrade
8 Lima Neto, Orientador e Presidente da Banca; Marcelo Rodrigo Portela Ferreira,
9 Coorientador; Tarciana Liberal Pereira de Araujo, Examinadora Interna ao Programa; e
10 Hemílio Fernandes Campos Coelho, Examinador Externo ao Programa. Dando início aos
11 trabalhos, o Presidente da Banca cumprimentou os presentes, comunicou a finalidade da
12 reunião e passou a palavra ao candidato para que fizesse, oralmente, a exposição do trabalho
13 de dissertação intitulado “*ANÁLISE DE FORMAS E ROBUSTEZ: MÉTODOS EXISTENTES*
14 *E ALGUMAS PROPOSIÇÕES*”. Concluída a exposição, o candidato foi arguido pela Banca
15 Examinadora, que emitiu o seguinte parecer: “*aprovado*”. Do ocorrido, eu, Gean Paulo P. M.
16 de Barros, secretário do Programa de Pós-Graduação em Modelagem Matemática e
17 Computacional (PPGMMC), lavrei a presente ata, que vai assinada por mim e pelos membros
18 da Banca Examinadora.

João Pessoa, 29 de agosto de 2023.

Gean Paulo Pereira Maurício de Barros
Secretário do PPGMMC
SIAPE 2326476

Prof. Dr. Eufrásio de Andrade Lima Neto
Orientador (PPGMMC)

Prof. Dr. Marcelo Rodrigo Portela Ferreira
Coorientador (PPGMMC)

Prof^a. Dr^a. Tarciana Liberal Pereira de Araujo
Examinadora Interna ao Programa (PPGMMC)

Prof. Dr. Hemílio Fernandes Campos Coelho
Examinador Externo ao Programa (UFPB)



Documento assinado digitalmente
 MARCELO RODRIGO PORTELA FERREIRA
Data: 27/09/2023 16:55:53-0300
Verifique em <https://validar.iti.gov.br>

Documento assinado digitalmente
 TARCIANA LIBERAL PEREIRA DE ARAUJO
Data: 29/09/2023 22:15:25-0300
Verifique em <https://validar.iti.gov.br>

Documento assinado digitalmente
 HEMILIO FERNANDES CAMPOS COELHO
Data: 27/09/2023 07:18:35-0300
Verifique em <https://validar.iti.gov.br>

ANÁLISE DE FORMAS E ROBUSTEZ: MÉTODOS EXISTENTES E
ALGUMAS PROPOSIÇÕES.

Rafael Oliveira do Nascimento

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE
PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL
(PPGMMC) DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL
DA PARAÍBA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A
OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM MODELAGEM
MATEMÁTICA E COMPUTACIONAL.

Examinada por:

Prof. Eufrásio de Andrade Lima Neto, D.Sc.

Prof. Marcelo Rodrigo Portela Ferreira, D.Sc.

Prof. Tarciana Liberal Pereira, D.Sc.

Prof. Hemílio Fernandes Campos Coelho, D.Sc.

JOÃO PESSOA, PB – BRASIL
AGOSTO DE 2023

Catálogo na publicação
Seção de Catalogação e Classificação

N244a Nascimento, Rafael Oliveira do.

Análise de forma e robustez : métodos existentes e algumas proposições / Rafael Oliveira do Nascimento. - João Pessoa, 2023.

71 f. : il.

Orientação: Eufrásio de Andrade Lima Neto.

Coorientação: Marcelo Rodrigo Portela Ferreira.

Dissertação (Mestrado) - UFPB/CI.

1. Análise estatística. 2. Análise de formas. 3. Procrustes. 4. Métodos robustos. 5. Kernel. I. Lima Neto, Eufrásio de Andrade. II. Ferreira, Marcelo Rodrigo Portela. III. Título.

UFPB/BC

CDU 519.23(043)

*Dedico este trabalho a aqueles
que me precederam e cuja
linhagem pertenço.
Que sua sabedoria e visão
aguçada permaneçam em mim
não só pela herança de sangue,
mas também, pela memória.*

Agradecimentos

A gratidão é um sentimento sublime, o que a torna difícil de ser descrita em palavras. No entanto, há muitos a quem devo agradecer, então é bom começar.

Quero, em primeiro lugar, expressar minha gratidão àquele que nos concedeu o espírito sem limite, que nos impulsiona, nos dá forças e ânimo para enfrentar a nossa luta diária.

Gostaria de agradecer também aos meus filhos, Cayo e Isabele, a quem amo de todo coração e cuja companhia tem sido a única coisa capaz de afastar o meu cansaço.

Agradeço também aos meus familiares por tudo que representam, principalmente a minha mãe Hilda Gomes de Oliveira, é desejo de meu coração que sua caminhada pela sombra não se repita e que o poder de sua visão não volte a esmorecer.

Aos professores Eufrásio e Marcelo, meus orientadores, expresso todo o meu carinho e apreço, pela acolhida, pela caminhada e pela vitória. Vocês são faróis que brilham com a intensidade de um sol, afastaram a escuridão e com ela a dúvida, com vocês aprendi a inutilidade da potência sem controle, a perseverar e a ter paciência.

Aos meus colegas Wanes, Joaquim e Erissandro pela amizade e companheirismo na jornada, pela ajuda nas horas de dificuldades, agradeço profundamente, que os desafios vindouros sejam dignos de vossas senhorias e que nossos caminhos se cruzem novamente.

Por fim, agradeço a todos os que, direta ou indiretamente, contribuíram para tornar este sonho realidade.

Resumo da Dissertação apresentada ao PPGMMC/CI/UFPB como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ANÁLISE DE FORMAS E ROBUSTEZ: MÉTODOS EXISTENTES E ALGUMAS PROPOSIÇÕES

Rafael Oliveira do Nascimento

Agosto/2023

Orientadores: Eufrásio de Andrade Lima Neto

Marcelo R. Portela Ferreira

Programa: Modelagem Matemática e Computacional

A análise estatística de formas é o estudo das variações de forma, tamanho e suas covariações. Embora a maioria das aplicações diga respeito à biologia, a análise estatística de formas tem se tornado uma poderosa ferramenta, com diversas aplicações nos campos da arqueologia, paleontologia, geografia, e medicina. A análise procustes é um dos métodos mais empregados na análise estatística de formas e representa um papel importante para mensurar, comparar e estimar a forma média de objetos. Entretanto, tal técnica é baseada no método de mínimos quadrados que é severamente afetado quando os dados apresentam observações aberrantes (*outliers*). Neste trabalho propomos um conjunto de métodos para análise procustes considerando alguns dos estimadores robustos existentes na literatura.

Palavra-chave: Análise de formas, procustes, métodos robustos, kernel.

Abstract of Dissertation presented to PPGMMC/CI/UFPB as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ROBUST PROCRUSTES ESTIMATORS FOR MEAN SHAPES

Rafael Oliveira do Nascimento

August/2023

Advisors: Eufrásio de Andrade Lima Neto

Marcelo R. Portela Ferreira

Program: Computational Mathematical Modelling

Statistical analysis of shapes is the study of variations in shape, size and their covariations. Although most applications relate to biology, the statistical analysis of shapes has become a powerful tool, with diverse applications in the fields of archeology, paleontology, geography, and medicine. Procrustes analysis is one of the most used methods in the statistical analysis of shapes and plays an important role in measuring, comparing and estimating the average shape of objects. However, such a technique is based on the least squares method, which is severely affected when the data have outliers. In this work we propose a set of methods for procrustes analysis considering some of the existing robust estimators in the literature.

Keywords: Shape analysis, procrustes, robust methods, kernel.

Sumário

Lista de Figuras	x
Lista de Tabelas	xi
1 Introdução	1
1.1 Objetivos	3
1.2 Estrutura do trabalho	3
2 Revisão bibliográfica	4
2.1 Análise Estatística de Formas	4
2.1.1 Marcos Anatômicos ou Pontos de Referência	5
2.1.2 Representação Matemática de Formas	6
2.1.3 Matriz de Helmert e Pré-forma	7
2.1.4 Forma Média	10
2.2 A Análise de Procrustes	11
2.3 Modelos de regressão	13
2.3.1 O Método dos Mínimos Quadrados	13
2.3.2 Regressão robusta	14
2.4 Estimadores robustos para análise de procustes	18
3 Métodos propostos	20
3.1 Decomposição em Valores Singulares	20
3.2 SVD e a análise procustes	21
3.2.1 Estimadores robustos para análise procrustes	22
3.3 Análise generalizada de procustes completa (<i>full GPA</i>) a partir de estimadores robustos	24
4 Aplicações	26
4.1 Características dos dados e a estruturação de cenários	26
4.2 Aplicação 1: dados da transcrição do dígito 3 por pessoas de idades variadas.	28
4.3 Aplicação 2: dados da vértebra T2 de ratos	30

4.3.1	Avaliação da influência do parâmetro de tolerância - GPA-M .	32
5	Considerações finais	34
	Referências Bibliográficas	35
A	Funções auxiliares e estimadores robustos	38
A.1	Funções Auxiliares	38
A.2	SVD robustas	42
A.3	Estimadores robustos para full OPA	49
A.4	Estimadores robustos para full GPA	52

Lista de Figuras

2.1	Marcos anatômicos no crânio de um macaco, (Dryden e Mardia, 2016, p. 5).	6
2.2	Pseudo-marcos, em amarelo, distribuídos sobre a vértebra T2 de um rato, (Dryden e Mardia, 2016, p. 5).	6
2.3	Diagrama para obtenção da forma a partir de uma configuração inicial, (Dryden e Mardia, 2016, p. 68).	8
2.4	Sobreposição da forma do crânio de gorilas através da análise de procrustes.	12
4.1	Distribuição dos marcos anatômicos em cada banco de dados, a Figura 4.1(a) corresponde ao banco 1, já 4.1(b) corresponde ao banco 2.	27
4.2	Inserção de outlier no banco de dados 2.	27
4.3	Sobreposição das formas médias procustes obtidas pelos estimadores GPA-M (a), GPA-MM (b) e GPA-Kernel (c) a forma média original.	28
4.4	Marcos anatômicos do banco de dados 1 após aplicação dos estimadores GPA (a), GPA-MM (b), GPA-MM (c) e GPA-Kernel (d).	29
4.5	Marcos anatômicos do banco 2 após aplicação dos estimadores GPA (a), GPA-M (b), GPA-MM (c) e GPA-Kernel (d).	30
4.6	Inserção de outliers, configurações com quatro (a) oito (b) doze (c), dezesseis (d), vinte (e) e vinte e quatro (f) outliers.	31

Lista de Tabelas

1.1	Algumas aplicações da análise de forma.(Cândido, 2017, p. 2)	2
2.1	Exemplos de funções kernel	16
4.1	Medida de ajuste dos modelos ao banco de dados 1.	29
4.2	Evolução das medidas de ajuste ao banco de dados 2 em cenários com outliers.	32
4.3	Medidas de ajuste com aumento do parâmetro de tolerância.	32

Capítulo 1

Introdução

A Análise Estatística de Formas (AEF) é o estudo das variações de forma, tamanho e suas covariações. A primeira menção sobre o tema é devida a [Kendall \(1977\)](#) que em seu trabalho "*The diffusion of shape*" introduziu uma nova representação de formas de objetos em espaços complexos projetados.

Posteriormente, [Kendall \(1984\)](#) apresentou um sistema de coordenadas que permitiu obter a forma de um objeto via pontos dispostos em seu contorno e que contribuiu decisivamente para a definição de forma como vemos hoje. Já a formulação matemática para o estudo das formas foi introduzida em [Bookstein \(1984\)](#) e [Bookstein \(1986\)](#).

Atualmente a AEF tem se tornado uma poderosa ferramenta para tratamento de dados de imagem em diversos segmentos, são exemplos de aplicações importantes o diagnóstico por imagens na medicina, e o reconhecimento por meio de informações biométricas no setor de segurança ([Cândido, 2017](#)). A Tabela 1.1 apresenta algumas dessas aplicações.

Com a formalização dos conceitos de morfometria e o desenvolvimento de técnicas computacionais e de informática, surgiu, ao longo do tempo, uma multiplicidade de métodos para AEF como a análise da matriz de distância euclidiana ([Lele e Richtsmeier, 1991](#)), análise elíptica de Fourier ([Lestrel, 1982](#)), e abordagens não baseadas em rótulos, como a voxel-based ([Ashburner e Friston, 2001](#)), principalmente aplicada em imagens cerebrais e a análise procrustes ([Bookstein, 2001](#));([Dryden e Mardia, 2016](#)).

Dentre as técnicas acima a análise procrustes é a mais difundida, sua estruturação matemática e estatística já são bem conhecidas o que a torna uma excelente escolha para estimar, mensurar e comparar formas médias. Nesta dissertação propomos um método baseado em regressão robusta para análise de procrustes. Adicionalmente, apresentaremos um algoritmo para estimação procrustes e, por fim, apresentaremos um estudo de simulação e uma aplicação a dados reais que servirá como parâmetro de comparação entre a abordagem proposta, baseada em estimadores robustos, e a

Tabela 1.1: Algumas aplicações da análise de forma. (Cândido, 2017, p. 2)

Área do Conhecimento	Exemplo de Aplicações
Neurociência	Taxonomia morfologia de neurônios, investigações sobre a relação, função e forma das células, comparação entre células de diferentes áreas corticais e de diferentes espécies, modelagem de células biologicamente realistas.
Análise de Documentos	Análise de documentos eletrônicos (textuais ou visuais) como em aplicações CBIR, sistemas OCR (<i>Optical Character Recognitions</i>), análise de dados em banco de dados multimídia.
Artes Visuais	Restauração de vídeo, efeitos especiais, monitoramento de vídeo, jogos, computação gráfica, síntese de imagens.
Medicina	Identificação de tumor, quantificação de mudança e deformação de estruturas anatômicas, análise numérica de cromossomos, identificação de doenças genéticas.
Biologia	Identificação de espécies, taxonomia, relação entre forma e função, comparativo de anatomias, citologia, identificação e contagem de células (como glóbulos brancos no sangue), caracterização de células e formas nucleares, crescimento e modificação de formas de estruturas, análise da forma de caminhar dos seres vivos.
Física	Aplicações envolvendo microscopia como a análise da trajetória, comportamento e distribuição de partículas em um meio ou material, análise de estruturas como polímeros e cristais, caracterização de grupos de estrelas ou análise de propriedades de corpos celestes em astronomia, análise do movimento de objetos macroscópicos (velocidade, aceleração, etc.).
Engenharia	Controle visual de qualidade de produtos em linha de produção, detecção de situações de perigo, interpretação (pelas máquinas) de desenhos feitos à mão, automação, robótica, sensoramento remoto.
Segurança	Detecção de impressão digital, reconhecimento facial e através do escaneamento da íris, verificação de assinatura e modo de andar de uma pessoa (biometria).
Agricultura	Controle de colheita, contagem de sementes, análise da maturação de frutos.

abordagem baseada em mínimos quadrados.

1.1 Objetivos

Objetivo geral:

Proposição, desenvolvimento teórico e implementação computacional de métodos de análise procrustes baseados em estimadores robustos.

Objetivos específicos:

- Desenvolver métodos de estimação baseados em modelos de regressão robusta para obtenção da forma média procrustes;
- Implementar os métodos desenvolvidos em linguagem R;
- Avaliar o desempenho dos métodos desenvolvidos com os métodos existentes a partir de aplicações a dados reais.

1.2 Estrutura do trabalho

Além deste capítulo introdutório, este trabalho está organizado da seguinte forma:

Capítulo 2: Nesse capítulo introduzimos os conceitos essenciais para o entendimento do trabalho. Esses conceitos visam apresentar as principais ideias de análise estatística de formas tais como formas, marcos anatômicos, pré-formas, assim como a ideia geral de robustez para modelos de regressão e sua conexão com os estimadores procrustes para forma média.

Capítulo 3: Nesse capítulo descrevemos a ideia geral adotada para implementação das técnicas propostas, os algoritmos desenvolvidos e suas respectivas funções.

Capítulo 4: Nesse capítulo comparamos o desempenho dos métodos desenvolvidos com os métodos existentes a partir de aplicações a bancos de dados presentes na literatura.

Capítulo 5: Nesse capítulo apresentamos as considerações finais e algumas perspectivas que podem servir como ponto de partida para trabalhos futuros.

Capítulo 2

Revisão bibliográfica

Neste capítulo apresentamos os principais conceitos da análise estatística de formas e sua representação matemática. Tratamos dos métodos para análise de procrustes e dos modelos de regressão utilizados, associando-os ao conceito de robustez. Por fim, associamos os estimadores robustos e análise de procrustes.

2.1 Análise Estatística de Formas

A análise estatística de formas se propõe a comparar formas de objetos a partir de medidas de distância. O primeiro a explorar esta área foi Kendall (1977), embora apenas nos trabalhos de [Kendall \(1984\)](#) e posteriormente de [Bookstein \(1986\)](#) é que seus conceitos foram devidamente formalizados.

Um dos tópicos mais relevante proposto por [Kendall \(1984\)](#) foi a elaboração de um sistema de coordenadas para obter a forma de um objeto via pontos de referência chamados de marcos anatômicos. Os marcos anatômicos têm suas coordenadas dispostas em um espaço não-euclidiano conhecido com "espaço de forma" ([Mardia e Gill, 1995](#), p. 1) cujas propriedades geométricas exigem a utilização de ferramentas adequadas.

Atualmente a palavra "forma" é utilizada quando nos referimos à aparência de um objeto. Entretanto, a análise estatística de formas diz respeito a todos os aspectos da análise de objetos em que as informações de locação, orientação e possivelmente de escala podem ser ignoradas.

Segundo [Dryden e Mardia \(2016\)](#), há três conceitos essenciais para a análise de formas, a saber:

- **Forma (Shape):** é toda a informação geométrica que permanece quando os efeitos de locação, escala e rotação de um objeto são filtrados;

- **Marcos anatômicos ou Pontos de referência (Landmarks):** são pontos de correspondência de cada objeto entre e dentro das populações;
- **Tamanho e forma (Size-and-shape):** é toda a informação geométrica que resta quando a locação e os efeitos de rotação são removidos de um objeto;

A forma de um objeto é invariável sob as transformações euclidianas de translação e rotação. Em outras palavras, dois objetos terão a mesma forma se pudermos estabelecer uma correspondência exata entre ambos removendo os efeitos de locação, escala e rotação.

2.1.1 Marcos Anatômicos ou Pontos de Referência

Existem três tipos básicos de pontos de referência: os científicos, os matemáticos e os pseudo-marcos.

Um marco científico é um ponto de correspondência entre objetos estabelecido por meio de características morfométricas geralmente estabelecidas a partir da análise de especialistas, estão relacionados a observações provenientes do desenvolvimento biológico ou a traços morfológicos característicos. Geralmente determinados por um perito ou especialista, estes pontos são comumente chamados de marcos anatômicos e designam partes homólogas em seres de mesmo grupo ou espécie (Dryden e Mardia, 2016).

Os marcos matemáticos são pontos definidos a partir de alguma propriedade matemática ou geométrica, como um ponto de alta curvatura ou um ponto extremo. A utilização de pontos de referência desse tipo é particularmente útil no reconhecimento e análise automatizados.

A Figura 2.1 ilustra a distribuição de 26 marcos anatômicos estabelecidos a partir da vista lateral de um crânio de macaco. A disposição dos pontos segue um critério morfo-biológico baseado na análise da estrutura craniana de diversos indivíduos de mesma espécie. O marco anatômico 25, por exemplo, não está numa região de alta curvatura, sendo necessária uma formulação matemática para determinar sua posição com algum nível de precisão.

Já os pseudo-marcos são pontos construídos em torno do esboço entre pontos de referência científicos ou matemáticos. Na Figura 2.2 vemos seis pontos de referência em regiões de alta curvatura (+) e quarenta e dois pseudo-marcos (+) marcados sobre o contorno da vértebra T2 de um rato. Nesse caso, a inserção dos pseudo-marcos permite construir a representação plana da vértebra, definindo o perfil do contorno que tende a se repetir em cada um dos indivíduos da população. Já os marcos em vermelho são todos pontos de alta curvatura e constituem o cerne da informação geométrica que compõe a forma da vértebra.

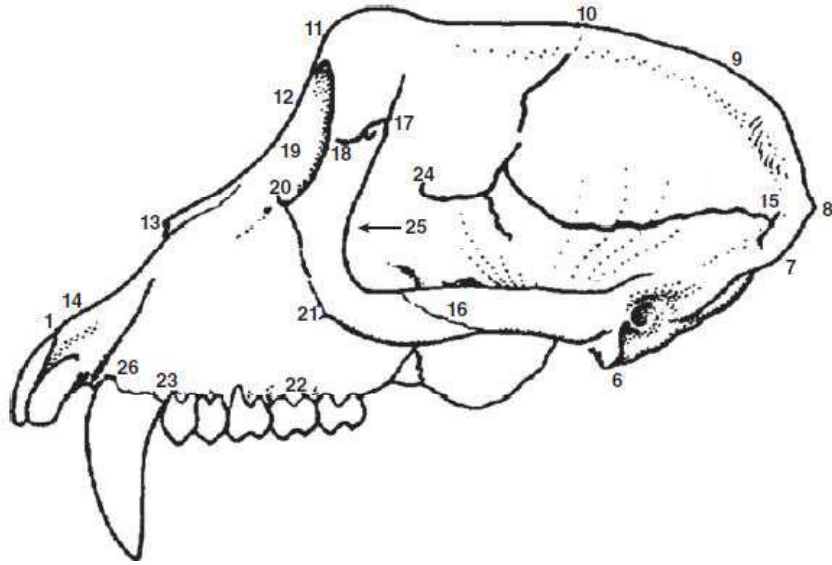


Figura 2.1: Marcos anatômicos no crânio de um macaco, (Dryden e Mardia, 2016, p. 5).

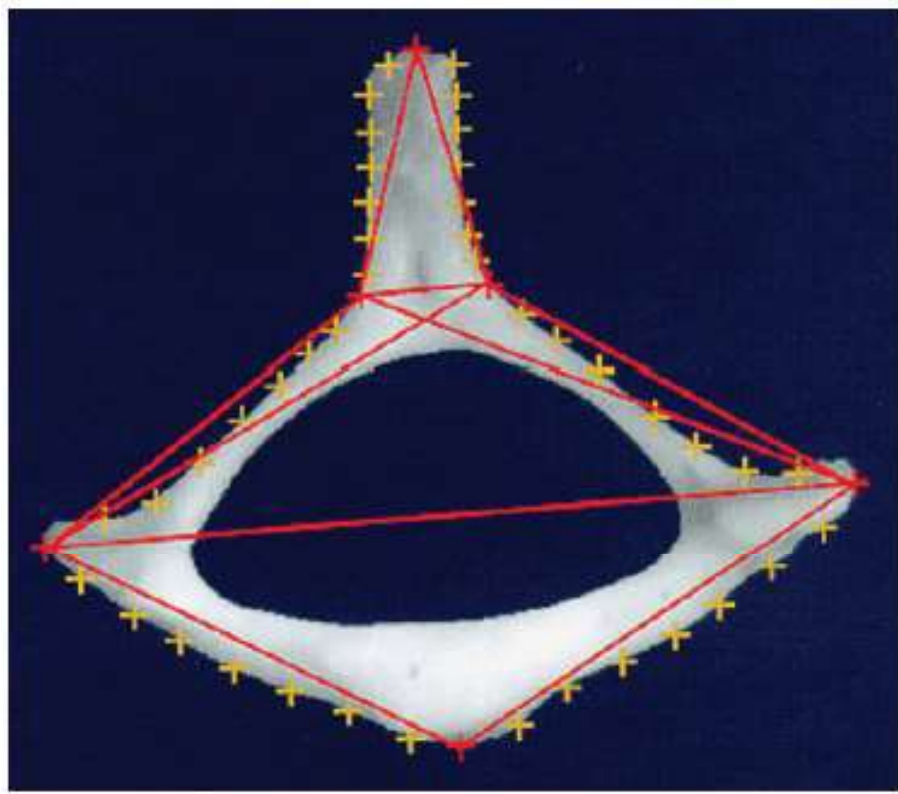


Figura 2.2: Pseudo-marcos, em amarelo, distribuídos sobre a vértebra T2 de um rato, (Dryden e Mardia, 2016, p. 5).

2.1.2 Representação Matemática de Formas

Considere uma matriz $X_{k \times m}$ de coordenadas cartesianas com k pontos de referência em m dimensões,

$$X = \begin{pmatrix} x_{1,1} & \dots & x_{1,m} \\ x_{2,1} & \dots & x_{2,m} \\ \vdots & \vdots & \vdots \\ x_{k,1} & \dots & x_{k,m} \end{pmatrix} \quad (2.1)$$

A matriz X é dita matriz de configuração de um conjunto de marcos em um objeto particular. O espaço de configurações para o objeto dado é formado por todas as coordenadas possíveis para os marcos em X .

Neste trabalho serão considerados os casos onde $k \geq 3$ e $m = 2$, o que corresponde as formas planas. Assim a matriz de configuração X na expressão 2.1 assumirá a forma,

$$X = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T = \begin{pmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ \vdots & \vdots \\ x_{k,1} & x_{k,2} \end{pmatrix} \quad (2.2)$$

Escrevendo a matriz X da expressão 2.2 como um vetor complexo de dimensão $k \times 1$ temos,

$$z = (z_{(1)}, \dots, z_{(k)})^T = (x_{1,1} + ix_{1,2}, \dots, x_{k,1} + ix_{k,2})^T \quad (2.3)$$

o qual corresponde as coordenadas complexas para os pontos de referência.

2.1.3 Matriz de Helmert e Pré-forma

Como dito anteriormente, a forma de um objeto é obtida quando os efeitos de locação, escala e de rotação são removidos. Evidentemente, a remoção de tais efeitos é feita por transformações geométricas com fases bem definidas que apresentam diferentes níveis de dificuldade.

A remoção da locação e escala são feitas por transformações lineares que promovem um redimensionamento para o tamanho da unidade, gerando, com isso, um objeto conhecido como "pré-forma". Já a remoção da rotação é mais difícil e pode ser feita através de um processo de otimização sobre as rotações. A Figura 2.3 apresenta o diagrama de etapas para obtenção da forma a partir de uma configuração inicial.

Para entender como o efeito de locação é filtrado é preciso definir, preliminar-

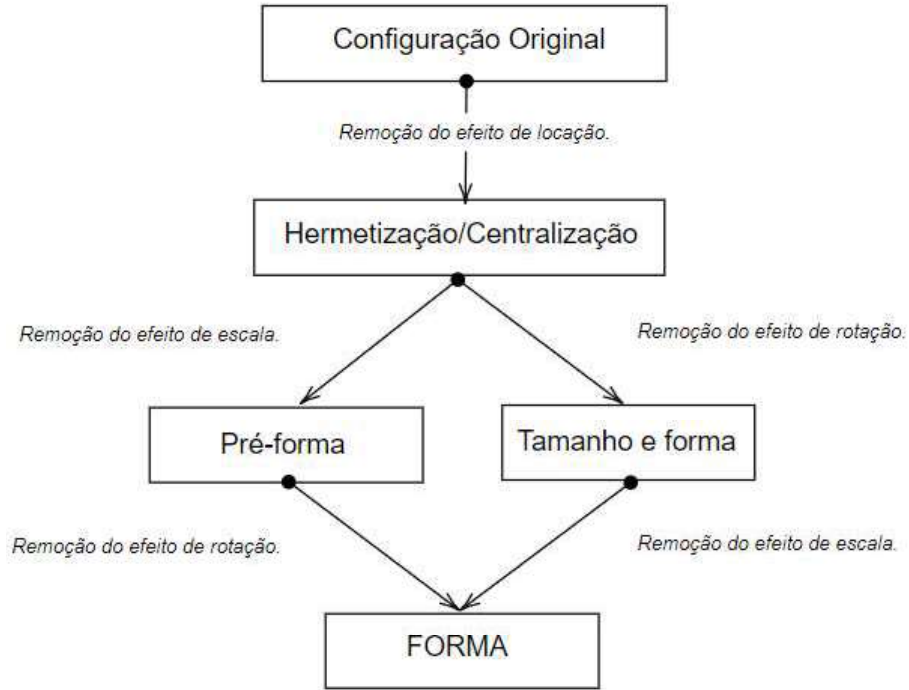


Figura 2.3: Diagrama para obtenção da forma a partir de uma configuração inicial, (Dryden e Mardia, 2016, p. 68).

mente, a submatriz de Helmert. A matriz H^F de ordem k é dita de Helmert quando os elementos da sua primeira linha tem forma $1/\sqrt{k}$ e os elementos das linhas restantes formam vetores ortogonais ao vetor formado pelos elementos da primeira linha.

A submatriz de Helmert, H , é obtida retirando-se a primeira linha de H^F . Ademais, a sua j -ésima linha é dada por $(h_j \dots h_j; -jh_j; 0 \dots 0)$ onde,

$$h_j = -\{j(j+1)\}^{1/2} \text{ com } j = 1, \dots, k-1.$$

A título de exemplo, para $k = 4$ a matriz de Helmert apresenta a forma,

$$H^F = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 & 0 \\ -1/\sqrt{6} & -1/\sqrt{6} & 2/6 & 0 \\ -1/\sqrt{12} & -1/\sqrt{12} & -1/\sqrt{12} & 3/\sqrt{12} \end{bmatrix}$$

Já a submatriz de Helmert apresenta a forma,

$$H = \begin{bmatrix} -1/\sqrt{2} & 1/\sqrt{2} & 0 & 0 \\ -1/\sqrt{6} & -1/\sqrt{6} & 2/6 & 0 \\ -1/\sqrt{12} & -1/\sqrt{12} & -1/\sqrt{12} & 3/\sqrt{12} \end{bmatrix}$$

Podemos retirar o efeito de locação multiplicando a equação 2.3 pela submatriz de Helmert. Assim,

$$w_{(k-1 \times 1)} = H_{(k-1 \times k)} z_{(k \times 1)}, \quad (2.4)$$

onde w representa a configuração z sem o efeito de locação denominada de configuração helmertizada (Gomes, 2020, p. 22).

A configuração centrada é obtida pré-multiplicando o vetor w pela transposta da submatriz de Helmert H^T . Assim,

$$H^T w = H^T H z = C * z.$$

Uma vez que a configuração inicial passa pelo processo de helmertização, o efeito de escala é removido dividindo a configuração w obtida na equação (2.4) pela sua norma $|w|$ obtendo, assim, a pré-forma da configuração original a partir da expressão

$$z_{(k-1 \times 1)} = \frac{w}{|w|} = \frac{w}{\sqrt{w * w}},$$

em que $w*$ é o transposto conjugado de w e $|\cdot|$ é a norma complexa de w .

A pré-forma da matriz X de dimensão m é dada por,

$$Z_{(k-1 \times m)} = \frac{H_{(k-1 \times k)} X_{(k \times m)}}{|H X|}, \quad (2.5)$$

a qual é invariante sob locação e escala em relação à configuração inicial.

Multiplicando o numerador e o denominador da equação (2.5) por H^T podemos obter as pré-formas centralizadas $Z_{c(k \times m)} = C_{(k \times k)} X_{(k \times m)} / |C X|$. Nesse caso, vale lembrar que z_c e z são matrizes diferentes sendo a primeira de dimensão $k \times m$ e a segunda de dimensão $(k-1) \times m$.

Gomes (2020) destaca que os espaços das pré-formas são todos os espaços possíveis das pré-formas z compostos pelos $(k-1)$ vetores sem os efeitos de locação e escala, bem como as vantagens em usar z ou z_c , que consistem em obter um vetor de posto completo com uma dimensão menor que k e uma representação das coordenadas cartesianas coerentes com a configuração original.

2.1.4 Forma Média

Estabelecido o conceito de pré-forma, podemos mencionar outro conceito central a morfometria que é o de forma média.

Considere uma amostra aleatória de configuração complexa com n objetos $(z_{(1)}^0, \dots, z_{(n)}^0)$. A forma média do grupo $\hat{\mu}$ pode ser obtida através do autovetor dominante da soma quadrática complexa da matriz produto

$$S = \sum_{i=1}^n z_i z_i^*,$$

em que $z_i = w_i / \|w_i\|$, $i = 1, \dots, n$ são as pré-formas, então $\hat{\mu}$ é o autovetor correspondente ao maior autovalor de S .

O efeito de rotação pode ser filtrado a partir de uma transformação que consiste na multiplicação da matriz $X_{(k \times m)}$ por uma matriz de rotação $\Gamma_{(m \times m)}$. Quando a matriz Γ satisfaz as condições $\Gamma^T \Gamma = \Gamma \Gamma^T = I_m$ e $\det(\Gamma) = 1$ ela é dita uma matriz ortogonal especial. O conjunto formado pelas matrizes ortogonais especiais é conhecido como grupo especial, geralmente denotado por $SO(m)$.

Para $m = 2$ dimensões, a matriz de rotação pode ser parametrizada por um único ângulo θ , $-\pi < \theta < \pi$ em radianos assumindo a forma

$$\Gamma = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix},$$

que permite a rotação no sentido horário sobre a origem.

Dryden e Mardia (2016) observam que as transformações euclidianas de similaridade de uma matriz de configurações X consistem no conjunto das matrizes X transladadas, rotacionadas e isotropicamente redimensionadas, ou seja,

$$\{\beta X \Gamma + 1_k \gamma : \Gamma \in SO(m), \gamma \in \mathbb{R}^m\}, \quad (2.6)$$

onde $\beta \in \mathbb{R}^+$ é a escala, Γ é uma matriz de rotação e γ é um m -vector de translação. Considerando $m = 2$ podemos reescrever a equação 2.6 em notação complexa. Sejam $k \geq 3$ pontos de referência em $\mathbb{C}$ e $z^0 = (z_{(1)}^0, \dots, z_{(k)}^0)^T$, a transformação euclidiana de similaridade de z^0 é dada por,

$$\{\eta z^0 + 1_k \xi : \eta = \beta e^{i\theta} \in \mathbb{C}, \xi \in \mathbb{C}\}, \quad (2.7)$$

onde $\beta \in \mathbb{R}^+$ a escala, $0 \leq \theta < 2\pi$ é o ângulo de rotação que gera uma rotação no sentido anti-horário em relação à origem e $\xi \in \mathbb{C}$ é o vetor de translação. A equação 2.7 é útil já que especifica as transformações euclidianas de similaridade

descritas por 2.6 como transformações lineares complexas, o que permite grandes simplificações na análise da forma em duas dimensões.

2.2 A Análise de Procrustes

Os métodos baseados em análise de procustes são úteis para estimar uma forma média e para explorar a estrutura da variabilidade da forma em um conjunto de dados. A técnica consiste em construir a correspondência de configurações com transformações de similaridade utilizando a distância euclidiana.

Curiosamente, o termo procustes tem origem na mitologia grega e designa um homem de estatura e força extraordinárias que vivia nas colinas de Ática, onde oferecia sua pousada a viajantes solitários. Segundo a tradição grega, Procusto convidava viajantes para dormirem em uma cama de ferro. Quando o viajante era menor que o comprimento da cama, ele o esticava até que coubesse na medida exata da cama. Nos casos em que o viajante era maior que o comprimento da cama, ele amputava seus membros para caber nas medidas exatas da cama. Em suma, a análise procustes procura, por meio de transformações de escala, translação e rotação, forçar as medidas para que elas se encaixem, promovendo, assim, um processo de sobreposição bem ajustado.

A Figura 2.4 ilustra a sobreposição das formas do crânio de um gorila adulto (em preto) e um gorila jovem (em vermelho) usando a análise procustes. Uma vez aplicada a técnica, percebemos que as formas se sobrepõem quase que perfeitamente. É possível observar ainda que a forma da estrutura craniana do gorila jovem é ligeiramente maior quando comparada com a do adulto, sendo compatível com fatores do desenvolvimento biológico de mamíferos em geral, como a compactação ou perda de massa óssea.

Dryden e Mardia (2016) apontam duas variações para análise estatística de procustes, a Análise ordinária de procustes (OPA) e a Análise generalizada de procustes (GPA). A OPA é utilizada para estabelecer a correspondência entre duas configurações. Considere o caso em que temos duas matrizes de configuração X_1 e X_2 compostas por k pontos de referências dispostos em m dimensões.

Definição 1: O método de OPA envolve a correspondência dos quadrados de duas configurações centradas utilizando as transformações de similaridade. A estimativa da similaridade dos parâmetros γ , Γ e β é realizada minimizando o quadrado da distância Euclidiana,

$$D_{OPA}^2(X_1, X_2) = \left\| X_2 - \beta X_1 \Gamma - 1_k \gamma^T \right\|^2 \quad (2.8)$$

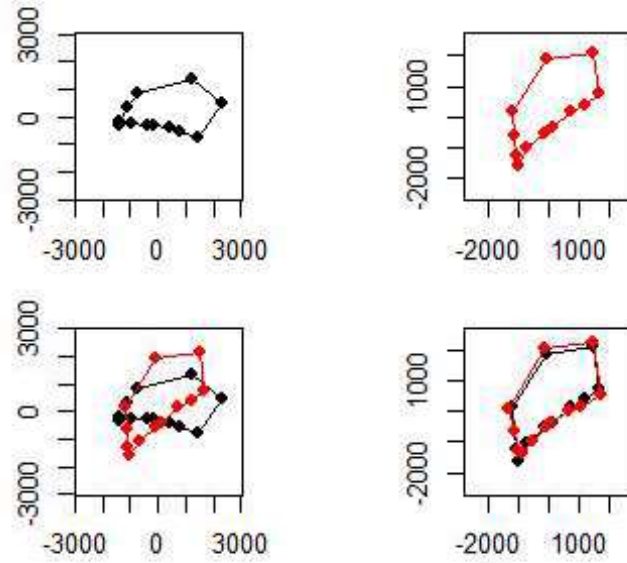


Figura 2.4: Sobreposição da forma do crânio de gorilas através da análise de procustes.

onde $\|X\| = \sqrt{\text{Tr}(X^T X)}$ é a norma euclidiana, $\Gamma_{(m \times m)}$ é uma matriz de rotação, $\beta > 0$ é um parâmetro de escala e $\gamma_{(m \times 1)}$ é um vetor de locação.

A técnica procustes baseada na equação 2.8 é chamada de Análise ordinária de procustes completa (Full OPA). A solução $(\hat{\gamma}, \hat{\beta}, \hat{\Gamma})$ apresenta valor mínimo quando $\hat{\gamma} = 0$, $\hat{\Gamma} = UV^T$ e $X_2^T X_1 = \|X_2\| \|X_1\| V \Lambda U^T$. Quando um bom número de matrizes de configurações estão disponíveis, podemos utilizar mínimos quadrados para obter uma forma média através da generalização da OPA.

Consideremos agora o caso geral em que as $n \geq 2$ matrizes de configuração $X_1, \dots, X_n$ estão disponíveis.

Definição 2: O método de Análise generalizada de procustes (GPA) consiste em filtrar os efeitos de locação, escala e rotação entre as configurações de modo a minimizar uma soma geral de quadrados das configurações centradas e rotacionadas, assim,

$$G(X_1, X_2, \dots, X_n) = \sum_{i=1}^n \left\| (\beta_i X_i \Gamma_i + 1_k \gamma_i^T) - \mu \right\|^2.$$

Quando uma amostra apresenta alguma homogeneidade na variação da forma, as medições estão frequentemente relacionadas. Por exemplo, as distâncias biológicas apresentam um elevado grau de intercorrelação, visto que as diferentes partes do organismo, muitas vezes, crescem harmoniosamente durante o desenvolvimento dos

indivíduos de uma população.

Uma vez verificada, a intercorrelação forte entre formas é possível associar a análise de procustes aos modelos de regressão. Na verdade, no processo de análise de procustes há necessariamente a utilização de técnicas de regressão.

2.3 Modelos de regressão

2.3.1 O Método dos Mínimos Quadrados

O método dos Mínimos Quadrados Ordinários (MQO) pode ser empregado para estimar os parâmetros de um modelo de regressão linear. Esta abordagem foi pioneiramente formalizada por Gauss (1809) e posteriormente promovida por Galton (1877). Uma das vantagens notáveis desse método é que ele não requer a suposição de uma distribuição de probabilidade para os erros do modelo. Assim, considere o modelo de regressão linear múltipla definido por

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} + e_i. \quad (2.9)$$

onde y_i representa a variável resposta, x_{ij} representa um conjunto de p variáveis independentes que explicam y_i , β_j representam $p + 1$ parâmetros a serem estimados e e_i representa o erro aleatório, com $E(e_i) = 0$, $Var(e_i) = \sigma^2$ e $Cov(e_i, e_j) = 0$, $\forall i \neq j$.

Em notação matricial, a expressão 2.9 assume a forma

$$y = X\beta + e, \quad (2.10)$$

onde y representa o vetor formado pelas variáveis resposta y_i , X é a matriz de variáveis independentes que explicam y , β representa o vetor dos $p + 1$ parâmetros a serem estimados e e é o vetor de erros aleatórios.

Observando 2.9 é possível perceber que quando erro aleatório e_i é pequeno, teremos uma boa estimativa para y_i . Mais ainda, se isolarmos e_i em 2.9 e considerarmos a soma dos erros de todas as observações obtemos a expressão,

$$\sum_{i=1}^n e_i = \sum_{i=1}^n (y_i - x_i^\top \beta).$$

Dada as características intrínsecas do modelo de regressão linear, os erros alea-

tórios e_i podem assumir valores positivos ou negativos, fazendo com que $\sum_{i=1}^n e_i = 0$. Para evitar esse fato, considera-se a soma do quadrado dos erros aleatórios, o que nos permite escrever

$$\sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 \quad (2.11)$$

O modelo de regressão clássica baseado em MQO consiste em minimizar 2.11 (Silva e Santos), o que resulta na função objetivo

$$\min \left\{ \sum_{i=1}^n e_i^2 \right\} = \min \left\{ \sum_{i=1}^n (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 \right\}$$

, cuja solução é dada por $\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T y$.

Fica implícito na construção acima que os erros aleatórios e_i tem mesmo peso na estimação dos coeficientes $\boldsymbol{\beta}$, isso faz com que o estimador $\hat{\boldsymbol{\beta}}$ seja sensível a outliers comprometendo assim a capacidade de predição do modelo.

2.3.2 Regressão robusta

Uma forma de reduzir a influência de *outliers* é definir uma função ρ que quando aplicada em e_i possa atribuir pesos diferenciados as observações que apresentam uma maior dispersão em torno da reta de regressão (Huber e Ronchetti, 2018).

De modo geral, os modelos robustos têm função objetivo que segue a forma

$$\min \left\{ \sum_{i=1}^n \rho(e_i) \right\} = \min \left\{ \sum_{i=1}^n \rho(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) \right\}.$$

Do ponto de vista matricial, o conceito de robustez consiste em incrementar $\hat{\boldsymbol{\beta}}$ por uma matriz de ponderação K , o que permite passar do modelo clássico de estimação para um modelo mais robusto baseado em estimadores da forma $\hat{\boldsymbol{\beta}} = (X^T K X)^{-1} X^T K y$.

Ao impor algumas condições a função ρ podemos atribuir pesos menores aos *outliers* gerando assim estimadores robustos. Apresentaremos a seguir os estimadores robustos que utilizaremos neste trabalho.

Estimador M-huber

Nos casos em que ρ passa pela origem, é monótona, não decrescente, é par e é não negativa, ela é dita um M-estimador. Um estimador para ρ bastante conhecido foi proposto por [Huber \(1973\)](#) e é dado por

$$\rho(e_i) = \begin{cases} e_i^2, & -k \leq e_i \leq k; \\ 2k \|e_i\| - k^2, & e_i < -k \text{ ou } e_i > k, \end{cases} \quad (2.12)$$

onde $k = 1.5 \times \hat{\sigma}$ e $\hat{\sigma} = 1.483 \times \text{med}(|r_{i1}|, \dots, |r_{im}|)$.

O M-estimador de Huber não possui forma analítica, sendo necessário o uso de um algoritmo para obter numericamente as estimativas referentes aos parâmetros do modelo.

Estimador baseado em LMS

Mesmo apresentando alguma robustez de início, o M-estimador obtido a partir de 2.12, tem desempenho afetado a medida que o número de *outliers* aumenta. Uma opção mais robusta pode ser obtida substituindo a soma do quadrado dos erros pela mediana do quadrado dos resíduos, obtendo assim a seguinte função objetivo:

$$\min \{ \text{med}(r_i^2) \} = \min \{ \text{med}(y_i - \beta_1 x_{i1} - \dots - \beta_{p-1} x_{ip-1} - \beta_p)^2 \}. \quad (2.13)$$

A expressão acima representa minimizar a mediana dos quadrados dos resíduos. O estimador obtido pela solução de 2.13 é obtido por um processo iterativo que consiste em criar subgrupos de observações e estimar o resíduo a partir de MQO, em seguida verifica-se qual dos subgrupos possui resíduos com menor mediana e calcula-se o estimador $\hat{\beta}_{LMS}$ ([Haykin, 2014](#)).

Apesar de o estimador $\hat{\beta}_{LMS}$ não ser único, a resistência da mediana a outliers o torna estável, ou seja, a variação entre grupos é pequena a cada iteração.

Estimadores baseados em funções kernel

Seja $X = \{x_1, x_2, \dots, x_n\}$ um conjunto não vazio onde $x_i \in \mathbb{R}^p$. Uma função $\kappa : X \rightarrow \mathbb{R}$ é chamada de definida positiva ou kernel de Mercer ([Friedman e Tibshirani, 2009](#)) se, e somente se, for simétrica, ou seja, $\kappa(x_i, x_k) = \kappa(x_k, x_i)$. Nesse caso vale a seguinte desigualdade:

$$\sum_{i=1}^n \sum_{k=1}^n c_i c_k \kappa(x_i, x_k) \geq 0, \quad \forall n \geq 2,$$

onde $c_r \in \mathbb{R}$, $\forall r = 1, 2, \dots, n$.

Seja $\Phi : X \rightarrow \mathcal{F}$ uma função não linear do espaço de entrada X para um espaço de características de alta dimensão $\mathcal{F}$. Aplicando a função Φ , ao produto interno $x_i^\top x_k$ do espaço de entrada, obtemos o produto interno $\Phi(x_i)^\top \Phi(x_k)$ no espaço de alta dimensão $\mathcal{F}$. A ideia chave aqui é que a função Φ não precisa ser especificada tendo em vista que cada operador kernel pode assumir a forma $\kappa(x_i, x_k) = \Phi(x_i)^\top \Phi(x_k)$.

Um dos aspectos mais relevantes dessas aplicações é que é possível calcular distâncias euclidianas em $\mathcal{F}$ sem que precisemos saber explicitamente a forma de Φ , bastando para isso usarmos o artifício da distância kernel.

A Tabela 2.1 apresenta três variações para a função κ .

Tabela 2.1: Exemplos de funções kernel

Kernel	Expressão
Gaussiano	$\kappa_G(x_i, x_k) = \exp\left(-\frac{\ x_i - x_k\ ^2}{2\gamma^2}\right)$
Exponencial	$\kappa_E(x_i, x_k) = \exp\left(-\frac{\ x_i - x_k\ }{\gamma}\right)$
Laplaciano	$\kappa_L(x_i, x_k) = \exp\left(-\frac{\ x_i - x_k\ }{\gamma}\right)$

O hiper-parâmetro de largura γ^2 assume um papel importante no desempenho dos kernels apresentados acima e precisa ser cuidadosamente ajustado ao problema em questão. Por exemplo, valores superestimados de γ^2 levam a uma projeção de linearidade de maior dimensão no comportamento da função exponencial. Por outro lado, para valores subestimados de γ^2 , a função não terá regularização e o limite de decisão será altamente sensível ao ruído nos dados.

Além disso, observe que a menos do quadrado da norma a exponencial no kernel está intimamente relacionada com o kernel gaussiano. Já o kernel laplaciano é completamente equivalente ao kernel exponencial, exceto por ser menos sensível a mudanças no hiper-parâmetro γ^2 .

O método de regressão robusta baseado em funções tipo exponencial (ETKRR) proposto por [Carvalho, Neto, e Ferreira \(2017\)](#) é inspirado no framework kernel. As estimativas dos parâmetros do modelo são obtidas através da minimização de uma função objetivo definida como a soma do quadrado dos erros calculados num espaço de alta dimensão $\mathcal{F}$ através da aplicação de uma função não linear Φ a variável de resposta y_i e sua média correspondente μ_i . A função é definida da seguinte maneira:

$$S = \sum_{i=1}^n \left\| \Phi(y_i) - \Phi(\mu_i) \right\|^2$$

onde $\mu = \beta + \beta x_{i1} + \dots + \beta x_{ip}$ representa a média das variáveis resposta y_i , que é estruturada a partir do conjunto de variáveis $x_{i1}, \dots, x_{ip}$ e de seus respectivos parâmetros $\beta_1, \beta_2, \dots, \beta_p$ a serem estimados.

Como a função não linear Φ não apresenta uma forma explícita, usa-se o artifício da distância kernel para estimar os parâmetros do modelo de regressão. Desse modo, a função objetivo passa a ter a forma:

$$S = \sum_{i=1}^n \{\kappa(y_i, y_i) - 2\kappa(y_i, \mu_i) + \kappa(\mu_i, \mu_i)\}$$

Se a função κ na equação acima é o kernel gaussiano (ver Tabela 2.1), Então $\kappa(y_i, y_i) = \kappa(\hat{\mu}_i, \hat{\mu}_i) = 1$ para $i = 1, 2, \dots, n$ o que nos permite escrever:

$$S = \sum_{i=1}^n 2 \left[1 - \kappa_G(y_i, \mu_i) \right], \quad \gamma > 0. \quad (2.14)$$

Avaliando a equação acima percebemos que se a distância entre y_i e μ_i ($i = 1, \dots, n$) for pequena a função objetivo tende a zero no modelo.

Diferenciando a equação 2.14 em relação ao vetor de parâmetros $\beta = (\beta_0, \dots, \beta_p)$, obtêm-se as equações normais para o modelo ETKRR, dadas por,

$$\frac{\partial S}{\partial \beta_r} = 2 \sum_{i=1}^n \kappa_G(y_i, \hat{\mu}_i) \frac{x_{ir}(y_i - \hat{\mu}_i)}{\gamma^2}, \quad r = 1, 2, \dots, p. \quad (2.15)$$

cuja solução é obtida por um processo iterativo baseado em mínimos quadrados ponderados conforme o modelo,

$$y^* = X^* \beta^* + e^*, \quad (2.16)$$

onde $y^* = K^{1/2}y$ é o vetor ponderado $n \times 1$ da variável de resposta, $X^* = K^{1/2}X$ é uma matriz ponderada de variáveis explicativas, β^* é o vetor dos parâmetros de regressão, $e^* = K^{1/2}e$ é o vetor ponderado $n \times 1$ de erros independentes e $K = (k_{11}, k_{22}, \dots, k_{nn})$ é a matriz kernel de pesos cuja diagonal k_{ii} é definida como $k_{ii} = K_G(y_i, \mu_i), \forall i = 1, 2, \dots, n$.

Rodrigues, Lourenço, e Mahmoud (2018) mencionaram que para $0 \leq \kappa_G(y_i, \mu_i) \leq 1$ as observações com maiores valores de resíduos recebem peso menor na estimação dos parâmetros, garantindo assim uma maior precisão na estimação dos parâmetros.

2.4 Estimadores robustos para análise de procustes

A análise de procustes utiliza transformações de similaridade para aproximar os pontos de referência minimizando a soma dos quadrados das distâncias. Ocorre que a abordagem clássica não é resistente a outliers havendo assim a necessidade de desenvolver métodos de ajuste resistentes.

Alguns estimadores robustos têm sido utilizados na adequação da OPA na análise da forma. Em particular, [Hyuna, Leeb, e Choil \(2018\)](#) apresentam adaptações da OPA baseadas no M-estimador e em LMS. Para entender a abordagem proposta, consideremos as matrizes X_1 e X_2 de pontos de referência com k configurações que podem ser sobrepostas a partir da minimização da seguinte função objetivo,

$$O(\alpha, \Gamma, \beta) = \sum_{i=1}^k \rho \left(\|r_i\| \right), \quad (2.17)$$

onde o resíduo $r_i = x_{(2)i} - \beta \Gamma^T x_{(2)i} - \alpha$, $x_{(1)i}$ e $x_{(2)i}$ são vetores de pontos de referência de X_1 e X_2 respectivamente, α é um vetor de locação, Γ é uma matriz de rotação, β é um parâmetro de escala e $\rho(\cdot)$ representa uma função de ponderação.

Sendo $\rho(\cdot)$ o M-estimador de Huber a equação 2.17 apresenta as seguintes soluções,

$$\hat{\alpha} = \frac{\left(X_2^T - \hat{\beta} \hat{\Gamma}^T X_1^T \right) W 1_k}{1_1^T W 1_k} \quad (2.18)$$

$$\hat{\Gamma} = U V^T \quad (2.19)$$

$$\hat{\beta} = \frac{\text{tr} \left[\hat{\Gamma}^T X_1^T W \left(I_k \frac{1_k 1_k^T W}{1_1^T W 1_k} \right) X_2 \right]}{\text{tr} \left[X_1^T W \left(I_k \frac{1_k 1_k^T W}{1_1^T W 1_k} \right) X_1 \right]} \quad (2.20)$$

onde $W = \text{diag}(w_1, \dots, w_k)$ é uma matriz diagonal com pesos $w_i = \psi(\|r_i\|)/\|r_i\|$, $i = 1, \dots, k$, $\psi(\cdot) = \rho(\cdot)'$ é a derivada de $\rho(\cdot)$, e $U, V \in SO(m)$ satisfazendo a decomposição em valores singulares

$$X_1^T W \left(I_k \frac{1_k 1_k^T W}{1_1^T W 1_k} \right) X_2 = U D_\lambda V^T$$

sendo D uma matriz diagonal de valores singulares.

A função objetivo $O(\alpha, \Gamma, \beta)$ pode ser minimizada por um processo iterativo

usando mínimos quadrados ponderados. O algoritmo apresenta duas etapas. Primeiro calculam-se os pesos através de $w_i = \psi(|r_i|)/|r_i|$ com $\rho(\cdot)' = \psi(\cdot)$ usando a matriz de peso $W = 1_k$ como valores iniciais. Em seguida, os parâmetros α , Γ e β são calculados por mínimos quadrados ponderados conforme as equações 2.18, 2.19 e 2.20, respectivamente. Paramos a iteração se a condição $|O(n+1) - O(n)| < \epsilon$ é satisfeita, onde $O(n)$ é o valor da função objetivo $O(\alpha, \Gamma, \beta)$ na n-ésima interação.

Ao contrário do estimador LS, os estimadores robustos limitam a influência de outliers e o ajuste resultante é mais próximo da forma média verdadeira. Rohlf e Slice (1990) consideraram o ajuste GPA usando a técnica RM e implementaram um estimador robusto para ajuste de forma de modo a comparar o LS GPA com o estimador proposto a partir de dados da asa de mosquitos.

Er (1998) sugeriu um método GPA robusto através do algoritmo EM e comparou o estimador com LS GPA visualizando as estimativas da forma média. Já Hyuna, Leeb, e Choil (2018) propõem a comparação entre dois algoritmos de ajuste GPA resistentes baseados no estimador M e LMS denominados de M GPA e LMS GPA.

Partindo da abordagem de Hyuna, Leeb, e Choil (2018), propomos um método robusto baseado em kernel para o cálculo da forma media na GPA.

Capítulo 3

Métodos propostos

Neste capítulo apresentaremos uma abordagem robusta para análise de procustes baseada em decomposição de valores singulares, os algoritmos que compõem a análise generalizada de Procustes na sua forma completa (Full GPA) e suas respectivas funções.

3.1 Decomposição em Valores Singulares

A decomposição em valores singulares ou *Singular Value Decomposition* (SVD) é uma poderosa ferramenta de álgebra linear com aplicações importantes em estatística. Formalmente, a técnica consiste em fatorar uma matriz $X_{n \times p}$ real ou complexa para obter a forma:

$$X = U \Sigma V^T,$$

onde $U_{n \times p}$ é uma matriz unitária real ou complexa, $\Sigma_{p \times p}$ é uma matriz cuja diagonal é composta por números reais não-negativos, e V^T é uma matriz unitária $p \times p$ real ou complexa.

Na SVD as entradas Σ_i da diagonal de Σ são os chamados valores singulares de X . Já as p colunas de U e as p colunas de V^T são os chamados vetores singulares à esquerda e vetores singulares à direita de X , respectivamente. Existe uma relação estreita entre a decomposição em valores singulares e a decomposição em autovalores de uma matriz X . Mais especificamente podemos dizer que:

- Os vetores singulares à esquerda de X são autovetores de XX^T .
- Os vetores singulares à direita de X são autovetores de $X^T X$.
- Os valores singulares não-nulos de X (ao longo da diagonal de Σ) são as raízes quadradas dos autovalores não-nulos de XX^T ou $X^T X$.

Escrevendo $\mathbf{u}_i$ e $\mathbf{v}_i$ para os i -ésimos autovetores à esquerda e à direita, respectivamente, e denotando por σ_i o i -ésimo autovalor, temos que

$$X = \sum_{i=1}^p \sigma_i \mathbf{u}_i \mathbf{v}_i^\top. \quad (3.1)$$

A tripla $(\sigma_i, \mathbf{u}_i, \mathbf{v}_i)$ pode, alternativamente, ser aproximada por um procedimento de mínimos quadrados.

Dentre as aplicações que envolvem a SVD está a resolução de sistemas lineares do tipo $Ax = b$. Como é sabido, sistemas desse tipo nem sempre apresentam solução exata, fazendo-se necessário encontrar um vetor solução por meio de um processo de minimização.

3.2 SVD e a análise procustes

O protocolo utilizado para uma sobreposição consiste em minimizar a soma do quadrado das distâncias entre pontos de referência similares, chamada de distância de procustes. A sobreposição de duas matrizes de configuração, X_1 e X_2 , envolve estimar três parâmetros α , Γ e β para minimizar a quantidade $d_F(X_1, X_2)$ na equação 2.6, assim

$$d_F(X_1, X_2) = \min \left\| X_2 - \beta X_1 \Gamma - \mathbf{1}_m \gamma^T \right\|,$$

onde $\beta > 0$ é um parâmetro de escala, $\Gamma_{(m \times m)}$ é uma matriz de rotação e $\gamma_{m \times 1}$ é um vetor de locação.

Pode-se filtrar o efeito de locação removendo as coordenadas do centroide X_{1c} e X_{2c} para configurações de X_1 e X_2 . As coordenadas do centroide definem a translação do parâmetro γ . Já para realizar a tradução, deve-se retirar o valor dessas coordenadas da configuração original, isso produz o centro da configuração.

A remoção da escala é feita dividindo as coordenadas pelo tamanho do objeto. Escalando a configuração centralizada e a configuração Helmertizada, respectivamente, obtemos a pré-forma Z e a pré-forma centrada Z_c .

Quanto aos efeitos de rotação, podemos avaliar a matriz de rotação Γ diretamente das configurações por operações matriciais. Para conseguir isso, devemos lembrar que a distância total de Procustes entre X_1 e X_2 é uma medida da diferença de forma entre essas configurações que deve satisfazer a seguinte equação:

$$d_F = \min \left\| Z_{2c} - \beta Z_{1c} \Gamma \right\|,$$

onde Z_{2c} e Z_{1c} são as configurações centralizadas de X_1 e X_2 dimensionadas para o

tamanho do centroide.

Embora as configurações tenham sido dimensionadas para tamanhos idênticos, pode-se ajustar o tamanho de uma das configurações para otimizar a minimização de distâncias entre os pontos de referência usando SVD. Assim temos,

$$\Gamma = UV^T,$$

onde U e V são matrizes de vetores singulares provenientes da decomposição em valores singulares do produto da transposição de Z_2 por Z_1 .

$$Z_2^T Z_1 = V \Delta U$$

O traço de Δ contém informações para calcular o parâmetro de escala β para ajustar a configuração Z_1 em Z_2 . Podemos obter distância procustes total usando a relação,

$$d_F = \sqrt{\left(1 - \left(\sum_{i=1}^k \lambda_i\right)^2\right)}$$

O parâmetro β que possibilita o melhor ajuste é dado pela equação 2.6 e tem a forma

$$\beta = \frac{\text{Traço}(Z_2^T Z_1 \Gamma)}{\text{Traço}(Z_2^T Z_1)}.$$

Por fim, a distância total de procustes entre X_1 e X_2 é dada por

$$d_F = \sqrt{\text{Traço}[(Z_2 - \beta Z_1 \Gamma)^T (Z_2 - \beta Z_1 \Gamma)]}.$$

No método de procustes descrito acima, a SVD de $Z_2^T Z_1$ é calculada a partir de mínimos quadrados ordinários, resultando em estimativas não robustas para U e V .

3.2.1 Estimadores robustos para análise procrustes

Estimativas robustas para U e V podem ser obtidas substituindo o OLS por outros estimadores robustos (Neto, 2023), obtendo-se, assim, formas para SVD robustas. A fim de ilustrar tal ideia, considere o seguinte algoritmo,

Algoritmo 1: Decomposição em Valores Singulares

Passo 1: Comece com uma estimativa inicial para o autovetor a esquerda U_1 pondo $U_1^{(0)}$ e $k = 0$;

Passo 2: Para cada coluna $j, j = 1, 2, \dots, p$, ajuste o coeficiente c_j da regressão de y_{ij} em $U_{i1}^{(k)}$;

Passo 3: Calcule a estimativa resultante do autovetor direito $V_1^{(k)} = \mathbf{c}/\|\mathbf{c}\|$, onde $\|\cdot\|$ é a norma euclidiana;

Passo 4: Usando esta estimativa do autovetor direito, redefina a estimativa do autovetor esquerdo. Para cada linha $i, i = 1, 2, \dots, n$, ajuste o coeficiente d_i da regressão de y_{ij} em $V_{j1}^{(k)}$;

Passo 5: Calcule a estimativa resultante do autovetor esquerdo $U_1^{(k+1)} = \mathbf{d}/\|\mathbf{d}\|$. Faça $k = k + 1$;

Passo 6: Iterar os passos de 2 a 5 até obter a convergência. Sejam U_1^* e V_1^* as estimativas finais obtidos para os autovetores à esquerda e à direita, respectivamente;

Passo 7: Encontre o autovalor λ_1 como o coeficiente da regressão de y_{ij} em $U_{i1}^* V_{j1}^*$;

Passo 8: Substitua Y por $Y - \lambda_1^* U_{i1}^* V_{j1}^{*T}$;

Passo 9: Reinicie a iteração até obter as p triplas.

No Algoritmo 1 o estimador OLS aplicado nos passos 2, 4 e 7 pode ser facilmente substituído por um estimador robusto. Desenvolvemos a função auxiliar `coef` (seção A.1 dos apêndices) que permite introduzir os estimadores robustos na decomposição em valores singulares a partir do seguinte algoritmo:

Algoritmo 2: Decomposição em Valores Singulares (Versão robusta)

Passo 1: Comece com uma estimativa inicial para o autovetor a esquerda U_1 pondo $U_1^{(0)}$ e $k = 0$;

Passo 2: Para cada coluna $j, j = 1, 2, \dots, p$, ajuste o coeficiente c_j da regressão de y_{ij} em $U_{i1}^{(k)}$ utilizando um método robusto;

Passo 3: Calcule a estimativa resultante do autovetor direito $V_1^{(k)} = \mathbf{c}/\|\mathbf{c}\|$, onde $\|\cdot\|$ é a norma euclidiana;

Passo 4: Usando esta estimativa do autovetor direito, redefina a estimativa do

autovetor esquerdo. Para cada linha $i, i = 1, 2, \dots, n$, ajuste o coeficiente d_i da regressão de y_{ij} em $V_{j1}^{(k)}$ utilizando um método robusto;

Passo 5: Calcule a estimativa resultante do autovetor esquerdo $U_1^{(k+1)} = \mathbf{d}/\|\mathbf{d}\|$. Faça $k = k + 1$;

Passo 6: Iterar os passos de 2 a 5 até obter a convergência. Sejam U_1^* e V_1^* as estimativas finais obtidos para os autovetores à esquerda e à direita, respectivamente;

Passo 7: Encontre o autovalor λ_1 como o coeficiente da regressão de y_{ij} em $U_{i1}^* V_{j1}^*$ utilizando um método robusto;

Passo 8: Substitua Y por $Y - \lambda_1^* U_{i1}^* V_{j1}^{*T}$;

Passo 9: Reinicie a iteração até obter as p triplas.

A substituição da SVD convencional por uma variação obtida a partir do algoritmo 2 permite o cálculo de parâmetros de entrada resistentes, gerando assim estimativas robustas para forma média procustes. Aqui cabe notar que, cada escolha de um modelo de regressão robusto na SVD resultará em um estimador robusto distinto tanto para full OPA quanto para full GPA, mais ainda, caso a função `coef` seja ajustada pelo modelo OLS, a SVD proposta será equivalente à convencional tonando-se, a última, um caso particular da primeira. as variações robustas para SVD podem ser encontradas na seção A.2 dos apêndices.

As versões robustas para os estimadores full OPA constantes no apêndice A.3 foram desenvolvidas substituindo a SVD convencional pelos estimadores robustos para SVD no algoritmo proposto por [Claude \(2008\)](#).

3.3 Análise generalizada de procustes completa (*full GPA*) a partir de estimadores robustos

Apesar da sobreposição de duas formas ser aparentemente simples, funções para sobreposição de mais de duas formas não são triviais. Evidentemente, A ideia básica para generalizar qualquer técnica de sobreposição consiste em definir uma forma média como referência, permitindo assim a visualização de variabilidade das formas que compõem a amostra ([Claude, 2008](#)).

O algoritmo 3 permite o cálculo da forma média através da full GPA,

Algoritmo 3

Passo 1: Extraia informações sobre o tamanho da matriz;

Passo 2: Crie uma matriz vazia para armazenar configurações dimensionadas e rotacionadas e um vetor inicial para armazenar o tamanho do centroide;

Passo 3: Traduza e dimensione as configurações para o tamanho da unidade;

Passo 4: Defina a partir de um modelo robusto a quantidade Q_m que deve ser minimizada;

Passo 5: Itere até que as diferenças entre as formas não diminuam;

Passo 6: Defina a forma média (M) ignorando a configuração que será rotacionada;

Passo 7: Faça uma sobreposição completa de procustes entre a forma média e a i -ésima configuração.

Note que, no passo 4, a quantidade Q_m a ser minimizada é obtida por um estimador full OPA. Os estimadores robustos para full GPA constantes na seção A.4 dos apêndices foram obtidas a partir dos estimadores full OPA robustos obtidos na subseção 3.2.1.

Capítulo 4

Aplicações

Neste capítulo apresentamos um breve comentário sobre a variabilidade de formas nos bancos de dados disponíveis em linguagem R especificamente na biblioteca Shapes e suas implicações na construção de cenários para avaliação dos estimadores desenvolvidos. Posteriormente apresentamos duas aplicações, sendo que a primeira consiste em avaliar o ajuste dos estimadores robustos para full GPA ao banco 1 (digit3.dat) que apresenta alta variabilidade das formas e a segunda que consiste em avaliar a resistência dos estimadores a outliers em cenários estruturados a partir do banco 2 (qlet2.dat).

4.1 Características dos dados e a estruturação de cenários

Na análise procustes, se duas configurações diferirem por um único marco anatômico, o algoritmo GPA atribuirá diferenças em toda estrutura da forma no momento da sobreposição. Desse modo, a variabilidade dos dados analisados tende a interferir no cálculo da forma média que, a depender do método, poderá variar expressivamente.

Na Figura 4.1 podemos visualizar a distribuição dos marcos anatômicos e a forma média em destaque nos dá uma ideia da variabilidade das formas em cada um dos bancos de dados.

No mesmo sentido, dados que apresentam baixa variabilidade são ideais para simulações que envolvem presença de outliers que podem ser gerados deslocando-se um ou mais marcos anatômicos da forma média. A Figura 4.2 ilustra a inserção de um outlier no banco de dados 2, as formas em 4.2(a) e 4.2(b) diferem apenas pelo deslocamento do marco 1, já as formas 4.2(c) e 4.2(d) nos dão uma ideia visual de do novo cenário, os marcos da forma média e da forma inserida estão destacados em vermelho.

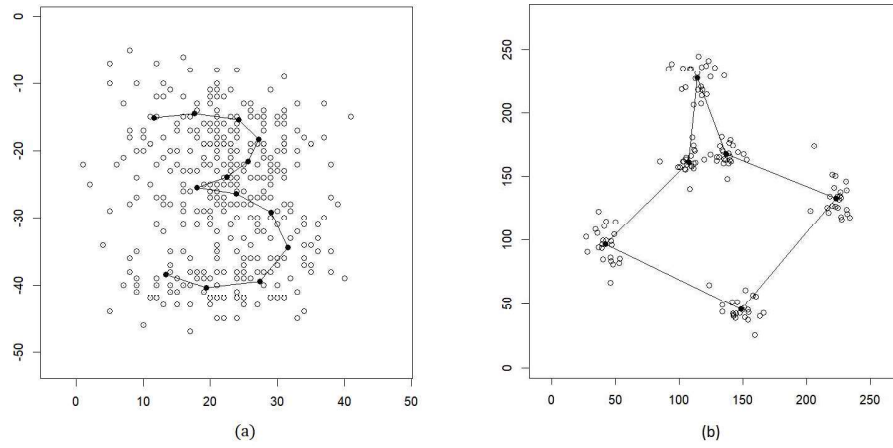


Figura 4.1: Distribuição dos marcos anatômicos em cada banco de dados, a Figura 4.1(a) corresponde ao banco 1, já 4.1(b) corresponde ao banco 2.

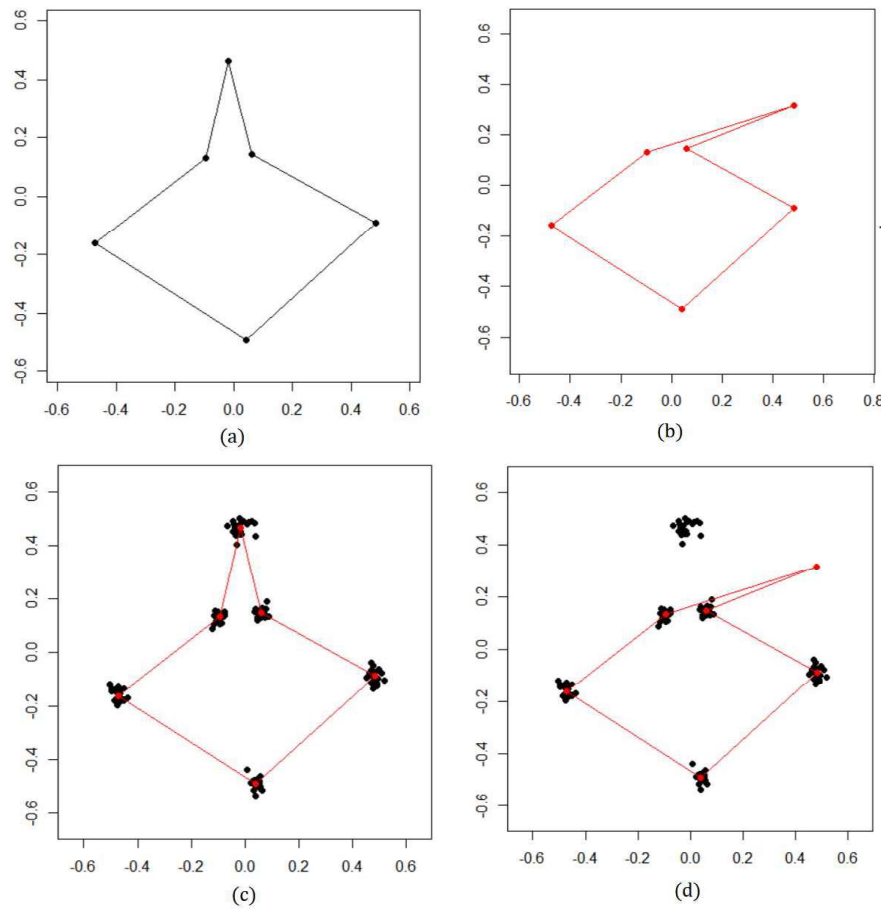


Figura 4.2: Inserção de outlier no banco de dados 2.

Nas Seções seguintes, abordaremos o ajuste dos modelos aplicando-os aos dados do banco 1 e a sensibilidade dos mesmos a outliers em cenários construídos a partir dos dados do banco 2. Adotaremos como medida de ajuste, o quadrado da norma

de Forbenius da matriz residual dada por $\epsilon = \|R\|_F^2 = \text{traço}(R^T R)$.

4.2 Aplicação 1: dados da transcrição do dígito 3 por pessoas de idades variadas.

O conjunto de dados do banco 1 corresponde a uma amostra de 30 configurações contendo 13 marcos anatômicos em 2 dimensões extraídos da transcrição do dígito 3 por pessoas de idades variadas. A principal característica dos dados desse banco consiste na alta variabilidade das suas configurações, o que, a depender do estimador robusto adotado, pode levar a distorções na forma média no processo de sobreposição. A figura 4.3 exibe a sobreposição entre a forma média procustes obtida por cada um dos métodos propostos a forma média procustes original.

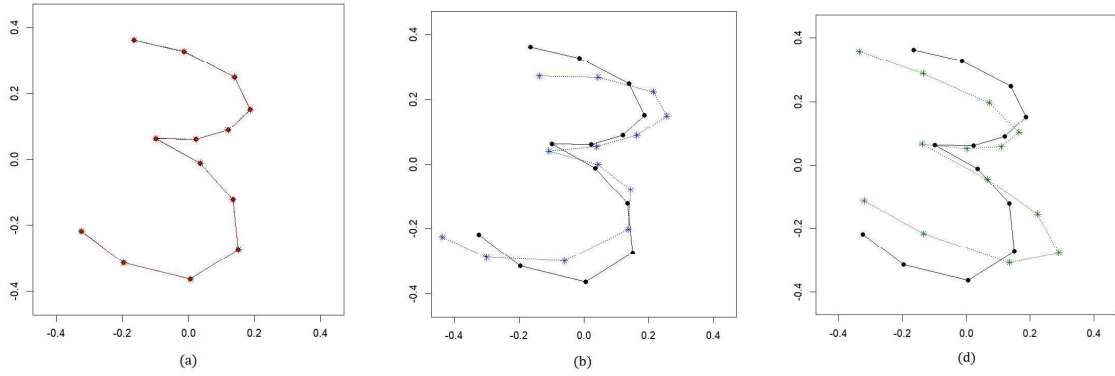


Figura 4.3: Sobreposição das formas médias procustes obtidas pelos estimadores GPA-M (a), GPA-MM (b) e GPA-Kernel (c) a forma média original.

Adotamos $\sigma = 0,01$ como parâmetro de tolerância inicial para todos os estimadores, o algoritmo 3 foi adaptado para os estimadores GPA-M, GPA-MM, GPA-Kernel e GPA-LMS sendo que o último não convergiu. Os valores e de ajuste entre as formas médias podem ser vistos na Tabela 4.1.

Pôde-se observar que as formas médias obtidas pelos estimadores GPA e GPA-M são similares (Figura 4.3(a)), sendo que erro absoluto $e = 0,00000391$ indica uma sobreposição quase perfeita. Entretanto, os resultados obtidos a partir dos estimadores robustos MM (Figura 4.3(b)) e do GPA-Kernel (Figura 4.3(c)) indicam que a variabilidade influencia negativamente na distribuição dos marcos anatômicos no processo de sobreposição para esses estimadores. A figura 4.4 mostra a distribuição dos marcos após a aplicação dos estimadores.

Por fim, mesmo tendo os efeitos relacionados a dispersão mitigados pela retirada dos efeitos de rotação, translação e escala, os estimadores GPA-MM ($e = 0.06982869$) e GPA-Kernel ($e = 0.13395170$) apresentaram erro absoluto alto em

Tabela 4.1: Medida de ajuste dos modelos ao banco de dados 1.

Modelo	e
GPA - M	0,00000391
GPA - MM	0.06982869
GPA - Kernel	0.13395170

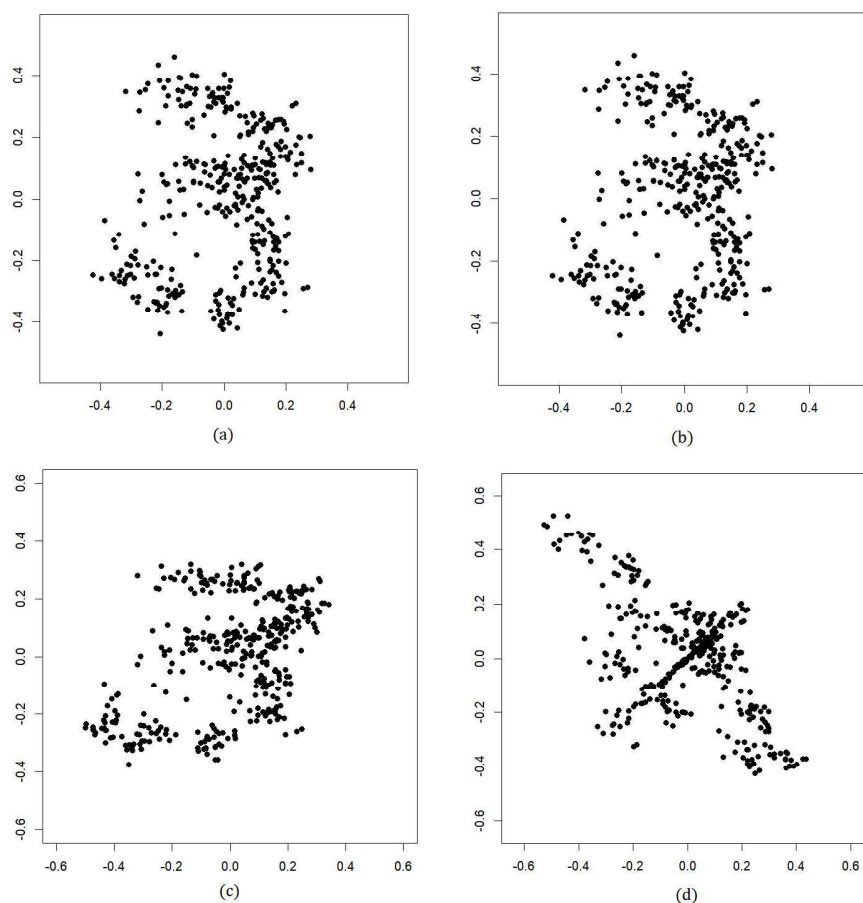


Figura 4.4: Marcos anatômicos do banco de dados 1 após aplicação dos estimadores GPA (a), GPA-MM (b), GPA-MM (c) e GPA-Kernel (d).

comparação ao estimador GPA-M além de apresentarem uma distorção nas formas após o processo de sobreposição que pode estar associada a alguma característica

intrínseca destes estimadores. O bom desempenho do estimador GPA-M para este banco de dados sinaliza um bom desempenho nos cenários avaliados na Seção a seguir.

4.3 Aplicação 2: dados da vértebra T2 de ratos

O conjunto de dados do banco 2 corresponde a uma amostra de 23 configurações contendo 6 marcos anatômicos em 2 dimensões obtidos a partir da análise da vértebra T2 de ratos. Em virtude da equivalência morfológica entre as vértebras, a distribuição dos marcos anatômicos apresentou um grau de homogeneidade maior em relação ao banco de dados anterior, o que pode ser visualizado na Figura 4.5.

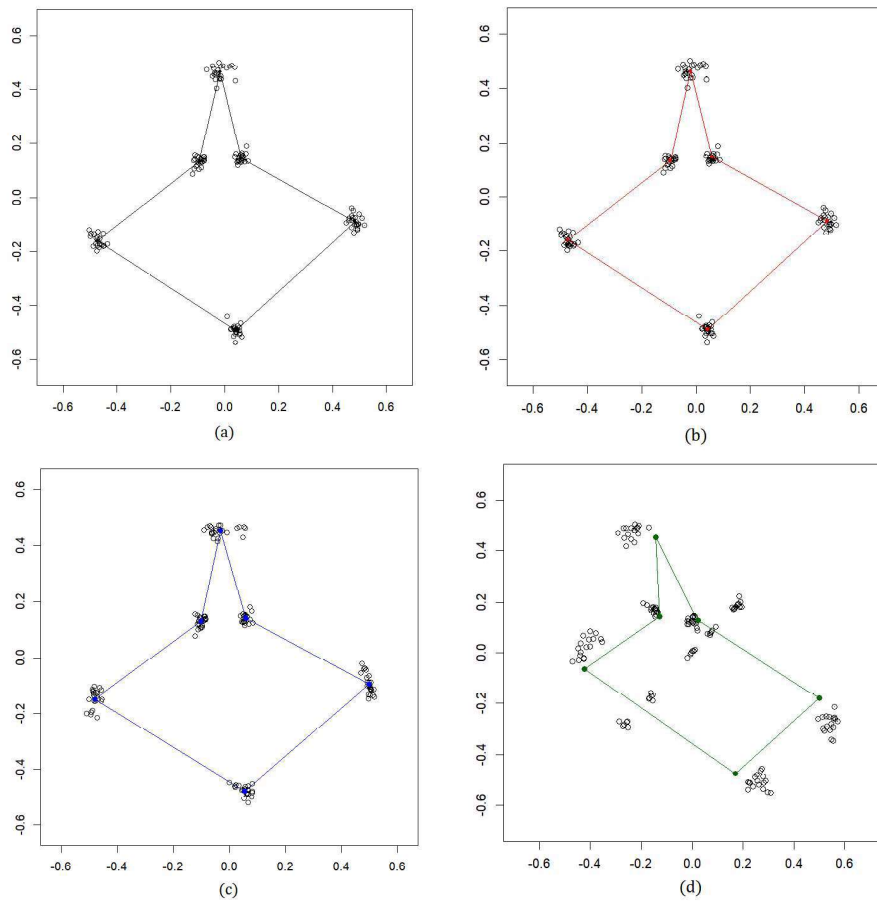


Figura 4.5: Marcos anatômicos do banco 2 após aplicação dos estimadores GPA (a), GPA-M (b), GPA-MM (c) e GPA-Kernel (d).

Para avaliação do desempenho dos estimadores na presença de outliers, foram construídos 24 cenários distintos. A inserção dos outliers se deu de maneira progressiva em cada um dos marcos anatômicos, a Figura 4.6 ilustra 6 destes cenários.

Note que o cenário 1 exposto em 4.6(a) apresenta a inserção de 4 outliers no marco anatômico 1, o que foi repetido para cada um dos marcos restantes, culminando no

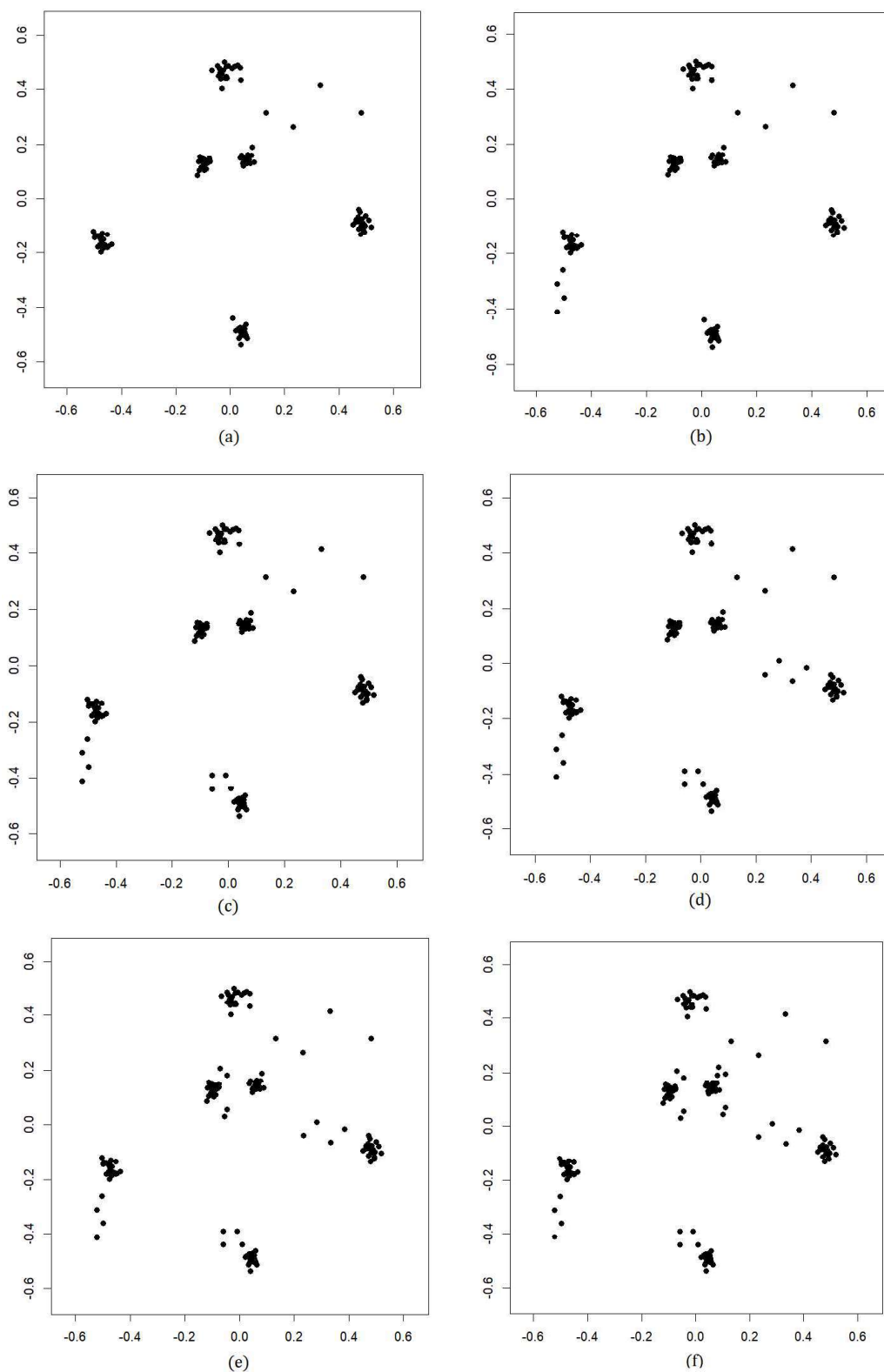


Figura 4.6: Inserção de outliers, configurações com quatro (a) oito (b) doze (c), dezesseis (d), vinte (e) e vinte e quatro (f) outliers.

cenário 6 apresentado em 4.6(f) onde é possível verificar 24 outliers. O desempenho

dos modelos para cada um destes cenários pode ser visualizado na Tabela 4.2,

Tabela 4.2: Evolução das medidas de ajuste ao banco de dados 2 em cenários com outliers.

Modelo	Cenário 1	Cenário 2	Cenário 3	Cenário 4	Cenário 5	Cenário 6
GPA	0.00145	0.00211	0.00245	0.00338	0.00333	0.00328
GPA - M	0.00145	0.00206	0.00238	0.00119	0.00105	0.00095
GPA - MM	0.00385	0.00515	0.00549	0.00697	0.00699	0.00696
GPA - Kernel	0.52348	0.52162	0.41943	0.47116	0.50243	0.51825

Ao que parece, a robustez transposta pelas SVD garante a resistência dos modelos testados, restando ainda, para conclusão da avaliação, a verificação dos parâmetros de tolerância que propiciem o melhor ajuste. Na subseção a seguir é apresentada uma breve avaliação da influência do parâmetro de tolerância no ajuste do estimador GPA-M.

4.3.1 Avaliação da influência do parâmetro de tolerância - GPA-M

De modo geral, o número de iterações não afeta significativamente o ajuste de nenhum dos modelos. Por outro lado, a medida que aumenta o número de formas, a possibilidade de variação na forma média para um do estimador que apresente um bom ajuste diminui, restando assim observar o comportamento do estimador quanto a variação do parâmetro de tolerância. Na Tabela 4.3 podemos verificar a evolução do ajuste do modelo GPA-M quando aumentado o parâmetro de tolerância.

Tabela 4.3: Medidas de ajuste com aumento do parâmetro de tolerância.

Tolerância	Cenário 1	Cenário 2	Cenário 3	Cenário 4	Cenário 5	Cenário 6
$\sigma = 0,01$	0.00145	0.00206	0.00238	0.00119	0.00105	0.00095
$\sigma = 0,0001$	0.00105	0.00114	0.00106	0.00122	0.00106	0.00101

Observando a Tabela 4.3 é possível verificar que alterações feitas no parâmetro de tolerância não apresentam ganho expressivo. Além disso, o aumento da precisão implica necessariamente em um custo computacional elevado, o que não se justifica tendo em vista que oscilação dos resultados sugere que o melhor ajuste não depende significativamente dos critérios de precisão adotados.

A aplicação dos estimadores propostos neste trabalho visa obter modelos robustos para a análise de procustes, incorporando algumas das técnicas de regressão existentes na literatura. A aplicação destes estimadores a bancos de dados com padrões

estatisticamente distintos se mostrou eficiente, sendo que estes grupos apresentaram uma diferença marcante no que diz respeito a disposição e variabilidade dos marcos anatômicos, onde o primeiro apresenta uma variabilidade maior de forma e comparação ao segundo que, por não apresentar esta característica, se mostrou propício a simulações com inserção de outliers. Nesse contexto, o estimador GPA-M apresentou um desempenho superior aos demais estimadores, figurando assim como opção interessante para aplicações a dados de análise de forma que apresentem outliers.

Capítulo 5

Considerações finais

Os estimadores robustos propostos neste trabalho visam obter formas médias resistentes na análise de procustes, integrando algumas das técnicas de regressão robustas existentes na literatura e a decomposição em valores singulares.

A avaliação dos métodos consistiu em utilizar os estimadores propostos em dois grupos com padrões de forma estatisticamente distintos. Estes grupos apresentaram uma diferença marcante no que diz respeito a disposição e variabilidade dos marcos anatômicos, sendo que o primeiro apresenta uma variabilidade maior de forma e comparação ao segundo que, por não apresentar esta característica, se mostrou propício a simulações com inserção de outliers.

Os resultados das aplicações indicam que estimador GPA-M se mostrou consistente, apresentando um bom desempenho tanto na aplicação 1 quanto na aplicação 2, figurando assim como uma boa opção de escolha no tratamento de formas. Já os estimadores GPA-MM e GPA-kernel apesar de se ajustarem aos dados, apresentaram um grau de distorção das formas no processo de sobreposição, o que comprometeu a estimativa da forma média procustes.

Por fim, cabe notar que o comportamento computacional dos estimadores GPA-MM, GPA-LMS e GPA-Kernel devem ser investigados tendo em vista que os mesmos, apresentaram um custo computacional elevado, o que pode estar relacionado a propagação de erros, a assimetria da matriz de rotação estimada na decomposição em valores ou a qualquer outro fator que possa ter causado as distorções das formas verificadas na full GPA quando do uso destes estimadores.

Referências Bibliográficas

- ASHBURNER, J., FRISTON, K., 2001, “Morfometria baseada em voxel, os métodos”, *Neuroimagem*, v. 11, pp. 805–821.
- BOOKSTEIN, F., 1984, “A statistical method for biological shape comparisons”, *Journal of Theoretical Biology*, v. 107, n. 3 (abril), pp. 475–520.
- BOOKSTEIN, F., 1986, “Size and Shape Spaces for Landmark Data in Two Dimensions”, *Statistical Science*, v. 1, n. 2 (may), pp. 181–222.
- BOOKSTEIN, F., 2001, *Combinando as ferramentas da Morfometria Geométrica*. In: Marcus, L.F., Corti, M., Loy, A., Naylor, G.J.P., Slice, D.E. (eds), v. 284. Springer.
- CARVALHO, F., NETO, E., FERREIRA, M., 2017, “A robust regression method based on exponential-type kernel functions”, *Neurocomputing*, , n. 234, pp. 58–74.
- CLAUDE, J., 2008, *Morphometrics with R*. John Wiley Sons. ISBN: 9781119072508.
- CÂNDIDO, D., 2017, “Agrupamento Espectral para Dados de Formas”, p. 58.
- DRYDEN, I. L., MARDIA, K. V., 2016, *Statistical shape analysis with applications in R*. Springer Science+Business Media. ISBN: 9780387777894.
- ER, F., 1998, “Robust methods in statistical shape analysis”, *University of Leeds*.
- FRIEDMAN, J. AND HASTIE, T., TIBSHIRANI, R., 2009, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer. ISBN: 9780387848570.
- GALTON, F., 1877, “Regression towards mediocrity in hereditary stature”, *Journal of the Anthropological Institute of Great Britain and Ireland*, , n. 8, pp. 356–362.

- GAUSS, C. F., 1809, “Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium”, *mpensis Dieterichianis*.
- GOMES, F., 2020, “Métodos de agrupamento para formas planas”, *Orientador: Getúlio José Amorim do Amaral Dissertação (Mestrado) – Universidade Federal de Pernambuco. CCEN. Estatística. Recife*, p. 66 p.
- HAYKIN, S., 2014, *Adaptive Filter Theory*. Pearson. ISBN: 9780132671453.
- HUBER, P. J., 1973, “Robust Statistics”, *Wiley Interdisciplinary Reviews: Computational Statistics*, , n. 1(3), pp. 195–206.
- HUBER, P., RONCHETTI, E., 2018, *Robust Statistics*. Wiley. ISBN: 9780470129906.
- HYUNA, G., LEEB, B. H., CHOIL, Y. S., 2018, “Resistant GPA algorithms based on the M and LMS estimation”, *Communications for Statistical Applications and Methods*, v. 25, n. 6 (Nov), pp. 673–685. ISSN: 2287-7843.
- KENDALL, D., 1984, “Shape manifolds, procrustean metrics and complex projective spaces”, *Bull. Land. Math. Soc.*, v. 16, pp. 81–121.
- KENDALL, D., 1977, “The DifFusion of Shape”, *Advances in Applied Probability*, , n. 9, pp. 428–430.
- LELE, S. R., RICHTSMEIER, J., 1991, “Euclidean Distance Matrix Analysis: A Coordinate-Free Approach for Comparing Biological Shapes Using Landmark Data”, *American Journal of Physical Anthropology*, , n. 86.
- LESTREL, P., 1982, “Um procedimento analítico fourier para descrever formas morfológicas complexas.” *Prog Clin Biol Res.*, , n. 101.
- MARDIA, K., GILL, C., 1995, “Proceedings in Current Issues in Statistical Shape Analysis”, *Department of Statistics, University of Leeds*, p. 234 p.
- NETO, E.A.L.;RODRIGUES, P., 2023, “Kernel robust singular value decomposition”, *Expert Systems with Application*, v. 211 (January), pp. 428–430.
- RODRIGUES, P. C., LOURENÇO, V., MAHMOUD, R., 2018, “A robust approach to singular spectrum analysis”, *SPECIAL ISSUE ARTICLE*, pp. 1437–1447.
- ROHLF, F., SLICE, D., 1990, “Extensions of the Procrustes method for the optimal superimposition of landmarks”, *Systematic Zoology*, 39, pp. 40–59.

SILVA, A. B., SANTOS, C. D., “Mínimos Quadrados Ordinários: Uma Abordagem Estatística para Análise de Regressão.” *Revista Brasileira de Estatística*, , n. 38(2) (março), pp. 123–136.

Apêndice A

Funções auxiliares e estimadores robustos

A.1 Funções Auxiliares

Hiperparâmetros para as funções Kernel

```
gauss.kern = function(a, b, s){as.vector(exp(-(1/s)*(a-b)^2))}
```

```
sigma2est = function(y1, y2, frac = .5){  
  n = length(y1)  
  m = floor(n*frac)  
  idx1 = sample(1:n, m, replace = T)  
  idx2 = sample(1:n, m, replace = T)  
  tmp = (y1[idx1] - y2[idx2])^2  
  mean(quantile(tmp[tmp != 0], probs = c(.9, .5, .1)))}
```

```
sigma2est2.new = function(y1, y2){  
  D = outer(y1,y2,-")  
  D = D^2  
  D.no.zero = D[which(!D == 0)]  
  median(D.no.zero)}
```

Funções auxiliares para análise de formas ([Claude, 2008](#))

```
centsiz = function(M){  
  p = dim(M)[1]  
  size = sqrt(sum(apply(M, 2, var))*(p-1))
```

```

list("centroid_size"= size,"scaled"= M/size)})

trans1 = function(M){scale(M,scale=F)}

fPsup = function(M1, M2) {
  k = ncol(M1)
  Z1 = trans1(centsiz(M1)[[2]])
  Z2 = trans1(centsiz(M2)[[2]])
  sv = svd(t(Z2)%*%Z1)
  U = sv$v; V = sv$u; Delt = sv$d
  sig = sign(det(t(Z2)%*%Z1))
  Delt[k]= sig*abs(Delt[k]) ; V[,k]= sig * V[,k]
  Gam = U%*%t(V)
  beta= sum(Delt)
  list(Mp1=beta*Z1%*%Gam,Mp2=Z2,rotation=Gam,
  scale=beta,DF=sqrt(1-beta^2))}

mshape = function(A){apply(A, c(1,2), mean)}

```

Funções **coef**

```

coef.M = function(y, x) { d = data.frame(y, x)
  f = as.formula("y ~ 0+x")
  modelo = rlm(f, data = d, method = "M")
  betahat = as.numeric(modelo$coefficients)
  return(betahat)

coef.LMS = function(y, x) {
  d = data.frame(y, x)
  f = as.formula("y ~ 0+x")
  modelo = lqs(f, data= d, method = "lms",
  subset, na.action, model = TRUE,
  x.ret = FALSE, y.ret = FALSE, contrasts = NULL)
  betahat = as.numeric(modelo$coefficients)
  return(betahat)}

coef.MM = function(y, x) { d = data.frame(y, x)
  f = as.formula("y ~ 0+x")
  modelo = rlm(f, data = d, method = "MM")
  betahat = as.numeric(modelo$coefficients)

```

```

return(betahat)

coef.k1 = function(y, x, tol = 1e-6, maxit = 100){
  x = as.matrix(x)
  n = nrow(x)
  betahat = solve(t(x)%*%x)%*%t(x)%*%y
  yhat = x%*%betahat
  s2 = sigma2est(y, yhat, frac = 1)
  K = gauss.kern(y, yhat, s2)
  S = sum(2-2*K)
  it = 1
  repeat {
    it = it+1
    betahat = solve(t(x)%*%diag(K)%*%x)%*%t(x)%*%diag(K)%*%y
    yhat = x%*%betahat
    K = gauss.kern(y, yhat, s2)
    S = c(S, sum(2-2*K))
    print(S[it])
    if (is.nan(S[it]) || is.na(S[it]) ||
        abs(S[it]-S[(it-1)]) <= tol ||
        it >= maxit) break }
  return(betahat)}

coef.k2 = function(y, x, tol = 1e-6, maxit = 100){
  x = as.matrix(x)
  n = nrow(x)
  betahat = solve(t(x)%*%x)%*%t(x)%*%y
  yhat = x%*%betahat
  s2 = sigma2est2.new(y, yhat)
  K = gauss.kern(y, yhat, s2)
  S = sum(2-2*K)
  it = 1
  repeat {
    it = it+1
    betahat = solve(t(x)%*%diag(K)%*%x)%*%t(x)%*%diag(K)%*%y
    yhat = x%*%betahat
    K = gauss.kern(y, yhat, s2)
    S = c(S, sum(2-2*K))
    if (abs(S[it]-S[(it-1)]) <= tol ||

```



```

    it >= maxit) break}
(return(betahat))}

coef.k3 = function(y, x, tol = 1e-6, maxit = 100){
  x = as.matrix(x)
  n = nrow(x)
  betahat = solve(t(x)%*%x)%*%t(x)%*%y
  yhat = x%*%betahat
  s2 = sum((y-yhat)^2)/(n-ncol(x))
  K = gauss.kern(y, yhat, s2)
  S = sum(2-2*K)
  it = 1
  repeat {
    it = it+1
    betahat = solve(t(x)%*%diag(K)%*%x)%*%t(x)%*%diag(K)%*%y
    yhat = x%*%betahat
    K = gauss.kern(y, yhat, s2)
    S = c(S, sum(2-2*K))
    print(S[it])
    if (is.nan(S[it]) || abs(S[it]-S[(it-1)]) <= tol ||
        it >= maxit) break}
  (return(betahat))}

coef.k4 = function(y, x, tol = 1e-6, maxit = 100){
  x = as.matrix(x)
  n = nrow(x)
  betahat = solve(t(x)%*%x)%*%t(x)%*%y
  yhat = as.numeric(x%*%betahat)
  s2 = h.select(y, yhat, method="aicc")
  K = gauss.kern(y, yhat, s2)
  S = sum(2-2*K)
  it = 1
  repeat {
    it = it+1
    betahat = solve(t(x)%*%diag(K)%*%x)%*%t(x)%*%diag(K)%*%y
    yhat = x%*%betahat
    K = gauss.kern(y, yhat, s2)
    S = c(S, sum(2-2*K))

```

```

        if (abs(S[it]-S[(it-1)]) <= tol || it >= maxit)
break}
    (return(betahat))}

```

A.2 SVD robustas

```

svd.M <- function(x, maxit = 300) {
  if (!requireNamespace("matrixStats", quietly = TRUE))
    stop("package matrixStats required but not available")
  print("SVD.K3")
  svdu <- matrix(NA, nrow = nrow(x), ncol = ncol(x))
  svdv <- matrix(NA, nrow = ncol(x), ncol = ncol(x))
  svdd <- rep(NA, ncol(x))
  for (k in 1:ncol(x)) {
    ak <- apply(abs(x), 1, median, na.rm = TRUE)
    ak <- ak/sqrt(sum(ak^2))
    converged <- FALSE
    it <- 0
    while (!converged) {
      akprev <- ak
      c <- apply(x, 2, coef.M, ak)
      bk <- c/sqrt(sum(c^2))
      d <- apply(x, 1, coef.M, bk)
      ak <- d/sqrt(sum(d^2))
      it <- it + 1
      print(it)
      if (sum((ak - akprev)^2) < 1e-6 ||
          it >= maxit)
        converged <- TRUE}
    eigenk.m <- function(x, a, b){
      new.x <- as.vector(x)
      new.ab <- as.vector(outer(a,b))
      eigenk <- coef.M(new.x, new.ab)
      (return(eigenk))}
    eigenk <- as.numeric(eigenk.m(x, ak, bk))
    x <- x - eigenk * ak %*% t(bk)
    rank.svd = rankMatrix(x)[1]
    svdu[, k] <- ak
  }
}

```

```

    svdv[, k] <- bk
    svdd[k] <- eigenk
  ret <- list()
  ret$d <- svdd
  ret$u <- svdu
  ret$v <- svdv
  return(ret)}

svd.MM <- function(x, maxit = 300) {
  if (!requireNamespace("matrixStats", quietly = TRUE))
    stop("package matrixStats required but not available")
  print("SVD.K3")
  svdu <- matrix(NA, nrow = nrow(x), ncol = ncol(x))
  svdv <- matrix(NA, nrow = ncol(x), ncol = ncol(x))
  svdd <- rep(NA, ncol(x))
  for (k in 1:ncol(x)) {
    ak <- apply(abs(x), 1, median, na.rm = TRUE)
    ak <- ak/sqrt(sum(ak^2))
    converged <- FALSE
    it <- 0
    while (!converged) {
      akprev <- ak
      c <- apply(x, 2, coef.MM, ak)
      bk <- c/sqrt(sum(c^2))
      d <- apply(x, 1, coef.MM, bk)
      ak <- d/sqrt(sum(d^2))
      it <- it + 1
      print(it)
      if (sum((ak - akprev)^2) < 1e-6 ||
          it >= maxit)
        converged <- TRUE}
    eigenk.m <- function(x, a, b){
      new.x <- as.vector(x)
      new.ab <- as.vector(outer(a,b))
      eigenk <- coef.MM(new.x, new.ab)
      (return(eigenk))}
    eigenk <- as.numeric(eigenk.m(x, ak, bk))
    x <- x - eigenk * ak %*% t(bk)
  rank.svd = rankMatrix(x)[1]

```

```

    svdu[, k] <- ak
    svdv[, k] <- bk
    svdd[k] <- eigenk
ret <- list()
ret$d <- svdd
ret$u <- svdu
ret$v <- svdv
return(ret)}

svd.LMS <- function(x, maxit = 300) {
  if (!requireNamespace("matrixStats", quietly = TRUE))
    stop("package matrixStats required but not available")
  print("SVD.K3")
  svdu <- matrix(NA, nrow = nrow(x), ncol = ncol(x))
  svdv <- matrix(NA, nrow = ncol(x), ncol = ncol(x))
  svdd <- rep(NA, ncol(x))
  for (k in 1:ncol(x)) {
    ak <- apply(abs(x), 1, median, na.rm = TRUE)
    ak <- ak/sqrt(sum(ak^2))
    converged <- FALSE
    it <- 0
    while (!converged) {
      akprev <- ak
      c <- apply(x, 2, coef.LMS, ak)
      bk <- c/sqrt(sum(c^2))
      d <- apply(x, 1, coef.LMS, bk)
      ak <- d/sqrt(sum(d^2))
      it <- it + 1
      print(it)
      if (sum((ak - akprev)^2) < 1e-6 ||
          it >= maxit)
        converged <- TRUE}
    eigenk.m <- function(x, a, b){
      new.x <- as.vector(x)
      new.ab <- as.vector(outer(a,b))
      eigenk <- coef.LMS(new.x, new.ab)
      (return(eigenk))}
    eigenk <- as.numeric(eigenk.m(x, ak, bk))
    x <- x - eigenk * ak %*% t(bk)
  }
}

```

```

    rank.svd = rankMatrix(x)[1]
    svdu[, k] <- ak
    svdv[, k] <- bk
    svdd[k] <- eigenk
  ret <- list()
  ret$d <- svdd
  ret$u <- svdu
  ret$v <- svdv
  return(ret)}

svd.K1 <- function(x, maxit = 300) {
  if (!requireNamespace("matrixStats", quietly = TRUE))
    stop("package matrixStats required but not available")
  print("SVD.K3")
  svdu <- matrix(NA, nrow = nrow(x), ncol = ncol(x))
  svdv <- matrix(NA, nrow = ncol(x), ncol = ncol(x))
  svdd <- rep(NA, ncol(x))
  for (k in 1:ncol(x)) {
    ak <- apply(abs(x), 1, median, na.rm = TRUE)
    ak <- ak/sqrt(sum(ak^2))
    converged <- FALSE
    it <- 0
    while (!converged) {
      akprev <- ak
      c <- apply(x, 2, coef.K1, ak)
      bk <- c/sqrt(sum(c^2))
      d <- apply(x, 1, coef.K1, bk)
      ak <- d/sqrt(sum(d^2))
      it <- it + 1
      print(it)
      if (sum((ak - akprev)^2) < 1e-6 ||
          it >= maxit)
        converged <- TRUE}
    eigenk.m <- function(x, a, b){
      new.x <- as.vector(x)
      new.ab <- as.vector(outer(a,b))
      eigenk <- coef.K1(new.x, new.ab)
      (return(eigenk))}
    eigenk <- as.numeric(eigenk.m(x, ak, bk))
  }
}

```

```

    x <- x - eigenk * ak %*% t(bk)
    rank.svd = rankMatrix(x)[1]
    svdu[, k] <- ak
    svdv[, k] <- bk
    svdd[k] <- eigenk
  }
  ret <- list()
  ret$d <- svdd
  ret$u <- svdu
  ret$v <- svdv
  return(ret)}

```

```

svd.K2 <- function(x, maxit = 300) {
  if (!requireNamespace("matrixStats", quietly = TRUE))
    stop("package matrixStats required but not available")
  print("SVD.K3")
  svdu <- matrix(NA, nrow = nrow(x), ncol = ncol(x))
  svdv <- matrix(NA, nrow = ncol(x), ncol = ncol(x))
  svdd <- rep(NA, ncol(x))
  for (k in 1:ncol(x)) {
    ak <- apply(abs(x), 1, median, na.rm = TRUE)
    ak <- ak/sqrt(sum(ak^2))
    converged <- FALSE
    it <- 0
    while (!converged) {
      akprev <- ak
      c <- apply(x, 2, coef.K2, ak)
      bk <- c/sqrt(sum(c^2))
      d <- apply(x, 1, coef.K2, bk)
      ak <- d/sqrt(sum(d^2))
      it <- it + 1
      print(it)
      if (sum((ak - akprev)^2) < 1e-6 ||
          it >= maxit)
        converged <- TRUE}
    eigenk.m <- function(x, a, b){
      new.x <- as.vector(x)
      new.ab <- as.vector(outer(a,b))
      eigenk <- coef.K2(new.x, new.ab)
      (return(eigenk))}
  }
}

```

```

    eigenk <- as.numeric(eigenk.m(x, ak, bk))
    x <- x - eigenk * ak %*% t(bk)
    rank.svd = rankMatrix(x)[1]
    svdu[, k] <- ak
    svdv[, k] <- bk
    svdd[k] <- eigenk
  }
  ret <- list()
  ret$d <- svdd
  ret$u <- svdu
  ret$v <- svdv
  return(ret) }

svd.K3 <- function(x, maxit = 300) {
  if (!requireNamespace("matrixStats", quietly = TRUE))
    stop("package matrixStats required but not available")
  print("SVD.K3")
  svdu <- matrix(NA, nrow = nrow(x), ncol = ncol(x))
  svdv <- matrix(NA, nrow = ncol(x), ncol = ncol(x))
  svdd <- rep(NA, ncol(x))
  for (k in 1:ncol(x)) {
    ak <- apply(abs(x), 1, median, na.rm = TRUE)
    ak <- ak/sqrt(sum(ak^2))
    converged <- FALSE
    it <- 0
    while (!converged) {
      akprev <- ak
      c <- apply(x, 2, coef.K3, ak)
      bk <- c/sqrt(sum(c^2))
      d <- apply(x, 1, coef.K3, bk)
      ak <- d/sqrt(sum(d^2))
      it <- it + 1
      print(it)
      if (sum((ak - akprev)^2) < 1e-6 ||
          it >= maxit)
        converged <- TRUE}
    eigenk.m <- function(x, a, b){
      new.x <-as.vector(x)
      new.ab <- as.vector(outer(a,b))
      eigenk <- coef.K3(new.x, new.ab)

```

```

    (return(eigenk)))}
    eigenk <- as.numeric(eigenk.m(x, ak, bk))
    x <- x - eigenk * ak %*% t(bk)
    rank.svd = rankMatrix(x)[1]
    svdu[, k] <- ak
    svdv[, k] <- bk
    svdd[k] <- eigenk
  ret <- list()
  ret$d <- svdd
  ret$u <- svdu
  ret$v <- svdv
  return(ret)}

svd.K4 <- function(x, maxit = 300) {
  if (!requireNamespace("matrixStats", quietly = TRUE))
    stop("package matrixStats required but not available")
  print("SVD.K3")
  svdu <- matrix(NA, nrow = nrow(x), ncol = ncol(x))
  svdv <- matrix(NA, nrow = ncol(x), ncol = ncol(x))
  svdd <- rep(NA, ncol(x))
  for (k in 1:ncol(x)) {
    ak <- apply(abs(x), 1, median, na.rm = TRUE)
    ak <- ak/sqrt(sum(ak^2))
    converged <- FALSE
    it <- 0
    while (!converged) {
      akprev <- ak
      c <- apply(x, 2, coef.K4, ak)
      bk <- c/sqrt(sum(c^2))
      d <- apply(x, 1, coef.K4, bk)
      ak <- d/sqrt(sum(d^2))
      it <- it + 1
      print(it)
      if (sum((ak - akprev)^2) < 1e-6 ||
          it >= maxit)
        converged <- TRUE}
    eigenk.m <- function(x, a, b){
      new.x <-as.vector(x)
      new.ab <- as.vector(outer(a,b))

```



```

    eigenk <- coef.K4(new.x, new.ab)
    (return(eigenk))}
    eigenk <- as.numeric(eigenk.m(x, ak, bk))
    x <- x - eigenk * ak %*% t(bk)
    rank.svd = rankMatrix(x)[1]
    svdu[, k] <- ak
    svdv[, k] <- bk
    svdd[k] <- eigenk
    ret <- list()
    ret$d <- svdd
    ret$u <- svdu
    ret$v <- svdv
    return(ret) }

```

A.3 Estimadores robustos para full OPA

```

fPsup.M = function(M1, M2) {
  k = ncol(M1)
  Z1 = trans1(centsiz(M1)[[2]])
  Z2 = trans1(centsiz(M2)[[2]])
  sv = svd.M(t(Z2)%*%Z1)
  U = sv$v; V = sv$u; Delt = sv$d
  sig = sign(det(t(Z2)%*%Z1))
  Delt[k]= sig*abs(Delt[k]) ; V[,k]= sig * V[,k]
  Gam = U%*%t(V)
  Gam[2,2] <-Gam[1,1]
  Gam[1,2] <- -Gam[2,1]
  beta= sum(Delt)
  list(Mp1=beta*Z1%*%Gam,Mp2=Z2,
  rotation=Gam,scale=beta,
  DF=sqrt(1-beta^2)) }

```

```

fPsup.MM = function(M1, M2) { k = ncol(M1)
  Z1 = trans1(centsiz(M1)[[2]])
  Z2 = trans1(centsiz(M2)[[2]])
  sv = svd.MM(t(Z2)%*%Z1)
  U = sv$v; V = sv$u; Delt = sv$d
  sig = sign(det(t(Z2)%*%Z1))

```

```

Delt[k]= sig*abs(Delt[k]) ; V[,k]= sig * V[,k]
Gam = U%*%t(V)
Gam[2,2] <-Gam[1,1]
Gam[1,2] <- -Gam[2,1]
beta= sum(Delt)
list(Mp1=beta*Z1%*%Gam,Mp2=Z2,
rotation=Gam,scale=beta,
DF=sqrt(1-beta^2))}

```

```

fPsup.LMS = function(M1, M2) {  k = ncol(M1)
  Z1 = trans1(centsiz(M1)[[2]])
  Z2 = trans1(centsiz(M2)[[2]])
  sv = svd.LMS(t(Z2)%*%Z1)
  U = sv$v; V = sv$u; Delt = sv$d
  sig = sign(det(t(Z2)%*%Z1))
  Delt[k]= sig*abs(Delt[k]) ; V[,k]= sig * V[,k]
  Gam = U%*%t(V)
  Gam[2,2] <-Gam[1,1]
  Gam[1,2] <- -Gam[2,1]
  beta= sum(Delt)
  list(Mp1=beta*Z1%*%Gam,Mp2=Z2,
rotation=Gam,scale=beta,
DF=sqrt(1-beta^2))}

```

```

fPsup.K1 = function(M1, M2) {  k = ncol(M1)
  Z1 = trans1(centsiz(M1)[[2]])
  Z2 = trans1(centsiz(M2)[[2]])
  sv = svd.K1(t(Z2)%*%Z1)
  U = sv$v; V = sv$u; Delt = sv$d
  sig = sign(det(t(Z2)%*%Z1))
  Delt[k]= sig*abs(Delt[k]) ; V[,k]= sig * V[,k]
  Gam = U%*%t(V)
  Gam[2,2] <-Gam[1,1]
  Gam[1,2] <- -Gam[2,1]
  beta= sum(Delt)
  list(Mp1=beta*Z1%*%Gam,Mp2=Z2,
rotation=Gam,scale=beta,
DF=sqrt(1-beta^2))}

```

```

fPsup.K2 = function(M1, M2) {  k = ncol(M1)

```

```

Z1 = trans1(centsiz(M1)[[2]])
Z2 = trans1(centsiz(M2)[[2]])
sv = svd.K2(t(Z2)%*%Z1)
U = sv$v; V = sv$u; Delt = sv$d
sig = sign(det(t(Z2)%*%Z1))
Delt[k]= sig*abs(Delt[k]) ; V[,k]= sig * V[,k]
Gam = U%*%t(V)
Gam[2,2] <-Gam[1,1]
Gam[1,2] <- -Gam[2,1]
beta= sum(Delt)
list(Mp1=beta*Z1%*%Gam,Mp2=Z2,
rotation=Gam,scale=beta,
DF=sqrt(1-beta^2))}

```

```

fPsup.K3 = function(M1, M2) { k = ncol(M1)
Z1 = trans1(centsiz(M1)[[2]])
Z2 = trans1(centsiz(M2)[[2]])
sv = svd.K3(t(Z2)%*%Z1)
U = sv$v; V = sv$u; Delt = sv$d
sig = sign(det(t(Z2)%*%Z1))
Delt[k]= sig*abs(Delt[k]) ; V[,k]= sig * V[,k]
Gam = U%*%t(V)
Gam[2,2] <-Gam[1,1]
Gam[1,2] <- -Gam[2,1]
beta= sum(Delt)
list(Mp1=beta*Z1%*%Gam,Mp2=Z2,
rotation=Gam,scale=beta,
DF=sqrt(1-beta^2))}

```

```

fPsup.K4 = function(M1, M2) { k = ncol(M1)
Z1 = trans1(centsiz(M1)[[2]])
Z2 = trans1(centsiz(M2)[[2]])
sv = svd.K4(t(Z2)%*%Z1)
U = sv$v; V = sv$u; Delt = sv$d
sig = sign(det(t(Z2)%*%Z1))
Delt[k]= sig*abs(Delt[k]) ; V[,k]= sig * V[,k]
Gam = U%*%t(V)
Gam[2,2] <-Gam[1,1]
Gam[1,2] <- -Gam[2,1]

```

```

beta= sum(Delt)
list (Mp1=beta*Z1%*%Gam,Mp2=Z2,
rotation=Gam,scale=beta,
DF=sqrt(1-beta^2)) }

```

A.4 Estimadores robustos para full GPA

```

GPA.M <- function(A, tol=1e-2){
  p<-dim(A)[1]; k<-dim(A)[2]; n<-dim(A)[3]
  temp2<-temp1<-array(NA, dim=c(p,k,n))
  Siz<-numeric(n)
  for (i in 1:n)
    {Acs<-centsiz(A[,i])
    Siz[i]<-Acs[[1]]
    temp1[,i]<-trans1(Acs[[2]])}
  iter<-0; sf<-NA
  M<-temp1[,1]
  for (i in 1:n)
    {temp1[,i]<-fPsup.m(temp1[,i],M)[[1]]}
  M<-mshape(temp1)
  Qm1<-dist(t(matrix(temp1,k*p,n)))
  Q<-sum(Qm1); iter<-0
  sc<-rep(1,n)
  while (abs(Q)>tol){
    for (i in 1:n){
      Z1<-temp1[,i]
      sv<-svd.M(t(M)%*%Z1)
      U<-sv$v; V<-sv$u; Delt<-sv$d
      sig<-sign(det(t(Z1)%*%M))
      Delt[k]<-sig*abs(Delt[k])
      V[,k]<-sig*V[,k]
      phi<-U%*%t(V)
      beta<-sum(Delt)
      temp1[,i]<-X<-sc[i]*Z1%*%phi}
    M<-mshape(temp1)
    for (i in 1:n)
      {sf[i]<-sqrt(sum(diag(temp1[,i]%*%t(M)))
/ (sum(diag(M%*%t(M))) *sum(diag(temp1[,i]

```

```

%%t(temp1[,i]))))
temp2[,i]<-sf[i]*temp1[,i]
M<-mshape(temp2)
sc<-sf*sc
Qm2<-dist(t(matrix(temp2,k*p,n)))
Q<-sum(Qm1)-sum(Qm2)
Qm1<-Qm2
iter=iter+1
temp1<-temp2}
list(rotated=temp2,iterationnumber=iter,
Q=Q,intereuclidean.dist=Qm2,
mshape=centsiz(mshape(temp2))[[2]], cent.size=Siz) }

GPA.MM <- function(A, tol=1e-2){
k<-dim(A)[2]; n<-dim(A)[3]
temp2<-temp1<-array(NA, dim=c(p,k,n))
Siz<-numeric(n)
for (i in 1:n)
{Acs<-centsiz(A[,i])
Siz[i]<-Acs[[1]]
temp1[,i]<-transl(Acs[[2]])}
iter<-0; sf<-NA
M<-temp1[,1]
for (i in 1:n)
{temp1[,i]<-fPsup.MM(temp1[,i],M)[[1]]}
M<-mshape(temp1)
Qm1<-dist(t(matrix(temp1,k*p,n)))
Q<-sum(Qm1); iter<-0
sc<-rep(1,n)
while (abs(Q)>tol){
for (i in 1:n){
Z1<-temp1[,i]
sv<-svd.MM(t(M)%*%Z1)
U<-sv$v; V<-sv$u; Delt<-sv$d
sig<-sign(det(t(Z1)%*%M))
Delt[k]<-sig*abs(Delt[k])
V[,k]<-sig*V[,k]
phi<-U%*%t(V)
beta<-sum(Delt)

```

```

    temp1[,i]<-X<-sc[i]*Z1%*%phi}
M<-mshape(temp1)
for (i in 1:n)
{sf[i]<-sqrt(sum(diag(temp1[,i]%*%t(M)))
/(sum(diag(M%*%t(M)))*sum(diag(temp1[,i]
%*%t(temp1[,i])))))
temp2[,i]<-sf[i]*temp1[,i]}
M<-mshape(temp2)
sc<-sf*sc
Qm2<-dist(t(matrix(temp2,k*p,n)))
Q<-sum(Qm1)-sum(Qm2)
Qm1<-Qm2
iter=iter+1
temp1<-temp2}
list(rotated=temp2,iterationnumber=iter,
Q=Q,intereuclidean.dist=Qm2,
mshape=centsiz(mshape(temp2))[[2]], cent.size=Siz)}

```

```

GPA.LMS <- function(A, tol=1e-2){
p<-dim(A)[1]; k<-dim(A)[2]; n<-dim(A)[3]
temp2<-temp1<-array(NA, dim=c(p,k,n))
Siz<-numeric(n)
for (i in 1:n)
{Acs<-centsiz(A[,i])
Siz[i]<-Acs[[1]]
temp1[,i]<-transl(Acs[[2]])}
iter<-0; sf<-NA
M<-temp1[,1]
for (i in 1:n)
{temp1[,i]<-fPsup.LMS(temp1[,i],M)[[1]]}
M<-mshape(temp1)
Qm1<-dist(t(matrix(temp1,k*p,n)))
Q<-sum(Qm1); iter<-0
sc<-rep(1,n)
while (abs(Q)>tol){
for (i in 1:n){
Z1<-temp1[,i]
sv<-svd.LMS(t(M)%*%Z1)
U<-sv$u; V<-sv$v; Delt<-sv$d

```

```

    sig<-sign(det(t(Z1)%*%M))
    Delt[k]<-sig*abs(Delt[k])
    V[,k]<-sig*V[,k]
    phi<-U%*%t(V)
    beta<-sum(Delt)
    temp1[,i]<-X<-sc[i]*Z1%*%phi}
M<-mshape(temp1)
for (i in 1:n)
{sf[i]<-sqrt(sum(diag(temp1[,i]%*%t(M))))
/(sum(diag(M%*%t(M)))*sum(diag(temp1[,i]
%*%t(temp1[,i]))))}
temp2[,i]<-sf[i]*temp1[,i]}
M<-mshape(temp2)
sc<-sf*sc
Qm2<-dist(t(matrix(temp2,k*p,n)))
Q<-sum(Qm1)-sum(Qm2)
Qm1<-Qm2
iter=iter+1
temp1<-temp2}
list(rotated=temp2,iterationnumber=iter,
Q=Q,intereuclidean.dist=Qm2,
mshape=centsiz(mshape(temp2))[[2]], cent.size=Siz)}

```

```

GPA.K1 <- function(A, tol=1e-2){
p<-dim(A)[1]; k<-dim(A)[2]; n<-dim(A)[3]
temp2<-temp1<-array(NA, dim=c(p,k,n))
Siz<-numeric(n)
for (i in 1:n)
{Acs<-centsiz(A[,i])
Siz[i]<-Acs[[1]]
temp1[,i]<-trans1(Acs[[2]])}
iter<-0; sf<-NA
M<-temp1[,1]
for (i in 1:n)
{temp1[,i]<-fPsup.K1(temp1[,i],M)[[1]]}
M<-mshape(temp1)
Qm1<-dist(t(matrix(temp1,k*p,n)))
Q<-sum(Qm1); iter<-0
sc<-rep(1,n)

```

```

while (abs(Q)>tol){
  for (i in 1:n){
    Z1<-temp1[,i]
    sv<-svd.K1(t(M)%*%Z1)
    U<-sv$v; V<-sv$u; Delt<-sv$d
    sig<-sign(det(t(Z1)%*%M))
    Delt[k]<-sig*abs(Delt[k])
    V[,k]<-sig*V[,k]
    phi<-U%*%t(V)
    beta<-sum(Delt)
    temp1[,i]<-X<-sc[i]*Z1%*%phi}
  M<-mshape(temp1)
  for (i in 1:n)
    {sf[i]<-sqrt(sum(diag(temp1[,i]%*%t(M)))
    /(sum(diag(M%*%t(M)))*sum(diag(temp1[,i]
    %*%t(temp1[,i])))))
    temp2[,i]<-sf[i]*temp1[,i]}
  M<-mshape(temp2)
  sc<-sf*sc
  Qm2<-dist(t(matrix(temp2,k*p,n)))
  Q<-sum(Qm1)-sum(Qm2)
  Qm1<-Qm2
  iter=iter+1
  temp1<-temp2}
list(rotated=temp2,iterationnumber=iter,
Q=Q,intereuclidean.dist=Qm2,
mshape=centsiz(mshape(temp2))[[2]], cent.size=Siz)}

```

```

GPA.K2 <- function(A, tol=1e-2){
  p<-dim(A)[1]; k<-dim(A)[2]; n<-dim(A)[3]
  temp2<-temp1<-array(NA, dim=c(p,k,n))
  Siz<-numeric(n)
  for (i in 1:n)
    {Acs<-centsiz(A[,i])
    Siz[i]<-Acs[[1]]
    temp1[,i]<-trans1(Acs[[2]])}
  iter<-0; sf<-NA
  M<-temp1[,1]
  for (i in 1:n)

```



```

{temp1[,i]<-fPsup.K2(temp1[,i],M)[[1]]}
M<-mshape(temp1)
Qm1<-dist(t(matrix(temp1,k*p,n)))
Q<-sum(Qm1); iter<-0
sc<-rep(1,n)
while (abs(Q)>tol){
  for (i in 1:n){
    Z1<-temp1[,i]
    sv<-svd.K2(t(M)%*%Z1)
    U<-sv$v; V<-sv$u; Delt<-sv$d
    sig<-sign(det(t(Z1)%*%M))
    Delt[k]<-sig*abs(Delt[k])
    V[,k]<-sig*V[,k]
    phi<-U%*%t(V)
    beta<-sum(Delt)
    temp1[,i]<-X<-sc[i]*Z1%*%phi}
  M<-mshape(temp1)
  for (i in 1:n)
    {sf[i]<-sqrt(sum(diag(temp1[,i]%*%t(M)))
    /(sum(diag(M)%*%t(M))*sum(diag(temp1[,i]
    %*%t(temp1[,i])))))
    temp2[,i]<-sf[i]*temp1[,i]}
  M<-mshape(temp2)
  sc<-sf*sc
  Qm2<-dist(t(matrix(temp2,k*p,n)))
  Q<-sum(Qm1)-sum(Qm2)
  Qm1<-Qm2
  iter=iter+1
  temp1<-temp2}
list(rotated=temp2,iterationnumber=iter,
Q=Q,intereuclidean.dist=Qm2,
mshape=centsiz(mshape(temp2))[[2]], cent.size=Siz)}

```

```

GPA.K3 <- function(A, tol=1e-2){
  p<-dim(A)[1]; k<-dim(A)[2]; n<-dim(A)[3]
  temp2<-temp1<-array(NA, dim=c(p,k,n))
  Siz<-numeric(n)
  for (i in 1:n)
    {Acs<-centsiz(A[,i])

```

```

Siz[i]<-Acs[[1]]
temp1[,i]<-transl(Acs[[2]])}
iter<-0; sf<-NA
M<-temp1[,1]
for (i in 1:n)
{temp1[,i]<-fPsup.K3(temp1[,i],M)[[1]]}
M<-mshape(temp1)
Qm1<-dist(t(matrix(temp1,k*p,n)))
Q<-sum(Qm1); iter<-0
sc<-rep(1,n)
while (abs(Q)>tol){
  for (i in 1:n){
    Z1<-temp1[,i]
    sv<-svd.K3(t(M)%*%Z1)
    U<-sv$v; V<-sv$u; Delt<-sv$d
    sig<-sign(det(t(Z1)%*%M))
    Delt[k]<-sig*abs(Delt[k])
    V[,k]<-sig*V[,k]
    phi<-U%*%t(V)
    beta<-sum(Delt)
    temp1[,i]<-X<-sc[i]*Z1%*%phi}
  M<-mshape(temp1)
  for (i in 1:n)
  {sf[i]<-sqrt(sum(diag(temp1[,i]%*%t(M)))
/(sum(diag(M%*%t(M)))*sum(diag(temp1[,i]
%*%t(temp1[,i])))))
  temp2[,i]<-sf[i]*temp1[,i]}
  M<-mshape(temp2)
  sc<-sf*sc
  Qm2<-dist(t(matrix(temp2,k*p,n)))
  Q<-sum(Qm1)-sum(Qm2)
  Qm1<-Qm2
  iter=iter+1
  temp1<-temp2}
list(rotated=temp2,iterationnumber=iter,
Q=Q,intereuclidean.dist=Qm2,
mshape=centsiz(mshape(temp2))[[2]], cent.size=Siz)}

```

```

GPA.K4 <- function(A, tol=1e-2){

```

```

p<-dim(A)[1]; k<-dim(A)[2]; n<-dim(A)[3]
temp2<-temp1<-array(NA, dim=c(p,k,n))
Siz<-numeric(n)
for (i in 1:n)
{Acs<-centsiz(A[,i])
Siz[i]<-Acs[[1]]
temp1[,i]<-transl(Acs[[2]])}
iter<-0; sf<-NA
M<-temp1[,1]
for (i in 1:n)
{temp1[,i]<-fPsup.K4(temp1[,i],M)[[1]]}
M<-mshape(temp1)
Qm1<-dist(t(matrix(temp1,k*p,n)))
Q<-sum(Qm1); iter<-0
sc<-rep(1,n)
while (abs(Q)>tol){
  for (i in 1:n){
    Z1<-temp1[,i]
    sv<-svd.K4(t(M)%*%Z1)
    U<-sv$v; V<-sv$u; Delt<-sv$d
    sig<-sign(det(t(Z1)%*%M))
    Delt[k]<-sig*abs(Delt[k])
    V[,k]<-sig*V[,k]
    phi<-U%*%t(V)
    beta<-sum(Delt)
    temp1[,i]<-X<-sc[i]*Z1%*%phi}
  M<-mshape(temp1)
  for (i in 1:n)
  {sf[i]<-sqrt(sum(diag(temp1[,i]%*%t(M)))
/ (sum(diag(M%*%t(M)))*sum(diag(temp1[,i]
%*%t(temp1[,i])))))
  temp2[,i]<-sf[i]*temp1[,i]}
  M<-mshape(temp2)
  sc<-sf*sc
  Qm2<-dist(t(matrix(temp2,k*p,n)))
  Q<-sum(Qm1)-sum(Qm2)
  Qm1<-Qm2
  iter=iter+1
  temp1<-temp2}

```

```
list(rotated=temp2, iterationnumber=iter,  
Q=Q, intereuclidean.dist=Qm2,  
mshape=centsiz(mshape(temp2))[[2]], cent.size=Siz)}
```