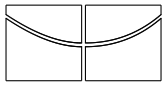


Universidade de Brasília

Instituto de Ciências Exatas

Departamento de Estatística

Dissertação de Mestrado

Implementação, análise e aplicação de algoritmos de agrupamento de dados superdimensionados, longitudinais e com amostras pequenas

por

Alex Pena Tosta da Silva

Orientador: Prof. Dr. George Freitas von Borries

Junho de 2012

Alex Pena Tosta da Silva

Implementação, análise e aplicação de algoritmos de agrupamento de dados superdimensionados, longitudinais e com amostras pequenas

Dissertação apresentada ao Departamento de Estatística do Instituto de Ciências Exatas da Universidadede de Brasília como requisito parcial à obtenção do título de Mestre em Estatística.

Universidade de Brasília

Brasília, Junho de 2012

Termo de Aprovação

Alex Pena Tosta da Silva

Implementação, análise e aplicação de algoritmos de agrupamento de dados superdimensionados, longitudinais e com amostras pequenas

Dissertação apresentada ao Departamento de Estatística do Instituto de Ciências Exatas da Universidadede de Brasília como requisito parcial à obtenção do título de Mestre em Estatística.

Data da defesa: 15 de Junho de 2012

Orientador:

Prof. George Freitas von Borries, PhD

Departamento de Estatística, UnB

Comissão Examinadora:

Prof. Geraldo da Silva e Souza, PhD

Departamento de Estatística, UnB

Profa. Joanlise Marco de Leon Andrade, PhD

Departamento de Estatística, UFRN

Brasília, Junho de 2012

Ficha Catalográfica

TOSTA, ALEX PENA DA SILVA
Implementação, análise e aplicação de algoritmos de agrupamento de dados superdimensionados, longitudinais e com amostras pequenas,
(UnB - IE, Mestre em Estatística, 2012).

Dissertação de Mestrado - Universidade de Brasília. Departamento de Estatística - Instituto de Ciências Exatas.

1. algoritmos de agrupamento 2. dados HDLLSS 3. estimação *jackknife*

4. análise de microarranjo 5. sinais de EEG

É concedida à Universidade de Brasília a permissão para reproduzir cópias desta dissertação de mestrado e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desta dissertação de mestrado pode ser reproduzida sem a autorização por escrito do autor.

Alex Pena

A Deus, por me mostrar o caminho

A meus pais

A meus irmãos

Agradecimentos

Primeiramente, agradeço a Deus, por estar sempre ao meu lado e me mostrar o caminho a ser seguido.

Aos meus pais, Ivan e Tânia, por me ensinarem desde cedo a importância do estudo, da dedicação e da ética. Pai e mãe, vocês são meus maiores exemplos de vida!

Aos meus queridos irmãos, Ivan Jr. e Alan, amigos de sangue e de espírito, por fazerem parte da minha vida. Agradeço, também, à Nádia, à Gisele e às minhas sobrinhas lindas, Paloma e Gabriela.

À minha namorada, Ingrid, por me dar alegria, equilíbrio e conforto necessários à consecução deste objetivo. Aos meus primos e amigos que me acompanharam em todos esses anos de estudo e entenderam minhas repetidas ausências.

Ao meu orientador e amigo, George von Borries, pela paciência e compreensão nos momentos difíceis, pela dedicação e pelo exemplo profissional e pessoal.

Gostaria de agradecer a todos os professores que fizeram parte da minha formação, em especial ao Prof. Lucio Vivaldi, meu orientador na graduação, à Profa. Cibele Queiroz, minha orientadora de Iniciação Científica, e ao Prof. Antônio Eduardo, pelas suas inspiradoras aulas de Inferência Estatística. Agradeço, também, aos funcionários do Departamento de Estatística pela dedicação diária aos alunos da pós-graduação.

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro que permitiu minha dedicação exclusiva aos estudos.

Finalmente, agradeço ao SAS Institute Brasil por permitir o uso do SAS por meio da parceria acadêmica com o Departamento de Estatística da Universidade de Brasília.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho analisa uma série de algoritmos destinados a agrupar variáveis em uma estrutura de dados superdimensionada, longitudinal e com amostras pequenas (do inglês, *High Dimensional Longitudinal Low Sample Size* - HDLLSS). Esses algoritmos utilizam como medida de similaridade o p-valor resultante de um teste de ausência de efeito simples de grupo em um delineamento fatorial com medidas repetidas no tempo.

Os testes não-paramétricos presentes em cada algoritmo serão estudados extensivamente por meio de simulações do erro do tipo I e curvas de poder do teste. Pesquisa bibliográfica dos métodos de agrupamento de dados HDLLSS mostra que a estimação da matriz de covariância é um grande problema em vários algoritmos. Neste trabalho, todas as simulações consideraram três formas distintas de estimação dessa matriz: Σ_i, Σ e Σ_G. Enquanto Σ_i utiliza as informações da i-ésima variável para estimar as matrizes, Σ utiliza todas as variáveis para a estimação de uma única matriz de covariâncias. O terceiro método considerado, Σ_G, estima uma matriz de covariâncias para cada grupo. Esse método apresentou melhores resultados por conseguir detectar a variabilidade entre os grupos com informação suficiente para uma boa qualidade de estimação.

Aplicações em dados de microarranjo e em sinais de eletroencefalograma (EEG) apresentam resultados promissores. Os estudos de simulação sugerem que os algoritmos de agrupamento propostos superam os métodos existentes na literatura destinados a detectar grupos em dados HDLLSS. Além disso, esses algoritmos possuem propriedades desejáveis como invariância a transformações monótonas nos dados e detecção automática do número de grupos amostrais.

Palavras Chave: *algoritmos de agrupamento, dados HDLLSS, análise de microarranjo, sinais de EEG, estimação jacknife.*

Abstract

This dissertation analyses a set of algorithms to cluster variables in high dimensional longitudinal low sample size (HDLLSS) data. These algorithms are based on the use of a p-value from a nonparametric test of no simple effect of group as a similarity measure for the clustering procedure.

The non-parametric tests in each algorithm were studied extensively by means of simulations of type I error and power curves. Investigation of recent literature in HDLLSS clustering algorithms shows that the covariance matrix estimation is a major problem. In this work, all simulations used three different ways of covariance matrix estimation: Σ_i, Σ and Σ_G. While Σ_i uses information from the i-th variable to estimate covariance matrices, Σ uses all variables for estimating a single covariance matrix for the data. The third method considered, Σ_G, estimates one covariance matrix for each group. This estimation method shows better results because it can detect the variability between the groups with sufficient information for a good quality estimation of time covariance structure.

Applications on microarray data and electroencephalogram (EEG) signals show promising results. The simulation studies reveal that the proposed clustering algorithms outperforms existing methods in the literature applied for detecting groups of HDLLSS data exhibiting high clustering accuracy and stability. Furthermore, these algorithms have desirable properties as invariance under monotone transformations and automatic detection of the number of sample groups.

Key words: *clustering algorithms, HDLLSS data analysis, microarray, EEG signals, jacknife estimation.*

Capítulo 1

Introdução

Devido ao avanço de novas tecnologias de coleta e armazenamento de dados, tornou-se necessário o desenvolvimento de métodos estatísticos capazes de extrair informações de grande quantidade de dados. Mineração de dados (*data mining*) e aprendizado estatístico (*statistical learning*) são duas das principais áreas de pesquisa que lidam com o problema citado.

A mineração de dados trabalha com um enorme número de observações e número relativamente pequeno de variáveis (Berry & Linoff, 1997). O aprendizado estatístico, por sua vez, possui número muito maior de variáveis em relação ao seu tamanho amostral[1]. Dessa forma, as especificidades existentes em cada uma dessas áreas conduzem a diferentes desafios a serem enfrentados pelos pesquisadores. O maior problema enfrentado quando se trabalha com bases de dados com muitas observações (mineração de dados) decorre do fato de que os testes estatísticos tradicionais são pouco conservativos. Já a análise de conjuntos de dados superdimensionados compromete os métodos inferenciais que decorrem da Teoria Assintótica.

Segundo Theodoridis & Koutroumbas (2009), o aprendizado estatístico divide-se em supervisionado e não-supervisionado. O aprendizado supervisionado requer a existência de um conjunto de dados de treinamento, e o modelo é construído com base nessa informação *a priori* dos dados. Exemplos de técnicas estatísticas utilizadas para esse propósito podem ser encontradas em Hastie et al. (2009) e incluem: modelos lineares para regressão e classificação, métodos de suavização Kernel, redes neurais,

[1]Definição que, neste trabalho, recebe a nomenclatura de estrutura superdimensionada de dados.

máquinas de suporte vetorial, entre outros.

Quando o objetivo do estudo é obter informações sobre uma grande quantidade de dados, mas não há qualquer tipo de informação *a priori* sobre o problema, então trata-se do caso de um aprendizado não-supervisionado. Dessa maneira, as técnicas estatísticas que são utilizadas nessa área possuem uma finalidade mais descritiva do que inferencial, tais como: análise de agrupamento, técnicas de redução de dimensão (análise de componentes principais, análise de componentes independentes, escalonamento multimensional), entre outras. Aplicações e detalhes de cada área de pesquisa podem ser encontrados em Theodoridis & Koutroumbas (2009).

Este trabalho analisa uma série de algoritmos cujo objetivo se concentra na identificação de grupos de variáveis em uma estrutura de dados superdimensionada, com poucas observações e com replicações ao longo do tempo. Esses dados são referidos na literatura como HDLLSS (*High Dimensional Longitudinal Low Sample Size*) e podem ser encontrados em estudos de triagem agrícola (Wang, 2004), dados de expressão gênica (Gillespie et al., 2010), dados de sinais elétricos do cérebro (von Borries et al., 2011), entre outros.

A dificuldade em se agrupar ou classificar dados com as características mencionadas origina-se tanto da multidimensionalidade (grande número de variáveis) dos dados, quanto do reduzido tamanho amostral disponível. Além disso, há a necessidade de se incorporar ao processo de agrupamento a presença da correlação entre pontos no tempo, característica inerente aos estudos longitudinais. Por todos esses motivos, faltava na literatura um procedimento que fosse capaz de agrupar dados HDLLSS de maneira eficaz.

Com o intuito de preencher essa lacuna, von Borries (2008) desenvolveu um algoritmo, denominado PPCLUSTEL, que utiliza como medida de similaridade o p-valor resultante de um teste de ausência de efeito simples de níveis de fatores em um delineamento fatorial. Esse procedimento é indicado quando a estrutura dos dados apresenta um elevado número de variáves e número fixo de pontos no tempo.

Em 2008, Wang desenvolveu um algoritmo de agrupamento similar ao PPCLUSTEL, denominado PCLUST, mas indicado quando a estrutura dos dados apresenta grande número de pontos no tempo e número fixo de variáveis. Diferentemente do PPCLUSTEL, que é baseado nos valores originais dos dados, o teste de hipótese presente

no PCLUST utiliza os postos dos valores dos dados.

Em von Borries & Wang (2009) foi apresentado um algoritmo, denominado PPCLUST, destinado a agrupar variáveis em uma estrutura de dados superdimensionada e com amostras pequenas. Esse algoritmo também utiliza os postos dos valores dos dados e apresenta uma construção teórica semelhante ao algoritmo PPCLUSTEL, o que sugere a possibilidade de estender o PPCLUST para uma versão a ser utilizada em dados superdimensionados, longitudinais e com amostras pequenas. Desta forma, este trabalho apresenta dois novos algoritmos na literatura: o PPCLUSTEL - R, versão baseada em postos do PPCLUSTEL, e o PPCLUSTEL - M, uma adaptação do PCLUST. Por serem baseados em postos, tanto o PPCLUSTEL - R quanto o PPCLUSTEL - M possuem a propriedade de serem invariantes a transformações monótonas[2], como as transformações logarítmica e exponencial. O PPCLUSTEL - M é um procedimento que agrega o teste de hipótese presente no PCLUST (grande número de pontos no tempo e número fixo de variáveis) com o algoritmo de agrupamento baseado em partições presente no PPCLUSTEL.

Os algoritmos apresentados permitem agrupar variáveis que se comportam de maneira semelhante ao longo do tempo. A ideia básica de todos os procedimentos é agrupar as variáveis do sistema de tal forma que, ao final do procedimento, a variância para cada ponto no tempo seja grande entre grupos distintos e seja pequena dentro de um mesmo grupo. Para este fim, será utilizada no processo de agrupamento a metodologia de Análise de Variância (ANOVA) com delineamento fatorial e medidas repetidas no tempo.

Tendo em vista a necessidade de se obter um estimador consistente para a matriz de covariâncias da estrutura temporal dos dados e a dificuldade na realização de tal estimação quando o tamanho amostral é reduzido, são sugeridas três variações dos algoritmos PPCLUSTEL, PPCLUSTEL - R e PPCLUSTEL - M que fornecem uma solução alternativa para os problemas citados. A discussão do problema da estimação da matriz de covariâncias tem um papel de destaque neste trabalho. São propostas diferentes formas de sua estimação, e apresentadas as suposições e restrições inerentes a cada escolha possível.

[2]São transformações que preservam a ordem das observações.

Como forma de testar a eficácia dos testes de hipótese presentes nos algoritmos PPCLUSTEL e PPCLUSTEL-R, serão realizadas simulações do erro do tipo I e curvas de poder considerando-se os seguintes cenários:

Estrutura dos dados: São consideradas diferentes situações para o número de variáveis, pontos no tempo e observações;

Distribuição dos dados: Cauda leve (normal multivariada), assimétrica (log-normal multivariada) e cauda pesada ($t_{\nu=10}$ de *student* multivariada);

Estimação de Σ: São sugeridas duas formas de estimação da estrutura de variância e covariância (Σ e Σ_i).

A estrutura deste trabalho está dividida da seguinte maneira: no Capítulo 2, é realizada uma revisão de literatura das principais técnicas que têm sido utilizadas para agrupar dados HDLLSS. No Capítulo 3, são apresentados e avaliados por simulações os testes presentes nos algoritmos PPCLUSTEL, PPCLUSTEL-R e PPCLUSTEL-M. No Capítulo 4, esses algoritmos são avaliados em dados simulados e aplicados em dados de expressão gênica e de sinais elétricos do cérebro. Finalmente, no Capítulo 5, são apresentadas conclusões e é dado um panorama de trabalhos futuros.

Capítulo 2

Revisão de Literatura

Neste capítulo, será feita uma revisão das principais técnicas estatísticas que têm sido utilizadas para agrupar dados HDLLSS nas áreas de genética - especificamente em análise de dados de expressão gênica - e de reconhecimento de padrões (*pattern recognition*) em dados de sinais elétricos do cérebro. Na seção 2.1, é feita uma breve revisão das classes mais tradicionais de algoritmos de agrupamento. O objetivo é dar ao leitor uma visão geral dos problemas que ocorrem ao se aplicar as técnicas tradicionais de agrupamento em dados HDLLSS. Na seção 2.2, é apresentada uma medida que mensura a qualidade de um agrupamento, o ARI (*Adjusted Rand Index*). Essa medida será utilizada nas simulações deste trabalho.

Análise de Microarranjo

Uma aplicação que tem tido bastante destaque na literatura de biotecnologia é a análise de dados de expressão gênica, ou, simplesmente, análise de microarranjo. Inicialmente, um gene pode ser definido como sendo um segmento da molécula de DNA, cuja sequência possui toda informação necessária para a síntese de proteína. As proteínas, por sua vez, são as unidades que formam componentes estruturais, tecidos e enzimas necessárias para as reações bioquímicas básicas de um organismo. Cada molécula de DNA contém uma grande quantidade de genes. Estima-se que o genoma humano possua por volta de 30 mil genes.

As instruções para a codificação de proteínas contidas nos genes são transmitidas indiretamente através do ácido ribonucleico mensageiro (mRNA), uma molécula in-

termediária similar ao DNA, porém possuidora de apenas uma fita. Em laboratório, o mRNA pode ser isolado e usado como molde para sintetizar o DNA complementar (cDNA), que é em geral usado para localizar genes em um mapa cromossômico. Desta forma, a tecnologia de microarranjo permite o monitoramento simultâneo da atividade estimada de milhares de transcritos (sequências de RNA) via quantificação de mRNA. A maior parte dos transcritos correspondem a sequências integrantes de genes conhecidos, por isso, essas quantidades, após normalizações e processos de controle de qualidade, são também conhecidas como expressões gênicas. Detalhes da metodologia desses estudos podem ser encontrados em Human Genome Program (2003), Allison et al. (2006) e em Carvalho (2008).

Basicamente, a tecnologia de microarranjo pode ser utilizada em estudos transversais ou longitudinais no tempo. Em estudos transversais, são estimadas as expressões de milhares de transcritos em um único ponto no tempo. Tais dados possuem a característica de serem superdimensionados (considera-se cada transcrito como uma variável) e com poucas repetições, por limitações de tempo e custo. Em estudos longitudinais, além de se ter dados superdimensionados e com poucas repetições, ainda há que se considerar uma estrutura temporal nos dados, o que aumenta consideravelmente o nível de complexidade das análises estatísticas.

O estudo de dados provenientes de microarranjo vem permitindo aos cientistas entender os mecanismos biológicos de diversos organismos com um nível de detalhamento que nunca fora possível. Alguns trabalhos que utilizaram com êxito a tecnologia de microarranjo incluem a determinação do ciclo celular de fungos (Gillespie et al., 2010) e a indentificação de genes causadores da leucemia aguda (Yi et al., 2009), linfoma (Alizadeh et al., 2000), câncer de mama (Perou et al., 1999), entre outros.

Uma das técnicas estatísticas que têm sido empregadas com grande frequência nessa área é a Análise de Agrupamento. O principal objetivo da aplicação de técnicas de agrupamento em dados de microarranjo consiste em descobrir os grupos de genes envolvidos em determinados processos biológicos de um organismo, pois isso fornece aos geneticistas o insumo necessário para que eles conheçam as relações existentes entre os genes. Em geral, genes que são relacionados reagem de forma semelhante quando submetidos a determinado estímulo e espera-se que a técnica de agrupamento seja capaz de discernir genes com reações distintas e agrupar os que possuam funções

similares em determinado processo biológico.

As técnicas de agrupamento tradicionais (k-*means*, técnicas hierárquicas, mapas auto-organizáveis e técnicas difusas - *fuzzy*) têm sido frequentemente utilizadas em análise de dados genômicos. Entretanto, esses algoritmos ignoram a correlação temporal dos dados, e, além disso, requerem a pré-especificação do número de grupos existentes. Alguns trabalhos recentes que têm considerado a correlação entre pontos no tempo incluem técnicas como: métodos de suavização (Ma et al., 2006; Storey et al., 2005; Serban & Wasserman, 2005; Liu & Yang, 2009); métodos bayesianos de agrupamento (Ramoni et al., 2002; Ma et al., 2008) e modelos baseados em misturas de distribuições (Fraley & Raftery, 2002; McNicholas & Murphy, 2010).

Os métodos bayesianos de agrupamento são baseados em modelos autorregressivos e requerem estacionariedade e a validade da propriedade de Markov (Wang et al., 2008), o que geralmente não ocorre na prática. Além disso, modelos autorregressivos requerem que o número de pontos no tempo seja muito grande, e isso também não se verifica em dados de microarranjo por limitações de custo, conforme mencionado anteriormente.

Alguns procedimentos que apresentam bons resultados na literatura se baseiam na modelagem de misturas de distribuições normais aos dados. No Capítulo 4, será apresentado um algoritmo, desenvolvido por Fraley & Raftery (2006), que utiliza esse tipo de metodologia. Esse algoritmo, denominado MCLUST, incorpora várias formas diferentes para a estrutura de correlação temporal dos dados e estima o número de grupos automaticamente utilizando o Critério de Informação de Bayes (BIC[1]). O contraponto desse tipo de procedimento é que a qualidade do agrupamento depende fortemente da normalidade dos dados, suposição difícil de ser satisfeita em casos reais.

Outro interesse dos pesquisadores em experimentos com microarranjo longitudinal é indentificar os genes responsáveis pelo ciclo celular dos organismos. De acordo com os geneticistas, espera-se que as expressões de tais genes tenham um comportamento periódico, acompanhando o processo biológico de uma etapa do ciclo celular. Os estudos provenientes desse tipo de interesse são conhecidos pela literatura como Análise de Dados Funcionais (*Functional Data Analysis*)[2]. Em Wang et al. (2008),

[1]Sigla da expressão em inglês: *Bayesian Information Criterion*.

[2]Também é possível encontrar a nomenclatura: Dados de Ciclo Celular (*Cell Cicle Data*).

por exemplo, os autores observaram o comportamento genético de uma planta, ao longo do tempo, sob diferentes condições de estresse (frio e calor, secura e umidade) e aplicaram o agrupamento para identificar os genes reponsáveis pelas alterações das expressões gênicas ocorridas pelos diferentes estímulos que foram aplicados.

Dessa forma, os algoritmos desenvolvidos para agrupar dados funcionais precisam considerar a natureza cíclica das curvas que serão agrupadas. Uma das técnicas mais utilizadas para esse propósito envolve a análise de séries de Fourier (Kim & Kim, 2008). Entretanto, a principal falha dos algoritmos baseados em séries de Fourier para agrupar dados está em ignorar a dependência temporal dos dados, que é uma das características desse tipo de experimento. Em uma abordagem diferente, Ma & Zhong (2008) ajustaram um modelo linear de efeitos mistos aos dados e usaram um algoritmo que seleciona os grupos com base na diferença em relação à média do modelo ajustado. Os efeitos aleatórios foram utilizados para incorporar a correlação temporal dos dados. Os autores relatam que o método apresentou bons resultados quando os grupos diferiam apenas com relação à sua média, mas quando a estrutura de covariância variava entre os grupos, então o algoritmo tornou-se instável.

Grande parte dos algoritmos de agrupamento mencionados aloca todos os genes em grupos. Entretanto, em estudos de microarranjo, alguns genes apresentam características que não são relacionadas a nenhum grupo específico[3] e se esses genes forem erroneamente alocados em algum grupo, então o seu padrão de expressão genético pode ficar prejudicado (Tsai & Qu, 2008). Uma das características dos algoritmos apresentados neste trabalho é a possível detecção de genes esporádicos, o que permite que sejam formados apenas os grupos com padrão de expressão gênico mais homogêneo.

Diferentemente do que ocorre com os demais métodos existentes na literatura, as medidas de similaridade dos algoritmos PPCLUSTEL-R e PPCLUSTEL - M são invariantes a transformações monótonas nos dados. Portanto, o agrupamento aplicado em dados com valores originais ou log transformados não se altera. Também não há qualquer suposição sobre a distribuição dos dados, o que o caracteriza como um procedimento não-paramétrico e invariante a transformações monótonas nos dados.

[3]São conhecidos pela literatura como genes esporádicos (*sporadic genes*).

No entanto, a maior vantagem dos algoritmos apresentados é que eles são destinados a captar as alterações com relação à distribuição dos grupos amostrais, não se limitando a variações apenas com relação à média dos dados, como acontece na maioria dos métodos existentes. Isso é possível pelo fato de a matriz de covariâncias dos dados ser estimada para cada fator, ou para cada grupo, conforme será visto no próximo capítulo.

Análise de Dados de Sinais Elétricos do Cérebro

A eletroencefalografia é o estudo do registro gráfico dos potenciais elétricos produzidos pelo encéfalo, e o exame por meio do qual esses registros são captados é chamado de eletroencefalograma (EEG). As correntes elétricas produzidas pelo cérebro são captadas através de eletrodos, os quais são aplicados no couro cabeludo, na superfície encefálica, ou até mesmo dentro da substância encefálica.

A análise de dados de EEG teve início na década de 30 com os notáveis trabalhos do cientista alemão Hans Berger[4] (1929) e, desde então, as metodologias de diagnóstico e previsão de doenças como epilepsia, encefalites e distúrbios metabólicos por meio do exame de EEG melhoraram consideravelmente. No passado, a interpretação do EEG era limitada à inspeção visual realizada por um neurologista que, baseado em informações prévias do paciente, tentava verificar anormalidades no resultado do exame e efetuar o diagnóstico. Atualmente, com o avanço da tecnologia computacional e dos métodos estatísticos, é possível quantificar as mudanças de padrões dos dados de EEG com maior precisão e confiabilidade.

No âmbito das pesquisas com dados de EEG, o estudo de sinais elétricos do cérebro em portadores de epilepsia é uma das áreas que mais têm tido publicações nos últimos anos. Os cientistas buscam, nesses casos, uma metodologia capaz de diagnosticar a doença, monitorá-la e, principalmente, prever a ocorrência de novos ataques. O processo de análise de sinais de EEG segue uma sequência de etapas, conforme o disposto na Figura 2.1, cujo objetivo principal é classificar o sinal em categorias de interesse. Como em casos de epilepsia o interesse maior é prever a ocorrência de ataques, então as categorias de classificação são: EEG normal e EEG em condição de

[4]Hans Berger (1873-1941) foi um psiquiatra reconhecido por obter a primeira imagem gráfica das correntes elétricas do cérebro em suas pesquisas sobre a psicofisiologia dos estados anímicos.

pré-ataque.

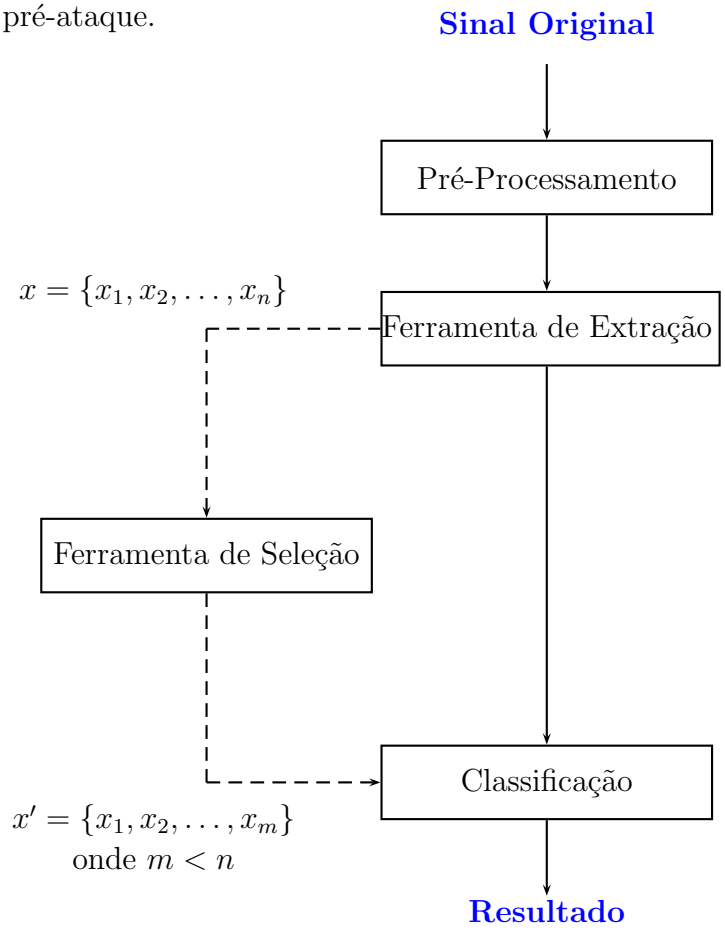


Figura 2.1: Os estágios básicos que envolvem um sistema de classificação de sinais de EEG.

Na etapa de pré-processamento, o sinal original passa por filtros para reduzir a quantidade de ruído resultante de interferências externas e internas (equipamento, eletrodos, etc.) ao experimento. Para explicar as técnicas utilizadas na etapa de **extração** é necessário compreender alguns conceitos de séries temporais. Uma série de tempo[5] pode ser caracterizada sob dois pontos de vista distintos: um considerando as informações contidas no domínio de tempo e outro no domínio de frequência (Cryer & Chan, 2009) dessa série. Por exemplo, o estudo das propriedades de autocorrelação de uma série faz parte do seu domínio de tempo. São exemplos de técnicas destinadas a analisar dados nesse domínio, os tradicionais modelos de análise de séries temporais descritas por Box & Jenkins (1976), como os modelos autorregressivos, integrados, de médias-móveis (ARIMA).

[5]Também será utilizada a nomenclatura de sinal, que são medições do potencial elétrico de uma unidade de estudo durante um determinado período, como os sinais de EEG.

De forma análoga, as técnicas que decorrem do estudo das propriedades de frequência de uma série de tempo fazem parte do seu domínio de frequência. O objetivo dessas técnicas é decompor o sinal em bandas de frequência minimizando a perda da informação contida no sinal original. A transformada de Fourier é um exemplo de uma técnica utilizada para esse propósito, onde a transformada possibilita transitar entre os domínios de tempo e frequência de uma série. A desvantagem desse método é que ao transitar do domínio de tempo para o domínio de frequência, perde-se qualquer informação referente ao domínio do tempo. Além disso, essa transformação é indicada apenas quando as séries são estacionárias (Subasi & Gursoy, 2010). A transformada de ondaletas (*wavelets*) é uma técnica que preserva tanto a informação do domínio de tempo, quanto a do domínio de frequências, além de ser indicada quando a série não suporta a condição de estacionariedade, como é o caso em EEG (Ocak, 2009). A análise do domínio de frequências de uma série temporal (análise espectral) é muito utilizada em áreas como econometria, acústica, engenharia de comunicação e ciências biomédicas.

Pelos motivos expostos, na etapa de extração dos dados, os autores geralmente utilizam a transformação de ondaletas nos dados para decompor o sinal em diferentes sub-bandas de frequência, facilitando o processo de classificação. A aplicação da transformada de ondaletas produz como saída um conjunto de coeficientes para cada sub-banda de frequência. Esses coeficientes são chamados de *vetores característicos* de um determinado sinal. Outra etapa presente em alguns trabalhos é a **seleção**, onde o objetivo é reduzir a dimensão dos vetores característicos que serão utilizados na **classificação**. Essa redução de dimensão visa amenizar possíveis redundâncias causadas pelo excesso de variáveis e para reduzir o tempo de processamento.

Os principais classificadores encontrados na literatura são os métodos que derivam de redes neurais artificiais (RNA), como o método MLPNN[6] (Orhan et al., 2011; Übeyli, 2009a, 2009b), o sistema de inferência neuro-*fuzzy* (Guler & Ubeyli, 2005), rede neural com função base radial (Aslan et al., 2008), o método LVQ[7] (Ocak, 2009), entre outros. Uma restrição dos métodos baseados em redes neurais artificiais é que o desempenho desse tipo de abordagem depende demasiadamente da técnica

[6] *Multilayer Perceptron Neural Network*

[7] *Learning Vector Quantization*

utilizada para extrair e reduzir a dimensão dos vetores característicos e do tamanho da amostra de treinamento. Máquinas de suporte vetorial (Subasi & Gursoy, 2010; Coutinho, 2011) e métodos de suavização (Kim et al., 2010) também são encontrados com frequência na literatura. Übeyli (2009) comparou o desempenho dos métodos baseados em redes neurais mais utilizados na literatura. Baseado nos resultados das comparações, o autor recomenda a utilização de máquinas de suporte vetorial para classificar sinais de EEG em epilépticos.

Neste trabalho, a classificação de um sinal de EEG será baseada nos grupos formados pelos algoritmos de agrupamento que serão apresentados. Em um primeiro momento, o método consiste em aplicar o algoritmo de agrupamento em um conjunto de sinais de EEG. Supondo que o procedimento aplicado seja eficaz em detectar o real padrão de grupos existentes na população, então é possível classificar um sinal desconhecido em alguns dos grupos formados, apenas repetindo o procedimento de agrupamento no conjunto de sinais de treinamento da etapa anterior, mas agora acrescentado o sinal desconhecido. A utilização da análise de agrupamento para classificar sinais de EEG limita-se na literatura à etapa de redução de dimensão do vetor característico, conforme realizado por Orhan et al. (2011), onde o autor utiliza o algoritmo *k-médias* nos coeficientes da transformada de ondaletas.

2.1 Métodos Tradicionais de Agrupamento

A análise de agrupamento consiste em um conjunto de técnicas exploratórias que desempenha um importante papel na compreensão de estruturas multivariadas de dados. Parte dessa compreensão ocorre por meio da identificação de grupos e padrões de comportamento das variáveis ou das observações em estudo.

As aplicações de agrupamento têm espaço em diversas áreas, tais como: nas ciências da vida (ex: biologia, zoologia); nas ciências sociais (ex: oncologia, sociologia, arqueologia); nas ciências médicas (psiquiatria, patologia); nas ciências da Terra (ex: geografia, geologia) e em muitos campos da engenharia (Anderberg, 1973). Tão diversos quanto as aplicações que o agrupamento possui são os objetivos sob os quais os pesquisadores utilizam essa técnica. Segundo Theodoridis & Koutroumbas (2009), existem quatro diretrizes básicas nas quais esse tipo de análise é utilizada.

São elas:

Redução dos dados: em muitos casos, a quantidade de dados disponíveis, N, é muito grande, o que torna sua análise e processamento difíceis. Nesses casos, a análise de agrupamento pode ser utilizada para agrupar os dados em um número representativo de grupos, m ($<< N$), e definir esses grupos como as novas unidades amostrais dos dados.

Estabelecimento de hipóteses: o usuário não possui um conhecimento prévio sobre os dados e aplica o agrupamento para verificar padrões nos dados e sugerir hipóteses de interesse. Essa é a ideia de se aplicar a análise de agrupamento em dados de microarranjo longitudinais, ou seja, aplicar uma técnica exploratória que permita encontrar padrões de comportamento gênico ao longo do tempo.

Testes de hipótese: o agrupamento é utilizado para verificação da validade de hipóteses específicas. Considere, por exemplo, a seguinte hipótese: "Profissionais mais bem remunerados falam mais de uma língua estrangeira". Aplicando-se a análise de agrupamento em um conjunto de dados representativo de trabalhadores é uma forma de verificar a veracidade dessa hipótese. Suponhamos que essa base de dados contenha informações sobre a remuneração, proficiência em mais de uma língua estrangeira e faixa etária, por exemplo. Se após a aplicação do algoritmo de agrupamento, um grupo for formado pelos profissionais com uma maior remuneração e que falam mais de uma língua (independentemente da sua faixa etária), então a hipótese foi validada pela análise de agrupamento.

Classificação baseada nos grupos: o agrupamento é realizado em um conjunto de dados e espera-se que os grupos formados tenham as características dos quais eles provieram. Dessa forma, caso se queira classificar um padrão desconhecido, então basta determinar em qual grupo ele teria mais chance de pertencer. Uma área que utiliza essa idéia de maneira intensiva é em reconhecimento de padrões (*Pattern Recognition*). Em uma das aplicações deste trabalho, utiliza-se a análise de agrupamento para classificar sinais de eletroencefalograma (EEG) seguindo a metodologia citada.

As técnicas de agrupamento tradicionais podem ser classificadas de acordo com a Figura 2.2. Basicamente, essas técnicas dividem-se entre procedimentos sólidos (*hard*) e difusos (*fuzzy*). A principal diferença entre elas é que, enquanto os procedimentos sólidos particionam a população em grupos disjuntos, onde cada elemento pertence somente a um grupo, as técnicas difusas podem alocar objetos em mais de um grupo possível, formando grupos com objetos em comum. Os procedimentos sólidos, por sua vez, podem ser classificados em métodos baseados em partição e métodos hierárquicos. Os primeiros buscam dividir de maneira ótima os objetos a serem alocados em um número fixo de grupos, enquanto que as técnicas hierárquicas, como o próprio nome sugere, seguem um processo onde os grupos formados em uma etapa do algoritmo dependem dos grupos formados na etapa anterior. Esse processo pode ocorrer ainda de forma divisiva (algoritmo inicia-se como um único grupo) ou aglomerativa (algoritmo inicia-se com n grupos).

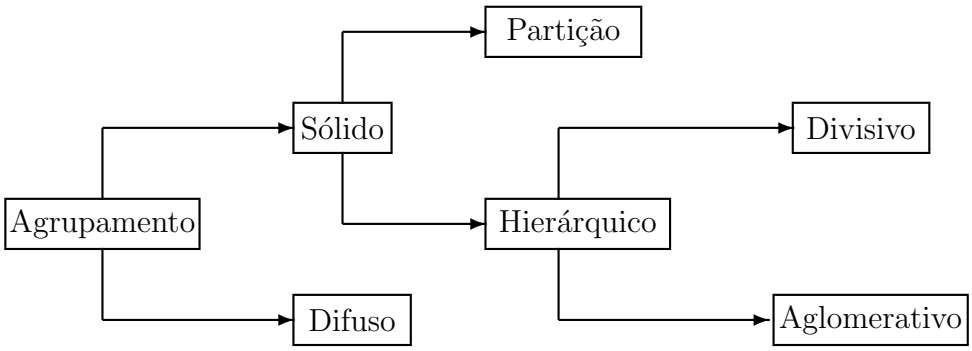


Figura 2.2: Diagrama dos algoritmos de agrupamento

Na seções 2.1.1 a 2.1.3 será feita uma breve descrição dos algoritmos mostrados na Figura 2.2. Na seção 2.2, é apresentada uma medida de qualidade de um agrupamento, chamada ARI (*Adjusted Rand Index*), que será utilizada nas simulações dos capítulos posteriores.

2.1.1 Técnicas hierárquicas

As técnicas de agrupamento hierárquicas podem ser classificadas em aglomerativas ou divisivas. Técnicas hierárquicas divisivas partem do princípio de que todos os elementos pertencem a um grupo e, no decorrer do processo, esses elementos são

separados de acordo com suas dissimilaridades, de modo que o processo seja finalizado quando cada elemento representa um conglomerado. Logo, se tivermos n elementos no início do processo, no final teremos n conglomerados.

Técnicas hierárquicas aglomerativas partem do pressuposto inverso. Inicialmente, cada elemento representa um conglomerado e, à medida que o processo de agrupamento se desenvolve, os elementos vão sendo agrupados até que todos os elementos estejam em um único conglomerado (*cluster*).

Em cada passo do algoritmo de agrupamento, os pares de elementos mais similares formam um novo conglomerado, de modo que apenas um novo conglomerado é formado em cada estágio. A propriedade da hierarquia é observada no processo porque cada novo conglomerado é um agrupamento de conglomerados formados anteriormente. Note que, uma vez unidos em um estágio do processo, dois elementos permanecerão no mesmo conglomerado em todos os estágios subsequentes.

Pelo fato de o processo parar quando há apenas um conglomerado contendo todos os elementos, no caso aglomerativo, ou cada elemento compor um conglomerado, no caso divisivo, a escolha do número final de grupos é muitas vezes subjetiva. Exemplos de técnicas hierárquicas:

Ligação Simples: a dissimilaridade entre dois grupos é expressa pela distância entre os dois vizinhos mais próximos. Vizinhos, aqui, são elementos pertencentes a conglomerados diferentes;

Ligação Completa: a dissimilaridade entre grupos é expressa pela distância entre os vizinhos mais distantes, ao contrário da Ligação Simples.

Método da Média das Distâncias: a dissimilaridade entre grupos é definida pela média das distâncias entre os pares de vizinhos.

Método de Ward: este método visa minimizar a perda de informação que ocorre ao juntar dois grupos. A perda de informação pode ser representada como um aumento da Soma dos Quadrados dos Erros (SQE), sendo o erro definido como o desvio de cada elemento à média do conglomerado (centróide). Se há a conglomerados, então

$$SQE = \sum_{i=1}^{a} \sum_{k=1}^{n_i} \left(X_{ik} - \bar{X}_k\right)' \left(X_{ik} - \bar{X}_k\right), \tag{2.1}$$

onde n_i é o número de elementos no i- ésimo grupo. Em cada estágio do processo, considera-se todos os pares de grupos possíveis, e a combinação escolhida é a que resulta no menor acréscimo de SQE ocorrido.

Método do Centróide: a distância entre os grupos é definida pela distância entre os centróides dos grupos que estão sendo comparados. Centróide é o ponto cujos componentes formam o vetor de médias.

Seja C_k o centróide do -ésimo grupo, então

$$C_k = \frac{X_1 + X_2 + \ldots + X_m}{m}, \tag{2.2}$$

onde m é o número de elementos no k-ésimo grupo.

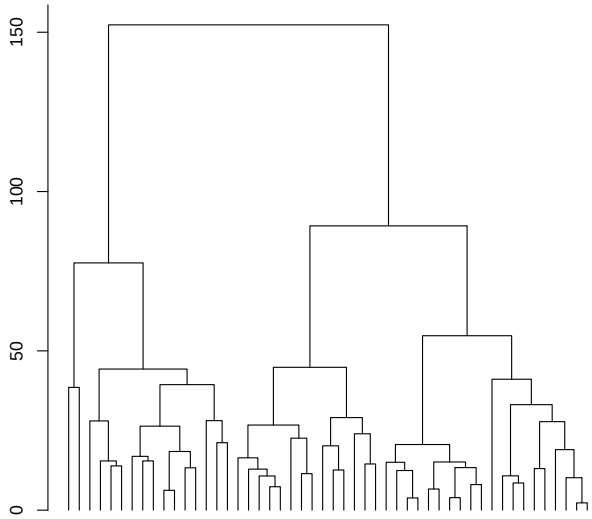


Figura 2.3: Dendograma aglomerativo obtido com o método da média das distâncias ($n = 50$).

O gráfico que exibe o andamento do processo de agrupamento hierárquico é o *Dendograma* (Figura 2.3). Quando a amostra é maior que determinado valor, esse

tipo de representação gráfica apresenta problemas de visualização. Reimann et al. (2008) indicam um limite amostral de $n < 100$ para sua melhor visualização.

Alguns algoritmos hierárquicos bastante utilizados são o AGNES (*Aglomerative Nesting*) e o DIANA (*Divisive Analysis*). Ambos estão disponíveis no pacote CLUSTER do *software* R. Detalhes sobre esses algoritmos podem ser encontrados em Kaufman & Rousseeuw (1990).

Um dos principais problemas dos algoritmos hierárquicos é causado pelo fato de que quando um objeto é erroneamente alocado em um determinado passo, ele não poderá ser realocado em passos subsequentes. Além disso, em agrupamentos de dados superdimensionados, o grande número de comparações necessárias para dois objetos formarem um grupo torna-os extremamente ineficientes em comparação a outros métodos.

2.1.2 Técnicas baseadas em partição

Técnicas de agrupamento baseadas em partição (*partitional clustering methods*) visam agrupar elementos por meio de um processo mais flexível, no sentido de que um item alocado inicialmente em um grupo possa ser realocado diversas vezes durante o andamento do processo de agrupamento. Essa é a principal característica que o difere dos métodos hierárquicos, onde essa flexibilidade ao longo do processo não é permitida.

Um dos algoritmos de partição mais utilizado é o k-médias (k-*means*). Proposto por MacQueen (1967), esse método consiste em dividir o conjunto de dados em k subconjuntos disjuntos, dado que k é um valor previamente fornecido pelo usuário. Obtém-se, então, o vetor de médias (centróides) $\hat{\mu}_i$, onde $i = 1, \ldots, k$ e aloca-se cada observação ao centróide mais próximo. Geralmente, utiliza-se a distância euclidiana para tal mensuração. O procedimento é repetido até que não haja rearranjos possíveis.

Uma vantagem desse algoritmo é sua simplicidade, podendo suportar bases de dados maiores do que as que são suportadas pelas técnicas hierárquicas. Entretanto, como utiliza-se da média amostral como medida para calcular os centróides, é demasiadamente afetado pela ocorrência de pontos discrepantes. Outro ponto negativo é que o número de grupos deve ser previamente fixado, o que de certa forma limita sua

aplicabilidade.

O algoritmo PAM (*Partitioning Around Medoids*) utiliza o mesmo processo do k-médias, mas em vez de calcular as médias dos subconjuntos, ele procura por objetos representativos, ou seja, objetos que possuam a menor dissimilaridade média para todos os outros objetos dentro de um mesmo protótipo de grupo. O fato de o PAM não utilizar as médias como centróides corrige a sensibilidade a pontos discrepantes presente no k-médias.

O ponto negativo de ambos os algoritmos baseados em patição aqui descritos é que quando utilizados em bases de dados que possuem muitas variáveis para serem agrupadas, como em dados de EEG, tornam-se extremamente lentos devido ao número de combinações possíveis para k grupos dentre milhares de sinais elétricos do cérebro.

Um algoritmo que procura resolver esse problema é denominado CLARA (*Clustering Large Applications*). Esse procedimento resume-se em dois passos básicos. Primeiramente, uma amostra aleatória simples é retirada dos dados e a ela aplica-se o algoritmo PAM, obtendo-se, portanto, k objetos representativos. Então, cada objeto não selecionado na amostra é alocado em um dos k objetos representativos àquele do qual esteja mais próximo. Após todos os elementos estarem alocados, obtém-se uma medida da qualidade do agrupamento calculando-se a distância média entre todos os objetos e seus respectivos objetos representativos. Depois de repetir todo esse processo cinco vezes é escolhido o agrupamento em que foi obtido a menor distância média.

Kaufman & Rousseeuw (1990) e Theodoridis & Koutroumbas (2009) detalham todos os algoritmos explicados nesta seção e apresentam ainda algumas variações dos mesmos. Segundo von Borries (2008), simulações têm mostrado que tais métodos continuam sendo lentos quando aplicados em dados superdimensionados como aqueles aos quais este trabalho se destina.

2.1.3 Técnicas difusas

Técnicas difusas (*fuzzy*) de agrupamento, ao contrário das técnicas sólidas (*hard*), que alocam os elementos no grupo mais próximo de acordo com algum critério (distância), fornecem a probabilidade de o objeto pertencer a cada um dos grupos. A soma das

probabilidades para cada objeto é igual a 1.

Assim como no método das k-médias, é necessário pré-especificar o número de grupos existentes nos dados. Em um banco de dados com n objetos a serem agrupados e g grupos, o objetivo do procedimento é minimizar a seguinte função (Mingoti, 2005):

$$J = \sum_{i=1}^{g} \sum_{j=1}^{n} (p_{ij})^{\lambda} d(X_j, V_i). \tag{2.3}$$

Sob a restrição de que

$$\sum_{i=1}^{g} p_{ij} = 1,$$

onde:

- V_i é o protótipo (semente ou centróide ponderado) do grupo i, $i = 1, \ldots, g$;
- $\lambda > 1$ é o parâmetro *fuzzy*;
- p_{ij} é a probabilidade de que o elemento X_j pertença ao grupo cujo protótipo é V_i;
- $d(.)$ é a distância escolhida pelo pesquisador (em geral, a distância euclidiana).

A função J é minimizada quando as probabilidades p_{ij} são iguais a:

$$p_{ij} = \left[\sum_{k=1}^{g} \left(\frac{d(X_j, V_i)}{d(X_j, V_k)} \right)^{\frac{2}{m-1}} \right]^{-1} \tag{2.4}$$

onde,

$$V_i = \frac{\sum_{j=1}^{n} (p_{ij})^{\lambda} \, X_j}{\sum_{j=1}^{n} (p_{ij})^{\lambda}} \tag{2.5}$$

O procedimento necessita das probabilidades p_{ij} iniciais e dos protótipos para dar início às iterações. De modo geral, cada valor inicial de p_{ij} é gerado por meio de uma distribuição uniforme entre 0 e 1.

Conforme o algoritmo se desenvolve, essas probabilidades vão se alterando, bem como os grupos determinados por cada protótipo, até que a diferença entre duas iteracões sucessivas seja menor que um valor predeterminado pelo usuário.

É interessante utilizar técnicas difusas, portanto, quando há suspeita de que alguns objetos são similares para mais de um grupo, isto é, estão nas adjacências de dois ou

mais grupos; ou até mesmo se determinado elemento não se assemelha a nenhum dos grupos. Cabe, portanto, ao pesquisador decidir se aloca cada objeto no grupo em que a probabilidade p_{ij} seja máxima ou se altera o número de grupos, por exemplo, para diminuir a incerteza na classificação.

Infelizmente, os pacotes computacionais que implementam esse algoritmo, como o FANNY (*Fuzzy Analysis Clustering*) do *software* R, tornam-se lentos e resultam em comparações difíceis de serem interpretadas quando aplicados em grandes bases de dados, como no caso de dados de EEG.

2.2 Avaliação da Qualidade do Agrupamento

Nos últimos 20 anos, o avanço das tecnologias de processamento de dados foi acompanhado de um significativo aumento na quantidade e variedade de algoritmos de agrupamento disponíveis. Tais avanços criaram a necessidade de se desenvolver medidas capazes de comparar o desempenho desses algoritmos e avaliar qual método é o mais apropriado.

Os métodos de avaliação da qualidade de um agrupamento dividem-se entre medidas internas e medidas externas. As *medidas internas* avaliam a variabilidade existente dentro de cada grupo e entre os grupos e são utilizadas quando não se sabe a correta estrutura dos grupos, como acontece em casos práticos. Em estudos de simulação, onde a correta estrutura dos grupos é conhecida, são utilizadas as chamadas *medidas externas*.

Uma medida externa que será extensivamente utilizada neste trabalho é o Índice de Rand ajustado, ou simplemente ARI. Criada por Hubert & Arabie (1985), essa medida é uma correção do RI (*Rand Index*, de Rand, 1971). O RI é uma medida que assume valores entre 0 e 1, mas caso os grupos estejam aleatoriamente dispostos, seu valor esperado não é igual a 0. Essa limitação impede a comparação de conjuntos de dados diferentes, uma vez que cada conjunto terá um respectivo valor esperado. Com isso, o trabalho de Hubert & Arabie (1985) foi adaptar o RI para que este tenha o valor esperado zero.

Para explicar como o ARI é construído, considere, inicialmente, um conjunto contendo todos os n objetos a serem agrupados, $S = \{O_1, O_2, \ldots, O_n\}$, e suponha que

$R = \{r_1, r_2, \ldots, r_k\}$ e $V = \{v_1, v_2, \ldots, v_c\}$ representem duas partições[8] dos objetos de S, de tal forma que $\cup_{i=1}^{k} r_i = S = \cup_{j=1}^{c} v_j$ e $r_i \cap r_{i'} = \emptyset = v_j \cap v_{j'}$ para $1 \leq i \neq i^{'} \leq r$ e $1 \leq j \neq j^{'} \leq c$. Em estudos de simulação, onde a real estrutura dos grupos é conhecida, a partição R representa os k grupos de referência que foram gerados no estudo, e V representa a partição onde foram obtidos c grupos por meio de algum algoritmo de agrupamento que se esteja interessado em estudar (k-médias, PAM, etc.). A Tabela 2.1 de contingência mostra os elementos em comum em cada grupo das duas partições em análise.

Partição		V				
	Grupo	v_1	v_2	$\ldots$	v_c	Total
R	r_1	n_{11}	n_{12}	$\ldots$	n_{1c}	$n_{1.}$
	r_2	n_{21}	n_{22}	$\ldots$	n_{2c}	$n_{2.}$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
	r_k	n_{k1}	n_{k2}	$\ldots$	n_{kc}	$n_{k.}$
Total		$n_{.1}$	$n_{.2}$	$\ldots$	$n_{.c}$	n

Tabela 2.1: Comparação das partições R e V.

Na Tabela 2.1, o valor de entrada n_{ij} representa o número de objetos que foram classificados tanto no grupo r_i, como no grupo v_j, com $i = 1, \ldots, k$, $j = 1, \ldots, c$ e $n_{i.} = \sum_{j=1}^{c}$, $n_{.j} = \sum_{i=1}^{k}$. Como forma de exemplificar a utilização dessa tabela, considere o seguinte conjunto $S = \{1, 2, 3, 4, 5, 6\}$ contendo os 6 objetos a serem agrupados. Agora, defina as partições $R = \{(1, 2, 3), (4, 5, 6)\}$ e $V = \{(4, 5), (2, 6), (1, 3)\}$. A Tabela 2.2 mostra a sobreposição de grupos dessas partições.

Do número total de combinações possíveis dois a dois entre os n elementos do conjunto S, podem existir quatro tipos diferentes de pares, são eles:

A - Par de objetos está alocado no mesmo grupo em R e em V;

B - Par de objetos está alocado no mesmo grupo em R e em grupos diferentes em V;

C - Par de objetos está alocado no mesmo grupo em V e em grupos diferentes em R;

[8]São subconjuntos disjuntos de um conjunto de dados qualquer.

Partição		V			
	Grupo	v_1	v_2	v_3	Total
R	r_1	0	1	2	3
	r_2	2	1	0	3
Total		2	2	2	6

Tabela 2.2: Exemplo de tabela de contingência utilizada para o cálculo do ARI.

D - Par de objetos está alocado em grupos diferentes em R e em V;

Essa decomposição das combinações dos pares de objetos conduz a uma representação alternativa da Tabela 2.1, baseada nos valores A, B, C e D, conforme mostra a Tabela 2.3.

Partição	V	
R	Par em um mesmo grupo	Par em grupos diferentes
Par em um mesmo grupo	A	B
Par em grupos diferentes	C	D

Tabela 2.3: Tabela de contingência 2×2 simplificada.

Os valores das quatro células da Tabela 2.3 podem ser calculados utilizando a notação apresentada na tabela de contingência (Santos & Embrechts, 2009), de acordo com as seguintes equações em (3.5).

$$A = \left(\sum_{i=1}^{k} \sum_{j=1}^{c} n_{ij}^2 - n \right) /2 \qquad B = \left(\sum_{i=1}^{k} n_{i.}^2 - \sum_{i=1}^{k} \sum_{j=1}^{c} n_{ij}^2 \right) /2 \tag{2.6a}$$

$$C = \left(\sum_{j=1}^{c} n_{.j}^2 - \sum_{i=1}^{k} \sum_{j=1}^{c} n_{ij}^2 \right) /2 \qquad D = \binom{n}{2} - a - b - c. \tag{2.6b}$$

Voltando ao exemplo, observe agora que os valores da tabela de contingência 2×2 são:

Ou seja, os dois pares que estão em um mesmo grupo tanto em R quanto em V são $A = \{(1,3), (4,5)\}$. Os quatro pares que estão no mesmo grupo de R, e em grupos

Partição	V	
R	Par em um mesmo grupo	Par em grupos diferentes
Par em um mesmo grupo	2	4
Par em grupos diferentes	1	8

diferentes de V são $B = \{(1,2), (2,3), (4,6), (5,6)\}$, e assim por diante. O Índice de Rand (RI) é calculado da seguinte forma:

$$RI = \frac{A + D}{A + B + C + D}. \tag{2.7}$$

Basicamente, como se pode perceber, essa medida se baseia no número de objetos que foram alocados juntos e separados tanto em R quanto em V. Conforme citado anteriormente, essa medida possui alguns problemas, o valor esperado do RI quando são comparadas duas partições aleatórias não é igual a zero. Além disso, o RI se aproxima de 1 à medida que o número de grupos das partições aumenta. Como forma de corrigir esse problema, Hubert & Arabie (1985) propuseram a seguinte alteração no índice:

$$ARI = \frac{RI - E(RI)}{\max(RI) - E(RI)}, \tag{2.8}$$

onde o valor esperado é calculado com base na distribuição hipergeométrica generalizada. Dessa forma, o ARI pode ser calculado por

$$ARI = \frac{\binom{n}{2}(A + D) - [(A + B)(A + C) + (C + D)(B + D)]}{\binom{n}{2}^2 - [(A + B)(A + C) + (C + D)(B + D)]} \tag{2.9}$$

ou

$$ARI = \frac{\sum_i \sum_j \binom{n_{ij}}{2} - \frac{\sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2}}{\binom{n}{2}}}{\frac{1}{2}\left[\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}\right] - \frac{\sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2}}{\binom{n}{2}}}. \tag{2.10}$$

O ARI calculado para o exemplo desta seção foi igual a 0,24. Assim como o RI, a medida em (2.10) varia entre 0 e 1, onde 1 indica que todos os objetos foram alocados

corretamente e 0 indica que os elementos foram alocados de forma aleatória, ou seja, que o agrupamento não foi eficaz em detectar os grupos existentes na população.

Milligan & Cooper (1986) realizaram um estudo onde diversos índices existentes para comparações de duas partições dos dados foram avaliados. Os autores recomendam fortemente o ARI como a medida que obteve os melhores resultados.

A macro SAS© para calcular o ARI, implementada por von Borries (2008), foi utilizada nas simulações deste trabalho.

Capítulo 3

Agrupamento de Dados HDLLSS com Base em p -Valores

3.1 Introdução

Suponhamos que se tenha observações de uma mistura de distribuições desconhecidas e que um grupo seja formado por todas as observações geradas a partir de uma mesma distribuição dessa mistura. As diferenças entre os grupos podem ser percebidas por meio de seus valores médios ou variâncias diferentes. Nos métodos apresentados neste capítulo, o problema de agrupamento é tratado como uma maneira de detectar uma diferença significativa na distribuição das observações de cada mistura.

Considere $X_{ijk} \sim F_{ij}(x)$ representando a observação k da variável i medida no tempo j, onde $i = 1, \ldots, a$, $j = 1, \ldots, b$ e $k = 1, \ldots, n_i$, com n_i sendo o número de replicações da variável i. Neste trabalho, n_i é assumido sempre pequeno. Os dados observados podem ser vistos como uma matriz com elementos X_{ijk}, conforme mostra a Tabela 3.1. No presente estudo, será tratado o problema de agrupar as variáveis desse tipo estrutura, considerando $a \to \infty$ com b fixo, e $b \to \infty$ com a fixo.

Para testar se os grupos são diferentes, basta verificar se as distribuições variam entre os grupos em algum ponto no tempo. A hipótese em estudo pode ser definida da seguinte forma:

$$H_0 : F_{ij}(x) = B_j(x), \tag{3.1}$$

para todo $i = 1, \ldots, a$ e $j = 1, \ldots, b$.

Como forma de testar a hipótese (3.1), Wang (2004) desenvolveu um teste não-paramétrico equivalente, que é baseado em um modelo linear de análise de variância com delineamento fatorial. Esse procedimento é similar ao usual teste F de análise de variância, porém a peculiaridade é sua aplicação em dados com a estrutura HDLLSS. A convergência assintótica da estatística desse teste ocorre quando $b \to \infty$ e o número de variáveis permanece fixo.

Seguindo a mesma ideia de testar homogeneidade de distribuições em dados super-dimensionados, longitudinais e com amostra pequena, von Borries (2008) estendeu o teste de Wang (2004) para o caso em que o número de variáveis é alto ($a \to \infty$), mas com o número de pontos no tempo fixo. Com o intuito de agrupar variáveis em dados que possuam a estrutura HDLLSS, von Borries (2008) desenvolveu um algoritmo que utiliza o p-valor resultante desse teste de hipótese como medida de similaridade no processo de agrupamento. Esse algoritmo, denominado PPCLUSTEL[1], particiona os dados sempre que o p-valor resultante do teste é menor que determinado limiar (*threshold*) pré-especificado pelo usuário.

Este trabalho propõe um novo método na literatura, chamado de agora em diante de PPCLUSTEL - M (do inglês: *modified*). O PPCLUSTEL - M propõe utilizar o p-valor do teste desenvolvido por Wang (2004) como medida de similaridade no algoritmo de agrupamento baseado em partições de von Borries (2008). Com isso, enquanto o PPCLUSTEL é indicado para agrupar dados com $a \to \infty$, amostra pequena e um número fixo de pontos no tempo; o PPCLUSTEL - M é indicado para o caso oposto, no qual os dados possuam grande número de observações no tempo ($b \to \infty$) e número fixo de variáveis.

Uma vantagem do PPCLUSTEL - M em relação ao PPCLUSTEL é que o teste foi desenvolvido para dados em postos e isso o permite ser invariante a transformações monótonas. Com o intuito de fazer com que o PPCLUSTEL seja robusto às situações em que os dados seguem distribuições assimétricas, é proposta, ainda, uma versão baseada em postos do PPCLUSTEL, o PPCLUSTEL - R (do inglês: *ranked*). O PPCLUSTEL - M utiliza o p-valor do teste desenvolvido por Wang (2004) como

[1] *p - Values Based Partitional Clustering of Longitudinal Data.*

Variável	Observação	Tempo			
		t_1	t_2	$\ldots$	t_b
1	1	X_{111}	X_{121}	$\ldots$	X_{1b1}
	2	X_{112}	X_{122}	$\ldots$	X_{1b2}
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	n_1	X_{11n_1}	X_{12n_1}	$\ldots$	X_{2bn_1}
2	1	X_{211}	X_{221}	$\ldots$	X_{2b1}
	2	X_{212}	X_{222}	$\ldots$	X_{2b2}
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	n_2	X_{21n_2}	X_{22n_2}	$\ldots$	X_{2bn_2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
a	1	X_{a11}	X_{a21}	$\ldots$	X_{ab1}
	2	X_{a12}	X_{a22}	$\ldots$	X_{ab2}
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	n_a	X_{a1n_a}	X_{a2n_a}	$\ldots$	X_{abn_a}

Tabela 3.1: Estrutura dos dados HDLLSS.

medida de similaridade no algoritmo de agrupamento baseado em partições de von Borries (2. Portanto, percebe-se que tais procedimentos diferem entre si apenas com relação às suas respectivas medidas de similaridade (representadas pelo p-valor), pois eles compartilham o mesmo algoritmo de agrupamento.

Este capítulo está dividido da seguinte maneira: O teste do PPCLUSTEL será apresentado na Seção 3.2, onde suas propriedades serão avaliadas, sob diferentes condições, por meio de simulações do erro do tipo I e curvas de poder. Ainda nessa seção, são feitas algumas considerações acerca das maneiras nas quais se calcula a matriz de covariâncias da estrutura temporal dos dados (necessária para se obter o p-valor do teste). Na Seção 3.3, o teste presente no PPCLUSTEL-R também será estudado por simulações para verificar se a convergência assintótica do teste do PPCLUSTEL continua válida com os dados originais sendo substituídos pelos seus respectivos postos. Na Seção 3.4, é apresentado o teste desenvolvido por Wang (2004) e é realizada uma comparação do PPCLUSTEL - M com o método de agrupamento proposto pela

autora, o PCLUST (Wang, 2008). Finalmente, na Seção 3.5, é apresentado o algoritmo de agrupamento por partição que foi desenvolvido por von Borries (2008) e que será utilizado neste trabalho. A Tabela 3.2 mostra os algoritmos apresentados neste capítulo.

Procedimento	Dados	Algoritmo	Condição de Convergência
PPCLUSTEL	Originais	von Borries (2008)	$a \to \infty$ e b fixo
PPCLUSTEL-R	Postos	von Borries (2008)	$a \to \infty$ e b fixo
PCLUST	Postos	Wang (2008)	$b \to \infty$ e a fixo
PPCLUSTEL-M	Postos	von Borries (2008)	$b \to \infty$ e a fixo

Tabela 3.2: Características dos algoritmos.

3.2 Estudo do Teste Presente no PPCLUSTEL (b fixo e $a \to \infty$)

Nesta seção é apresentado o teste desenvolvido por von Borries (2008), que fornecerá a medida de similaridade do algoritmo de agrupamento PPCLUSTEL. Considere, inicialmente, o problema de se ajustar aos valores de entrada da Tabela 3.1, denotado por X_{ijk}, o seguinte modelo

$$X_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{ijk}, \tag{3.2}$$

onde $i = 1, \ldots, a$; $j = 1, \ldots, b$; $k = 1, \ldots, n_i$ com $E(\epsilon_{ijk}) = 0$ e $X_{ijk} \sim F_{ij}$ arbitrário. Vale ressaltar a inexistência de qualquer tipo de restrição quanto à distribuição dos dados. Portanto, percebe-se que as variáveis da estrutura apresentada na Tabela 3.1 serão representadas pelo i - ésimo nível do fator do modelo (3.2).

Uma vez definido o modelo a ser ajustado aos dados, então é possível utilizar o teste de ausência de efeito simples de fator desse modelo como forma equivalente para testar a hipótese 3.1. A hipótese não-paramétrica de ausência de efeito simples de fator do modelo (3.2) pode ser escrita como

$$H_0(\phi) : \phi_{ij} = \alpha_i + \gamma_{ij} = 0, \text{para todo } i \text{ e } j. \tag{3.3}$$

onde a usual restrição $\sum_{i=1}^{a} \alpha_i = \sum_{j=1}^{b} \beta_j = \sum_{i=1}^{a} \gamma_{ij} = \sum_{j=1}^{b} \gamma_{ij} = 0$ se faz necessária. Dessa maneira, daqui por diante será com base na hipótese 3.3 que testamos a homogeneidade de distribuições ao longo do tempo.

Quanto à estrutura de covariância dos dados, assuma que

$$cov(X_{ijk}, X_{i'j'k'}) = \begin{cases} \sigma_{ijj'} & \text{se } i = i',\ k = k' \\ 0 & \text{se } i \neq i',\ k \neq k', \end{cases} \tag{3.4}$$

o que implica que observações para o mesmo nível do fator e sujeito são correlacionadas, enquanto observações para níveis e/ou diferentes sujeitos são não correlacionadas. Ou seja, a estrutura de covariância descreve somente o comportamento temporal dos dados.

Considere, ainda, a usual notação utilizada para as estatísticas em análise de variância:

$$\bar{X}_{ij.} = \frac{1}{n_i} \sum_{k=1}^{n_i} X_{ijk} \qquad \tilde{X}_{.j.} = \frac{1}{a} \sum_{i=1}^{a} \bar{X}_{ij.} \tag{3.5a}$$

$$\bar{X}_{...} = \frac{1}{N} \sum_{i=1}^{a} \sum_{j=1}^{b} \sum_{k=1}^{n_i} X_{ijk} \qquad \tilde{X}_{...} = \frac{1}{ab} \sum_{i=1}^{a} \sum_{j=1}^{b} \bar{X}_{ij.}, \tag{3.5b}$$

onde $N = b \sum_{i=1}^{a} n_i$.

Agora, defina a estatística da soma dos quadrados médios de tratamento,

$$MS\varphi = \frac{1}{(a-1)b} \sum_{i=1}^{a} \sum_{j=1}^{b} (\bar{X}_{ij.} - \tilde{X}_{.j.})^2, \tag{3.6}$$

e da soma dos quadrados médios do erro

$$MSE = \frac{1}{ab} \sum_{i=1}^{a} \sum_{j=1}^{b} \sum_{k=1}^{n_i} \frac{(X_{ijk} - \bar{X}_{ij.})^2}{n_i(n_i - 1)}. \tag{3.7}$$

von Borries (2008) demonstra em seu trabalho que, sob a condição de que o número de níveis de fatores tende ao infinito ($a \to \infty$) e o número de pontos no tempo é fixo, a distribuição amostral da estatística $F_\phi = \sqrt{ab}(MS\varphi - MSE)$ converge em distribuição para a distribuição gaussiana. O resultado é descrito pelo seguinte Teorema:

Teorema 3.1. *(Teste de ausência de efeito simples de um grupo de níveis do fator.)*

Seja $X_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{ijk}$, *onde* μ *é o efeito geral,* α_i, $i = 1, \ldots, a$, *o efeito médio do fator,* β_j, $j = 1, \ldots, b$; *o efeito do ponto no tempo,* γ_{ij} *o efeito interação fator-tempo e* ϵ_{ijk} *algum erro com distribuição arbitrária* F_{ij}, *para todo* $k = 1, \ldots, n_i$.

Seja também, $H_0(\phi) : \phi_{ij} = \alpha_i + \gamma_{ij} = 0$ *satisfeita. Se as observações* X_{ijk} *têm momento central finito* $(2+\delta)(\delta > 0)$, *e o número de repetições é pequeno, com* $n_i \geq 2$ *e limitado, observado para um número fixo* b *de pontos no tempo,*

$$F_\phi = \sqrt{ab}(MS\varphi - MSE) \xrightarrow{d} N\left(0, \lim_{a\to\infty} \frac{2}{ab} \sum_{i=1}^{a} \frac{1}{n_i(n_i-1)} \sum_{j=1}^{b} \sum_{j'=1}^{b} \sigma^2_{ijj'}\right) \tag{3.8}$$

com $a \to \infty$.

3.2.1 A Estimação da Matriz Σ

Um aspecto importante deste trabalho é que, conforme será verificado nas simulações das próximas seções, a qualidade do agrupamento está diretamente relacionada à correta estimação dos parâmetros de covariância da estrutura temporal do Modelo (3.2). Isso acontece porque para se obter o p-valor do Teste (3.3) é necessário estimar a variância assintótica da estatística F_ϕ, ou seja, $\sigma^2_{ijj'}$ para todo $i \neq i'$. Mas, como definido na Equação (3.4), $cov(X_{ijk}, X_{i'j'k'}) = \sigma_{ijj'}$, o que significa que devemos estimar o quadrado da matriz de covariâncias usual, elemento a elemento.

Resultado 3.1. *Seja a amostra aleatória* $x_1, \ldots, x_n$, *o estimador*

$$\sigma^4_n = \left(\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}\right)^2,$$

e $\sigma^4_{n(i)}$, *com* $i = 1, \ldots, n$, *sendo calculado como* σ^4_n, *mas retirando-se a observação* x_i *da amostra. O estimador* $jackknife$ *de* σ^4, $\hat{\sigma}^4_{jack}$, *é dado por,*

$$\hat{\sigma}^4_{jack} = n\sigma^4_n - \frac{n-1}{n} \sum_{i=1}^{n} \sigma^4_{n(i)}. \tag{3.9}$$

De acordo com Pawitan (2001), o estimador (3.9) possui um viés menor que σ^4_n, motivo pelo qual ele foi utilizado no cálculo das variâncias e covariâncias $\sigma^2_{ijj'}$.

O maior problema da estimação de $\sigma^2_{ijj'}$ é a limitação do número de replicações (n_i) disponíveis para cada nível de fator, o que impossibilita a estimação precisa da matriz de covariância. Para se ter uma ideia, com apenas 5 pontos no tempo é necessária a estimação de 15 parâmetros de covariância[2].

Uma possível solução para esse problema é supor algumas condições sobre o comportamento dessa estrutura. Considerando que a estrutura de covariância temporal seja a mesma em todos os níveis de fatores, ou seja, $\sigma_{1jj'} = \sigma_{2jj'} = \ldots = \sigma_{ajj'}$, então é possível utilizar todas as $N_a = \sum_{i=1}^{a} n_i$ observações para estimar apenas uma única matriz $\sigma_{ijj'} = \sigma_{jj'}, \forall\, i$. Essa suposição resolve o problema do tamanho amostral, pois como N_a depende do número de níveis de fator e que, por sua vez, é sempre um número alto ($a \to \infty$), então há a garantia de informação suficiente para a estimação de $\sigma_{jj'}$. Entretanto, cria-se um problema diferente, pois caso os grupos difiram entre si com relação à estrutura de covariância, então o algoritmo não conseguirá captar tal diferença, uma vez que as estimativas foram fixadas para cada nível de fator.

Portanto, existem duas possibilidades para a maneira em que a estrutura de covariância se expressa nos dados: uma é essa estrutura ser diferente para cada fator, implicando na estimação de a matrizes Σ_i (representação matricial de $\hat{\sigma}_{ijj'}$), e a outra é a estrutura ser a mesma para todos os fatores, implicando na estimação de apenas uma matriz Σ para todo o conjunto de dados. Vale lembrar que a matriz Σ representa a estrutura temporal dos dados, portanto sua dimensão é dada pelo número de pontos no tempo (b).

Neste trabalho, considera-se ambas as possibilidades em todas as etapas de avaliação do desempenho do Teste (3.3). A escolha da maneira correta de representar os dados em casos práticos dependerá do conhecimento do pesquisador sobre o problema. Caso haja indícios de que o comportamento dos fatores ao longo do tempo seja igual para todos os fatores, então utiliza-se Σ; se esse comportamento for diferente para cada fator, utiliza-se Σ_i. A possibilidade de escolha da forma de estimação da matriz de covariâncias permite ao usuário mais flexibilidade em encontrar o algoritmo que se aplique melhor às suas necessidades. No Capítulo 4, será proposto ainda um terceiro método de se estimar Σ que visa solucionar tanto o problema do tamanho amostral,

[2]Como a matriz de covariância é simétrica, então o número de parâmetros distintos dessa matriz é igual a $b \times (b+1)/2$.

quanto o de detectar mudanças significativas entre as matrizes de covariâncias dos grupos.

3.2.2 O Erro do Tipo I

Nesta seção, será estudado o erro do tipo I do Teste (3.3). Esse tipo de estudo permite que se tenha uma estimativa do erro que o algoritmo pode cometer ao realizar o procedimento de agrupamento. Especificamente, o que será avaliado nas simulações é a chance de o algoritmo detectar a existência de ao menos um nível de fator com distribuição distinta das demais, quando na realidade todos os níveis seguem a mesma distribuição.

Além do objetivo citado, outra razão que justifica tal análise é a necessidade de saber quantos níveis de fatores e quantos pontos no tempo são necessários para que a convergência em (3.8) ocorra. De acordo com a condição exigida pelo teste, espera-se que quanto maior o valor de a e menor o de b, melhor será o resultado. Como forma de verificar essa afirmação, foram considerados os seguintes cenários nas simulações:

Número de níveis do fator (a): 20, 100, 500 e 1000;

Número de pontos no tempo (b): 5, 10, 20 e 40;

Número de observações (n_i): 3, 5, 10 e 20;

Nível nominal: 1%, 5% e 10%.

Como o teste não requer nenhuma especificação sobre a distribuição dos dados, também foram consideradas simulações com as distribuições normal multivariada, log-normal multivariada e t de *student* multivariada com 10 graus de liberdade para verificar como o teste se comporta quando os dados possuem assimetria (log-normal) e cauda pesada (t de student).

Antes de os resultados serem apresentados, faremos uma breve descrição das distribuições que serão utilizadas nas simulações. Considere, inicialmente, o seguinte vetor aleatório p - dimensional:

$$\mathbf{X}' = [X_1, X_2, \ldots, X_p] \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

onde $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ é a notação usual da normal multivariada com média $\boldsymbol{\mu} \in \mathbb{R}^p$ e matriz de covariâncias $\boldsymbol{\Sigma} = (\sigma_{ij})$. Ao efetuarmos a transformação $\mathbf{Y} = \exp(\mathbf{X})$, obtemos que o vetor aleatório $\mathbf{Y}$ segue a distribuição log-normal multivariada, cuja função densidade é dada por:

$$f(\mathbf{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{p}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \left(\prod_{i=1}^{p} y_i \right)^{-1} \exp\left[-\frac{(\log \mathbf{y} - \boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\log \mathbf{y} - \boldsymbol{\mu})}{2} \right], \quad (3.10)$$

onde log é o operador do logaritmo neperiano. O valor esperado e a variância do vetor aleatório $\mathbf{Y}$ são dados por:

$$E(\mathbf{Y}) = \boldsymbol{\kappa}' = [\kappa_1, \kappa_2, \ldots, \kappa_p] \quad \kappa_i = \exp(\mu_i + \frac{\sigma_{ii}}{2}), \quad (3.11)$$

$$cov(\mathbf{Y}) = \boldsymbol{\Phi} = (\phi_{ij}) \qquad \phi_{ij} = \exp\left[(\mu_i + \mu_j) + (\sigma_{ii} + \sigma_{jj})/2\right] \left[\exp(\sigma_{ij}) - 1\right].$$

Agora, seja o vetor aleatório $\mathbf{Z}' = [Z_1, Z_2, \ldots, Z_p]$ seguindo a distribuição t de student multivariada com $\nu > 0$ graus de liberdade (g.l.), média $\boldsymbol{\lambda} \in \mathbb{R}^p$ e parâmetro de escala $\boldsymbol{\Phi}$, uma matriz $p \times p$ simétrica e positiva-definida com diagonais unitárias e elementos ρ_{ij} (correlação linear entre os vetores Z_i e Z_j). Se a variável aleatória $\nu S^2/\sigma^2$ tem distribuição qui-quadrado com ν graus de liberdade, então a função densidade de probabilidade da distribuição $t_p(\nu, \boldsymbol{\lambda}, \boldsymbol{\Phi})$ pode ser obtida efetuando-se a seguinte transformação:

$$\mathbf{Z} = S^{-1}(\mathbf{X} - \boldsymbol{\mu}) + \boldsymbol{\lambda} \quad (3.12)$$

O valor esperado e a variância do vetor aleatório $\mathbf{Z}$ são:

$$\text{se } \nu \geq 2, \qquad E(\mathbf{Z}) = \boldsymbol{\lambda}' = [\lambda_1, \lambda_2, \ldots, \lambda_p], \quad (3.13a)$$

$$\text{e se } \nu \geq 3, \qquad var(\mathbf{Z}) = \frac{\nu}{\nu - 2}, \quad (3.13b)$$

$$cov(Z_i, Z_j) = \frac{\nu}{\nu - 2} \rho_{ij}, \quad \forall i, j, \ i \neq j. \quad (3.13c)$$

A geração dos vetores b-dimensionais $X_{ik} = (X_{i1k}, X_{i2k}, \ldots, X_{ibk})$ com as três distribuições multivariadas foram realizadas no SAS/IML, versão 9.2. A normal e t

multivariadas foram geradas com os módulos RANDNORMAL e RANDMVT, respectivamente. Para obter vetores com distribuição log-normal multivariada, conforme visto, bastou aplicar a transformação exponencial nos vetores aleatórios com distribuição normal multivariada.

Os componentes do vetor de médias foram iguais a $\mu_j = \cos(\pi\ (j+1)),\ j = 1, \ldots, b$. Os componentes da matriz de covariâncias Σ possuem variância unitária e covariância decrescendo para observações mais distantes no tempo, seguindo

$$\sigma_{ijj'} = 1 - |j - j'| \times 0,2.$$

O número de replicações do processo de simulação foi igual a 1500 para cada combinação de cenários.

As Figuras 3.1, 3.2 e 3.3 mostram os resultados simulados do erro do tipo I com 20 e 1000 níveis de fatores para as três distribuições multivariadas citadas, estimativas Σ e Σ_i e nível nominal de 5%. Os resultados com 100 e 500 níveis de fatores foram omitidos por apresentarem resultados semelhantes, mas encontram-se no Anexo A.

Percebe-se nesses gráficos um mesmo padrão: enquanto com a estimação de Σ os resultados foram aparentemente independentes do tamanho amostral, com a estimação Σ_i o erro do tipo I aproxima-se do valor nominal fixado à medida que o tamanho amostral aumenta e o número de pontos no tempo diminui. O cenário ideal para o teste é aquele em que $a = 1000$, $b < 20$, e estimação Σ da matriz de covariâncias, pois, nesses casos, as estimativas do erro do tipo I foram próximas do valor nominal estabelecido. Ainda assim, os resultados mostram que com apenas 20 níveis de fatores, o erro do tipo I foi próximo do nível nominal para as distribuições normal e t_{10} multivariadas e estimação Σ.

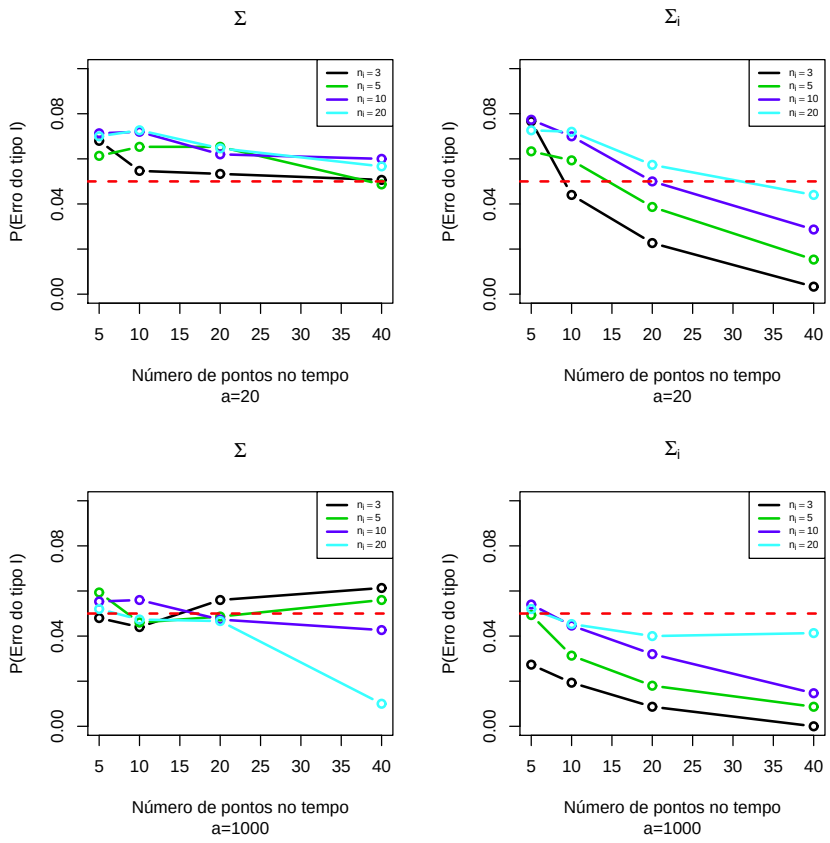


Figura 3.1: Erro do tipo I ($a = 20$ e $a = 1000$) com nível nominal 5% para a distribuição **normal multivariada** e com estimação Σ e Σ_i.

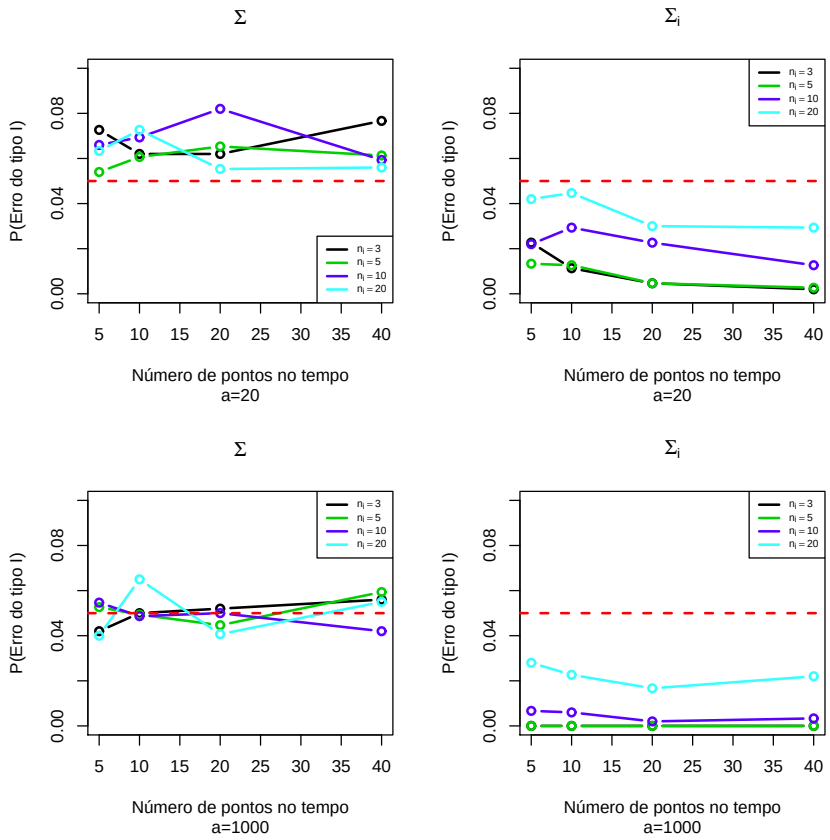


Figura 3.2: Erro do tipo I ($a = 20$ e $a = 1000$) com nível nominal 5% para a distribuição **log-normal multivariada** e com estimação Σ e Σ_i.

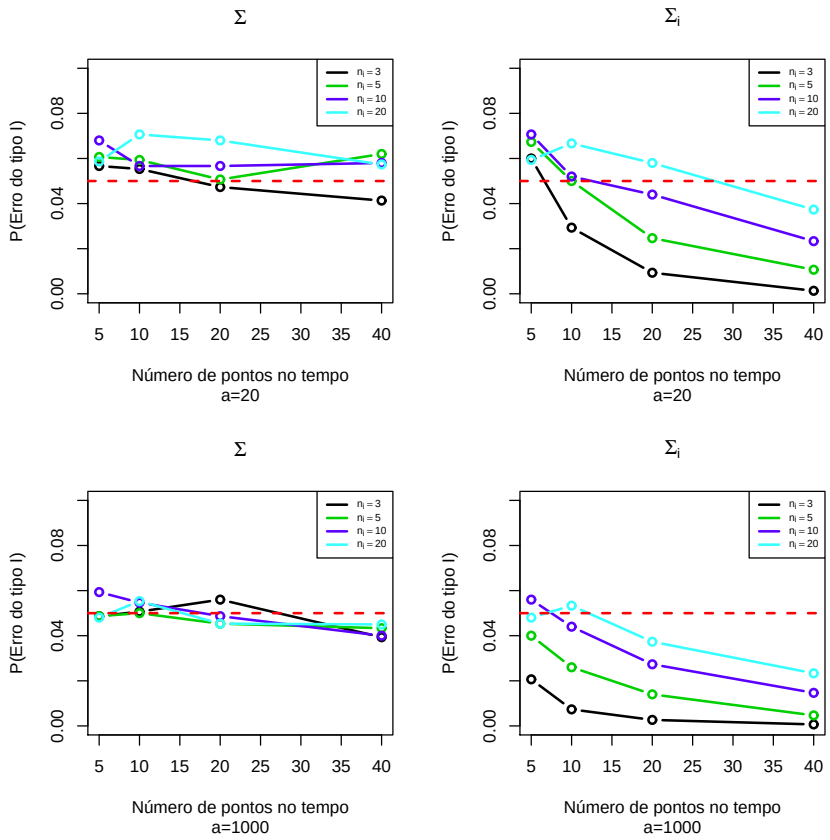


Figura 3.3: Erro do tipo I ($a = 20$ e $a = 1000$) com nível nominal 5% para a distribuição t **multivariada com 10 graus de liberdade** e com estimação Σ e Σ_i .

3.2.3 Poder do Teste

Para estudar o poder do teste dado pelo Teorema 3.2, foram geradas bases de dados seguindo as mesmas três distribuições multivariadas do erro do tipo I. Quanto à escolha do número de níveis de fator e pontos no tempo utilizados nas simulações do poder do teste houve a preocupao em considerar um cenrio intermedirio de favorecimento ao critrio de convergncia do teste. Baseado nos resultados do erro do tipo I, foi visto que o cenrio ideal para o teste é aquele em que o número de níveis de fatores é igual a 1000 e $b < 20$. Com isso, o cenário intermediário escolhido foi $a = 500$ níveis de fator, $b = 20$ pontos no tempo e $n_i = 5$ observações.

Como forma de avaliar a capacidade do teste em detectar desvios de H_0, adicionou-se um termo d na média, que varia de 0 a 2, de forma que a nova média fica sendo $\cos(\pi \times (j+1)) + d$, $j = 1, \ldots, b$. Como antes, a estrutura de variância-covariância é $\sigma_{ijj_1} = 1 - |j - j_1| \times 0,2$.

As curvas do poder foram obtidas em três situações para cada distribuição: (1) 12 fatores deslocados pelo fator d; (2) 25 fatores deslocados pelo fator d e (3) 50 fatores deslocados pelo fator d.

A Figura 3.4 mostra o comportamento do poder do teste com as três distribuições multivariadas e as duas formas de estimação da matriz de covariâncias. O teste apresenta uma boa capacidade de detecção de pequenos deslocamentos em uma fração razoavelmente pequena dos dados (5% como é o caso da Figura 3.4). Apenas quando os dados são assimétricos (caso da log-normal, linha vermelha tracejada) que a diferença precisa ser um pouco maior para que o teste a detecte.

As Figuras 3.5, 3.6 e 3.7 mostram as curvas de poder do teste considerando $2,5\%$, 5% e 10% do número total de níveis de fatores deslocados pelo fator d. Percebe-se em todos os gráficos que é necessário um deslocamento d ligeiramente maior para que o teste detecte a diferença entre os grupos quando é utilizado Σ_i.

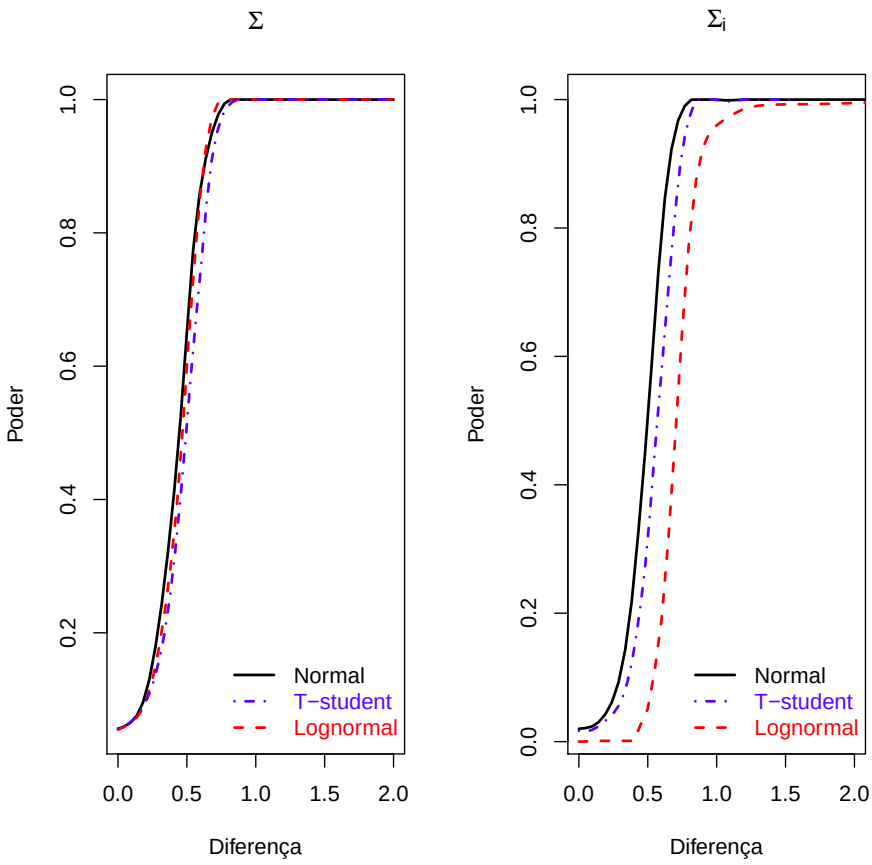


Figura 3.4: Poder do teste considerando as três distribuições e as duas formas de estimação ao nível 0,05.

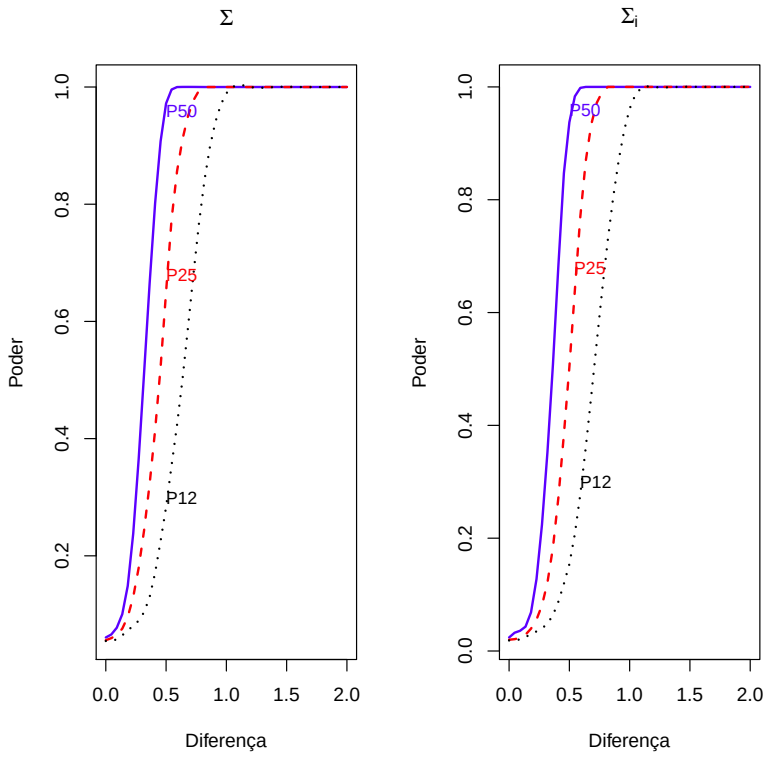


Figura 3.5: Poder do teste considerando 12, 25 e 50 fatores deslocados, as duas formas de estimação com a distribuição normal multivariada dos dados.

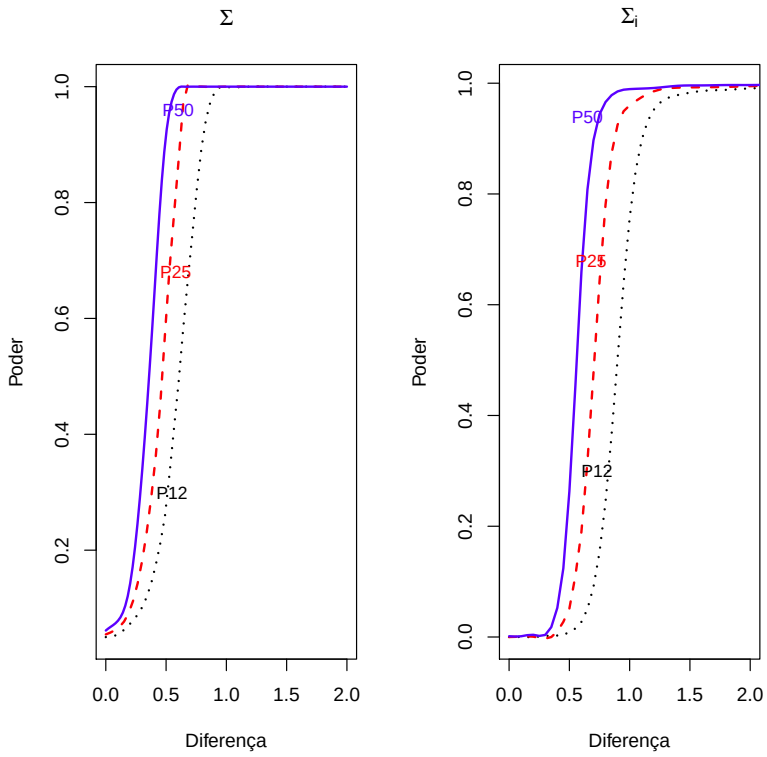


Figura 3.6: Poder do teste considerando 12, 25 e 50 fatores deslocados, as duas formas de estimação com a distribuição log-normal multivariada dos dados.

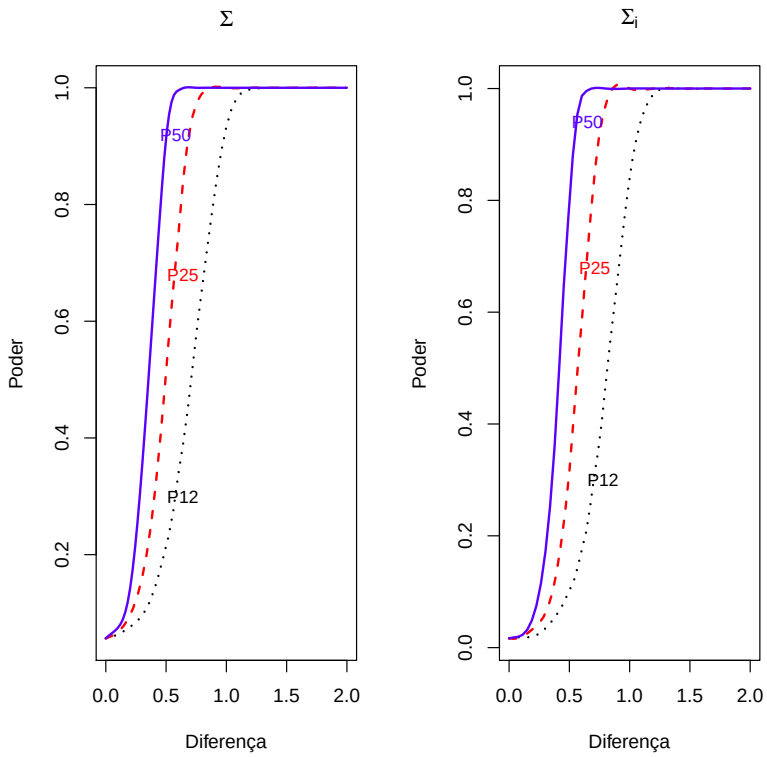


Figura 3.7: Poder do teste considerando 12, 25 e 50 fatores deslocados, as duas formas de estimação com a distribuição $t_{v=10}$ multivariada dos dados.

3.3 Estudo do Teste Presente no PPCLUSTEL-R

Conforme visto na análise do poder do teste presente no PPCLUSTEL, quando os dados seguem uma distribuição assimétrica, como é o caso da distribuição log-normal, então o teste tem menos poder em detectar desvios de H_0 se comparado às demais distribuições estudadas. Nesta seção, será apresentado o teste de uma versão do PPCLUSTEL que possibilita corrigir o problema mencionado. Essa versão recebe o nome de PPCLUSTEL-R[3] e possui a mesma construção do PPCLUSTEL. A única diferença entre eles é que o PPCLUSTEL-R realiza o agrupamento com os dados em postos.

Portanto, considere que R_{ijk} é a representação baseada em postos da observação X_{ijk} e $\tilde{R}_{.j.} = a^{-1}\sum_{i=1}^{a}\bar{R}_{ij.}$. Defina as estatísticas:

[3]Do inglês: p-Values Based Partitional Clustering of Longitudinal Ranked Data.

$$MS\varphi = \frac{1}{(a-1)b}\sum_{i=1}^{a}\sum_{j=1}^{b}(\bar{R}_{ij.} - \tilde{R}_{.j.})^2; \tag{3.14}$$

$$MSE = \frac{1}{ab}\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n_i}\frac{(R_{ijk} - \bar{R}_{ij.})^2}{n_i(n_i-1)}, \tag{3.15}$$

e de forma semelhante ao Teorema 3.2, obtém-se o seguinte resultado idêntico a (3.8):

$$F_\phi = \sqrt{ab}(MS\varphi - MSE) \xrightarrow{d} N\left(0, \lim_{a\to\infty}\frac{2}{ab}\sum_{i=1}^{a}\frac{1}{n_i(n_i-1)}\sum_{j=1}^{b}\sum_{j'=1}^{b}\sigma^2_{ijj'}\right) \tag{3.16}$$

com $a \to \infty$.

Como não há demonstração formal da convergência (3.16), então, na próxima seção, a validade dessa convergência será verificada por meio de simulações. Ainda será realizado um estudo do erro do tipo I e poder do teste, seguindo a mesma metodologia utilizada com o teste presente no PPCLUSTEL.

3.3.1 Simulação da Convergência

Nesta seção, será verificada se a substituição dos dados originais pelos seus respectivos postos mantém a convergência da distribuição amostral da estatística F_ϕ em (3.16). Para tal investigação, simulou-se bases de dados com 20, 100, 500 e 1000 níveis de fatores e 5, 10, 20 e 40 pontos no tempo, com 5 replicações para cada nível do fator. Foi utilizada a forma de estimação da matriz de covariâncias: Σ_i.

Os dados foram gerados da distribuição normal multivariada com a mesma média e matriz de covariância apresentada na Subseção 3.2.2. Em cada caso, foi realizado um teste de aderência à distribuição normal de Shapiro-Wilk (1965). Esse teste foi escolhido com base nos resultados obtidos por Razali & Wah (2011), onde os autores compararam o teste de Shapiro-Wilk com outros testes de aderência à distribuição normal, como Kolmogorov-Smirnov, Lilliefors e Anderson-Darling e comprovaram que ele é o mais poderoso em detectar normalidade nos dados. Seja,

$$\eta = \lim_{a\to\infty}\frac{2}{ab}\sum_{i=1}^{a}\frac{1}{n_i(n_i-1)}\sum_{j=1}^{b}\sum_{j'=1}^{b}\sigma^2_{ijj'}. \tag{3.17}$$

Então, a hipótese testada foi a seguinte:

$$H_0 : F'_{\phi} = \sqrt{ab}\,\frac{(MS\varphi - MSE)}{\sqrt{\eta}} \sim N\,(0,1)\,. \tag{3.18}$$

Ainda de acordo com Razali & Wah (2011), o poder de detecção do teste de Shapiro-Wilk quando os dados seguem distribuições simétricas diferentes da normal dependem muito do tamanho amostral. Por esse motivo, as simulações consideraram duas situações: 60 e 200 replicações do procedimento.

Os resultados da simulação são apresentados na Tabela 3.3. Tanto com 60, quanto com 200 repetições, o p-valor do teste W de Shapiro-Wilk rejeitou a hipótese de normalidade a 5% de significância apenas quando o número de níveis de fator foi igual a 20. Rejeição cuja causa provável seja o reduzido número de níveis de fator e/ou elevado número de pontos no tempo (condições desfavoráveis à convergência do teste).

Portanto, esses resultados corroboram a afirmação de que o teste do PPCLUSTEL -R possui a mesma convergência que o teste do PPCLUSTEL e que esta se verifica com u número de níveis de fator é maior que 100.

Parâmetros		60 rep.		200 rep.	
a	b	W	$P(W < w)$	W	$P(W < w)$
20	5	0,946	0,010*	0,982	0,011*
20	10	0,983	0,575	0,982	0,013*
20	20	0,986	0,705	0,990	0,160
20	40	0,955	0,026*	0,988	0,101
100	5	0,969	0,135	0,997	0,947
100	10	0,992	0,965	0,996	0,878
100	20	0,993	0,987	0,986	0,053**
100	40	0,983	0,574	0,988	0,099**
500	5	0,973	0,199	0,994	0,574
500	10	0,991	0,937	0,992	0,365
500	20	0,982	0,540	0,995	0,713
500	40	0,971	0,162	0,993	0,458
1000	5	0,983	0,569	0,995	0,736
1000	10	0,977	0,327	0,987	0,072**
1000	20	0,994	0,991	0,994	0,549
1000	40	0,991	0,952	0,992	0,370

Tabela 3.3: Resultado do teste de normalidade de Shapiro-Wilk. (**) Significativo a 5%. (*) Significativo a 10%.

3.3.2 O Erro do Tipo I

Para o estudo do erro do tipo I do teste presente no PPCLUSTEL-R utilizou-se as mesmas condições das simulações da Subseção 3.2.2. A Figura 3.8 mostra os erros do tipo I estimados para as distribuições normal e log-normal multivariadas e a Figura 3.9 mostra os erros do tipo I estimados para a distribuição t_{10} multivariada. Os resultados foram semelhantes aos obtidos com o PPCLUSTEL, exceto pelo esperado fato de a log-normal multivariada ter apresentado resultados iguais aos da normal multivariada por ser uma transformação monótona desta.

Portanto, observa-se nos gráficos o mesmo padrão: enquanto com a estimação de Σ os resultados foram aparentemente independentes do tamanho amostral, com a estimação Σ_i o erro do tipo I aproxima-se do valor nominal fixado à medida que o tamanho amostral aumenta e o número de pontos no tempo diminui. As tabelas com todas as simulações encontram-se no Anexo A.

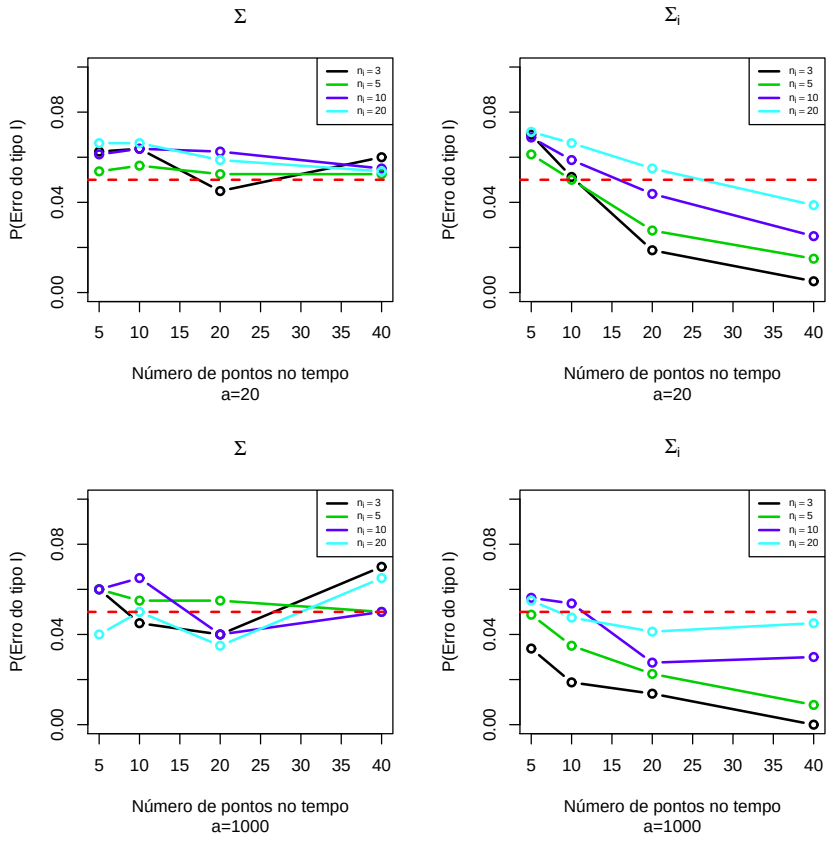


Figura 3.8: Erro do tipo I ($a = 20$ e $a = 1000$) com nível nominal 5% para as distribuições **normal multivariada** e **log-normal multivariada**.

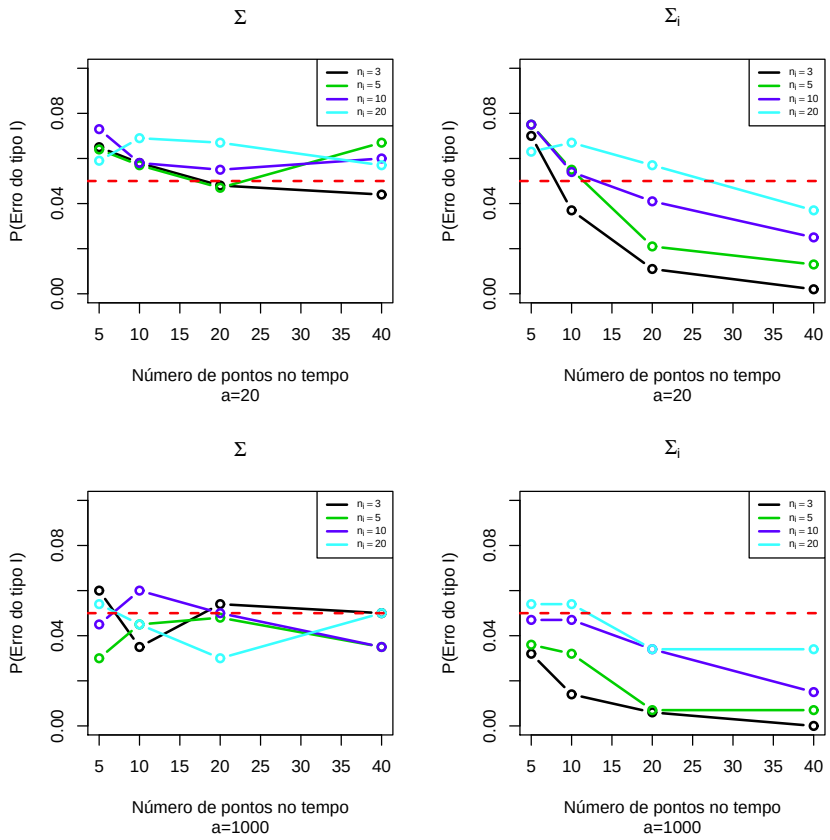


Figura 3.9: Erro do tipo I ($a = 20$ e $a = 1000$) com nível nominal 5% para a distribuição t **multivariada com 10 graus de liberdade** .

3.3.3 Poder do Teste

Para estudar o poder do teste dado pelo Teorema 3.2, foram geradas bases de dados seguindo as mesmas condições apresentadas na Subseção 3.2.3. As curvas de poder do teste presente no PPCLUSTEL-R foram muito semelhantes às Figuras 3.5 a 3.7, e, portanto, foram omitidas desta parte do trabalho.

A Figura 3.10 mostra o comportamento do poder do teste presente no algoritmo PPCLUSTEL-R com as três distribuições multivariadas e as duas formas de estimação da matriz de covariâncias. Percebe-se que as curvas de poder quando os dados seguem as distribuições normal e log-normal são coincidentes, o que corrige a deficiência apresentada no PPCLUSTEL em detectar desvios quando os dados são assimétricos. A Figura mostra que o teste apresenta uma boa capacidade de detecção de pequenos deslocamentos em uma fração razoavelmente pequena dos dados (apenas 25 níveis de fator deslocados por d, o que representa 5% do número total de níveis de fator).

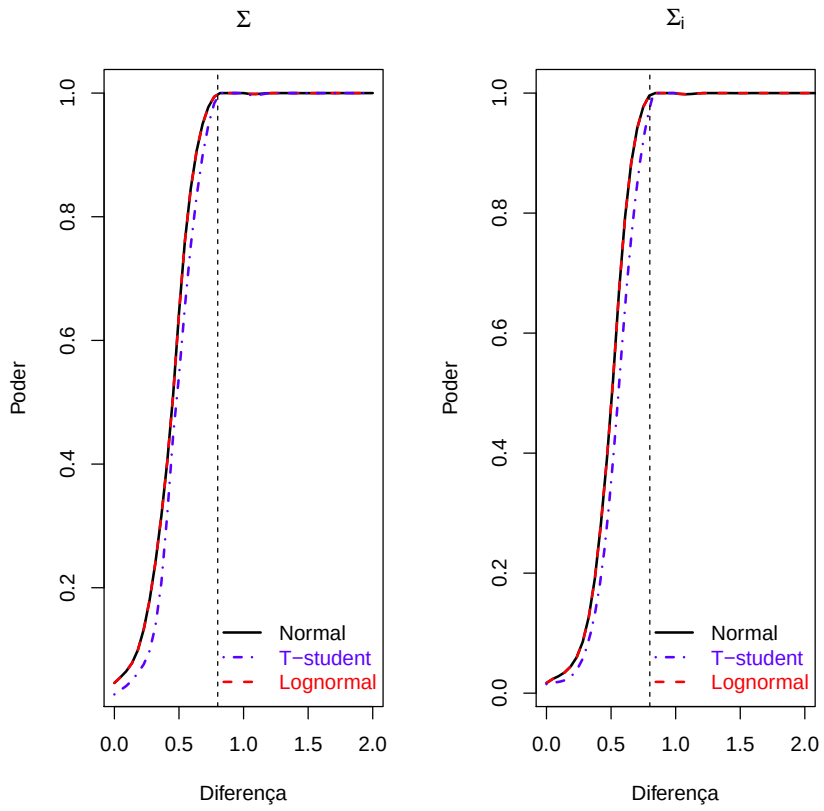


Figura 3.10: Poder do teste considerando as três distribuições e as duas formas de estimação ao nível 0,05. .

3.4 O Teste Presente no PPCLUSTEL - M ($b \to \infty$ e a fixo)

Nesta seção, será apresentado um teste de hipóteses desenvolvido por Wang (2004), cujo objetivo é o mesmo do teste (3.3) utilizado no PPCLUSTEL, mas diferentemente deste, agora o procedimento é indicado para uma estrutura de dados onde o número de níveis de fator (a) é fixo, há poucas observações disponíveis (n_i) e o número de pontos no tempo (b) é alto.

O teste é construído sob a suposição de que as séries de tempo correspondentes a diferentes indivíduos são independentes e satisfazem uma condição α-*mixing*. De acordo com essa condição, para alguma sequência $\alpha_m \to 0$,

$$|P(A \cap B) - P(A)P(B)| \leq \alpha_m, \tag{3.19}$$

vale para todo $A \in \sigma(X_{i1k}, \ldots, X_{ilk})$, $B \in \sigma(X_{i,l+m,k}, X_{i,l+m+1,k}, \ldots)$, e todo i, k, e $l \geq 1$, onde $\sigma(.)$ é o espaço σ-álgebra gerado pelas variáveis aleatórias. Basicamente, essa suposição diz que a correlação entre observações de uma mesma curva reduz à medida que aumenta o lapso de tempo m. De acordo com a autora, a suposição α-*mixing* para a estrutura temporal dos dados não é um requisito restritivo, visto que muitos modelos tradicionais de séries de tempo como os processos ARCH (*Autoregressive Conditional Heteroscedasticity*) e AR aditivos com variáveis exógenas também satisfazem essa condição.

Seja R_{ijk} a representação baseada em postos para a observação X_{ijk} e $\tilde{R}_{.j.} = a^{-1}\sum_{i=1}^{a} \bar{R}_{ij.}$. Considere o mesmo modelo (3.2), e defina novamente as estatísticas:

$$MS\varphi = \frac{1}{(a-1)b}\sum_{i=1}^{a}\sum_{j=1}^{b}(\bar{R}_{ij.} - \tilde{R}_{.j.})^2 \tag{3.20}$$

$$MSE = \frac{1}{ab}\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n_i}\frac{(R_{ijk} - \bar{R}_{ij.})^2}{n_i(n_i-1)} \tag{3.21}$$

e

$$F_\phi = \frac{MS\varphi}{MSE} \tag{3.22}$$

Teorema 3.2. *(Wang, 2004) Assuma cada curva X_{ijk}, $j = 1, 2, \ldots$, como sendo* α-mixing *com $\alpha_m = O(m^{-5})$. Seja $N = \sum_{i=1}^{a} n_i b$ e $\sigma_{ijj'} = cov(X_{ijk}, X_{ij'k})$, defina*

$$\zeta_1 = \frac{2}{a^2 b} \sum_{j=1}^{b} \sum_{j'=1}^{b} \sum_{i=1}^{a} \frac{\sigma_{ijj'}^2}{n_i \left(n_i - 1\right)} \tag{3.23}$$

$$\zeta_2 = \frac{2}{a^2 b} \sum_{j=1}^{b} \sum_{j'=1}^{b} \sum_{i \neq i'}^{a} \frac{\sigma_{ijj'}\ \sigma_{i'jj'}}{n_i\ n_{i'}}. \tag{3.24}$$

Dessa forma, se $n_i \geq 2$, então quando $b \to \infty$ e a permanece fixo, com

$$\sigma^2 = \lim_{b \to \infty} \frac{E(MSE)}{N^2}, \qquad \tau^2 = \lim_{b \to \infty} \frac{\zeta_1 + \zeta_2/(a-1)^2}{N^4}.$$

Então, sob H_0 vale,

$$\sqrt{b}(F_\phi - 1) \xrightarrow{d} N\ (0,\ \tau^2/\sigma^4) \tag{3.25}$$

Seguindo a mesma ideia de agrupar variáveis em dados HDLLSS utilizando o p-valor resultante do teste (3.1) como medida de similaridade (von Borries, 2008), Wang (2008) desenvolveu um algoritmo similar ao PPCLUSTEL, denominado PCLUST. Este algoritmo utiliza a mesma ideia de particionar os dados sempre que o p-valor encontrado for menor que determinado limiar pré-especificado. Entretanto, há algumas diferenças importantes entre os algoritmos com relação à estimação da matriz Σ que devem ser destacadas. O PCLUST não utiliza nenhuma técnica específica para reduzir o viés inerente à estimação de $\sigma_{ijj'}^2$. Além disso, o algoritmo não propõe nenhuma solução ao problema da limitação do tamanho amostral para a estimação desse valor. Perceba que esse problema é ainda mais crônico no PCLUST, visto que seu teste converge apenas com muitos pontos no tempo ($b \to \infty$).

Levando-se em consideração os aspectos mencionados, optou-se por adequar o teste (3.25) ao algoritmo do PPCLUSTEL, pois este consegue lidar relativamente bem com os problemas citados, conforme será visto no capítulo 4. Esse novo algoritmo recebe, então, a nomenclatura PPCLUSTEL - M. Na próxima seção, o PPCLUSTEL - M será comparado ao PCLUST para verificar se os algoritmos se equivalem, e se o método de estimação da estrutura de covariâncias proposto é capaz de superar um

algoritmo que não o utiliza. Vale ressaltar que o erro do tipo I e poder do Teste (3.25) não serão apresentados, pois já foram estudados no trabalho de Wang (2004).

3.4.1 Comparação entre o PPCLUSTEL - M e o PCLUST

Nesta seção, a metodologia utilizada por Wang (2008) para verificar o desempenho do PCLUST em dados simulados será reproduzida com o PPCLUSTEL - M. O objetivo é verificar se a estimação *jackknife* de $\sigma^2_{ijj'}$, em conjunto com o algoritmo de agrupamento do PPCLUSTEL produzem melhores resultados que o PCLUST, uma vez que as medidas de similaridade dos algoritmos são idênticas.

Em estudos com microarranjo longitudinal é comum encontrar genes que apresentam um padrão de expressão genético invariante ao longo do tempo. Esse padrão é referido pela literatura de biotecnologia (Ramoni et al., 2002) como curvas planas (*flat curves*), e podem ser representadas estatisticamente como eventos independentes e identicamente distribuídos (i.i.d.) de uma variável aleatória qualquer (ruído branco). Como o objetivo do estudo de microarranjo é encontrar grupos de genes que apresentem padrões de atividade gênica semelhantes em determinado processo biológico, então a presença de genes que não se relacionem com nenhum outro gene do modelo apenas prejudica a eficiência do agrupamento.

Alguns algoritmos como o Cats (Serban & Wasserman, 2005) e o PCLUST (Wang et. al., 2008) possuem metodologias para detectar as curvas planas, de forma que uma vez detectadas, elas sejam removidas da base de dados antes da aplicação do algoritmo de agrupamento propriamente dito. Diferentemente desses, o PPCLUSTEL-M não retira as curvas planas antes de o algoritmo ser aplicado, entretanto, espera-se que esse procedimento seja capaz de diferenciá-las das demais curvas agrupando-as em separado.

Dessa forma, os autores dos trabalhos supracitados simulam as curvas planas como amostras independentes e identicamente distribuídas de uma normal com média ν e variância σ^2_0, onde $\nu \sim Unif(-3;\, 3)$ e $\sigma^2_0 \sim Unif(1,2;\, 1,4)$. Foram geradas 1400 curvas desse tipo. Os grupos foram gerados com os seguintes vetores de médias :

Grupo 1: 150 curvas com média,

$$\mu_j^1 = 3\,min\left\{\left(\frac{2-5j}{2}\right), \left[\left(\frac{5j-2}{3}\right)^2 + sen(\frac{5\pi j}{2})\right]\right\}, \quad j = 1, \ldots, b;$$

Grupo 2: 150 curvas com média $\mu_j^2 = -\mu_j^1$, $j = 1 \ldots b$;

Grupo 3: 150 curvas com média $\mu_j^3 = cos(2\pi j)$, $j = 1 \ldots b$;

Grupo 4: 150 curvas com média $\mu_j^4 = -\mu_j^3$, $j = 1 \ldots b$.

Esses grupos também foram utilizados em Serban & Wasserman (2005), entretanto, os autores consideraram apenas observações independentes no tempo, ou seja, $\Sigma_{b\times b} = I$, onde I é a matriz identidade e b, o número de pontos no tempo. Nesta etapa do trabalho, será considerado o caso correlacionado. Seguindo a estrutura de covariância utilizada nas simulações dos trabalhos de Wang et al. (2008), Fan & Zhang (2000) e Wu & Chiang (2000), temos:

$$\Sigma_{b\times b} = cov(X_{ijk}, X_{i'j'k'}) = \begin{cases} \sigma_j \sigma_{j'}\, e^{-|j-j'|/b}, & \text{se } i = i',\ k = k' \\ 0, & \text{se } i \neq i',\ k \neq k', \end{cases}$$

onde b é o número de pontos no tempo do experimento e $\sigma_j \sim Unif(1,2;\ 1,4)$. De acordo com essa estrutura de covariância, a correlação entre pontos sucessivos é alta e decresce à medida que o intervalo de tempo aumenta. Portanto, foram simuladas bases de dados segundo as distribuições normal e log-normal multivariadas, com 2000 níveis de fatores, 25 pontos no tempo e 3 repetições. Como forma de comparar a estabilidade dos algoritmos, o processo de simulação foi repetido 200 vezes. A Tabela 3.4 apresenta os resultados dos procedimentos. O índice ARI, visto na Seção 2.2, foi calculado comparando o agrupamento obtido com a real estrutura simulada em cada um das 200 simulações realizadas.

Percebe-se que o PPCLUSTEL - M apresentou um maior índice de acertos que o PCLUST, pois o ARI mediano foi igual 0,994, valor próximo de 1, que é o máximo que o índice pode atingir. Conforme o esperado, os resultados do PPCLUSTEL - M foram iguais para as distribuições normal e log-normal multivariadas. Outro ponto a favor do PPCLUSTEL-M é que este apresentou maior estabilidade, que pode ser verificada comparando-se os desvios-padrão (D.P.) e as distâncias interquartílicas ($q_3 - q_1$)

Algoritmo	Distribuição	Mín.	q_1	Mediana	q_3	Máx.	D.P.
PCLUST	Normal	0,882	0,902	0,907	0,913	0,925	0,008
	Log-normal	0,838	0,882	0,889	0,896	0,92	0,012
PPCLUSTEL - M	Normal	0,985	0,992	0,994	0,994	0,997	0,002
	Log-normal	0,985	0,992	0,994	0,994	0,997	0,002

Tabela 3.4: Resumo do ARI com 200 replicações para os algoritmos PPCLUSTEL-M e PCLUST. Nível de significância dos testes foi de 5%.

calculadas. Vale ressaltar que foi utilizada a estimação Σ_i para a estrutura de covariâncias, o que indica, nesse caso, que a qualidade da estimação dos parâmentros $\sigma^2_{ijj'}$ com um reduzido tamanho amostral foi suficiente para garantir um bom desempenho do agrupamento. Portanto, conclui-se que a redução do viés obtida com o estimador *jackknife* de Σ_i juntamente com o algoritmo de agrupamento desenvolvido por von Borries (2008) são os responsáveis pela superioridade do PPCLUSTEL - M em relação ao PCLUST.

3.5 O Algoritmo de Agrupamento por Partição

Os p-valores obtidos aplicando-se um dos testes que foram explicados podem ser utilizados como medidas de similaridade em um algoritmo de agrupamento baseado em partições com dados HDLLSS. O algoritmo que será apresentado nesta seção foi desenvolvido e implementado[4] por von Borries.

Este procedimento busca iterativamente testar se cada grupo, o qual contém múltiplos níveis de fatores, possui homogeneidade (medida de similaridade) acima de um limiar, determinado pelo nível de confiança do teste. A partição ocorre sempre que a similaridade estiver abaixo desse limiar, ou seja, sempre que o teste 3.1 for rejeitado. Ao final do procedimento, as partições remanescentes serão os grupos formados. Uma vantagem do método de partição é que ele permite reduzir o número de comparações a serem feitas, o que evita a necessidade de lidar com o problema de comparações múltiplas.

Seja g o índice do grupo em que o teste está sendo aplicado, $D1$ contém as ob-

[4]Linguagem Macro SAS© Versão 9.2

servações no grupo g e nf é o número de fatores em $D1$. O algoritmo é descrito abaixo em sete passos e um diagrama do procedimento está representado na Figura 3.11.

1. Seja $g = 1$, $D1 = Dados$.
2. Calcule a mediana para cada fator em $D1$.
3. Ordene os fatores em $D1$ pelas suas medianas.
4. Teste $D1$.
   - 4.1 Se H_0 não é rejeitada: algoritmo termina.
   - 4.2 Se H_0 é rejeitada: vá para o Passo 5.
5. Retire um subconjunto de $D1$ e chame de $D2$.
6. Calcule o número de fatores em $D2$ e chame de nf.
   - 6.1 Se $nf = 1$:
     - 6.1.1 Aloque fator em $D2$ para o grupo 0.
     - 6.1.2 Remova os fatores em $D2$ de $D1$.
     - 6.1.3 Se nf em $D1 = 0$, então algoritmo termina.
     - 6.1.4 Se nf em $D1 > 0$, então faça $D2 = D1$ e vá para o Passo 7.
7. Teste $D2$.
   - 7.1 Se H_0 não é rejeitada:
     - 7.1.1 Atribua fatores de $D2$ para o grupo g.
     - 7.1.2 Faça $g = g + 1$.
     - 7.1.3 Remova os fatores de $D1$ em $D2$.
     - 7.1.4 Se nf em $D1 = 0$ então algoritmo termina.
     - 7.1.5 Se nf em $D1 > 0$ então faça:
       - A. Teste se cada fator em $D1$ pertence ao novo grupo assinalado. Remova o fator de $D1$ quando H_0 não é rejeitada e o coloque nesse novo grupo.

B. Faça $D2 = D1$ para os fatores remanescentes em $D1$ e retorne ao Passo 7.

7.2 Se H_0 é rejeitada:

7.2.1 Retire um subconjunto de $D2$ e chame de $D3$.

7.2.2 Retorne para $D1$ todos os fatores que não estão em $D3$.

7.2.3 Faça $D2 = D3$ e remova $D3$.

7.2.4 Retorne ao Passo 7.

Ao utilizarmos as medidas de similaridade provenientes dos três testes não-paramétricos, apresentados neste capítulo, juntamente com o algoritmo de agrupamento baseado em partições da Figura 3.11, obtemos os seguintes procedimentos capazes de agrupar dados HDLLSS: PPCLUSTEL, PPCLUSTEL - R e PPCLUSTEL - M. Esses procedimentos possuem as seguintes propriedades:

1. Especificação automática do número de grupos: Os algoritmos não exigem que o número de grupos seja pré-especificado. O algoritmo determina o número de grupos automaticamente pela especificação de um nível de significância limiar que será comparado com os p-valores para testar a hipótese de homogeneidade de distribuição;

2. Menor preocupação com problemas de comparação múltipla: Esta é uma preocupação menos importante no PPCLUSTEL, pois o teste é aplicado em grupos de variáveis, e não em comparações individuais, o que reduz drasticamente o número total de comparações;

3. Flexibilidade em trabalhar com dados desbalanceados e com pequenas amostras: O algoritmo funciona tanto com dados balanceados quanto desbalanceados. A única exigência é que o número de repetições por variável seja pelo menos 2. Não há necessidade de que todas as variáveis tenham o mesmo número de repetições.

4. Flexibilidade na escolha do comportamento da estrutura de covariância dos dados: A possibilidade de escolha da forma de estimação da matriz de covariâncias (Σ ou Σ_i) permite ao usuário maior flexibilidade em adequar o algoritmo aos conhecimentos prévios que ele possa ter do comportamento dessa estrutura.

Além da mencionada propriedade de invariância a transformações monótonas, outra importante característica dos algoritmos PPCLUSTEL-R e PPCLUSTEL-M decorrente do fato de serem baseados em postos é que eles são robustos à presença de valores discrepantes nos dados.

A maioria dos algoritmos existentes destinados a agrupar dados HDLLSS alocam todos os objetos nos grupos. Entretanto, em dados de microarranjo, a presença de genes esporádicos pode levar a uma interpretação errônea dos resultados obtidos. Uma vantagem do algoritmo apresentado é que ele permite a identificação dos genes esporádicos, ou seja, variáveis que não se relacionam com nenhum outro grupo criado. Esses objetos são alocados em um grupo 0 ao final do procedimento.

A escolha do nível de significância dos testes (limiar) fica a cargo do usuário. Quanto menor for esse limiar (próximo de 0), mais conservador o algoritmo será em detectar diferenças entre grupos, e, consequentemente, a tendência será a formação de menos grupos ao final do procedimento. Por outro lado, quanto maior o limiar escolhido (próximo de 1), menos conservador será o algoritmo, levando à formação de um número maior de grupos. As escolhas de diferentes limiares resultam em partições distintas, o que, de certa forma, limita a propriedade de detecção automática do número de grupos.

Entretanto, conforme verificado em estudos de simulação, ao repetirmos o algoritmo diversas vezes mudando os limiares estabelecidos, verifica-se que, em determinado ponto, ocorre uma homogeneização dos grupos encontrados. Portanto, sugere-se que se escolha o limiar que estabilize os resultados obtidos.

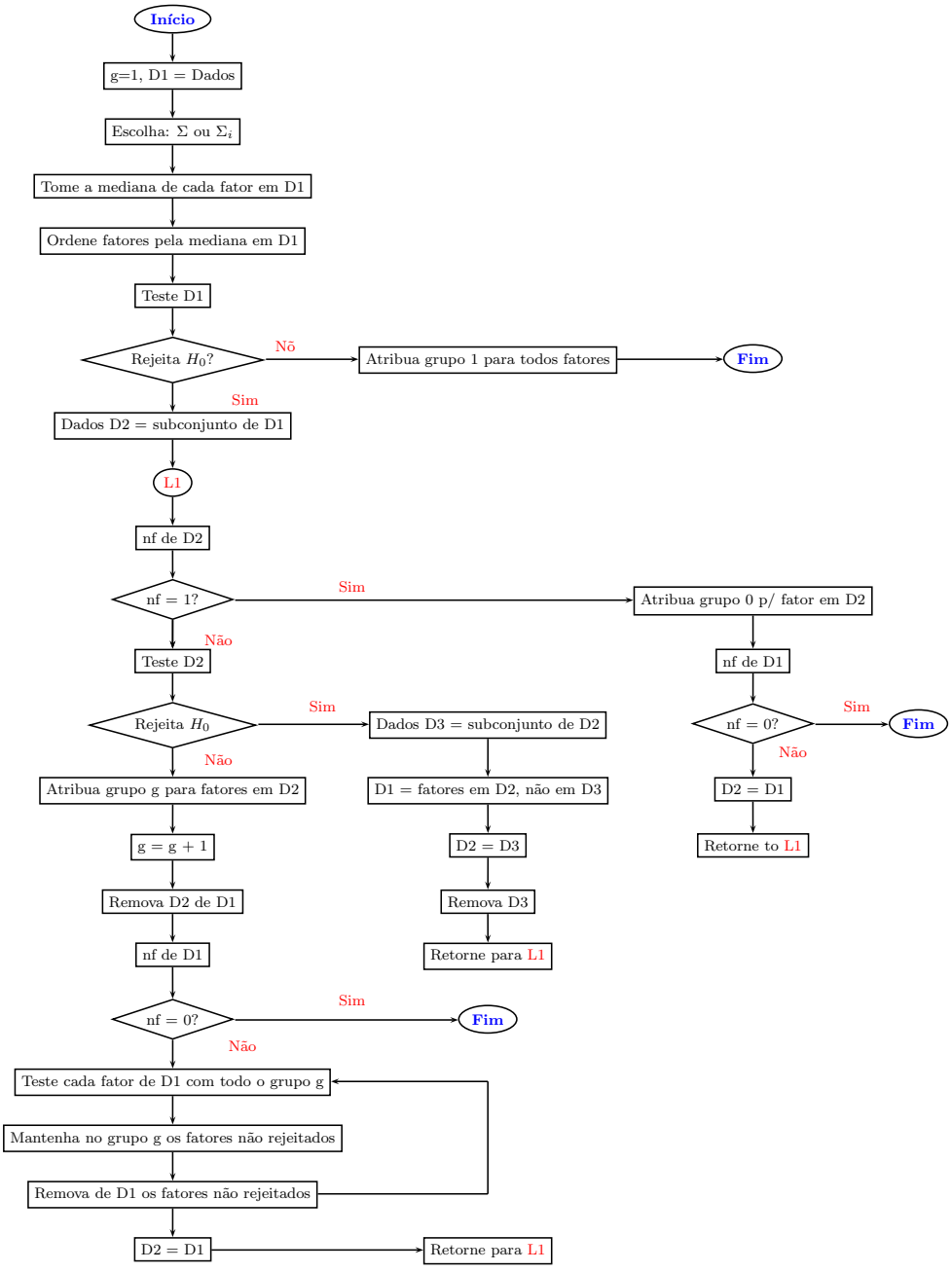


Figura 3.11: Diagrama PPCLUSTEL (von Borries, 2008, com modificações). No diagrama, nf significa "número de fatores" e g significa "nome do grupo". Grupo 0 é o grupo reservado aos fatores que não foram alocados em nenhum outro grupo criado.

Capítulo 4

Simulação e Aplicação dos Algoritmos

Nesta seção, será realizado um estudo do desempenho dos algoritmos apresentados em dados simulados. Foram gerados grupos que diferem entre si com relação aos seus respectivos vetores de médias e possuem a mesma estrutura de covariâncias. Primeiramente, será apresentado um algoritmo de agrupamento baseado em misturas de distribuições normais, o MCLUST. Esse procedimento é muito utilizado em agrupamento e se comporta relativamente bem em dados HDLLSS, conforme será visto.

4.1 Resultados em Dados Simulados

4.1.1 O Algoritmo MCLUST

Quando um conjunto de dados é formado por grupos que provêm de distribuições de probabilidade diferentes, então a modelagem desses dados pode ser feita utilizando-se a metodologia de mistura de distribuições.

Em uma abordagem de agrupamentos, esse método pode ser classificado como um procedimento hierárquico aglomerativo que modela uma mistura de distribuições aos dados para fornecer a probabilidade de um objeto i pertencer ao grupo k. Percebe-se, portanto, a mesma ideia de atribuir probabilidades existentes nas técnicas difusas (*fuzzy*), explicadas na Seção 2.1.3.

Dessa forma, considera-se que os dados seguem a seguinte função densidade de probabilidade

$$f(x) = \sum_{i=1}^{g} \pi_i \, f_i(x), \tag{4.1}$$

onde os dados a serem agrupados são representados por x, π_i é a probabilidade de uma observação pertencer à i-ésima componente ($\pi_i \in [0,\ 1]$ e $\sum_{i=1}^{g} \pi_i = 1$) e f_i é a função densidade de probabilidade dos elementos do grupo i.

O pacote MCLUST[1], desenvolvido por Fraley & Raftery (2002, 2006, 2007), assume que os dados seguem uma mistura de distribuições normais. Portanto,

$$f(x) = \sum_{i=1}^{g} \pi_i \, f_i(x|\ \mu_i, \Sigma_i), \tag{4.2}$$

onde μ_i é a média e Σ_i a matriz de covariâncias do i-ésimo grupo, cuja função densidade de probabilidade f_i segue a distribuição gaussiana. Os parâmetros da normal são estimados pelo algoritmo *Expectation-Maximization* (EM), e o objeto é alocado ao componente no qual possui a maior probabilidade de pertencer. Ao final do procedimento, os componentes da mistura que foram estimados são os grupos formados.

Os autores definem um conjunto de formas para Σ_i que fornecem informações sobre o volume, forma e orientação dos componentes. Considere a decomposição espectral da matriz $\Sigma_i = \lambda_i D_i A_i D_i^T$, onde D_k é a matriz ortogonal de autovetores de Σ_i, A_i é uma matriz diagonal cujos elementos são proporcionais aos autovalores de Σ_i, e λ_i é um escalar.

A orientação dos componentes principais de Σ_i é determinada por D_i, enquanto A_i determina a forma das curvas de nível da densidade do i-ésimo grupo; $\lambda_i \propto \lambda_i^d |A_i|$ especifica o volume da elipsoide, sendo d é o número de dimensões dos dados.

A Tabela 4.1 mostra as opções disponíveis no pacote para a escolha da estrutura dos dados. Caso o usuário não fixe uma dessas opções, então o algoritmo utiliza o critério de informação bayesiano (BIC) para a escolha do melhor modelo.

Segundo Fraley e Raftery (2006), a deficiência do MCLUST está no fato de o algoritmo EM falhar ao estimar os parâmetros quando há singularidade na matriz de covariâncias ou quando os grupos contiverem poucas observações. Outro ponto a ser

[1]Disponível no *software R*

Modelo	**Distribuição**	**Volume**	**Forma**	**Orientação**
λI	Esférica	fixo	fixa	NA
$\lambda_i I$	Esférica	variável	fixa	NA
λA	Diagonal	fixo	fixa	Eixos coord.
$\lambda_i A$	Diagonal	variável	fixa	Eixos coord.
λA_i	Diagonal	fixo	variável	Eixos coord.
$\lambda_i A_i$	Diagonal	variável	variável	Eixos coord.
λDAD^T	Elipsoidal	fixo	fixa	fixa
$\lambda D_i A D_i^T$	Elipsoidal	fixo	fixa	variável
$\lambda_i D_i A D_i^T$	Elipsoidal	variável	fixa	variável
$\lambda_i D_i A_i D_i^T$	Elipsoidal	variável	variável	variável

Tabela 4.1: Possíveis estruturas dos dados consideradas pelo pacote MCLUST.

considerado é a suposição de normalidade dos dados, que nem sempre é satisfeita ou é de difícil detecção (caso multivariado).

4.1.2 Desempenho Comparativo dos Algoritmos

Nesta seção, utilizou-se o índice de Rand ajustado (ARI), apresentado na seção 2.2, para comparar a eficiência dos algoritmos apresentados em dados simulados. Em Wang (2008), o PCLUST foi comparado com diversos algoritmos existentes na literatura, tais como o MCLUST, k-means, diana, fanny, energy, SSClust, SOM, entre outros. Como o MCLUST apresentou bons resultados frente ao PCLUST, que é um procedimento cuja construção é semelhante aos métodos deste trabalho, e por ser um algoritmo muito popular e eficiente, então ele foi utilizado nas simulações desta seção para servir de comparação aos métodos derivados do PPCLUSTEL que foram apresentados no Capítulo 3.

Uma característica do MCLUST é que ele não foi projetado para agrupar dados com valores de entrada X_{ijk}, como o disposto na Tabela 3.1. Caso isso ocorra, então o algoritmo não distinguirá as repetições de um mesmo nível de fator, pois ele interpreta cada linha da base de dados como sendo uma unidade amostral e cada coluna como uma variável. Dessa forma, para adequar a estrutura X_{ijk} ao MCLUST, utilizou-se as médias $\bar{X}_{ij.}$ como os valores de entrada do algoritmo.

A base de dados que simula a estrutura HDLLSS foi construída levando-se em consideração as seguinte situações: 1.000 e 4.000 variáveis (níveis de fator); 5, 10 e 20 pontos no tempo; 3 e 5 observações; distribuições normal, lognormal e t (10 graus de liberdade) multivariadas. As variáveis foram divididas em 5 grupos que diferem apenas em relação à sua média. Cada grupo formado possui vetor de médias μ_b e matriz de covariâncias $\Sigma_{b\times b}$. As médias μ_j, com $j = 1, 2, \ldots, b$, de cada grupo foram geradas de acordo com o seguinte esquema:

Grupo 1: 20% do número total de níveis de fator com média $\mu_j = cos\,(\pi(j+1))$;

Grupo 2: 20% do número total de níveis de fator com média $\mu_j = cos\left(\frac{\pi(j+1)}{10}\right) + 3$;

Grupo 3: 20% do número total de níveis de fator com média $\mu_j = sin(\pi(j+1)/2)$;

Grupo 4: 20% do número total de níveis de fator com média $\mu_j = j - 4$;

Grupo 5: 20% do número total de níveis de fator com média $\mu_j = j/4$.

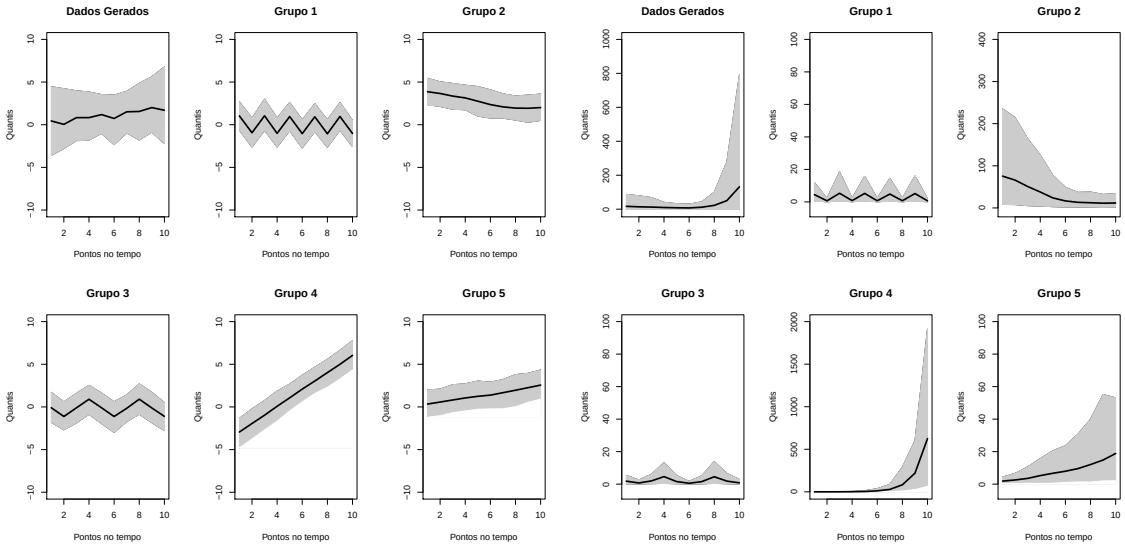
Os elementos da matriz de covariâncias foram os mesmos de simulações anteriores, ou seja,

$$\sigma_{ijj'} = 1 - |j - j'| \times 0,2.$$

A Figura 4.1 ilustra os dados com as misturas juntas (painel superior esquerdo) e os 5 grupos gerados separadamente para as três distribuições consideradas. Percebe-se que os cinco grupos gerados pela normal e $t_{v=10}$ multivariada possuem formas bem definidas. Já os grupos formados pela log-normal destacam-se pela grande variabilidade dos grupos.

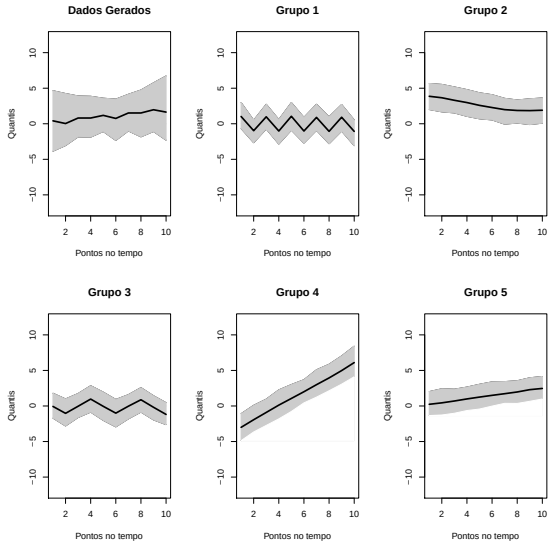
Esse padrão se reflete nos resultados do agrupamento. A Tabela 4.2 mostra o índice de Rand médio resultante do agrupamento de 200 simulações para cada cenário considerado. As Figuras 4.2 e 4.3 mostram os *box-plots* dos ARI's resultantes das simulações com: 3 observações; 5, 10 e 20 pontos no tempo; 1.000 e 4.000 variáveis, respectivamente. Em todas as simulações, o limiar escolhido foi de 5% (ponto a partir do qual o teste é rejeitado). De maneira geral, todos os algoritmos identificaram corretamente os grupos gerados pelas distribuições simétricas (normal multivariada e t multivariada), em especial o MCLUST, por ser um algoritmo que modela mistura de normais para realizar o agrupamento. Para esses casos, quando o número de variáveis aumenta e o número de pontos no tempo fica em torno de 10, os algoritmos são equivalentes.

Quando os dados seguem uma distribuição assimétrica (log-normal), percebe-se que os algoritmos com estimação Σ_i superam os procedimentos que utilizam a estimação Σ. Isso ocorre porque ao estimar uma matriz de covariância para cada variável, esses procedimentos captam melhor a grande variabilidade existente em cada grupo e, nesse caso, se torna o fator preponderante para detectá-los. De maneira geral, nota-se um melhor desempenho dos algoritmos PPCLUSTEL(Σ e Σ_i) e PPCLUSTEL-R (Σ e Σ_i) ao passarmos de 1.000 para 4.000 variáveis. Além do aumento do índice de Rand médio, também é possível perceber a diminuição da variância dos índices resultantes das simulações quando o número de variáveis aumenta (algoritmos mais estáveis).



(a) Normal Multivariada

(b) Log-normal Multivariada



(c) t_{10} Multivariada

Figura 4.1: Média, 5º e 95º percentil para os dados agregados e para os 5 grupos simulados considerando as distribuições: normal multivariada (a), log-normal multivariada (b) e t multivariada (c), com 1000 fatores, 10 pontos no tempo e 3 observações.

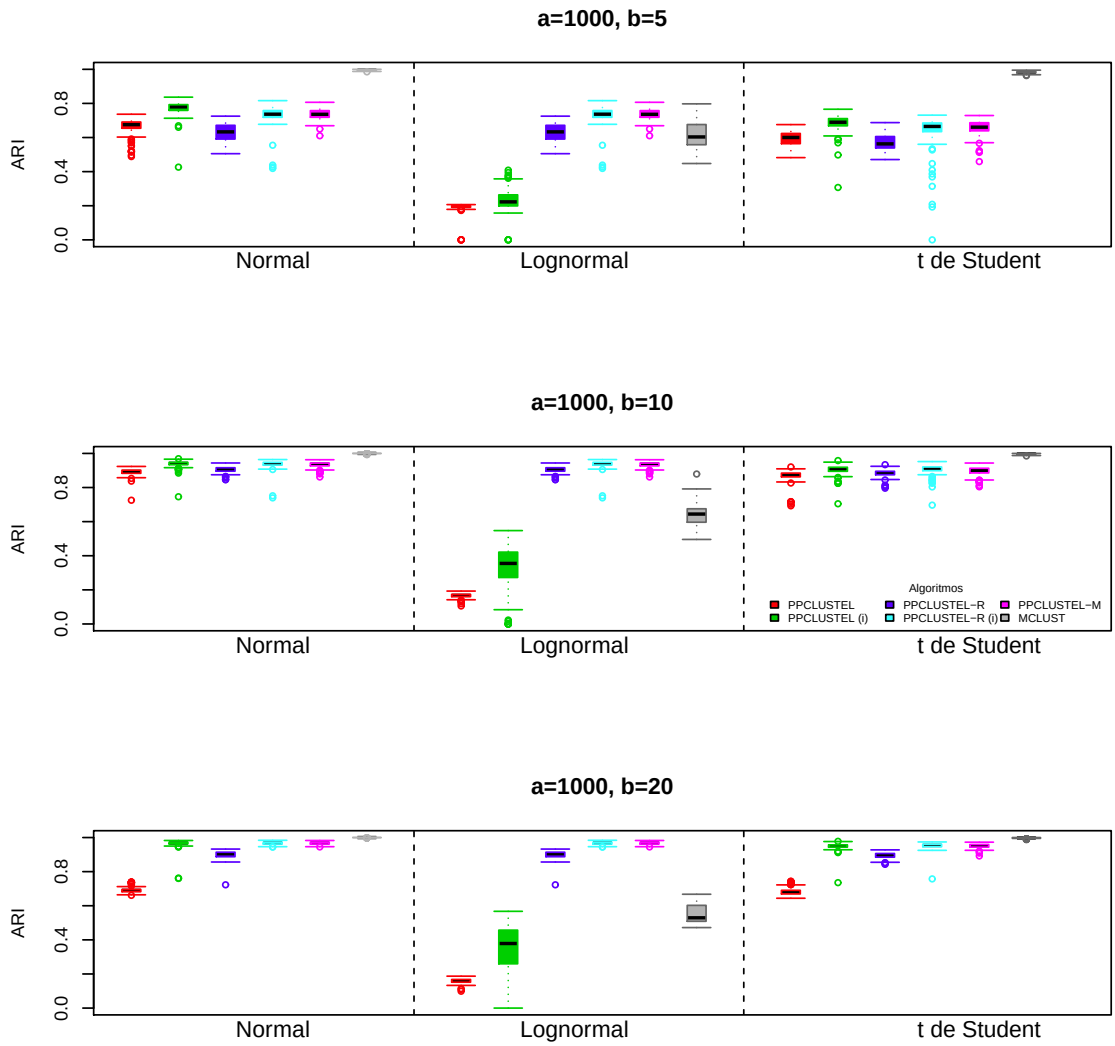


Figura 4.2: Boxplot's com os índices de Rand ajustados (ARI) considerando os algoritmos PPCLUSTEL (Σ), PPCLUSTEL (Σ_i), PPCLUSTEL-R (Σ), PPCLUSTEL-R (Σ_i), PPCLUSTEL-M e MCLUST, nessa ordem; as três distribuições multivariadas; 3 observações; 1000 variáveis e 5, 10 e 20 pontos no tempo.

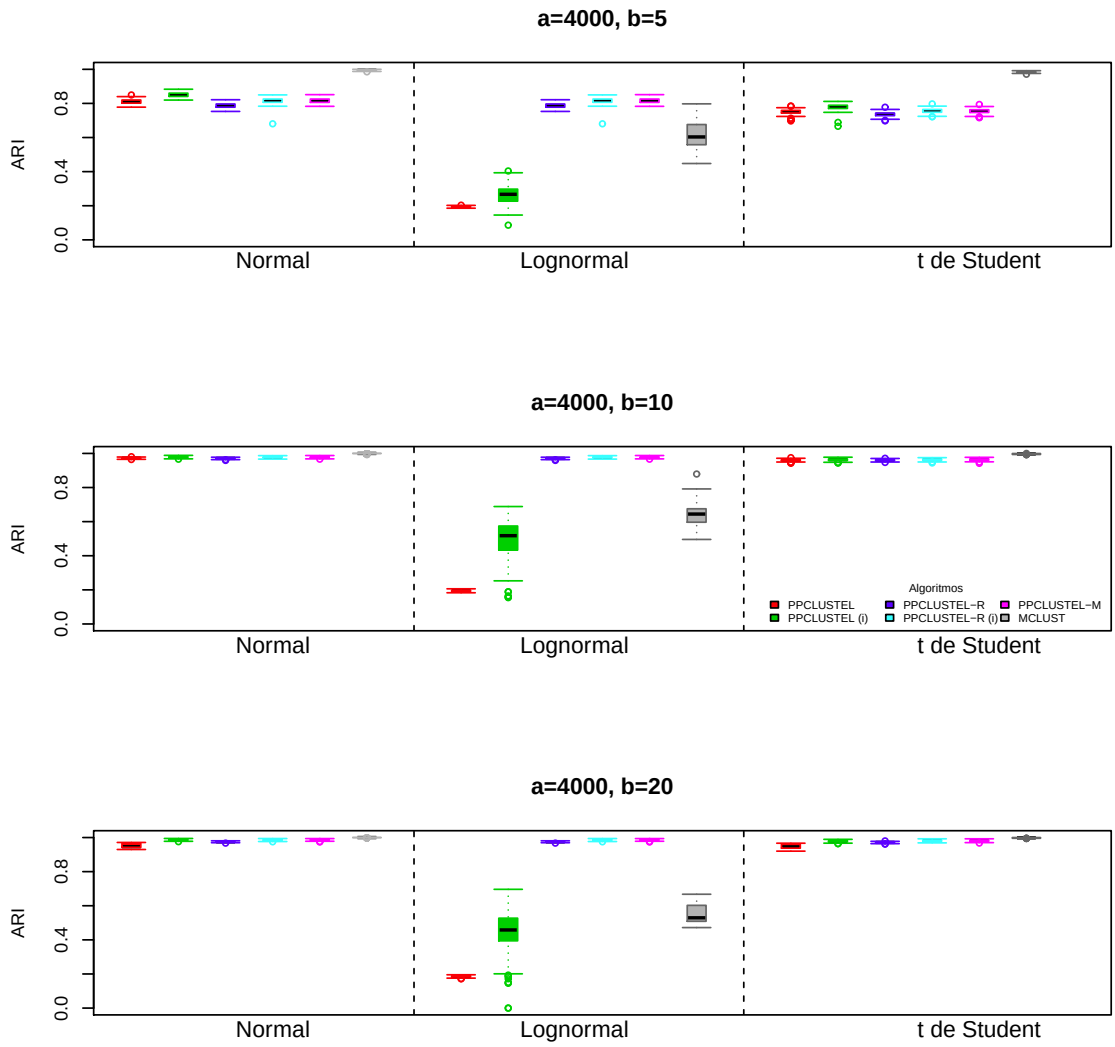


Figura 4.3: Boxplot's com os índices de Rand ajustados (ARI) considerando os algoritmos PPCLUSTEL (Σ), PPCLUSTEL (Σ_i), PPCLUSTEL-R (Σ), PPCLUSTEL-R (Σ_i), PPCLUSTEL-M e MCLUST, nessa ordem; as três distribuições multivariadas; 3 observações; 4000 variáveis e 5, 10 e 20 pontos no tempo.

Tabela 4.2: Resultado das simulações considerando 200 replicações e as distribuições multivariadas: normal, lognormal e t (10 g.l.). Para cada caso apresentado foram calculadas as médias dos ARI's.

PARÂMETROS			**PPCLUSTEL**						**PPCLUSTEL-R**				**PPCLUSTEL-M**		**MCLUST**		
			Σ			Σ_i			Σ		Σ_i						
a	b	n_i	*Normal*	*LogN*	T	*Normal*	*LogN*	T	*N/LogN*	T	*N/LogN*	T	*N/LogN*	T	*Normal*	*LogN*	T
1000	5	3	0,666	0,190	0,594	0,775	0,232	0,686	0,632	0,564	0,733	0,654	0,738	0,66	0,994	0,617	0,982
1000	5	5	0,866	0,197	0,822	0,914	0,412	0,857	0,874	0,833	0,908	0,859	0,905	0,861	1,000	0,657	0,998
1000	10	3	0,892	0,166	0,870	0,938	0,334	0,905	0,905	0,883	0,937	0,901	0,937	0,901	0,999	0,635	0,996
1000	10	5	0,942	0,191	0,938	0,963	0,579	0,951	0,951	0,947	0,965	0,962	0,968	0,962	1,000	0,696	0,999
1000	20	3	0,691	0,160	0,685	0,965	0,343	0,950	0,901	0,895	0,967	0,953	0,967	0,952	1,000	0,558	0,997
1000	20	5	0,778	0,181	0,776	0,983	0,426	0,974	0,932	0,934	0,983	0,972	0,983	0,979	1,000	0,561	0,999
4000	5	3	0,810	0,194	0,750	0,851	0,264	0,780	0,788	0,735	0,815	0,751	0,816	0,755	0,994	0,617	0,983
4000	5	5	0,942	0,202	0,906	0,964	0,515	0,927	0,941	0,911	0,958	0,928	0,958	0,928	1,000	0,657	0,998
4000	10	3	0,972	0,194	0,960	0,980	0,496	0,963	0,971	0,960	0,978	0,964	0,978	0,963	0,999	0,635	0,996
4000	10	5	0,983	0,283	0,982	0,988	0,735	0,985	0,984	0,983	0,984	0,986	0,988	0,986	1,000	0,696	0,999
4000	20	3	0,952	0,186	0,948	0,987	0,457	0,980	0,975	0,971	0,986	0,981	0,987	0,981	1,000	0,558	0,998
4000	20	5	0,975	0,197	0,974	0,993	0,628	0,990	0,984	0,984	0,993	0,991	0,993	0,991	1,000	0,561	0,999

Deve-se atentar ao fato de que, à exceção do MCLUST, os algoritmos apresentaram melhores resultados quando o número de pontos no tempo e o número de variáveis aumentaram. Portanto, ao analisarmos esse comportamento em conjunto com os resultados das simulações dos erros do tipo I dos testes presentes no PPCLUSTEL e PPCLUSTEL-R, podemos sugerir que as restrições de convergência desses testes não impactaram de maneira significativa a qualidade do agrupamento. Caso essas restrições tivessem exercido o efeito esperado, então o PPCLUSTEL e o PPCLUSTEL-R teriam apresentado melhores resultados nas simulações com 5 pontos no tempo (1.000 ou 4.000 varivéis) do que com 20 pontos no tempo e, no caso do PPCLUSTEL-M, teria ocorrido o efeito contrário. Dessa maneira, conclui-se que os algoritmos são robustos às situações que desfavorecem a convergência do teste.

Conforme visto ao longo do trabalho, os algoritmos com estimação Σ levam em consideração apenas a diferença com relação à média dos grupos. Os algoritmos com estimação Σ_i, por sua vez, também são sensíveis às variações da estrutura de covariâncias entre os grupos, porém essa propriedade é limitada em razão da falta de informação disponível para a eficaz estimação dos parâmetros dessas matrizes. Com o intuito de corrigir esses problemas, foi criado um novo procedimento de estimação que funciona como um método intermediário entre as formas Σ e Σ_i. Esse método, denotado por Σ_G, se baseia na ideia de permitir que o algoritmo estime diferentes covariâncias para variáveis (níveis de fator) em diferentes grupos e impor uma covariância constante para variáveis de um mesmo grupo. O método consiste em calcular a matriz de covariâncias com estimação Σ toda vez que o teste é aplicado em alguma partição do algoritmo. Dessa forma, as estimativas vão se atualizando à medida que o algoritmo avança e, ao final do procedimento, cada grupo terá uma estimativa da estrutura de covariâncias temporais própria.

Essa forma de estimação se assemelha ao método Σ no sentido de que o algoritmo continua sendo capaz de detectar apenas mudanças com relação à média das variáveis, pois as estimativas da matriz de covariâncias são fixadas para variáveis que estão sendo testadas em determinada partição do algoritmo. Por outro lado, esse método também utiliza a ideia presente na estimação Σ_i ao permitir certa flexibilidade para caracterizar a estrutura de covariâncias dos dados. A vantagem dessa forma de estimação é que, na maioria das vezes, o teste é aplicado em um número alto de

variáveis, de forma que há a garantia de informação suficiente para a estimação dos parâmetros de Σ_G.

PARÂM.		**PPCLUSTEL**			**PPCLUSTEL-R**				**PPCLUSTEL-M**			
		Σ_G			Σ_i		Σ_G		Σ_i		Σ_G	
a	b	N	LN	T	N/LN	T	N/LN	T	N/LN	T	N/LN	T
1000	5	0,80	0,37	0,72	0,73	0,65	0,76	0,69	0,74	0,7	0,76	0,69
1000	10	0,95	0,74	0,92	0,94	0,90	0,95	0,92	0,94	0,90	0,94	0,93
1000	20	0,96	0,66	0,96	0,97	0,95	0,97	0,96	0,97	0,95	0,97	0,96

Tabela 4.3: Resultado das simulações dos algoritmos com estimação Σ_G considerando 200 replicações, $n_i = 3$, e as distribuições multivariadas: normal, lognormal e t (10 g.l.). Para cada caso apresentado foram calculadas as médias dos ARI's.

Do ponto de vista metodológico, é razoável supor que cada grupo apresente uma estrutura de covariância própria, mas para verificar se essa suposição realmente se traduz em melhoria de desempenho, repetiu-se as simulações com os algoritmos PPCLUSTEL, PPCLUSTEL-R e PPCLUSTEL-M considerando a estimação Σ_G. A Tabela 4.3 apresenta os resultados dos índices de Rand ajustado (ARI). Foram consideradas nas simulações: 1000 variáveis; 5, 10 e 20 pontos no tempo; 3 observações e as 3 distribuições multivariadas.

Em geral, em todos os casos considerados, os procedimentos Σ_G foram superiores aos Σ_i. O algoritmo em que a melhoria foi mais acentuada foi o PPCLUSTEL. Por exemplo, note que a média dos ARI's do PPLCUSTEL (Σ_i) para $a = 1000$, $b = 10$ e $n_i = 3$ passou de 0,334 (Tabela 4.2) para 0,74 com o PPCLUSTEL (Σ_G).

O Tempo de Processamento

Em todas as simulações deste trabalho utilizou-se uma máquina com a seguinte configuração: processador Intel Core i5, CPU 760 @ 2.80 GHz, com 4GB de memória RAM. Plataforma: Windows 7 Ultimate, com arquitetura 64 bits. Todos os algoritmos foram implementados em linguagem macro de SAS©, versão 9.2, e estão disponíveis no Anexo C.

A Tabela 4.4 apresenta o tempo que os algoritmos levaram para realizar o agrupamento em uma base de dados com 4.000 variáveis, 10 pontos no tempo e 3 observações.

Foram selecionados apenas os algoritmos que apresentaram melhor desempenho no agrupamento de dados simulados, por isso não serão apresentados os procedimentos com estimação Σ.

Algoritmos	**Forma de Estimação**	
	Σ_i	Σ_G
PPCLUSTEL	17s	19min 43s
PPCLUSTEL-R	18s	19min 45s
PPCLUSTEL-M	1min 52s	25min 26s
MCLUST	1 min 13s	

Tabela 4.4: Tempo de processamento dos algoritmos que apresentaram os melhores resultados. Estudo verificado em uma base de dados com 4000 variáveis, 10 pontos no tempo e 3 observações.

Percebe-se que o tempo de processamento dos algoritmos depende basicamente da forma de estimação da matriz de covariâncias. Um importante aspecto que se observa é que o método *jackknife* é extremamente dispendioso computacionalmente quando há um grande número de observações disponíveis para a estimação dos parâmetros de covariância. Por esse motivo, os algoritmos que utilizam as formas de estimação Σ e Σ_G são mais demorados em comparação àqueles que utilizam Σ_i. Além disso, o método Σ_G, conforme explicado, atualiza as estimativas da matriz de covariâncias ao longo do procedimento, o que colabora com o baixo rendimento de processamento.

A Tabela 4.4 mostra ainda o tempo de processamento do algoritmo MCLUST[2]. Apesar de ser relativamente rápido, este algoritmo acusa problemas de alocação de memória quando aplicado em bases de dados com mais de 10.000 variáveis.

[2]Algoritmo rodado no *software* no R, Versão 2.13, 64 bits.

4.2 Resultados em Dados Reais

Nesta seção, os algoritmos que apresentaram os melhores resultados no estudo de simulação serão aplicados em dados de microarranjo e de sinais elétricos do cérebro proveniente de eletroencefalograma.

4.2.1 Análise de Microarranjo

Os dados utilizados nesta seção foram coletados por Gillespie et al. (2011). Foram estudadas três replicações de uma cepa contendo leveduras do tipo selvagem (*wild-type*) e uma cepa contendo leveduras portadoras de mutação sensível a variações de temperatura (*cdc13-1*). As expressões gênicas das leveduras foram amostradas inicialmente a 23 °C, e, então, 1, 2, 3 e 4 horas após um aumento de temperatura para 30 °C. Ao todo foram observados 10.928 genes de leveduras em cinco pontos no tempo, com três replicações e dois tipos de condições experimentais: cepas de leveduras sem acréscimo de temperatura (testemunha) e cepas com alteração de temperatura (tratamento). Esses dados estão disponíveis na base de dados *ArrayExpress*, com número de acesso: E-MEXP-1551.

O objetivo desse estudo é identificar grupos de genes de levedura que se expressaram de forma semelhante ao longo do tempo. Em outras palavras, deseja-se encontrar os genes que reagiram de forma semelhante quando submetidos à elevação de temperatura. No trabalho de Gillespie et al. (2011), os autores, primeiramente, reduziram a dimensão do estudo, selecionando os 50 genes que apresentaram uma maior variabilidade em sua expressão ao longo do tempo. Essa redução de dimensão possibilitou aos autores a utilização das técnicas de agrupamento tradicionais. Os algoritmos apresentadas neste trabalho foram projetados para lidar com a multidimensionalidade, portanto, não há necessidade de reduzir a dimensão dos dados, e, consequentemente, evita a perda desnecessária de informações.

Para verificar o efeito do tratamento na expressão gênica das leveduras portadoras da mutação *cdc13-1* foi necessário padronizar as observações da base com tratamento em relação às observações da base de dados de controle.

Seja X_{ijk} as expressões dos genes de leveduras mutantes e Y_{ijk} as expressões das leveduras do tipo selvagem, Gillespie et al. (2011) padronizou as expressões dos genes

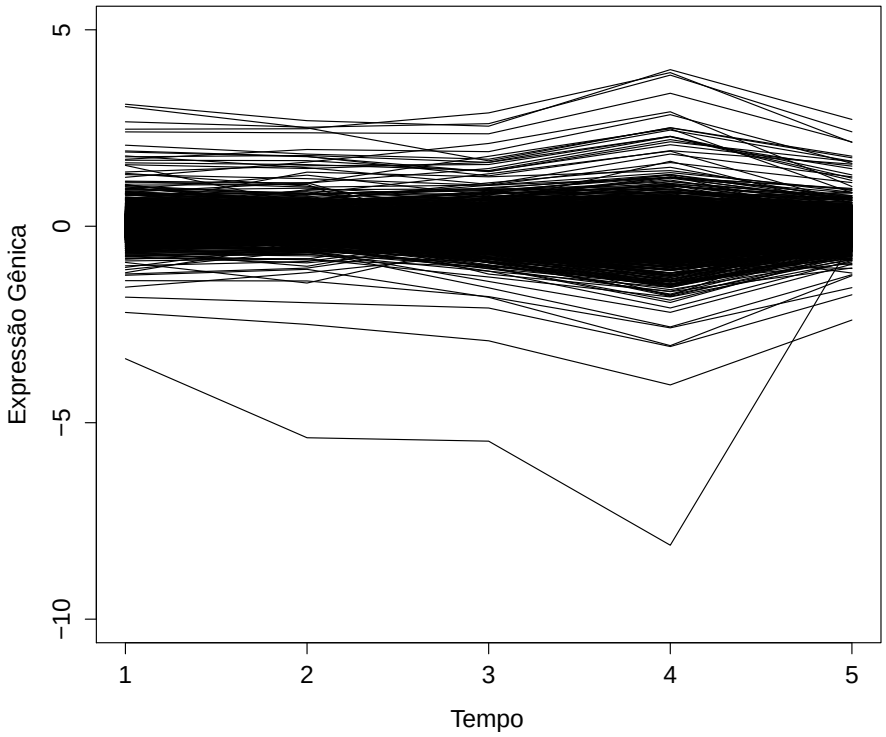


Figura 4.4: Expressão dos 10.928 genes ao longo de cinco mensurações no tempo.

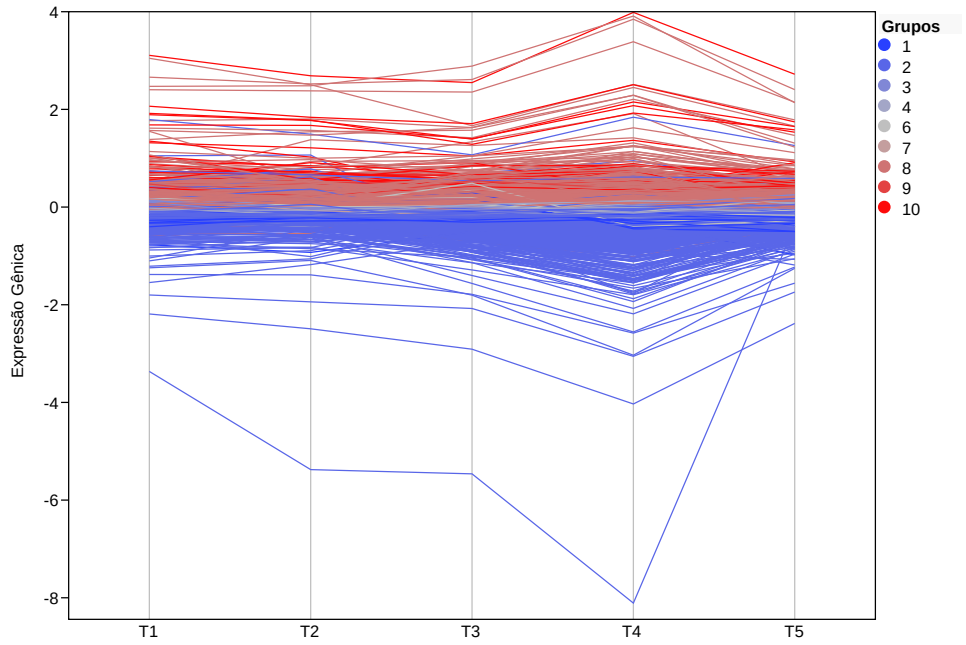
de leveduras mutantes em relação à média de todas as observações de controle, ou seja, $Y_{ijk} - \bar{X}_{...}$. Entretanto, como foi verificado que cada gene possui um nível de expressão gênico distinto, então preferimos utilizar a padronização de acordo com a mediana de cada gene, ou seja, $Y_{ijk} - \widetilde{X}_{i..}$, onde $\widetilde{X}_{i..}$ é a mediana do i-ésimo gene.

A Figura 4.4 mostra o efeito da temperatura nas expressões dos genes ao longo do tempo. Percebe-se que a maior parte dos genes concentrou-se em torno de zero durante todo o período em análise, ou seja, não reagiram ao aumento da temperatura.

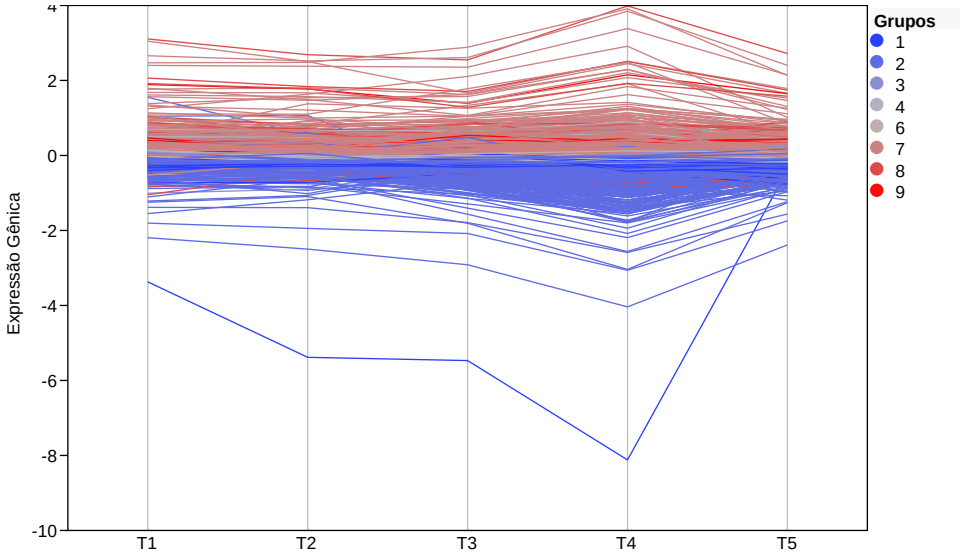
A Figura 4.5 mostra o resultado do agrupamento aplicando-se os procedimentos PPCLUSTEL - R e PPCLUSTEL - M. Utilizou-se em cada um dos algoritmos o procedimento descrito na Seção 3.5 para a escolha do limiar. O algoritmo foi aplicado com os limiares 10^{-1}, 10^{-2}, 10^{-3} até 10^{-8}. Verificou-se em ambos os casos que a partir de 10^{-3} o padrão dos grupos era o mesmo. Sempre haviam três grupos predominantes, o maior deles (sempre em torno de 60% do total genes) representando os genes que não reagiram ao longo do tempo. E os outros dois grupos cada um com mais ou menos

20% dos genes. Para facilitar a visualização, o grupo central contendo os genes que não se expressaram foram retirados do gráfico.

A semelhança dos resultados obtidos com os dois algoritmos nas simulações da seção anterior se confirma ao analisarmos os grupos formados por esses procedimentos. Enquanto o PPCLUSTEL - R encontrou dez grupos, o PPCLUSTEL - M encontrou nove. Percebe-se que ambos os procedimentos separaram basicamente os genes com expressão acima e abaixo de zero.



(a) PPCLUSTEL - R com limiar $\alpha = 10^{-5}$.



(b) PPCLUSTEL - M com limiar $\alpha = 10^{-8}$.

Figura 4.5: Agrupamento obtido com os respectivos algoritmos. Forma de estimação utilizada (Σ_G).

4.2.2 Análise de EEG

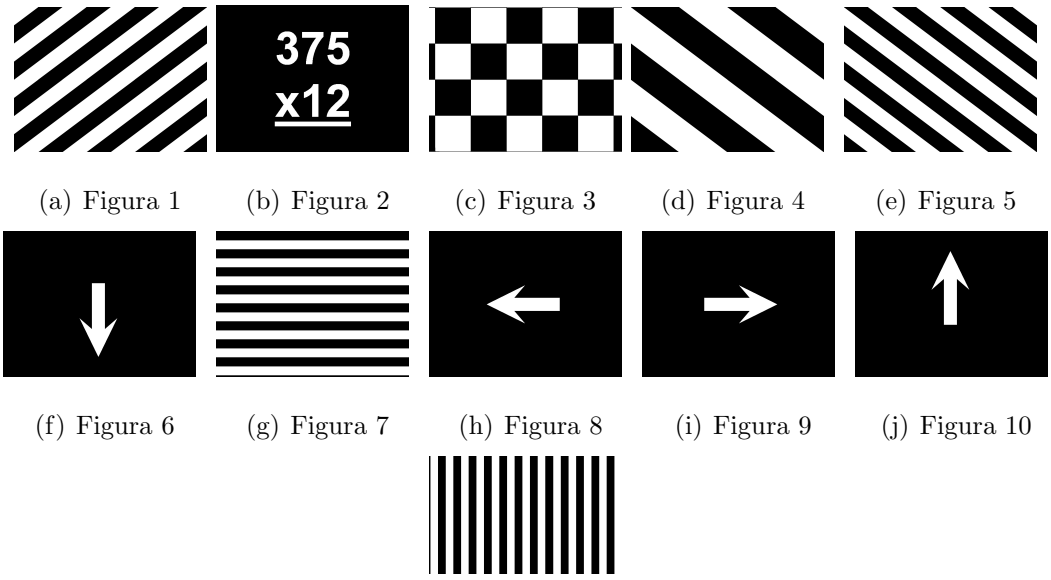
A base de dados utilizada nesta seção foi coletada no laboratório *Multi-Sensing-Processing and Learning* (MSPL) da Universidade do Texas em El Passo. Os dados consistem de sinais elétricos do cérebro (eletroencefalograma) captados por eletrodos em indivíduos submetidos a determinados estímulos visuais e auditivos. Uma breve descrição do protocolo de pesquisa será detalhada a seguir.

Figura 4.6: Indivíduo durante o experimento de coleta dos dados de EEG pelo MSPL Lab. - UTEP.

Inicialmente, é realizada uma triagem dos voluntários, sendo selecionados apenas aqueles em plenas condições físicas e mentais. Qualquer problema de saúde detectado eliminaria o participante. A Figura 4.6 mostra um voluntário com a touca utilizada para a coleta dos dados. Percebe-se que o método utilizado é não-invasivo, o que não oferece riscos aos participantes.

Cada touca possui 128 eletrodos que são igualmente posicionados nas cabeças dos participantes. O experimento se inicia com o indivíduo observando 11 figuras (Figura 4.7) e mais três estímulos auditivos (100 Hz, 2000 Hz e 44100 Hz). Cada estímulo é repetido cinco vezes e cada repetição dura 4 segundos. A ordem dos estímulos é aleatória.

Para cada repetição dos estímulos é registrado o sinal elétrico com duração de quatro segundos. Neste momento, a transformada de Fourier é utilizada para determinar a frequência máxima (f_{max}) do sinal registrado. Esta frequência encontrada determina a taxa máxima de reamostragem (Tx):



(a) Figura 1 (b) Figura 2 (c) Figura 3 (d) Figura 4 (e) Figura 5

(f) Figura 6 (g) Figura 7 (h) Figura 8 (i) Figura 9 (j) Figura 10

(k) Figura 11

Figura 4.7: Figuras observadas durante o experimento para a coleta dos sinais EEG.

$$Tx \leq \frac{1}{2f_{max}}.$$

A taxa de reamostragem em processamento de sinais é o intervalo de tempo usado para registrar os pontos do sinal, assim um sinal contínuo é transformado em um conjunto discreto de pontos. Dessa forma, a frequência máxima obtida pela transformada vai determinar o número de pontos usados para representar o sinal preservando as informações essenciais contidas no sinal original. Maiores detalhes da metodologia de pesquisa estão disponíveis em Frondana (2011).

A proposta do estudo com os sinais de EEG é desenvolver uma metodologia capaz de classificar corretamente um sinal desconhecido em alguma classe previamente estabelecida por uma amostra de treinamento (aprendizado supervisionado). Cada classe do estudo corresponde a um determinado estímulo recebido pelo participante (sons, imagens, dor, etc.).

Esse estudo encontra-se, ainda, em fase inicial, onde estão sendo testadas as primeiras técnicas estatísticas de classificação. Entre os trabalhos realizados até o momento, Coutinho (2011) foi o que obteve resultados mais expressivos. Esse autor obteve um bom índice de acertos utilizando a técnica *Support Vector Machine* como classificadora. Além disso, esse autor identificou que os eletrodos que fornecem os

sinais mais expressivos para o melhor desempenho da classificação são os que estão na parte frontal da cabeça, que é a região do cérebro responsável pela visão. Outro trabalho de destaque foi o de Frondana (2011), onde o método de cadeias de Markov ocultas foi utilizado como classificador. A autora verificou que o desempenho do procedimento melhora consideravelmente quando a técnica é aplicada em subgrupos de figuras.

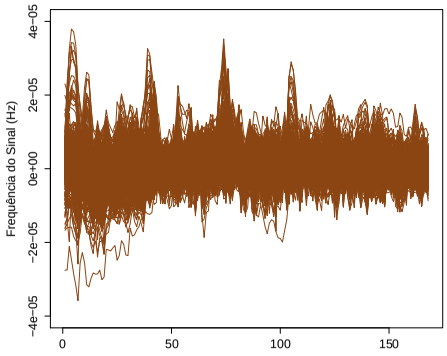
Para adequar a estrutura dos dados de EEG à estrutura HDLLSS da Tabela 3.1 considerou-se como variável cada combinação entre estímulo e eletrodo. Como temos 168 pontos no tempo, 128 eletrodos e mais 14 estímulos, então a base de dados HDLLSS contém 1.792 variáveis (cada figura com 128 eletrodos), com 5 repetições avaliadas ao longo dos 168 pontos no tempo.

O algoritmo utilizado foi o PPCLUSTEL-M (Σ_G), pois este é indicado para lidar com grande número de pontos no tempo. Apesar de o resultado das simulações ter dado indício de que as condições de convergência dos testes não exercem um papel preponderante para um melhor desempenho dos algoritmos, justifica-se a escolha desse algoritmo pelo fato de não ter sido abordado nas referidas simulações nenhum caso com mais de 20 pontos no tempo.

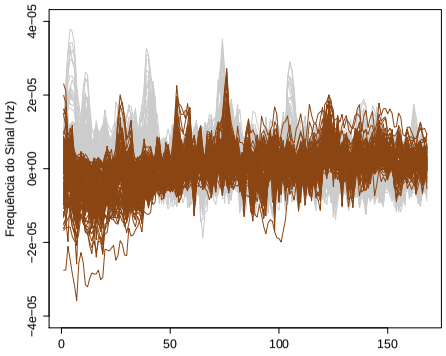
Em uma primeira análise, verificou-se o PPCLUSTEL-M não conseguia detectar os grupos de estímulos corretamente. Entretanto, para diversos limiares diferentes, um subgrupo de cinco estímulos se destacava dos demais: som com frequência 100 Hz, som com frequência 2000 Hz, figura 1, figura 5 e figura 7. A Figura 4.8 mostra os sinais de EEG de cada um desses estímulos. É possível perceber a completa ausência de padrão dos sinais.

Para nos certificarmos que o algoritmo realmente consegue detectar as diferenças entre os estímulos citados, repetimos o procedimento considerando apenas o subgrupo mostrado na Figura 4.8.

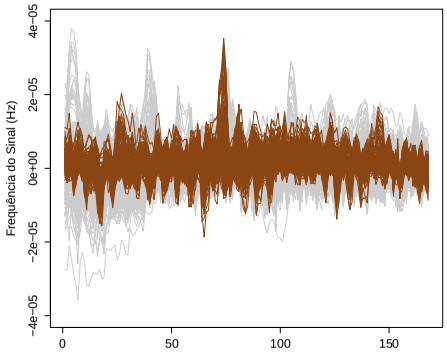
O resultado do agrupamento está disposto na Tabela 4.5. Nessa tabela pode ser verificada a sobreposição das figuras com os grupos encontrados. Percebe-se que do total de cinco estímulos, o algoritmo conseguiu detectar bem três deles: o som de 100 Hz (estímulo 1), a figura 1 (estímulo 3) e a figura 7 (estímulo 5). A figura 5 (estímulo 4) teve a maior parte dos sinais alocados no Grupo 4, grupo identificado ao estímulo da figura 1, o que mostra a dificuldade que o algoritmo encontrou em diferenciar essas



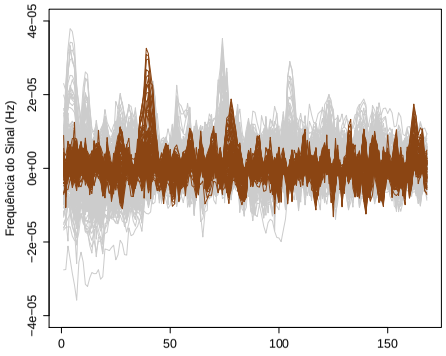
(a) Todos os Sinais



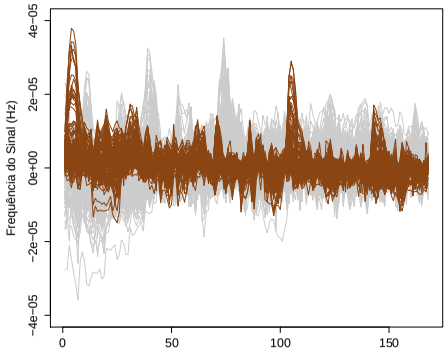
(b) Som 100 Hz.



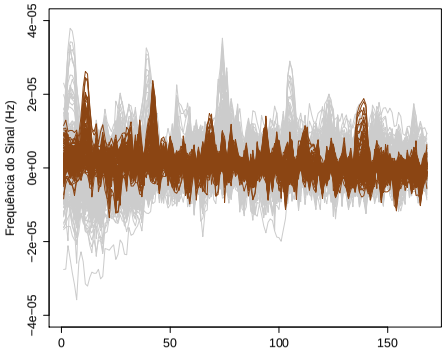
(c) Som 2000 Hz.



(d) Figura 1



(e) Figura 5



(f) Figura 7

Figura 4.8: Sinais de EEG obtidos dos cinco estímulos utilizados no agrupamento.

figuras, provavelmente, por serem muito parecidas.

	Freq. / Linha (%) / Col. (%)	Grupos 1	2	3	4	5	**Total**
Estímulos	1	**103**	0	0	11	14	128
		80,47	0	0	8.59	10,94	
		93,64	0	0	4,58	12,28	
	2	4	68	7	49	0	128
		3,125	53,12	5,47	38,28	0	
		3,64	53,13	14,58	20,42	0	
	3	0	25	0	**103**	0	128
		0	19,53	0	**80,47**	0	
		0	19,53	0	42,92	0	
	4	0	33	39	52	4	128
		0	25,78	30,47	40,63	3,13	
		0	25,78	**81,25**	21,67	3,51	
	5	3	2	2	25	**96**	128
		2,34	1,56	1,56	19,53	75	
		2,73	1,56	4,17	10,42	**84,21**	
	Total	110	128	48	240	114	640

Tabela 4.5: Resultado do agrupamento com o algoritmo PPCLUSTEL-M. Foi considerada a forma de estimação Σ_G e o limiar $\alpha = 0{,}1$.

Capítulo 5

Conclusão

Neste trabalho, foram propostos dois novos algoritmos na literatura: o PPCLUSTEL - R e o PPCLUSTEL - M. O PPCLUSTEL - R se destina a agrupar dados longitudinais, com grande número de variáveis e um número fixo de pontos no tempo. Já o PPCLUSTEL - M é indicado para agrupar dados onde o número de variáveis é fixo e o número de pontos no tempo, alto. Ambos os procedimentos se baseiam na utilização do p-valor resultante de um teste de ausência de efeito simples de grupo em um delineamento fatorial como medida de similaridade em agrupamento de dados HDLLSS. Baseado nas possibilidades de estimação da matriz de covariância dos dados, foram sugeridas três variações desses algoritmos: o PPCLUSTEL (Σ_i, Σ, Σ_G), o PPCLUSTEL-R (Σ_i, Σ, Σ_G) e o PPCLUSTEL-M (Σ_i, Σ, Σ_G).

Os resultados apresentados mostraram que a eficiência dos algoritmos não depende das condições de convergência dos testes[1], mas da distribuição dos dados e, principalmente, da qualidade de estimação da matriz de covariâncias. Por esse motivo, este trabalho se concentrou na elaboração de uma solução para o problema da falta de informação disponível para a estimação daquela matriz, levando-se em consideração a necessidade de detectar as mudanças na estrutura de covariâncias dos grupos.

A forma de estimação Σ_i estima uma matriz de covariâncias para cada variável. Essa forma de estimação possibilita ao algoritmo detectar as mudanças com relação à estrutura de covariâncias dos grupos, porém essa capacidade é limitada em virtude

[1]Condições de convergência do PPCLUSTEL/PPCLUSTEL-R e PPCLUSTEL-M: $a \to \infty$, b fixo e $b \to \infty$, a fixo, respectivamente.

da falta de replicações disponíveis em cada variável (n_i pequeno). Os métodos Σ e Σ_G supõem que um grupo de variáveis possuem um mesmo comportamento ao longo do tempo. Isso permite utilizar uma maior quantidade de informações para a estimação da matriz de covariâncias. Ao passo que a estimação Σ considera que todas as variáveis do sistema possuem a mesma estrutura de covariâncias, a estimação Σ_G considera que cada grupo possui uma estrutura de covariâncias distinta.

Foram apresentados os resultados das simulações do erro do tipo I e do poder do teste PPCLUSTEL E PPCLUSTEL-R, em diferentes situações. Em ambos os procedimentos o erro do tipo I apresentou resultados satisfatórios para Σ e uma grande dependência do tamanho amostral para a manutenção do nível nominal quando é utilizado Σ_i. As curvas de poder mostraram que o teste é eficaz em detectar pequenas diferenças na média mesmo quando a distribuição dos dados é assimétrica (caso da log-normal) e com cauda pesada ($t_{v=10}$ de student). Como o PPCLUSTEL-R utiliza os dados organizados em postos, então foi realizado um estudo onde se verificou a validade da convergência do teste em diferentes situações. Os resultados mostraram que a convergência do PPCLUSTEL continua ocorrendo para o PPCLUSTEL-R e que esta se verifica mesmo para as situações diferentes das quais o teste foi projetado.

Quanto ao desempenho dos procedimentos em dados simulados, verificou-se que os algoritmos que utilizam a estimação Σ_G superam os procedimentos com estimação Σ e Σ_i, e que o MCLUST tende a superar ambos os algoritmos apenas quando os dados seguem distribuições simétricas e o número de pontos no tempo gira em torno de 5. Pelo fato de estimar uma matriz de variâncias e covariâncias para cada grupo, os procedimentos (Σ_G) são mais indicados quando há grande variabilidade entre os grupos, ou quando não se tem certeza sobre esse comportamento.

Os resultados da aplicação em sinais de EEG mostraram ser possível, ainda que em estágio exploratório, a identificação, por meio de sinais de EEG, de alguns estímulos recebidos por um indivíduo . Ressaltamos que, por ser uma pesquisa ainda incipiente, o método de coleta e extração dos dados ainda precisa ser aperfeiçoado para que o procedimento de agrupamento seja eficaz.

5.1 Estudos Futuros

Como pudemos perceber, existem diversos problemas que surgem ao lidarmos com dados HDLLSS. Neste trabalho, apenas uma parte desses problemas foram contornados e muitos outros ainda precisam ser considerados. Portanto, esperamos que esse trabalho sirva como incentivo a futuros trabalhos. Com esse intuito, sugerimos os seguintes tópicos como propostas a pesquisas futuras:

- Verificação do desempenho dos testes não-paramétricos propostos em outras distribuições contínuas e, também, discretas, como: Laplace, Cauchy, Poisson, Gama, enter outras;

- Estudo do impacto no desempenho do teste (erro do tipo I e poder) e no algoritmo de agrupamento ao se utilizar nas simulações grupos gerados com diferentes estruturas de covariâncias;

- Em estudos de microarranjo, uma grande preocupação dos geneticistas está em verificar as relações existentes entre genes, em um processo conhecido como regulação gênica. Uma possível solução para essa questão seria estender os testes presentes no PPCLUSTEL e PPCLUSTEL-R para considerar, além da estrutura de covariância temporal, a covariância entre variáveis.

- Comparação do PPCLUSTEL - R e do PPCLUSTEL - M em simulações que considerem situações mais favoráveis ao PPCLUSTEL - M, ou seja, com um número maior de pontos no tempo.

- Desenvolvimento de uma metodologia para a escolha do limiar que forneça o agrupamento mais verossímil. Uma possibilidade seria utilizar uma medida externa de avaliação de agrupamento, como a *Root Mean Square* (RMS), que se baseia nas somas dos quadrados dentro e entre cada grupo, para comparar as partições obtidas com diferentes limiares.

- No âmbito das pesquisas com sinais de EEG, considerar estímulos que produzam reações mais distinguíveis no padrão elétrico cerebral do indivíduo.

Referências Bibliográficas

ALIZADEH, A., EISEN M. B., DAIS R. E. (2000). Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, **403**, 503511.

ALLISON, D. B.; G. P. PAGE; BEASLEY, T. M.; e EDWARDS J. W.; . *DNA microarrays and related genomics techniques: design, analysis, and interpretation of experiments*. Chapman and Hall/CRC, 2006.

ANDERBERG, M. R. *Cluster Analysis for Applications*. Academic Press, 1973.

ASLAN, K.; BOZDEMIR, H.; SAHIN, C.; OGULATA, S. N.; EROL, R. (2008). A Radial Basis Function Neural Network Model for Classification of Epilepsy Using EEG Signals. *Journal Medical System,* **32**, 403-408.

BERGER, H. (1929). ber das Elektroenkephalogram des Menschen. *Arch. f. Psychiatry,* **87**, 527-570.

BERRY, M. J. A.; LINOFF, G. *Data mining techniques for marketing, sales and customer support*. John Wiley & Sons, Ltd, 1997.

BOX, G. E. P.; JENKINS, G. M.. *Time Series Analysis: forecasting and control*. San Francisco, Holden-Day, 1976.

CARVALHO, B. S. *Análise de Microarranjos empregando ferramentas do Projeto Bioconductor*. 18 SINAPE/Associação Brasileira de Estatstica - ABE, 2008.

CORDUAS, M.; PICCOLO, D. (2008). Time series clustering and classification by the regressive metric. *Computacional Statistics & Data Analysis,* **52**, 1860-1872.

COUTINHO, M. *Classificação de dados de Eletroencefalografia (EEG) usando Support Vector Machine (SVM)*. Monografia, Universidade de Brasília, 2011.

CRYER, J. D.; CHAN K-S. *Time Series Analysis: With Applications in R.* Springer, Second Edition, 2009.

FAN, J.; ZHANG, J. (2000). Two-step estimation of functinal linear models with applications to longitudinal data. *Journal of Royal Statistical Society, B* **62,** 303-322.

FIGUEIREDO, C. O. *O uso de misturas de distribuições normais no agrupamento de dados superdimensionados com amostras pequenas.* Monografia, Universidade de Brasília, 2008.

FRALEY, C.; RAFTERY, A. E. (2002). Model-based clustering, discriminant analysis, and density estimation. *J. American Statistical Association,* **97,** 611-631.

FRALEY, C.; RAFTERY, A. E. (2006). Mclust version 3.0: an R package for normal mixture modeling and model-based clustering. Technical Report No. 504, *University of Washington.*

FRALEY, C.; RAFTERY, A. E. (2007). Model-based Methods of Classification: Using the mclust Software in Chemometrics. *Journal of Statistical Software.* **18**, Issue 6.

FRONDANA, I. M. *Classificação de biopotenciais via cadeias de Markov ocultas.* Dissertação (Mestrado em Estatística), Universidade de Brasília, 2011.

GAN, G.; MA, C.; WU, J. (2007). Data Clustering: Theory, Algorithms and Applications. *SIAM.*

GILLESPIE, C. S.; LEI, G.; BOYS, R. J.; GREENALL, A.; WILKINSON, D. J. (2011). Analysing time course microarray data using Bioconductor: a case study using yeast2 Affymetrix arrays. *Oxford Bioinformatics,* **3**:81.

GULER, I.; UBEYLI, E. D. (2005). Adaptive neuro-fuzzy inference system for classification. *Journal of Neuroscience Methods*,**148**, 113121.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The elements of statistical learning: data mining, inference and prediction.* Springer, Second edition, 2009.

HUBERT, L.; ARABIE, P. (1985). Comparing partitions.*Journal of Classification,* **2**, 193-218.

JOO, Y.; CASELLA, G.; HOBERT, J. (2010). Bayesian model-based tight clustering for time course data. *Computational Statistics,* **25, No 1**, 17-38.

JOHNSON, R. A.; WICHERN, D. W. *Applied Multivariate Statistical Analysis.* Prentice Hall, Sixth Edition, 2002

KANNATHAL, N.; ACHARYA, U. R.; LIM, C. M.; SADASIVAN, P. K. (2005). Characterization of EEG: A comparative study. *Computer Methods and Programs in Biomedicine,* **80**, 17-23.

KAUFMAN, L.; ROUSSEEUW, P. J. *Finding groups in data: An introduction to cluster analysis.* Wiley Interscience, 1990.

KIM, J.; KIM, H. (2008). Clustering of change patterns using Fourier coefficients. *Oxford Bioinformatics,* **Vol 24, No 2**, 184-191.

KIM, S. B.; RATTAKORN, P.; PENG, Y. B. (2010). An effective clustering procedure of neuronal response profiles in graded thermal stimulation. *Expert Systems with Applications,* **37**, 5818-5826.

KRIEGEL, H-P.; KRGER, P.; ZIMEK, A. (2009). Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-Based Clustering, and Correlation Clustering. *ACM Transactions on Knowledge Discovery from Data,* **3**, Issue 1.

KRISHNA, R.; LI C-T.; BUCHANAN-WOLLASTON, V. (2010). A temporal precedence based clustering method for gene expression microarray data. *BMC Bioinformatics,* **11**:68.

LEE, N. K.; WANG, D. (2011). SOMEA: self-organizing map based extraction algorithm for DNA motif identification with heterogeneous model. *BMC Bioinformatics,* **12**(Suppl 1):S16.

LIU, X.; YANG, M. C. K. (2009). Simultaneous curve registration and clustering for functional data. *Computational Statistics and Data Analysis,* **53**, 1361-1376.

MA, P.; CASTILLO-DAVIS, C. I.; ZHONG, W.; LIU, J. S. (2006). A data-driven clustering method for time course gene expression data. *Nucleic Acids Research,* **34***(4)* 1261-1269.

MA, P.; ZHONG, W. (2008). Penalized Clustering of Large Scale Functional Data with Multiple Covariates. *Journal of the American Statistical Association,* **103**, No. 482, 625-636.

MA, P.; ZHONG, W.; FENG, Y.; LIU J. S. (2008). Bayesian Functional Data Clustering for Temporal Microarray Data. *International Journal of Plant Genomics,* **ID: 231897** .

MACQUEEN, J. B. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of fifth Berkeley symposium on mathematical statistics and probability.*

MAHARAJ, E. A.; ALONSO, A. M. (2007). Discrimination of locally stationary time series using wavelets. *Computacional Statistics & Data Analysis,* **52**, 879-895.

McNICHOLAS, P. D.; MURPHY, T. B. (2010). Model-based clustering of microarray expression data via latent Gaussian mixture models. *Oxford Bioinformatics,* **Vol 26, No 21**, 2705-2712.

MILLIGAN, G.; COOPER, M. C. (1986). A study of the comparability of external criteria for hierarchical cluster analysis. *Multivariate Behavioral Research,* **21**.

MINGOTI, S. A.*Análise de dados através de métodos de estatística multivariada: uma abordagem aplicada.* Editora UFMG, 2005.

OCAK, H. (2009). Automatic detection of epileptic seizures in EEG using discrete wavelet transform and approximate entropy. *Expert Systems with Applications,* **36**, 2027-2036.

OMBAO, H.; HO, M. R. (2006). Time-dependent frequency domain principal components analysis of multichannel non-stationary signals. *Computacional Statistics & Data Analysis,* **50**, 2339-2360.

ORHAN, U.; HEKIM, M.; OZER, M. (2011). EEG signals classification using the K-means clustering and a multilayer perceptron neural network model. *Expert Systems with Applications,* **38**, 13475-13481.

PAWITAN, Y. *In all likelihood: statistical modeling and inference using likelihood. Oxford, 2001.*

PEROU C. M.; JEFFREY S. S.; van de RIJN M.; REES C. A.; EISEN M. B.; TOSS D. T.; PERGAMENSCHIKOV A.; WILLIAMS C. F.; ZHU S.X.; LEE J. C. F.; LASHKARI D.; SHALON D.; BROWN P. O.; BOTSTEIN D. (1999). Distinctive gene expression patterns in human mammary epithelial cells and breast cancers. *Proc Natl Acad Sci USA*, **96**, 9212 - 9217.

PING, M.; ZHONG, W. (2008). Penalized Clustering of Large Scale Functional Data with Multiple Covariates. *Journal of American Statistical Association,* **103(482),** 625-636.

RAMONI, M.; SEBASTIANI, P.; KOHANE, P. (2002). Cluster analysis of gene expreesion dynamics. *Proc. Natl Acad. Sci. USA.* **99**.

RAND, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association,* **66**, 846-850.

RAZALI, N. M.; WAH, Y. B. (2011). Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics,* **2**, 21-33.

REIMANN, C.; FILZMOSER, P.; GARRET, R. G.; DUTTER, R. *Statistical Data Analysis Explained: Applied Environmental Statistics with R.* John Wiley & Sons, Ltd., 2008.

RUTKOWSKI, T. M.; CICHOCKI, A.; TANAKA, T.; RALESCU, A. L.; MANDIE, D. P. (2009). Clustering of Spectral Patterns based on EMD Components of EEG Channels with Applications to Neurophysiological Signals Separation. *Lecture Notes in Computer Science,* **Vol 5506**, 453-460.

SANTOS, J. M., EMBRECHTS. M. (2009). On the use of the Adjusted Rand Index as a Metric for Evaluating Supervised Classification. *Artificial Neural Networks - ICANN*, 175 - 184.

SERBAN, N.; WASSERMAN, L. (2005). Cats: clustering after transformation and smoothing. *Journal of the American Statistical Association,* **100**, 990-999.

SHAPIRO, S.S.; WILK, M. (1965). An analysis of variance test for normality (complete samples). *Biometrika,* **52**, 591611.

STOREY, J.; XIAO, W.; LEEK, J.; TOMPKINS, R.; DAVIS, R. (2005). Significance analysis of time course microarray experiments. *Proc. Natl Acad. Sci. USA.* **102**.

SUBASI, A.; ERÇELEBI, E. (2005). Classification of EEG signals using neural network and logistic regression. *Computer Methods and Programs in Biomedicine,* **78**, 87-99.

SUBASI, A. (2007). EEG signal classification using wavelet feature extraction and a mixture of expert model. *Expert Systems with Applications,* **32**, 1084-1093.

SUBASI, A.; GURSOY, M. I. (2010). EEG signal classification using PCA, ICA, LDA and support vector machine. *Expert Systems with Applications,***Vol 37**, 8659-8666.

THEODORIDIS, S.; KOUTROUMBAS, K. *Pattern recognition.* Fourth ed., Academic Press, 2009.

TSAI, C-A.; CHEN, J. J. (2009). Multivariate analysis of variance test for gene set analysis. *Oxford Bioinformatics,* **Vol 25, No 7**, 897-903.

TSAI, G-F.; QU, A. (2008). Testing the significance of cell-cycle patterns in time-course microarray data using nonparametric quadratic inference functions. *Computational Statistics and Data Analysis,* **52**, 1387-1398.

ÜBEYLI, E. D. (2009). Decision support systems for time-varying biomedical signals: EEG signals classification. *Expert Systems with Applications,***Vol 36**, 5818-5826.

ÜBEYLI, E. D. (2009). Combined neural network model employing wavelet coefficients for EEG signals classification. *Digital Signal Processing,***19**, 297-308.

VON BORRIES, G. F. *Partition clustering of high dimensional low sample size data based on p-values.* Tese (Doutorado em Estatstica) - Department of Statistics, Kansas State University, 2008.

VON BORRIES, G. F.; WANG, H. (2009). Partition clustering of high dimensional low sample size data based on p-values. *Computational Statistics and Data Analysis,* **53**, 3987-3998.

WANG, H.; NEILL, J.; MILLER, F. (2008). Nonparametric Clustering of Functional Data. *Statistics and Its Inference,* **1**, 47-62.

WANG, H. *Testing in Multi-factor heteroscedastic anova and repeated measures designs with large number of levels.* Tese (Doutorado em Estatstica) - Department of Statistics, The Pennsylvania State University, 2004.

WU, C.; CHIANG, C. (2000). Kernel smoothing on varying coefficient models with longitudinal dependent variable. *Statistica Sinica,* **10**, 433-456.

YI, S-G.; JOO, Y-J.; PARK, T. (2009). Rank-based clustering analysis for the time-course microarray data. *Journal of Bioinformatics and Computational Biology,* **Vol 7, No 1**, 75-91.

Apêndice A

Erros do Tipo I - PPCLUSTEL

A.1 Normal Multivariada

Parâmetros		PPCLUSTEL				PPCLUSTEL (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,015	0,021	0,023	0,026	0,024	0,028	0,029	0,027
20	10	0,013	0,021	0,023	0,019	0,014	0,017	0,023	0,019
20	20	0,017	0,019	0,017	0,019	0,003	0,006	0,015	0,017
20	40	0,011	0,015	0,012	0,018	0,000	0,001	0,005	0,009
100	5	0,016	0,015	0,019	0,011	0,013	0,015	0,021	0,012
100	10	0,011	0,017	0,011	0,016	0,004	0,013	0,010	0,015
100	20	0,009	0,015	0,009	0,014	0,001	0,005	0,007	0,009
100	40	0,013	0,010	0,013	0,015	0	0,001	0,005	0,007
500	5	0,013	0,016	0,010	0,011	0,005	0,011	0,010	0,011
500	10	0,013	0,010	0,017	0,012	0,003	0,007	0,007	0,011
500	20	0,009	0,010	0,008	0,008	0,000	0,002	0,005	0,005
500	40	0,011	0,011	0,009	0,011	0,000	0,001	0,001	0,004
1000	5	0,006	0,007	0,007	0,012	0,004	0,005	0,007	0,012
1000	10	0,011	0,010	0,009	0,013	0,003	0,005	0,008	0,011
1000	20	0,013	0,011	0,011	0,011	0	0,002	0,006	0,008
1000	40	0,011	0,011	0,007	0,003	0	0,001	0,001	0,002

Tabela A.1: Erro do tipo I obtido com 1500 simulações para a distribuição normal multivariada, $\alpha = 0,01$.

Parâmetros		PPCLUSTEL				PPCLUSTEL (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,068	0,061	0,071	0,070	0,077	0,063	0,077	0,073
20	10	0,055	0,065	0,072	0,073	0,044	0,059	0,070	0,072
20	20	0,053	0,065	0,062	0,065	0,023	0,039	0,050	0,057
20	40	0,051	0,049	0,060	0,057	0,003	0,015	0,029	0,044
100	5	0,050	0,057	0,061	0,053	0,043	0,056	0,061	0,053
100	10	0,045	0,053	0,057	0,056	0,022	0,038	0,051	0,055
100	20	0,054	0,057	0,049	0,053	0,007	0,026	0,033	0,048
100	40	0,053	0,043	0,059	0,055	0,002	0,006	0,023	0,041
500	5	0,046	0,063	0,049	0,045	0,027	0,055	0,048	0,045
500	10	0,045	0,048	0,045	0,053	0,015	0,033	0,041	0,050
500	20	0,052	0,043	0,042	0,042	0,008	0,016	0,028	0,038
500	40	0,053	0,053	0,053	0,055	0,001	0,005	0,019	0,043
1000	5	0,048	0,059	0,055	0,052	0,027	0,049	0,054	0,052
1000	10	0,044	0,046	0,056	0,047	0,019	0,031	0,045	0,045
1000	20	0,056	0,049	0,047	0,047	0,009	0,018	0,032	0,040
1000	40	0,061	0,056	0,043	0,010	0	0,009	0,015	0,041

Tabela A.2: Erro do tipo I obtido com 1500 simulações para a distribuição normal multivariada, $\alpha = 0,05$.

Parâmetros		PPCLUSTEL				PPCLUSTEL (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,121	0,103	0,118	0,122	0,116	0,108	0,125	0,129
20	10	0,109	0,113	0,124	0,128	0,083	0,103	0,121	0,125
20	20	0,103	0,118	0,117	0,111	0,048	0,078	0,099	0,106
20	40	0,103	0,100	0,109	0,103	0,019	0,033	0,073	0,083
100	5	0,103	0,108	0,111	0,101	0,086	0,104	0,110	0,102
100	10	0,097	0,117	0,106	0,103	0,055	0,093	0,097	0,101
100	20	0,101	0,111	0,101	0,105	0,034	0,065	0,074	0,098
100	40	0,102	0,097	0,101	0,113	0,012	0,027	0,065	0,088
500	5	0,087	0,113	0,107	0,102	0,061	0,103	0,101	0,102
500	10	0,091	0,099	0,098	0,113	0,048	0,076	0,085	0,107
500	20	0,099	0,087	0,093	0,093	0,033	0,047	0,069	0,080
500	40	0,109	0,110	0,085	0,105	0,009	0,025	0,057	0,086
1000	5	0,106	0,120	0,107	0,099	0,067	0,108	0,107	0,099
1000	10	0,089	0,106	0,106	0,097	0,045	0,075	0,098	0,093
1000	20	0,105	0,103	0,093	0,091	0,035	0,051	0,073	0,081
1000	40	0,123	0,111	0,098	0,019	0,008	0,025	0,045	0,083

Tabela A.3: Erro do tipo I obtido com 1500 simulações para a distribuição normal multivariada, $\alpha = 0,1$.

A.2 Lognormal Multivariada

Parâmetros		PPCLUSTEL				PPCLUSTEL (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	10	0,025	0,027	0,024	0,030	0,002	0,001	0,005	0,012
20	20	0,019	0,019	0,031	0,017	0,001	0,000	0,001	0,008
20	40	0,031	0,020	0,021	0,019	0	0,000	0,001	0,007
100	5	0,021	0,023	0,013	0,015	0	0,001	0,002	0,004
100	10	0,025	0,020	0,015	0,009	0	0,001	0,001	0,003
100	20	0,018	0,020	0,015	0,011	0	0	0	0,002
100	40	0,022	0,013	0,015	0,015	0	0	0,001	0,002
500	5	0,018	0,015	0,010	0,011	0	0	0,001	0,003
500	10	0,012	0,014	0,009	0,008	0	0	0,001	0,002
500	20	0,011	0,013	0,009	0,005	0	0	0	0,002
500	40	0,013	0,010	0,010	0,012	0	0	0	0,004
1000	5	0,013	0,015	0,011	0,005	0	0	0,001	0,003
1000	10	0,011	0,011	0,011	0,015	0	0	0	0,001
1000	20	0,010	0,007	0,007	0,007	0	0	0	0,001
1000	40	0,012	0,011	0,009	0,015	0	0	0	0,002

Tabela A.4: Erro do tipo I obtido com 1500 simulações para a distribuição lognormal multivariada, $\alpha = 0,01$.

Parâmetros		PPCLUSTEL				PPCLUSTEL (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,073	0,054	0,066	0,063	0,023	0,013	0,022	0,042
20	10	0,062	0,061	0,069	0,073	0,011	0,013	0,029	0,045
20	20	0,062	0,065	0,082	0,055	0,005	0,005	0,023	0,030
20	40	0,077	0,061	0,059	0,056	0,002	0,003	0,013	0,029
100	5	0,062	0,055	0,059	0,055	0,006	0,006	0,014	0,025
100	10	0,061	0,053	0,057	0,049	0,002	0,003	0,009	0,025
100	20	0,050	0,054	0,053	0,044	0	0,001	0,007	0,021
100	40	0,064	0,049	0,055	0,055	0	0	0,006	0,023
500	5	0,057	0,058	0,055	0,055	0,001	0	0,005	0,029
500	10	0,051	0,051	0,050	0,055	0	0,001	0,004	0,031
500	20	0,041	0,050	0,044	0,037	0	0,001	0,004	0,020
500	40	0,058	0,043	0,051	0,050	0	0	0,003	0,021
1000	5	0,042	0,053	0,055	0,059	0	0	0,007	0,028
1000	10	0,050	0,049	0,049	0,04	0	0	0,006	0,023
1000	20	0,052	0,045	0,050	0,041	0	0	0,002	0,017
1000	40	0,056	0,059	0,042	0,055	0	0	0,003	0,022

Tabela A.5: Erro do tipo I obtido com 1500 simulações para a distribuição lognormal multivariada, $\alpha = 0,05$.

Parâmetros		PPCLUSTEL				PPCLUSTEL (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,107	0,095	0,113	0,111	0,039	0,029	0,060	0,083
20	10	0,105	0,099	0,114	0,112	0,022	0,025	0,049	0,081
20	20	0,101	0,121	0,135	0,103	0,009	0,018	0,051	0,069
20	40	0,117	0,118	0,110	0,106	0,007	0,011	0,033	0,060
100	5	0,107	0,104	0,099	0,101	0,009	0,013	0,039	0,069
100	10	0,093	0,100	0,099	0,088	0,008	0,010	0,028	0,055
100	20	0,097	0,106	0,094	0,091	0,001	0,006	0,029	0,053
100	40	0,113	0,099	0,101	0,099	0,001	0,001	0,023	0,057
500	5	0,098	0,101	0,095	0,105	0,001	0,004	0,029	0,067
500	10	0,097	0,087	0,098	0,100	0,001	0,003	0,021	0,065
500	20	0,086	0,103	0,092	0,081	0,001	0,002	0,016	0,044
500	40	0,098	0,109	0,107	0,097	0	0,001	0,016	0,051
1000	5	0,085	0,102	0,104	0,098	0,001	0,003	0,027	0,062
1000	10	0,095	0,095	0,103	0,095	0	0,003	0,022	0,067
1000	20	0,104	0,095	0,091	0,083	0	0,001	0,015	0,045
1000	40	0,110	0,098	0,087	0,095	0	0	0,013	0,059

Tabela A.6: Erro do tipo I obtido com 1500 simulações para a distribuição lognormal multivariada, $\alpha = 0,1$.

A.3 $t_{v=10}$ Multivariada

Parâmetros		PPCLUSTEL				PPCLUSTEL (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,014	0,017	0,013	0,018	0,020	0,020	0,021	0,019
20	10	0,012	0,019	0,019	0,022	0,009	0,016	0,017	0,023
20	20	0,011	0,016	0,014	0,020	0,001	0,003	0,011	0,017
20	40	0,007	0,015	0,014	0,011	0	0,000	0,003	0,006
100	5	0,011	0,019	0,015	0,017	0,008	0,017	0,015	0,019
100	10	0,007	0,015	0,015	0,019	0,001	0,008	0,012	0,017
100	20	0,009	0,013	0,010	0,012	0,000	0,002	0,006	0,008
100	40	0,007	0,013	0,008	0,006	0,000	0,000	0,002	0,003
500	5	0,010	0,009	0,017	0,012	0,003	0,008	0,015	0,012
500	10	0,011	0,011	0,011	0,009	0,001	0,004	0,010	0,009
500	20	0,007	0,012	0,011	0,006	0	0,003	0,003	0,006
500	40	0,006	0,009	0,013	0,011	0	0	0,003	0,005
1000	5	0,011	0,010	0,013	0,013	0,003	0,006	0,010	0,013
1000	10	0,006	0,008	0,012	0,015	0	0,003	0,009	0,013
1000	20	0,009	0,009	0,006	0,010	0	0,001	0,002	0,006
1000	40	0,007	0,010	0,009	0,025	0	0	0	0,004

Tabela A.7: Erro do tipo I obtido com 1500 simulações para a distribuição $t_{v=10}$ multivariada, $\alpha = 0,01$.

Parâmetros		PPCLUSTEL				PPCLUSTEL (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,057	0,061	0,068	0,059	0,060	0,067	0,071	0,059
20	10	0,055	0,059	0,057	0,071	0,029	0,050	0,052	0,067
20	20	0,047	0,051	0,057	0,068	0,009	0,025	0,044	0,058
20	40	0,041	0,062	0,058	0,057	0,001	0,011	0,023	0,037
100	5	0,059	0,055	0,064	0,059	0,036	0,048	0,065	0,061
100	10	0,053	0,062	0,059	0,071	0,013	0,039	0,051	0,069
100	20	0,057	0,050	0,052	0,060	0,006	0,018	0,036	0,050
100	40	0,042	0,061	0,047	0,049	0,000	0,007	0,017	0,029
500	5	0,057	0,053	0,051	0,064	0,019	0,041	0,047	0,064
500	10	0,049	0,057	0,059	0,052	0,013	0,034	0,051	0,046
500	20	0,051	0,057	0,049	0,043	0,003	0,017	0,031	0,033
500	40	0,045	0,041	0,044	0,045	0,000	0,004	0,015	0,024
1000	5	0,049	0,049	0,059	0,048	0,040	0,056	0,048	0,007
1000	10	0,051	0,050	0,055	0,055	0,026	0,044	0,053	0,003
1000	20	0,056	0,045	0,049	0,045	0,014	0,027	0,037	0,001
1000	40	0,039	0,043	0,040	0,045	0,001	0,005	0,015	0,023

Tabela A.8: Erro do tipo I obtido com 1500 simulações para a distribuição $t_{v=10}$ multivariada, $\alpha = 0,05$.

Parâmetros		PPCLUSTEL				PPCLUSTEL (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,112	0,106	0,115	0,099	0,100	0,101	0,117	0,101
20	10	0,099	0,103	0,107	0,117	0,063	0,084	0,097	0,114
20	20	0,089	0,114	0,115	0,109	0,029	0,055	0,084	0,101
20	40	0,091	0,101	0,108	0,112	0,007	0,035	0,059	0,082
100	5	0,124	0,103	0,103	0,115	0,088	0,089	0,099	0,115
100	10	0,097	0,109	0,115	0,125	0,045	0,081	0,101	0,118
100	20	0,108	0,097	0,092	0,109	0,022	0,046	0,070	0,097
100	40	0,088	0,113	0,097	0,097	0,005	0,029	0,045	0,069
500	5	0,095	0,109	0,103	0,113	0,068	0,090	0,097	0,113
500	10	0,098	0,105	0,101	0,101	0,035	0,071	0,087	0,099
500	20	0,101	0,104	0,098	0,097	0,013	0,047	0,067	0,078
500	40	0,086	0,098	0,091	0,085	0,003	0,019	0,043	0,061
1000	5	0,092	0,093	0,105	0,103	0,055	0,077	0,099	0,099
1000	10	0,095	0,107	0,101	0,106	0,037	0,069	0,087	0,100
1000	20	0,106	0,095	0,093	0,088	0,020	0,040	0,072	0,073
1000	40	0,083	0,078	0,084	0,105	0,001	0,013	0,038	0,063

Tabela A.9: Erro do tipo I obtido com 1500 simulações para a distribuição $t_{v=10}$ multivariada, $\alpha = 0,1$.

Apêndice B

Erros do Tipo I - PPCLUSTEL-R

B.1 Normal Multivariada

Parâmetros		PPCLUSTEL-R				PPCLUSTEL-R (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,018	0,019	0,022	0,023	0,025	0,029	0,024	0,025
20	10	0,014	0,017	0,019	0,021	0,013	0,017	0,020	0,022
20	20	0,012	0,015	0,019	0,019	0,002	0,003	0,016	0,016
20	40	0,014	0,014	0,012	0,020	0,000	0,000	0,004	0,010
100	5	0,015	0,012	0,018	0,012	0,014	0,013	0,020	0,012
100	10	0,009	0,017	0,014	0,016	0,004	0,012	0,013	0,014
100	20	0,009	0,017	0,011	0,016	0,001	0,009	0,006	0,010
100	40	0,018	0,010	0,013	0,016	0,000	0,000	0,005	0,006
500	5	0,009	0,007	0,014	0,007	0,005	0,006	0,014	0,007
500	10	0,014	0,007	0,010	0,011	0,005	0,005	0,007	0,010
500	20	0,015	0,007	0,010	0,008	0,001	0,003	0,003	0,007
500	40	0,011	0,012	0,009	0,015	0,000	0,001	0,001	0,004
1000	5	0,015	0,020	0,010	0,010	0,004	0,012	0,008	0,012
1000	10	0,010	0,020	0,009	0,015	0,002	0,007	0,007	0,011
1000	20	0,020	0,010	0,005	0,010	0,000	0,003	0,004	0,009
1000	40	0,010	0,005	0,015	0,015	0,000	0,000	0,002	0,003

Tabela B.1: Erro do tipo I obtido com 1000 simulações para a distribuição normal multivariada, $\alpha = 0,01$.

Parâmetros		PPCLUSTEL-R				PPCLUSTEL-R (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,061	0,054	0,063	0,066	0,069	0,061	0,072	0,073
20	10	0,060	0,056	0,065	0,071	0,046	0,052	0,060	0,071
20	20	0,038	0,050	0,061	0,058	0,017	0,027	0,043	0,055
20	40	0,055	0,051	0,056	0,058	0,005	0,012	0,025	0,043
100	5	0,044	0,049	0,050	0,053	0,038	0,046	0,051	0,053
100	10	0,042	0,049	0,058	0,058	0,021	0,037	0,051	0,053
100	20	0,052	0,060	0,056	0,058	0,008	0,031	0,036	0,051
100	40	0,054	0,033	0,067	0,055	0,003	0,005	0,024	0,036
500	5	0,046	0,052	0,050	0,045	0,032	0,051	0,050	0,045
500	10	0,048	0,050	0,045	0,053	0,020	0,033	0,039	0,050
500	20	0,051	0,046	0,044	0,041	0,011	0,017	0,029	0,035
500	40	0,050	0,052	0,052	0,080	0,001	0,003	0,022	0,045
1000	5	0,060	0,060	0,060	0,040	0,034	0,055	0,055	0,052
1000	10	0,045	0,055	0,058	0,050	0,023	0,035	0,050	0,046
1000	20	0,040	0,055	0,040	0,035	0,014	0,022	0,026	0,039
1000	40	0,070	0,050	0,050	0,065	0	0,007	0,014	0,037

Tabela B.2: Erro do tipo I obtido com 1000 simulações para a distribuição normal multivariada, $\alpha = 0,05$.

Parâmetros		PPCLUSTEL-R				PPCLUSTEL-R (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,107	0,102	0,112	0,111	0,116	0,106	0,122	0,115
20	10	0,105	0,112	0,112	0,122	0,085	0,100	0,108	0,119
20	20	0,094	0,108	0,107	0,114	0,033	0,069	0,088	0,107
20	40	0,100	0,098	0,105	0,099	0,018	0,035	0,067	0,076
100	5	0,104	0,105	0,099	0,098	0,091	0,103	0,100	0,100
100	10	0,089	0,115	0,101	0,102	0,057	0,088	0,096	0,098
100	20	0,092	0,116	0,099	0,110	0,035	0,064	0,073	0,098
100	40	0,114	0,096	0,104	0,107	0,015	0,023	0,068	0,077
500	5	0,098	0,104	0,112	0,094	0,073	0,099	0,110	0,094
500	10	0,088	0,099	0,097	0,110	0,053	0,078	0,086	0,106
500	20	0,099	0,097	0,095	0,094	0,035	0,049	0,069	0,078
500	40	0,110	0,099	0,082	0,135	0,009	0,024	0,054	0,085
1000	5	0,120	0,135	0,130	0,100	0,075	0,122	0,108	0,098
1000	10	0,075	0,105	0,118	0,105	0,053	0,084	0,112	0,094
1000	20	0,070	0,145	0,090	0,090	0,034	0,062	0,070	0,085
1000	40	0,120	0,090	0,120	0,095	0,007	0,021	0,046	0,076

Tabela B.3: Erro do tipo I obtido com 1000 simulações para a distribuição normal multivariada, $\alpha = 0,1$.

B.2 Lognormal Multivariada

Parâmetros		PPCLUSTEL-R				PPCLUSTEL-R (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,018	0,019	0,022	0,023	0,025	0,029	0,024	0,025
20	10	0,014	0,017	0,019	0,021	0,013	0,017	0,02	0,022
20	20	0,012	0,015	0,019	0,019	0,002	0,003	0,016	0,016
20	40	0,014	0,014	0,012	0,02	0	0	0,004	0,01
100	5	0,015	0,012	0,018	0,012	0,014	0,013	0,02	0,012
100	10	0,009	0,017	0,014	0,016	0,004	0,012	0,013	0,014
100	20	0,009	0,017	0,011	0,016	0,001	0,009	0,006	0,01
100	40	0,018	0,01	0,013	0,016	0	0	0,005	0,006
500	5	0,009	0,007	0,014	0,007	0,005	0,006	0,014	0,007
500	10	0,014	0,007	0,01	0,011	0,005	0,005	0,007	0,01
500	20	0,015	0,007	0,01	0,008	0,001	0,003	0,003	0,007
500	40	0,011	0,012	0,009	0,015	0	0,001	0,001	0,004
1000	5	0,015	0,02	0,01	0,01	0,004	0,012	0,008	0,012
1000	10	0,01	0,02	0,009	0,015	0,002	0,007	0,007	0,011
1000	20	0,02	0,01	0,005	0,01	0	0,003	0,004	0,009
1000	40	0,01	0,005	0,015	0,015	0	0	0,002	0,003

Tabela B.4: Erro do tipo I obtido com 1000 simulações para a distribuição lognormal multivariada, $\alpha = 0,01$.

Parâmetros		PPCLUSTEL-R				PPCLUSTEL-R (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,061	0,054	0,063	0,066	0,069	0,061	0,072	0,073
20	10	0,06	0,056	0,065	0,071	0,046	0,052	0,06	0,071
20	20	0,038	0,05	0,061	0,058	0,017	0,027	0,043	0,055
20	40	0,055	0,051	0,056	0,058	0,005	0,012	0,025	0,043
100	5	0,044	0,049	0,05	0,053	0,038	0,046	0,051	0,053
100	10	0,042	0,049	0,058	0,058	0,021	0,037	0,051	0,053
100	20	0,052	0,06	0,056	0,058	0,008	0,031	0,036	0,051
100	40	0,054	0,033	0,067	0,055	0,003	0,005	0,024	0,036
500	5	0,046	0,052	0,05	0,045	0,032	0,051	0,05	0,045
500	10	0,048	0,05	0,045	0,053	0,02	0,033	0,039	0,05
500	20	0,051	0,046	0,044	0,041	0,011	0,017	0,029	0,035
500	40	0,05	0,052	0,052	0,08	0,001	0,003	0,022	0,045
1000	5	0,06	0,06	0,06	0,04	0,034	0,055	0,055	0,052
1000	10	0,045	0,055	0,058	0,05	0,023	0,035	0,05	0,046
1000	20	0,04	0,055	0,04	0,035	0,014	0,022	0,026	0,039
1000	40	0,07	0,05	0,05	0,065	0	0,007	0,014	0,037

Tabela B.5: Erro do tipo I obtido com 1000 simulações para a distribuição lognormal multivariada, $\alpha = 0,05$.

Parâmetros		PPCLUSTEL-R				PPCLUSTEL-R (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,107	0,102	0,112	0,111	0,116	0,106	0,122	0,115
20	10	0,105	0,112	0,112	0,122	0,085	0,1	0,108	0,119
20	20	0,094	0,108	0,107	0,114	0,033	0,069	0,088	0,107
20	40	0,1	0,098	0,105	0,099	0,018	0,035	0,067	0,076
100	5	0,104	0,105	0,099	0,098	0,091	0,103	0,1	0,1
100	10	0,089	0,115	0,101	0,102	0,057	0,088	0,096	0,098
100	20	0,092	0,116	0,099	0,11	0,035	0,064	0,073	0,098
100	40	0,114	0,096	0,104	0,107	0,015	0,023	0,068	0,077
500	5	0,098	0,104	0,112	0,094	0,073	0,099	0,11	0,094
500	10	0,088	0,099	0,097	0,11	0,053	0,078	0,086	0,106
500	20	0,099	0,097	0,095	0,094	0,035	0,049	0,069	0,078
500	40	0,11	0,099	0,082	0,135	0,009	0,024	0,054	0,085
1000	5	0,12	0,135	0,13	0,1	0,075	0,122	0,108	0,098
1000	10	0,075	0,105	0,118	0,105	0,053	0,084	0,112	0,094
1000	20	0,07	0,145	0,09	0,09	0,034	0,062	0,07	0,085
1000	40	0,12	0,09	0,12	0,095	0,007	0,021	0,046	0,076

Tabela B.6: Erro do tipo I obtido com 1000 simulações para a distribuição lognormal multivariada, $\alpha = 0,1$.

B.3 $t_{v=10}$ Multivariada

Parâmetros		PPCLUSTEL				PPCLUSTEL (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,017	0,018	0,021	0,021	0,027	0,028	0,026	0,023
20	10	0,01	0,018	0,016	0,023	0,009	0,016	0,015	0,023
20	20	0,007	0,014	0,015	0,018	0,002	0,004	0,008	0,015
20	40	0,01	0,016	0,012	0,013	0	0,001	0,004	0,006
100	5	0,011	0,017	0,016	0,017	0,009	0,017	0,017	0,018
100	10	0,006	0,015	0,012	0,018	0	0,01	0,011	0,017
100	20	0,01	0,007	0,012	0,012	0	0,001	0,009	0,006
100	40	0,006	0,012	0,01	0,007	0	0	0,002	0,005
500	5	0,008	0,013	0,015	0,012	0,006	0,009	0,014	0,012
500	10	0,008	0,01	0,017	0,01	0,003	0,006	0,011	0,009
500	20	0,008	0,007	0,012	0,005	0	0,002	0,004	0,005
500	40	0	0,005	0,005	0,015	0	0	0,004	0,005
1000	5	0,015	0,01	0,02	0,014	0,005	0,007	0,014	0,014
1000	10	0,01	0,015	0,005	0,02	0,002	0,003	0,009	0,014
1000	20	0,009	0,004	0,005	0,005	0	0,001	0,002	0,006
1000	40	0,01	0	0,005	0,025	0	0	0,001	0,004

Tabela B.7: Erro do tipo I obtido com 1000 simulações para a distribuição $t_{v=10}$ multivariada, $\alpha = 0,01$.

Parâmetros		PPCLUSTEL				PPCLUSTEL (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,065	0,064	0,073	0,059	0,07	0,075	0,075	0,063
20	10	0,058	0,057	0,058	0,069	0,037	0,055	0,054	0,067
20	20	0,048	0,047	0,055	0,067	0,011	0,021	0,041	0,057
20	40	0,044	0,067	0,06	0,057	0,002	0,013	0,025	0,037
100	5	0,065	0,059	0,069	0,067	0,048	0,056	0,068	0,068
100	10	0,053	0,057	0,054	0,072	0,024	0,044	0,049	0,069
100	20	0,049	0,048	0,04	0,057	0,007	0,013	0,03	0,043
100	40	0,043	0,054	0,058	0,058	0	0,007	0,022	0,029
500	5	0,059	0,06	0,052	0,056	0,036	0,052	0,052	0,056
500	10	0,05	0,06	0,063	0,054	0,017	0,038	0,049	0,051
500	20	0,04	0,042	0,043	0,04	0,005	0,015	0,03	0,03
500	40	0,03	0,025	0,06	0,06	0,002	0,004	0,019	0,027
1000	5	0,06	0,03	0,045	0,054	0,032	0,036	0,047	0,054
1000	10	0,035	0,045	0,06	0,045	0,014	0,032	0,047	0,054
1000	20	0,054	0,048	0,05	0,03	0,006	0,007	0,034	0,034
1000	40	0,05	0,035	0,035	0,05	0	0,007	0,015	0,034

Tabela B.8: Erro do tipo I obtido com 1000 simulações para a distribuição $t_{v=10}$ multivariada, $\alpha = 0,05$.

Parâmetros		PPCLUSTEL				PPCLUSTEL (Σ_i)			
		n_i				n_i			
a	b	3	5	10	20	3	5	10	20
20	5	0,124	0,117	0,126	0,104	0,118	0,116	0,132	0,107
20	10	0,095	0,106	0,11	0,121	0,069	0,087	0,102	0,117
20	20	0,088	0,094	0,1	0,118	0,035	0,054	0,082	0,104
20	40	0,094	0,116	0,106	0,119	0,014	0,041	0,064	0,087
100	5	0,13	0,106	0,117	0,116	0,105	0,1	0,117	0,118
100	10	0,101	0,107	0,106	0,122	0,06	0,082	0,094	0,116
100	20	0,1	0,092	0,095	0,103	0,023	0,047	0,063	0,093
100	40	0,091	0,115	0,104	0,105	0,002	0,03	0,057	0,077
500	5	0,111	0,11	0,105	0,105	0,088	0,106	0,102	0,105
500	10	0,094	0,106	0,106	0,099	0,049	0,085	0,089	0,098
500	20	0,086	0,092	0,094	0,08	0,016	0,041	0,061	0,077
500	40	0,055	0,075	0,115	0,13	0,004	0,018	0,044	0,069
1000	5	0,115	0,07	0,08	0,097	0,071	0,082	0,094	0,097
1000	10	0,085	0,1	0,095	0,08	0,05	0,074	0,096	0,11
1000	20	0,11	0,099	0,105	0,085	0,028	0,041	0,067	0,073
1000	40	0,08	0,095	0,1	0,11	0,002	0,016	0,035	0,067

Tabela B.9: Erro do tipo I obtido com 1000 simulações para a distribuição $t_{v=10}$ multivariada, $\alpha = 0,1$.

Apêndice C

Programações em SAS

C.1 Estimação *jackknife* de Σ

C.1.1 Estimação Σ

```
%macro s4_jackknife_f(dataset);
proc iml;

use &dataset;
read all into dat;

*Function to remove row "i" from a matrix "x";
start delrow(x,i);
return(x[setdif(1:nrow(x),i),]);
finish delrow;



sigma4_est_jck=J(1,&b,1);
do i=1 to &a;
dat_i=dat[loc(dat[,1]=i),3:%eval(2+&b)];

n=nrow(dat_i);
b=ncol(dat_i);
```

```
aux=dat_i - J(n,1)*(dat_i[+,]/n); *Compute covariance matrix;
sigma=t(aux)*aux/(n);*changed, denominator n-1;
sigma4=sigma##2;

sum_sigma4_i=J(b,b,0);
do k=1 to n;
dat_i_jck=delrow(dat_i,k);

aux=dat_i_jck - J(nrow(dat_i_jck),1)*(dat_i_jck[+,]/nrow(dat_i_jck));
sigma_i=t(aux)*aux/(nrow(dat_i_jck));*changed, denominator n-1;
sigma4_i=sigma_i##2;
sum_sigma4_i=sum_sigma4_i+sigma4_i;
end;

sigma4_est_jck_i=n#sigma4-((n-1)/n)#sum_sigma4_i;
sigma4_est_jck_i=sigma4_est_jck_i#(sigma4_est_jck_i>0);
sigma4_est_jck=sigma4_est_jck//sigma4_est_jck_i;
end;
sigma4_est_jck=delrow(sigma4_est_jck,1);



*DATA BASE FORMAT;
time = 1;
do i = 2 to &b;
time = time // i;
end;

time = j(&a,1) @ time;
treatment = j(&b,1)*1;
do treat = 2 to &a;
treatment = treatment// j(&b,1) * treat;
```

```
end;
temp = treatment || time || sigma4_est_jck;

name1 = {treatid time};
name2 = ("t1":"t&b");
names = name1 || name2;

create temp3 from temp[colname=names];
append from temp;
close temp3;
quit;
%mend s4_jackknife_f;
```

C.1.2 Estimação Σ_i

```
%macro s4_jackknife(dataset);*modified: deleted parameters "a" and "b";
proc iml;

use generated;
read all into dat;

*Function to remove row "i" from a matrix "x";
start delrow(x,i);
return(x[setdif(1:nrow(x),i),]);
finish delrow;

dat_i=dat[,3:%eval(2+&b)];

n=nrow(dat_i);
b=ncol(dat_i);

*Computing jackknife estimator of sigma^4;
```

```
SampleMean=dat_i[:,];
aux=dat_i - repeat(SampleMean,n); *Compute covariance matrix;
sigma=t(aux)*aux/(n);*changed, denominator n-1;
sigma4=sigma##2;


sum_sigma4_i=J(b,b,0);
do k=1 to n;
dat_i_jck=delrow(dat_i,k);
n1=nrow(dat_i_jck);
SampleMean=dat_i_jck[:,];
aux=dat_i_jck - repeat(SampleMean,n1);*Compute covariance matrix;
sigma_i=t(aux)*aux/(n1);*changed, denominator n-1;
sigma4_i=sigma_i##2;
sum_sigma4_i=sum_sigma4_i+sigma4_i;
end;

sigma4_est_jck=n#sigma4-((n-1)/n)#sum_sigma4_i;
sigma4_est_jck=sigma4_est_jck#(sigma4_est_jck>0);
sigma4_est_jck = j(&a,1) @ sigma4_est_jck;

*DATA BASE FORMAT;
time = 1;
do i = 2 to &b;
time = time // i;
end;

time = j(&a,1) @ time;
treatment = j(&b,1)*1;
do treat = 2 to &a;
treatment = treatment// j(&b,1) * treat;
end;
```

```
temp = treatment || time || sigma4_est_jck;

name1 = {treatid time};
name2 = ("t1":"t&b");
names = name1 || name2;

create temp3 from temp[colname=names];
append from temp;
close temp3;
quit;
%mend s4_jackknife;
```

C.2 PPCLUSTEL - R (Σ ou Σ_i)

```
%macro ppclustelr(datafile=,treatment=,subject=,tmin=,tmax=);
proc iml;
  start rankmiss(matrix1);
   matrixm = matrix1 = .;
nmiss = matrixm[,+];
nmiss = nmiss[+,];
miss = -9999999999;
matrix1 = choose(matrix1=.,miss,matrix1);
matrix1 = ranktie(matrix1);
matrix1 = matrix1 - nmiss;
matrix1 = choose(matrix1<=0,.,matrix1);
  finish;

  use &datafile;
  read all var{&treatment &subject} into factor;
  read all var("&tmin":"&tmax") into expression;
  b = ncol(expression);
```

```
  call rankmiss(expression);

  name1 = {treatid subjectid};
  name2 = ("&tmin":"&tmax");
  names = name1 || name2;

  rt = factor || expression;
  create datasetf1 from rt[colname=names];
  append from rt;

  create nb var{b};
  append from b;
quit;


data nb;
  set nb;
  call symput('b',b);
run;

*STEP 1 ;
proc freq data=datasetf1 noprint;
  tables treatid / out=nitreat(keep=treatid count);
run;

proc means data=nitreat sum noprint;
  var count;
  output out=sumni sum=sumni;
run;

data _null_;
```

```
  set sumni;
  call symput('sumni',sumni);
run;



*STEP 3;
data datasetf2 (drop= &tmin - &tmax);
  set datasetf1;
  array num(*) &tmin - &tmax;
  do time=1 to dim(num);
    exp = num[time];
  output;
  end;
run;

*STEP 4;
proc means data=datasetf2 median noprint;
  var exp;
  by treatid;
  output out=result1(keep=treatid median) median=median;
run;

*STEP 5;
proc sort data=datasetf2; by treatid time;
run;

*STEP 6;
proc means data=datasetf2 mean noprint;
  var exp;
  by treatid time;
  output out=result2(keep=treatid time rbijp) mean=rbijp;
run;
```

```
*STEP 7;
data temp1;
  merge datasetf2 result2;
  by treatid time;
run;

data temp2;
  merge temp1 nitreat;
  by treatid;
  msepartial = (exp - rbijp)**2 / (&b*count*(count-1));
run;

proc means data=temp2 sum noprint;
  var msepartial;
  by treatid;
  output out=result3(keep=treatid msep) sum=msep;
run;

%if &sgmest=0 %then %s4_jackknife(datasetf1);
%if &sgmest=1 %then %s4_jackknife_f(datasetf1);

data temp4 (drop= &tmin - &tmax);
  set temp3;
  array num(*) &tmin - &tmax;
  do timecov=1 to dim(num);
    cov = num[timecov];
cov2 = cov;  *modified;
  output;
  end;
run;
```

```
*STEP 9;
proc means data=temp4 sum noprint;
  var cov2;
  by treatid;
  output out=result4(keep=treatid cov2p) sum=cov2p;
run;

data result4(keep=treatid cov2p);
  merge result4 nitreat;
  by treatid;
  cov2p = 2 * cov2p / (&b*count*(count-1));
run;

*STEP 10;
proc means data=datasetf1 mean noprint;
  var &tmin - &tmax;
  by treatid;
  output out=result5(keep=treatid &tmin - &tmax) mean=&tmin - &tmax;
run;

*STEP 11;
data prepmatrix;
  merge result1 result3 result4 result5;
  by treatid;
run;

*STEP 12;
proc iml;

* Routine to sort lines of matrix according to column of medians;
*STEP 12.2;
start sortmed(matrix);
```

```
call sortndx(ms,matrix,{2});
matrix = matrix[ms,];
finish;

*STEP 12.3;
start sortcenter(matrix,tst);
   nr = nrow(matrix);
nc = ncol(matrix);
if tst = 1 then do;
   m2 = matrix;
end;
else if tst > 1 then do;
ed = tst-1;
    m1 = matrix[1:ed,];
m2 = matrix[tst:nr,];
end;
nr2 = nrow(m2);
  j1 = j(nr2,1) * 0;
c1 = int(0.35*nr2);
c2 = int(0.65*nr2);
call sort(m2,1);
do i = 1 to nr2;
   if i < c1 then j1[i,] = 1;
if i > c2 then j1[i,] = 1;
end;
m2 = m2 || j1;
nc2 = nc + 1;
call sort(m2,nc2);
       m2 = m2[,1:nc];
if tst = 1 then matrix = m2;
else if tst > 1 then matrix = m1 // m2;
finish;
```

```
* Main Routine for one-way anova routine when we have
  a large number of factors;

*STEP 12.4;
    start anovalongrank(resp,a,b,pv);
cts = 5;
cte = 4 + b;
xijp = resp[,cts:cte];
        xpjp = xijp[+,]/a;
k1 = j(a,1);
k2 = j(b,1);
k = k1 * k2';
  xpjp = k # xpjp;
       mst = (xijp - xpjp) ## 2;
mst = mst[,+];
mst = mst[+,]/((a-1)*b);
       mse = resp[,3];
mse = mse[+,] / a;
* fr = mst/mse; * changed;
cov2 = resp[,4];
cov2 = cov2[+,] / a;
       pv = 1 - probnorm(sqrt(a*b)*(mst - mse)/sqrt(cov2)); * changed;
finish;

*STEP 12.5;
   start indtest(exp,b,st,alpha,a,colgr,colts,g);
   exp[,colts] = 0; count = 0;
   do i=1 to a;
  if exp[i,colgr]=g then exp[i,colts] = 1;
end;
do i=st to a;
```

```
   if exp[i,colts] = 0 then do;
kwtest = 0;
exp[i,colts] = 1;
n = sum(exp[,colts]);
if kwtest = 0 then do;
             expt = exp[loc(exp[,colts]),];
call anovalongrank(expt,n,b,pv);
if (pv > alpha) then exp[i,colgr] = g;
else if (pv <= alpha) then exp[i,colts] = 0;
end;
end;
end;
do i=1 to a;
   if exp[i,colgr] <= g then count = count + 1;
end;
st = count + 1;
call sort(exp,colgr);
kwtest = 0;
finish;

*STEP 12.1;
   use prepmatrix;
   read all into mdata;
   alpha = &alpha;
   g = 1;
ktest = 0;
a = nrow(mdata);
b = &b;
ncole = ncol(mdata);
call sortmed(mdata);

    colid = 1;
```

```
colgr = ncole + 1;
colts = ncole + 2;
j1 = j(a,1);
mdata = mdata || j1 * 9999;
mdata = mdata || j1 * 0;
tstart = 1; tend = a;
call sortcenter(mdata,tstart);
    call anovalongrank(mdata,a,b,pv);

   pv2 = pv;
   if (pv > alpha) then do;
   mdata[,colgr] = 1;
   end;
if (pv <= alpha) then do;
L = (tend - tstart + 1)/2;
tend = tstart + int(L-1);
do while (tstart <= a);
   if (pv2 > alpha) then g = g + 1;
call sortcenter(mdata,tstart);
mdata[,colts] = 0;
do i = 1 to a;
   if (tstart <= i) then do;
  if ( i <= tend) then mdata[i,colts] = 1;
end;
end;
n = sum(mdata[,colts]);
if (n = 1) then do;
   mdata[tstart,colgr] = 0;
tstart = tend + 1;
tend = a;
ktest = -1;
end;
```

```
exp = mdata[loc(mdata[,colts]),];
if ktest = 0 then call anovalongrank(exp,n,b,pv);
pv2 = pv;
if (pv > alpha) then do;
   do i = 1 to a;
if (tstart <= i) then do;
  if ( i <= tend) then mdata[i,colgr] = g;
end;
end;
tstart = tend + 1;
tend = a;
st = tstart;  exp = mdata;
call indtest(exp,b,st,alpha,a,colgr,colts,g);
tstart = st;
mdata = exp;
end;
else if (pv <= alpha) then do;
L = (tend - tstart + 1)*0.9;
tend = tstart + int(L-1);
end;
end;
end;

exp = mdata[,colid];
exp = exp || mdata[,colgr] || mdata[,2];

create groupclass var{treatid group mediange};
append from exp;

free all;
quit;
```

```
*STEP 13;
proc sort data=groupclass;
  by group;
run;

data groupclass groupclass0;
  set groupclass;
  if group = 0 then output groupclass0;
  else if group ^= 0 then output groupclass;
run;

proc means data=groupclass median noprint;
  var mediange;
  by group;
  output out=newgroupclass median=med;
run;

proc sort data=newgroupclass;
  by med;
run;

data newgroupclass(keep=group sgroup);
  set newgroupclass;
  if group=0 then delete;
  sgroup = _N_;
run;

proc sort data=newgroupclass;
  by group;
run;
```

```
data groups;
  merge groupclass newgroupclass;
  by group;
run;

data groups(keep=treatid group);
  set groupclass0 groups;
  if group=0 then sgroup=0;
  rename group = og;
  rename sgroup = group;
run;

proc datasets nolist;
  delete datasetf1 datasetf2 groupclass groupclass0
         nb newgroupclass nigene prepmatrix result1-result5
         sumni temp1-temp5;
run;

data groupclass;
  set groups;
  rename treatid=idobs;
run;

proc datasets nolist;
   delete groups;
run;

proc freq data=groupclass noprint;
tables group / out=grfr;
run;

data datanew;
```

```
  set &datafile;
  idobs = &treatment;
run;

proc sort data=groupclass;
  by idobs;
run;

data datanew;
  merge datanew groupclass;
  by idobs;
run;

proc sort data=groupclass;
by group;
run;
%mend ppclustelr;
```

C.3 PPCLUSTEL - R (Σ_G)

```
%macro ppclustel_rg(dataset,treatment,subject,tmin,tmax,alpha,datasetout);



proc iml;
  start rankmiss(matrix1);
   matrixm = matrix1 = .;
nmiss = matrixm[,+];
nmiss = nmiss[+,];
miss = -9999999999;
```

```
matrix1 = choose(matrix1=.,miss,matrix1);
matrix1 = ranktie(matrix1);
matrix1 = matrix1 - nmiss;
matrix1 = choose(matrix1<=0,.,matrix1);
  finish;

  use &dataset;
  read all var{&treatment &subject} into factor;
  read all var("&tmin":"&tmax") into expression;
  b = ncol(expression);

  call rankmiss(expression);

  name1 = {treatid subjectid};
  name2 = ("&tmin":"&tmax");
  names = name1 || name2;

  rt = factor || expression;
  create datasetf1 from rt[colname=names];
  append from rt;

  create nb var{b};
  append from b;
quit;


data nb;
  set nb;
  call symput('b',b);
run;

*STEP 1 ;
```

```
proc freq data=datasetf1 noprint;
  tables treatid / out=nitreat(keep=treatid count);
run;

proc means data=nitreat sum noprint;
  var count;
  output out=sumni sum=sumni;
run;

data _null_;
  set sumni;
  call symput('sumni',sumni);
run;



*STEP 3;
data datasetf2 (drop= &tmin - &tmax);
  set datasetf1;
  array num(*) &tmin - &tmax;
  do time=1 to dim(num);
    exp = num[time];
  output;
  end;
run;

*STEP 4;
proc means data=datasetf2 median noprint;
  var exp;
  by treatid;
  output out=result1(keep=treatid median) median=median;
run;
```

```
*STEP 5;
proc sort data=datasetf2; by treatid time;
run;

*STEP 6;
proc means data=datasetf2 mean noprint;
  var exp;
  by treatid time;
  output out=result2(keep=treatid time rbijp) mean=rbijp;
run;

*STEP 7;
data temp1;
  merge datasetf2 result2;
  by treatid time;
run;

data temp2;
  merge temp1 nitreat;
  by treatid;
  msepartial = (exp - rbijp)**2 / (&b*count*(count-1));
run;

proc means data=temp2 sum noprint;
  var msepartial;
  by treatid;
  output out=result3(keep=treatid msep) sum=msep;
run;


*STEP 10;
```

```
proc means data=datasetf1 mean noprint;
  var &tmin - &tmax;
  by treatid;
  output out=result5(keep=treatid &tmin - &tmax) mean=&tmin - &tmax;
run;

*STEP 11;
data prepmatrix;
  merge result1 result3 result5;
  by treatid;
run;

proc iml;

*Function to remove row "i" from a matrix "x";
start delrow(x,i);
return(x[setdif(1:nrow(x),i),]);
finish delrow;


/* start delcol(x,i);*/
/*   return(x[,setdif(1:ncol(x),i)]);*/
/* finish delcol;*/


* Routine to sort lines of matrix according to column of medians;
*STEP 12.2;
start sortmed(matrix);
call sortndx(ms,matrix,{2});
matrix = matrix[ms,];
finish;
```

```
*STEP 12.3;
start sortcenter(matrix,tst);

    nr = nrow(matrix);
nc = ncol(matrix);
if tst = 1 then do;
   m2 = matrix;
end;
else if tst > 1 then do;
ed = tst-1;
    m1 = matrix[1:ed,];
m2 = matrix[tst:nr,];
end;
nr2 = nrow(m2);
  j1 = j(nr2,1) * 0;
c1 = int(0.35*nr2);
c2 = int(0.65*nr2);
call sort(m2,1);
do i = 1 to nr2;
   if i < c1 then j1[i,] = 1;
if i > c2 then j1[i,] = 1;
end;
m2 = m2 || j1;
nc2 = nc + 1;
call sort(m2,nc2);
       m2 = m2[,1:nc];
if tst = 1 then matrix = m2;
else if tst > 1 then matrix = m1 // m2;
finish;

* Main Routine for one-way anova routine when we have
  a large number of factors;
```

```
*STEP 12.4;
    start anovalongrank(resp,a,b,pv);
use datasetf1;
read all into dat;

use nitreat;
read all into ni_vec;

newdat=j(1,b+2);
ni_vec_sub=1;
do j=1 to a;
tmp=dat[LOC(dat[,1]=resp[j,1]),];
newdat=newdat//tmp;
tmp=ni_vec[loc(ni_vec[,1]=resp[j,1]),2];
ni_vec_sub=ni_vec_sub//tmp;
end;

newdat=delrow(newdat,1);
ni_vec_sub=delrow(ni_vec_sub,1);

dat_i=newdat[,3:(2+b)];

n=nrow(dat_i);

*Computing jackknife estimator of sigma^4;
SampleMean=dat_i[:,];
aux=dat_i - repeat(SampleMean,n); *Compute covariance matrix;
sigma=t(aux)*aux/(n);*changed, denominator n-1;
sigma4=sigma##2;

sum_sigma4_i=J(b,b,0);
```

```
do k=1 to n;
dat_i_jck=delrow(dat_i,k);
n1=nrow(dat_i_jck);
SampleMean=dat_i_jck[:,];
aux=dat_i_jck - repeat(SampleMean,n1);*Compute covariance matrix;
sigma_i=t(aux)*aux/(n1);*changed, denominator n-1;
sigma4_i=sigma_i##2;
sum_sigma4_i=sum_sigma4_i+sigma4_i;
end;

sigma4_est_jck=n#sigma4-((n-1)/n)#sum_sigma4_i;
sigma4_est_jck=sigma4_est_jck#(sigma4_est_jck>0);

cov=sigma4_est_jck[+,+];
cov2 = j(a,1) @ cov;
denominador=1/(ni_vec_sub#(ni_vec_sub-1)#b);

cov2=2#cov2#denominador;

cts = 4;
cte = 3 + b;
xijp = resp[,cts:cte];
        xpjp = xijp[+,]/a;
k1 = j(a,1);
k2 = j(b,1);
k = k1 * k2';
  xpjp = k # xpjp;
       mst = (xijp - xpjp) ## 2;
mst = mst[,+];
mst = mst[+,]/((a-1)*b);
       mse = resp[,3];
mse = mse[+,] / a;
```

```
* fr = mst/mse; * changed;
cov2 = cov2[+] / a;
       pv = 1 - probnorm(sqrt(a*b)*(mst - mse)/sqrt(cov2)); * changed;
store cov;
finish;




    start anovalongrank2(resp,a,b,pv);
use datasetf1;
read all into dat;

use nitreat;
read all into ni_vec;

ni_vec_sub=1;
do j=1 to a;
tmp=ni_vec[loc(ni_vec[,1]=resp[j,1]),2];
ni_vec_sub=ni_vec_sub//tmp;
end;

ni_vec_sub=delrow(ni_vec_sub,1);

load cov;
cov = j(a,1) @ cov;
denominador=1/(ni_vec_sub#(ni_vec_sub-1)#b);

cov2=2#cov#denominador;

cts = 4;
```

```
cte = 3 + b;
xijp = resp[,cts:cte];
        xpjp = xijp[+,]/a;
k1 = j(a,1);
k2 = j(b,1);
k = k1 * k2‘;
  xpjp = k # xpjp;
       mst = (xijp - xpjp) ## 2;
mst = mst[,+];
mst = mst[+,]/((a-1)*b);
       mse = resp[,3];
mse = mse[+,] / a;
* fr = mst/mse; * changed;
cov2 = cov2[+] / a;
       pv = 1 - probnorm(sqrt(a*b)*(mst - mse)/sqrt(cov2)); * changed;

finish anovalongrank2;



*STEP 12.5;
   start indtest(exp,b,st,alpha,a,colgr,colts,g);
   exp[,colts] = 0; count = 0;
   do i=1 to a;
  if exp[i,colgr]=g then exp[i,colts] = 1;
end;
do i=st to a;
   if exp[i,colts] = 0 then do;
kwtest = 0;
exp[i,colts] = 1;
n = sum(exp[,colts]);
if kwtest = 0 then do;
             expt = exp[loc(exp[,colts]),];
```

```
call anovalongrank2(expt,n,b,pv);
if (pv > alpha) then exp[i,colgr] = g;
else if (pv <= alpha) then exp[i,colts] = 0;
end;
end;
end;
do i=1 to a;
   if exp[i,colgr] <= g then count = count + 1;
end;
st = count + 1;
call sort(exp,colgr);
kwtest = 0;
finish;

*STEP 12.1;
   use prepmatrix;
   read all into mdata;
   alpha = &alpha;
   g = 1;
ktest = 0;
a = nrow(mdata);
b = &b;
ncole = ncol(mdata);
call sortmed(mdata);

    colid = 1;
colgr = ncole + 1;
colts = ncole + 2;
j1 = j(a,1);
mdata = mdata || j1 * 9999;
mdata = mdata || j1 * 0;
tstart = 1; tend = a;
```

```
call sortcenter(mdata,tstart);
    call anovalongrank(mdata,a,b,pv);


   pv2 = pv;
   if (pv > alpha) then do;
   mdata[,colgr] = 1;
   end;
if (pv <= alpha) then do;
L = (tend - tstart + 1)/2;
tend = tstart + int(L-1);
do while (tstart <= a);
   if (pv2 > alpha) then g = g + 1;
call sortcenter(mdata,tstart);
mdata[,colts] = 0;
do i = 1 to a;
   if (tstart <= i) then do;
  if ( i <= tend) then mdata[i,colts] = 1;
end;
end;
n = sum(mdata[,colts]);
if (n = 1) then do;
   mdata[tstart,colgr] = 0;
tstart = tend + 1;
tend = a;
ktest = -1;
end;
exp = mdata[loc(mdata[,colts]),];
if ktest = 0 then call anovalongrank(exp,n,b,pv);
pv2 = pv;
if (pv > alpha) then do;
   do i = 1 to a;
```

```
if (tstart <= i) then do;
  if ( i <= tend) then mdata[i,colgr] = g;
end;
end;
tstart = tend + 1;
tend = a;
st = tstart;  exp = mdata;
call indtest(exp,b,st,alpha,a,colgr,colts,g);
tstart = st;
mdata = exp;
end;
else if (pv <= alpha) then do;
L = (tend - tstart + 1)*0.9;
tend = tstart + int(L-1);
end;
end;
end;

exp = mdata[,colid];
exp = exp || mdata[,colgr] || mdata[,2];

create groupclass var{treatid group mediange};
append from exp;

free all;
quit;



*STEP 13;
proc sort data=groupclass;
  by group;
run;
```

```
data groupclass groupclass0;
  set groupclass;
  if group = 0 then output groupclass0;
  else if group ^= 0 then output groupclass;
run;

proc means data=groupclass median noprint;
  var mediange;
  by group;
  output out=newgroupclass median=med;
run;

proc sort data=newgroupclass;
  by med;
run;

data newgroupclass(keep=group sgroup);
  set newgroupclass;
  if group=0 then delete;
  sgroup = _N_;
run;

proc sort data=newgroupclass;
  by group;
run;

data groups;
  merge groupclass newgroupclass;
  by group;
run;
```

```
data groups(keep=treatid group);
  set groupclass0 groups;
  if group=0 then sgroup=0;
  rename group = og;
  rename sgroup = group;
run;
/**/
/*proc datasets nolist;  */
/*  delete datasetf1 datasetf2 groupclass groupclass0*/
/*         nb newgroupclass nigene prepmatrix result1-result5*/
/*         sumni temp1-temp5;  */
/*run;*/

data groupclass2;
  set groups;
  rename treatid=idobs;
run;

proc datasets nolist;
   delete groups;
run;

proc freq data=groupclass2 noprint;
tables group / out=grfr;
run;

data datanew;
  set &dataset;
  idobs = &treatment;
run;

proc sort data=groupclass2;
```

```
  by idobs;
run;

data datanew;
  merge datanew groupclass2;
  by idobs;
run;

proc sort data=groupclass2;
by group;
run;

data &datasetout; set groupclass2; run;
%mend ppclustel_rg;
```

C.4 PPCLUSTEL - M (Σ_i)

```
%macro vector_zeta;
proc iml;

use datasetf1;
read all into dat;
nfac=max(dat[,1]);
%let a=nfac;
*Function to remove row "i" from a matrix "x";
start delrow(x,i);
return(x[setdif(1:nrow(x),i),]);
finish delrow;

*Inicializando vetores que sero utilizados no looping;
ni_vec=J(&a,1,1);
sigma4_est_jck=J(1,&b,1);
```

```
sigma_est=J(1,&b,1);
zeta1=J(&a,1,1);

do i=1 to &a;
dat_i=dat[loc(dat[,1]=i),3:%eval(2+&b)];

n=nrow(dat_i);
ni_vec[i]=n;
b=ncol(dat_i);

aux=dat_i - J(n,1)*(dat_i[+,]/n); *Compute covariance matrix;
sigma=t(aux)*aux/(n);*changed, denominator n-1;
sigma4=sigma##2;

sum_sigma4_i=J(b,b,0);
do k=1 to n;
dat_i_jck=delrow(dat_i,k);

aux=dat_i_jck - J(nrow(dat_i_jck),1)*(dat_i_jck[+,]/nrow(dat_i_jck));
sigma_i=t(aux)*aux/(nrow(dat_i_jck));*changed, denominator n-1;
sigma4_i=sigma_i##2;
sum_sigma4_i=sum_sigma4_i+sigma4_i;
end;
sigma_est=sigma_est//sigma;

sigma4_est_jck_i=n#sigma4-((n-1)/n)#sum_sigma4_i;
sigma4_est_jck_i=sigma4_est_jck_i#(sigma4_est_jck_i>0);
    *Calculando valores de zeta1;
denominador=(ni_vec[i]-1)#(ni_vec[i]);
prod2=sigma4_est_jck_i/(denominador);
zeta1[i]=prod2[+,+];
end;
```

```
zeta1=(2/&b)#zeta1;
sigma_est=delrow(sigma_est,1);


*Calculando valores de zeta2;
treatid=shape(1:&a,&a,1);
treatment =treatid@j(&b,1);
ni = ni_vec@j(&b,1);

sigma_mod= treatment ||ni|| sigma_est;


zeta2=J(&a,1,1);
do i=1 to &a;
sigma_ni_i=sigma_mod[loc(sigma_mod[,1]=i),3:%eval(2+&b)]
#(1/sigma_mod[loc(sigma_mod[,1]=i),2]);
sigma_ni_i_vec=repeat(sigma_ni_i,&a-1,1);

sigma_ni_noi_vec=sigma_mod[loc(sigma_mod[,1]^=i),3:%eval(2+&b)]
#(1/sigma_mod[loc(sigma_mod[,1]^=i),2]);
prod=sigma_ni_i_vec#sigma_ni_noi_vec;
zeta2[i]=prod[+,+];
end;
zeta2=(2/&b)#zeta2;

treatid=1:&a;
create zeta var{treatid ni_vec zeta1 zeta2};
append;
close zeta;

quit;
%mend vector_zeta;
```

```
*****************************************************************
PROGRAM:  PPCLUSTEL_M.SAS
DESCRIPTION: CLUSTERING OF LONGITUDINAL MICROARRAY DATA
VERSION: 1.0.0
UPDATE: 1-2-2012
SYSTEM: SAS WINDOWS 9.1.3
MODULES: BASE - IML - STAT
MACRO:          PPCLUSTEL_M AND VECTOR_ZETA
NOTE:           FINISHED OF PPCLUSTEL_M
AUTHOR: ALEX PENA TOSTA DA SILVA
UNIVERSIDADE DE BRASILIA - MASTER CANDIDATE

*****************************************************************;

*libname sasmac 'c:\sasmacros';
*options mstored sasmstore=sasmac;

*options source source2 notes stimer mprint symbolgen mlogic;
options nosource nosource2 nonotes nostimer nomprint nosymbolgen nomlogic;
%macro ppclustel_m(dataset,treatment,subject,tmin,tmax);

proc iml;
  start rankmiss(matrix1);
   matrixm = matrix1 = .;
nmiss = matrixm[,+];
nmiss = nmiss[+,];
miss = -9999999999;
matrix1 = choose(matrix1=.,miss,matrix1);
matrix1 = ranktie(matrix1);
matrix1 = matrix1 - nmiss;
```

```
matrix1 = choose(matrix1<=0,.,matrix1);
  finish;

  use &dataset;
  read all var{&treatment &subject} into factor;
  read all var("&tmin":"&tmax") into expression;
  b = ncol(expression);

  call rankmiss(expression);

  name1 = {treatid subjectid};
  name2 = ("&tmin":"&tmax");
  names = name1 || name2;

  rt = factor || expression;
  create datasetf1 from rt[colname=names];
  append from rt;

  create nb var{b};
  append from b;
quit;



data nb;
  set nb;
  call symput('b',b);
run;

*STEP 1 ;
proc freq data=datasetf1 noprint;
  tables treatid / out=nitreat(keep=treatid count);
run;
```

```
proc means data=nitreat sum noprint;
  var count;
  output out=sumni sum=sumni;
run;

data _null_;
  set sumni;
  call symput('sumni',sumni);
run;

*STEP 3;
data datasetf2 (drop= &tmin - &tmax);
  set datasetf1;
  array num(*) &tmin - &tmax;
  do time=1 to dim(num);
    exp = num[time];
  output;
  end;
run;

*STEP 4;
proc means data=datasetf2 median noprint;
  var exp;
  by treatid;
  output out=result1(keep=treatid median) median=median;
run;

*STEP 5;
proc sort data=datasetf2; by treatid time;
run;
```

```
*STEP 6;
proc means data=datasetf2 mean noprint;
  var exp;
  by treatid time;
  output out=result2(keep=treatid time rbijp) mean=rbijp;
run;

*STEP 7;
data temp1;
  merge datasetf2 result2;
  by treatid time;
run;

data temp2;
  merge temp1 nitreat;
  by treatid;
  msepartial = (exp - rbijp)**2 / (&b*count*(count-1));
run;

proc means data=temp2 sum noprint;
  var msepartial;
  by treatid;
  output out=result3(keep=treatid msep) sum=msep;
run;

%vector_zeta;

*STEP 10;
proc means data=datasetf1 mean noprint;
  var &tmin - &tmax;
  by treatid;
  output out=result4(keep=treatid &tmin - &tmax) mean=&tmin - &tmax;
```

```
run;

*STEP 11;
data prepmatrix;
  merge zeta result1 result3 result4 ;
  by treatid;
run;

*STEP 12;
proc iml;

* Routine to sort lines of matrix according to column of medians;
*STEP 12.2;
start sortmed(matrix);
call sortndx(ms,matrix,{5});
matrix = matrix[ms,];
finish;

*STEP 12.3;
start sortcenter(matrix,tst);
   nr = nrow(matrix);
nc = ncol(matrix);
if tst = 1 then do;
   m2 = matrix;
end;
else if tst > 1 then do;
ed = tst-1;
    m1 = matrix[1:ed,];
m2 = matrix[tst:nr,];
end;
nr2 = nrow(m2);
  j1 = j(nr2,1) * 0;
```

```
c1 = int(0.35*nr2);
c2 = int(0.65*nr2);
call sort(m2,1);
do i = 1 to nr2;
   if i < c1 then j1[i,] = 1;
if i > c2 then j1[i,] = 1;
end;
m2 = m2 || j1;
nc2 = nc + 1;
call sort(m2,nc2);
       m2 = m2[,1:nc];
if tst = 1 then matrix = m2;
else if tst > 1 then matrix = m1 // m2;
finish;

* Main Routine for one-way anova routine when we have
  a large number of factors;

*STEP 12.4;
start anovalongrank(resp,a,b,pv);
cts = 7;
cte = 6 + b;
xijp = resp[,cts:cte];
        xpjp = xijp[+,]/a;
k1 = j(a,1);
k2 = j(b,1);
k = k1 * k2';
  xpjp = k # xpjp;
       mst = (xijp - xpjp) ## 2;
mst = mst[,+];
mst = mst[+,]/((a-1)*b);
       mse = resp[,6];
```

```
mse = mse[+,] / a;

* fr = mst/mse; * changed;
zeta1=resp[,3];
zeta2=resp[,4];
zeta1 = zeta1[+,] / a##2;
zeta2 = zeta2[+,] / a##2;

*Montando covariancia da aproximao assinttica;
*Denominador;
N2=(resp[+,2]#b)##2;
sig4=(mse/N2)##2;
*Numerador;
tau2=zeta1+zeta2/(a-1)##2;
tau2=tau2/(N2##2);

cov=tau2/sig4;

Fg=mst/mse;

       pv = 1 - probnorm(sqrt(b)*(Fg-1)/sqrt(cov)); * changed;
finish;

*STEP 12.5;
   start indtest(exp,b,st,alpha,a,colgr,colts,g);
   exp[,colts] = 0; count = 0;
   do i=1 to a;
  if exp[i,colgr]=g then exp[i,colts] = 1;
end;
do i=st to a;
   if exp[i,colts] = 0 then do;
kwtest = 0;
```

```
exp[i,colts] = 1;
n = sum(exp[,colts]);
if kwtest = 0 then do;
             expt = exp[loc(exp[,colts]),];
call anovalongrank(expt,n,b,pv);
if (pv > alpha) then exp[i,colgr] = g;
else if (pv <= alpha) then exp[i,colts] = 0;
end;
end;
end;
do i=1 to a;
   if exp[i,colgr] <= g then count = count + 1;
end;
st = count + 1;
call sort(exp,colgr);
kwtest = 0;
finish;

*STEP 12.1;
   use prepmatrix;
   read all into mdata;
   alpha = &alpha;
   g = 1;
ktest = 0;
a = nrow(mdata);
b = &b;
ncole = ncol(mdata);
call sortmed(mdata);

    colid = 1;
colgr = ncole + 1;
colts = ncole + 2;
```

```
j1 = j(a,1);
mdata = mdata || j1 * 9999;
mdata = mdata || j1 * 0;
tstart = 1; tend = a;
call sortcenter(mdata,tstart);
    call anovalongrank(mdata,a,b,pv);

   pv2 = pv;
   if (pv > alpha) then do;
   mdata[,colgr] = 1;
   end;
if (pv <= alpha) then do;
L = (tend - tstart + 1)/2;
tend = tstart + int(L-1);
do while (tstart <= a);
   if (pv2 > alpha) then g = g + 1;
call sortcenter(mdata,tstart);
mdata[,colts] = 0;
do i = 1 to a;
   if (tstart <= i) then do;
  if ( i <= tend) then mdata[i,colts] = 1;
end;
end;
n = sum(mdata[,colts]);
if (n = 1) then do;
   mdata[tstart,colgr] = 0;
tstart = tend + 1;
tend = a;
ktest = -1;
end;
exp = mdata[loc(mdata[,colts]),];
if ktest = 0 then call anovalongrank(exp,n,b,pv);
```

```
pv2 = pv;
if (pv > alpha) then do;
   do i = 1 to a;
if (tstart <= i) then do;
  if ( i <= tend) then mdata[i,colgr] = g;
end;
end;
tstart = tend + 1;
tend = a;
st = tstart;  exp = mdata;
call indtest(exp,b,st,alpha,a,colgr,colts,g);
tstart = st;
mdata = exp;
end;
else if (pv <= alpha) then do;
L = (tend - tstart + 1)*0.9;
tend = tstart + int(L-1);
end;
end;
end;

exp = mdata[,colid];
exp = exp || mdata[,colgr] || mdata[,2];

create groupclass var{treatid group mediange};
append from exp;

free all;
quit;



*STEP 13;
```

```
proc sort data=groupclass;
  by group;
run;

data groupclass groupclass0;
  set groupclass;
  if group = 0 then output groupclass0;
  else if group ^= 0 then output groupclass;
run;

proc means data=groupclass median noprint;
  var mediange;
  by group;
  output out=newgroupclass median=med;
run;

proc sort data=newgroupclass;
  by med;
run;

data newgroupclass(keep=group sgroup);
  set newgroupclass;
  if group=0 then delete;
  sgroup = _N_;
run;

proc sort data=newgroupclass;
  by group;
run;

data groups;
  merge groupclass newgroupclass;
```

```
  by group;
run;

data groups(keep=treatid group);
  set groupclass0 groups;
  if group=0 then sgroup=0;
  rename group = og;
  rename sgroup = group;
run;

proc datasets nolist;
  delete datasetf1 datasetf2 groupclass groupclass0
         nb newgroupclass nigene prepmatrix result1-result5
         sumni temp1-temp5;
run;

data groupclass;
  set groups;
  rename treatid=idobs;
run;

proc datasets nolist;
   delete groups;
run;

proc freq data=groupclass noprint;
tables group / out=grfr;
run;

data datanew;
  set &dataset;
  idobs = &treatment;
```

```
run;

proc sort data=groupclass;
  by idobs;
run;

data datanew;
  merge datanew groupclass;
  by idobs;
run;

proc sort data=groupclass;
by group;
run;

%mend ppclustel_m;
```

C.5 PPCLUSTEL - M (Σ_G)

```
%macro ppclustel_mg(dataset,treatment,subject,tmin,tmax,alpha, datasetout);



proc iml;
  start rankmiss(matrix1);
   matrixm = matrix1 = .;
nmiss = matrixm[,+];
nmiss = nmiss[+,];
miss = -9999999999;
matrix1 = choose(matrix1=.,miss,matrix1);
matrix1 = ranktie(matrix1);
matrix1 = matrix1 - nmiss;
matrix1 = choose(matrix1<=0,.,matrix1);
```

```
  finish;

  use &dataset;
  read all var{&treatment &subject} into factor;
  read all var("&tmin":"&tmax") into expression;
  b = ncol(expression);

  call rankmiss(expression);

  name1 = {treatid subjectid};
  name2 = ("&tmin":"&tmax");
  names = name1 || name2;

  rt = factor || expression;
  create datasetf1 from rt[colname=names];
  append from rt;

  create nb var{b};
  append from b;
quit;


data nb;
  set nb;
  call symput('b',b);
run;


*STEP 1 ;
proc freq data=datasetf1 noprint;
  tables treatid / out=nitreat(keep=treatid count);
run;
```

```
proc means data=nitreat sum noprint;
  var count;
  output out=sumni sum=sumni;
run;

data _null_;
  set sumni;
  call symput('sumni',sumni);
run;

*STEP 3;
data datasetf2 (drop= &tmin - &tmax);
  set datasetf1;
  array num(*) &tmin - &tmax;
  do time=1 to dim(num);
    exp = num[time];
  output;
  end;
run;

*STEP 4;
proc means data=datasetf2 median noprint;
  var exp;
  by treatid;
  output out=result1(keep=treatid median) median=median;
run;

*STEP 5;
proc sort data=datasetf2; by treatid time;
run;
```

```
*STEP 6;
proc means data=datasetf2 mean noprint;
  var exp;
  by treatid time;
  output out=result2(keep=treatid time rbijp) mean=rbijp;
run;

*STEP 7;
data temp1;
  merge datasetf2 result2;
  by treatid time;
run;

data temp2;
  merge temp1 nitreat;
  by treatid;
  msepartial = (exp - rbijp)**2 / (&b*count*(count-1));
run;

proc means data=temp2 sum noprint;
  var msepartial;
  by treatid;
  output out=result3(keep=treatid msep) sum=msep;
run;

*STEP 10;
proc means data=datasetf1 mean noprint;
  var &tmin - &tmax;
  by treatid;
  output out=result4(keep=treatid &tmin - &tmax) mean=&tmin - &tmax;
run;
```

```
*STEP 11;
data prepmatrix;
  merge result1 result3 result4 ;
  by treatid;
run;

*STEP 12;
proc iml;
*Function to remove row "i" from a matrix "x";
start delrow(x,i);
return(x[setdif(1:nrow(x),i),]);
finish delrow;

* Routine to sort lines of matrix according to column of medians;
*STEP 12.2;
start sortmed(matrix);
call sortndx(ms,matrix,{2});
matrix = matrix[ms,];
finish;

*STEP 12.3;
start sortcenter(matrix,tst);
   nr = nrow(matrix);
nc = ncol(matrix);
if tst = 1 then do;
   m2 = matrix;
end;
else if tst > 1 then do;
ed = tst-1;
    m1 = matrix[1:ed,];
m2 = matrix[tst:nr,];
end;
```

```
nr2 = nrow(m2);
  j1 = j(nr2,1) * 0;
c1 = int(0.35*nr2);
c2 = int(0.65*nr2);
call sort(m2,1);
do i = 1 to nr2;
   if i < c1 then j1[i,] = 1;
if i > c2 then j1[i,] = 1;
end;
m2 = m2 || j1;
nc2 = nc + 1;
call sort(m2,nc2);
       m2 = m2[,1:nc];
if tst = 1 then matrix = m2;
else if tst > 1 then matrix = m1 // m2;
finish;

* Main Routine for one-way anova routine when we have
  a large number of factors;

*STEP 12.4;
start anovalongrank(resp,a,b,pv);
use datasetf1;
read all into dat;
use nitreat;
read all into ni_vec;

newdat=j(1,b+2);
ni_vec_sub=1;
do j=1 to a;
tmp=dat[LOC(dat[,1]=resp[j,1]),];
newdat=newdat//tmp;
```

```
tmp=ni_vec[loc(ni_vec[,1]=resp[j,1]),2];
ni_vec_sub=ni_vec_sub//tmp;
end;


newdat=delrow(newdat,1);
ni_vec_sub=delrow(ni_vec_sub,1);

dat2=newdat[,3:(2+b)];
n=nrow(dat2);

*Computing jackknife estimator of sigma^4;
SampleMean=dat2[:,];
aux=dat2 - repeat(SampleMean,n); *Compute covariance matrix;
sigma=t(aux)*aux/(n);*changed, denominator n-1;
s2=t(aux)*aux/(n-1);
sigma4=sigma##2;


sum_sigma4_i=J(b,b,0);
do k=1 to n;
dat_i_jck=delrow(dat2,k);
n1=nrow(dat_i_jck);
SampleMean=dat_i_jck[:,];
aux=dat_i_jck - repeat(SampleMean,n1);*Compute covariance matrix;
sigma_i=t(aux)*aux/(n1);*changed, denominator n-1;
sigma4_i=sigma_i##2;
sum_sigma4_i=sum_sigma4_i+sigma4_i;
end;

sigma4_est_jck_i=n#sigma4-((n-1)/n)#sum_sigma4_i;
sigma4_est_jck_i=sigma4_est_jck_i#(sigma4_est_jck_i>0);*assigning zero to negati
```

```
    *Calculando valores de zeta1;
s4_jck=sigma4_est_jck_i[+,+];
numerador_vec = s4_jck@j(a,1);
denominador=(ni_vec_sub-1)#(ni_vec_sub);
prod2=numerador_vec/denominador;
zeta1=prod2[+,+];
zeta1=(2/&b)#zeta1;

*Calculando valores de zeta2;

numerador=(s2#s2)[+,+];
numerador_vec = numerador@j(a-1,1);



zeta2=0;
do i=1 to a;
ni=ni_vec_sub[i];
ni_vec1=repeat(ni,a-1,1);

noi=delrow(ni_vec_sub,i);

denom=1/(ni#noi);
prod=numerador_vec#denom;
zeta2=zeta2+prod[+];
end;

zeta2=(2/b)#zeta2;

cts = 4;
cte = 3 + b;
xijp = resp[,cts:cte];
```

```
        xpjp = xijp[+,]/a;
k1 = j(a,1);
k2 = j(b,1);
k = k1 * k2‘;
  xpjp = k # xpjp;
       mst = (xijp - xpjp) ## 2;
mst = mst[,+];
mst = mst[+,]/((a-1)*b);
       mse = resp[,3];
mse = mse[+,] / a;

zeta1 = zeta1/ a##2;
zeta2 = zeta2/ a##2;
*Montando covariancia da aproximao assinttica;
*Denominador;
N2=(ni_vec_sub[+]#b)##2;
sig4=(mse/N2)##2;
*Numerador;
tau2=zeta1+zeta2/(a-1)##2;
tau2=tau2/(N2##2);

cov=tau2/sig4;
Fg=mst/mse;
       pv = 1 - probnorm(sqrt(b)*(Fg-1)/sqrt(cov)); * changed;
store s2 s4_jck;
finish;


    start anovalongrank2(resp,a,b,pv);
use datasetf1;
read all into dat;
```

```
use nitreat;
read all into ni_vec;

ni_vec_sub=1;
do j=1 to a;
tmp=ni_vec[loc(ni_vec[,1]=resp[j,1]),2];
ni_vec_sub=ni_vec_sub//tmp;
end;

ni_vec_sub=delrow(ni_vec_sub,1);

load s2 s4_jck;

numerador_vec = s4_jck@j(a,1);
denominador=(ni_vec_sub-1)#(ni_vec_sub);
prod2=numerador_vec/denominador;
zeta1=prod2[+,+];
zeta1=(2/b)#zeta1;


numerador=(s2#s2)[+,+];
numerador_vec = numerador@j(a-1,1);


zeta2=0;
do i=1 to a;
ni=ni_vec_sub[i];
ni_vec1=repeat(ni,a-1,1);

noi=delrow(ni_vec_sub,i);

denom=1/(ni#noi);
```

```
prod=numerador_vec#denom;
zeta2=zeta2+prod[+];
end;

zeta2=(2/b)#zeta2;

cts = 4;
cte = 3 + b;
xijp = resp[,cts:cte];
        xpjp = xijp[+,]/a;
k1 = j(a,1);
k2 = j(b,1);
k = k1 * k2‘;
  xpjp = k # xpjp;
       mst = (xijp - xpjp) ## 2;
mst = mst[,+];
mst = mst[+,]/((a-1)*b);
       mse = resp[,3];
mse = mse[+,] / a;

zeta1 = zeta1/ a##2;
zeta2 = zeta2/ a##2;
*Montando covariancia da aproximao assinttica;
*Denominador;
N2=(ni_vec_sub[+]#b)##2;
sig4=(mse/N2)##2;
tau2=zeta1+zeta2/(a-1)##2;
tau2=tau2/(N2##2);

cov=tau2/sig4;
Fg=mst/mse;
       pv = 1 - probnorm(sqrt(b)*(Fg-1)/sqrt(cov)); * changed;
```

```
finish anovalongrank2;



*STEP 12.5;
   start indtest(exp,b,st,alpha,a,colgr,colts,g);
   exp[,colts] = 0; count = 0;
   do i=1 to a;
  if exp[i,colgr]=g then exp[i,colts] = 1;
end;
do i=st to a;
   if exp[i,colts] = 0 then do;
kwtest = 0;
exp[i,colts] = 1;
n = sum(exp[,colts]);
if kwtest = 0 then do;
             expt = exp[loc(exp[,colts]),];
call anovalongrank2(expt,n,b,pv);
if (pv > alpha) then exp[i,colgr] = g;
else if (pv <= alpha) then exp[i,colts] = 0;
end;
end;
end;
do i=1 to a;
   if exp[i,colgr] <= g then count = count + 1;
end;
st = count + 1;
call sort(exp,colgr);
kwtest = 0;
finish;

*STEP 12.1;
   use prepmatrix;
```

```
   read all into mdata;

   alpha = &alpha;
   g = 1;
ktest = 0;
a = nrow(mdata);
b = &b;
ncole = ncol(mdata);
call sortmed(mdata);

    colid = 1;
colgr = ncole + 1;
colts = ncole + 2;
j1 = j(a,1);
mdata = mdata || j1 * 9999;
mdata = mdata || j1 * 0;
tstart = 1; tend = a;

call sortcenter(mdata,tstart);
    call anovalongrank(mdata,a,b,pv);

   pv2 = pv;
   if (pv > alpha) then do;
   mdata[,colgr] = 1;
   end;
if (pv <= alpha) then do;
L = (tend - tstart + 1)/2;
tend = tstart + int(L-1);
do while (tstart <= a);
   if (pv2 > alpha) then g = g + 1;
call sortcenter(mdata,tstart);
mdata[,colts] = 0;
```

```
do i = 1 to a;
   if (tstart <= i) then do;
  if ( i <= tend) then mdata[i,colts] = 1;
end;
end;
n = sum(mdata[,colts]);
if (n = 1) then do;
   mdata[tstart,colgr] = 0;
tstart = tend + 1;
tend = a;
ktest = -1;
end;
exp = mdata[loc(mdata[,colts]),];
if ktest = 0 then call anovalongrank(exp,n,b,pv);
pv2 = pv;
if (pv > alpha) then do;
   do i = 1 to a;
if (tstart <= i) then do;
  if ( i <= tend) then mdata[i,colgr] = g;
end;
end;
tstart = tend + 1;
tend = a;
st = tstart;  exp = mdata;
call indtest(exp,b,st,alpha,a,colgr,colts,g);
tstart = st;
mdata = exp;
end;
else if (pv <= alpha) then do;
L = (tend - tstart + 1)*0.9;
tend = tstart + int(L-1);
end;
```

```
end;
end;
exp = mdata[,colid];
exp = exp || mdata[,colgr] || mdata[,2];

create groupclass var{treatid group mediange};
append from exp;

free all;
quit;



*STEP 13;
proc sort data=groupclass;
  by group;
run;
data groupclass groupclass0;
  set groupclass;
  if group = 0 then output groupclass0;
  else if group ^= 0 then output groupclass;
run;

proc means data=groupclass median noprint;
  var mediange;
  by group;
  output out=newgroupclass median=med;
run;

proc sort data=newgroupclass;
  by med;
run;
```

```
data newgroupclass(keep=group sgroup);
  set newgroupclass;
  if group=0 then delete;
  sgroup = _N_;
run;

proc sort data=newgroupclass;
  by group;
run;

data groups;
  merge groupclass newgroupclass;
  by group;
run;

data groups(keep=treatid group);
  set groupclass0 groups;
  if group=0 then sgroup=0;
  rename group = og;
  rename sgroup = group;
run;

/*proc datasets nolist;  */
/*  delete datasetf1 datasetf2 groupclass groupclass0*/
/*         nb newgroupclass nigene prepmatrix result1-result5*/
/*         sumni temp1-temp5;  */
/*run;*/

data groupclass2;
  set groups;
  rename treatid=idobs;
run;
```

```
proc datasets nolist;
   delete groups;
run;

proc freq data=groupclass2 noprint;
tables group / out=grfr;
run;

data datanew;
  set &dataset;
  idobs = &treatment;
run;

proc sort data=groupclass2;
  by idobs;
run;

data datanew;
  merge datanew groupclass2;
  by idobs;
run;

proc sort data=groupclass2;
by group;
run;

data &datasetout; set groupclass2; run;
%mend ppclustel_mg;
```