UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE ESTATÍSTICA

Pedro Henrique de Mesquita Pacheco

Modelagem de dados longitudinais complexos no R: desenvolvimento de um pacote estatístico

Juiz de Fora
2021

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Pacheco, Pedro.
Modelagem de dados longitudinais complexos no R: desenvolvimento de um pacote estatístico / Pedro Henrique de Mesquita Pacheco. – 2021.
94 f. : il.

Orientador: Marcel de Toledo Vieira
Trabalho de Conclusão de Curso – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Departamento de Estatística, 2021.

1. Modelos Longitudinais. 2. Amostragem Complexa. 3. Programação em R. I. Vieira, Marcel, orient. II. Título.

Pedro Henrique de Mesquita Pacheco

Modelagem de dados longitudinais complexos no R: desenvolvimento de um pacote estatístico

Trabalho de conclusão de curso apresentado ao Departamento de Estatística da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de bacharel em Estatística

Orientador: Prof. Dr. Marcel de Toledo Vieira

Juiz de Fora

2021

Pedro Henrique de Mesquita Pacheco

Modelagem de dados longitudinais complexos no R: desenvolvimento de um pacote estatístico

Trabalho de conclusão de curso apresentado ao Departamento de Estatística da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de bacharel em Estatística

Aprovado em 08 de setembro de 2021

BANCA EXAMINADORA

Prof. Dr. Marcel de Toledo Vieira - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Augusto Carvalho Souza
Universidade Federal de Juiz de Fora

Prof. Dr. Pedro Luis do Nascimento Silva
Escola Nacional de Ciências Estatísticas

Dedico este trabalho a minha Senhora e Mãe, a sempre Virgem Maria, a quem pertence tudo o que tenho e sou.

AGRADECIMENTOS

Acima de tudo, agradeço à Deus pelo dom da minha vida e por todas as bençãos que recebi e recebo todos os dias de Suas mãos. Foi graças a Sua Divina Providência que hoje sou quem sou. Agradeço aos anjos e santos, por intercederem junto a Deus por mim. Em especial, meus queridos Santo Agostinho, Santa Teresa D'Ávila e São João Paulo II, que me mostraram o verdadeiro caminho para a sabedoria. Ao meu Glorioso São José, que sempre me protegeu mesmo nos momentos mais difíceis e aos santos Tomás de Aquino e Antônio, meus eternos professores.

Agradeço aos meus pais, por sempre me educarem na fé e na razão, me ensinando que a família é, e sempre será, a base que sustenta o homem. Agradeço profundamente por doarem suas próprias vidas em favor da minha, por meio dos sacrifícios que a paternidade e maternidade lhes exigiram. Em particular, ao meu pai, por me ensinar a desenvolver um pensamento crítico e matemático, e à minha mãe, por me ensinar o valor do serviço e da doação.

Aos meus familiares e amigos, por todo o apoio que me concederam nestes quatro anos de caminhada. Tudo isto só foi possível graças aos móveis, panelas, potes, utensílios, e etc que me foram doados por cada um de vocês para compor minha casa. Em especial, agradeço minhas três queridas avós, Cida, Zúlcar e Cidinha, por todo o amor que me deram em toda a minha vida. Agradeço ainda ao meu grande amigo Roberto, que sempre se prontificou em ajudar a transportar minhas mudanças, seja para onde fosse, e a minha madrinha Imaculada, por ser minha segunda mãe.

Agradeço a minha querida namorada, Gabriela, que tive a felicidade de conhecer durante a faculdade, por ser minha companheira em todos os momentos, me consolando em todas as dificuldades e me aturando quando fui casmurro, mas sempre me incentivando a ser um homem cada vez melhor. Sua presença foi e é fundamental para o meu crescimento pessoal, espiritual e profissional. Agradeço também ao meu irmão, Paulo, pelas longas conversas e debates, buscando sempre refinar nossos conhecimentos, e por me introduzir no ramo da tecnologia, me ensinando e me apaixonando por computadores. Essas duas pessoas se tornaram meus verdadeiros braços direito e esquerdo.

Ao Marcel, meu admirável orientador e amigo, por ter acreditado em mim e me ter confiado tarefas desafiadoras ao longo destes anos. Não tenho palavras para descrever minha profunda gratidão, mas ressalto que sua pessoa foi, e continua sendo, minha maior fonte de inspiração profissional. Aos professores Augusto e Lupércio, também meus amigos, que me acolheram na estatística como grandes tutores e me guiaram ao longo destes anos. Aos três, obrigado por me ensinarem o valor de nossa profissão.

Aos demais professores do departamento de estatística da UFJF que tive a honra de poder ser aluno, cujos nomes gostaria de citar individualmente: Ângela, Camila, Clécio,

Gustavo, Henrique, Joaquim, Ronaldo, Thiago e Tufi. Cada um marcou a minha vida sob algum aspecto e serei eternamente grato por isso. Em especial, agradeço ao professor Clécio por todos os desafios propostos ao longo destes anos, sempre me incentivando a subir mais um degrau na busca pelo conhecimento. Agradeço também aos professores do departamento de matemática: Beatriz, Flaviana, Lucy e Sandro, por me acolherem principalmente nos primeiros anos da minha formação profissional. Nada disto seria possível sem a contribuição de cada um deles.

Agradeço a todos os amigos que fiz durante esses anos morando em Juiz de Fora, pela companhia que a mim fizeram, dando suporte para vencer todas as dificuldades diárias. Em especial, agradeço aos amigos: Bruno, Fábio, Gulliver, Raphael e Walter, por todas as noites na Universidade, e não eram poucas, estudando, conversando, mas principalmente rindo! Com eles vivi histórias que guardarei na memória pelo resto da vida.

Agradeço à Universidade Federal de Juiz de Fora por fornecer toda a estrutura necessária para a minha formação profissional. Em especial, agradeço à PROPP e à PROEX, pelo fornecimento de bolsas de Iniciação Científica e Extensão. Graças a essas duas pró-reitorias, tive a oportunidade de contar com o auxílio de bolsas de estudo em todos os períodos da faculdade e isto foi fundamental para o meu sustento em Juiz de Fora.

Finalmente, entrego tudo o que vivi ao longo destes anos em Juiz de Fora, em especial este trabalho, a minha querida Mãe, Nossa Senhora, a quem pertence todas as minhas obras, pedindo que ela continue sempre amparando e conduzindo a minha vida, assim como a de todos que estiveram nesta caminhada comigo. Muito obrigado!

A matemática como expressão da mente humana reflete a vontade ativa, a razão contemplativa e o desejo de perfeição estética. (COURANT E ROBBINS, 1941).

RESUMO

O surgimento dos computadores digitais a partir da segunda metade do século XX revolucionou o uso da estatística, pois, possibilitou que metodologias, antes inviáveis de serem colocadas em prática, fossem exploradas. Em particular, pesquisas longitudinais começaram a ser cada mais utilizadas em um contexto envolvendo grandes pesquisas. Surgiram então problemas sobre a incorporação do plano amostral das pesquisas às metodologias matemáticas e também a tradução de tais abstrações matemáticas para termos práticos computacionais. No que diz respeito a modelos longitudinais de efeitos fixos e aleatórios e modelos longitudinais de covariância estruturada, ambos envolvendo o plano amostral complexo, o presente trabalho visa apresentar técnicas computacionais capazes de solucionar a segunda parte do problema da modelagem estatística destacado anteriormente, por meio da discussão e criação de um pacote na linguagem de programação R, denominado 'clm'. Ao longo deste estudo foram discutidas boas práticas da programação que visam o desenvolvimento de um código limpo e sustentável e também possíveis estratégias a serem adotadas para solucionar o problema de implementação dos modelos longitudinais complexos, como por exemplo, o uso de fluxogramas. Ao final, foram apresentados os resultados de tais práticas analisando as principais funções do pacote desenvolvido e também uma aplicação exemplificando seu uso, buscando destacar os benefícios proporcionados por este trabalho.

Palavras-chave: Modelos Longitudinais. Amostragem Complexa. Programação em R.

ABSTRACT

The emergence of digital computers in the second half of the 20st century revolutionized the use of statistics, as it made it possible for methodologies, which were previously unfeasible to be put into practice, to be explored. In particular, longitudinal surveys began to be increasingly used in a context of large surveys. Problems then arose about incorporating the sampling design of the survey into mathematical methodologies and also the translation of such mathematical abstractions into practical computational terms. With regard to fixed and random effects longitudinal models and structured covariance longitudinal models, both involving the complex sampling scheme, the present work aims to present computational techniques capable of solving the second part of the statistical modeling problem highlighted above, through the discussion and creation of a package in the R programming language, called 'clm'. Throughout the study, good programming practices aimed at the development of a clean and sustainable code were discussed, as well as possible strategies to be adopted as the solution for the problem of implementing longitudinal models, such as the use of flowcharts. In the end, the results of such practices were presented analyzing the main functions of the developed package and also an application exemplifying its use, seeking to highlight the benefits provided by this work.

Keywords: Longitudinal Models. Complex Sampling Design. R Programming.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 PESQUISAS LONGITUDINAIS

Desde o princípio da espécie humana o homem se destacou por sua capacidade de abstrair a realidade a um nível mais elevado que os demais animais. O desenvolvimento da racionalidade humana possibilitou que o homem construísse relações com a natureza cada vez mais sofisticadas, como, por exemplo, a formulação teórica e prática da matemática, que o ajudou a superar grandes obstáculos. Foi nessa tentativa de abstrair a realidade na qual o homem está inserido é que a estatística foi desenvolvida (Courant e Robbins, 1941).

De forma especial, a segunda metade do século XX foi marcada por grandes mudanças na sociedade. Em particular, o surgimento dos computadores revolucionou o uso da estatística, pois, possibilitou que metodologias, antes inviáveis de serem colocadas em prática, fossem exploradas. Entre elas, as pesquisas longitudinais começaram a se destacar e vem sendo amplamente utilizadas nos últimos tempos para o estudo de fenômenos sociais, demográficos, políticos e econômicos, além de estudos epidemiológicos, das áreas agrícola e industrial (Vieira, 2012).

Ao se idealizar uma pesquisa é necessário ter em mente qual o tipo de estudo será implementado. Na estatística existem diferentes classificações para os tipos de estudos, dentre elas podemos dividir as pesquisas em transversais e longitudinais. Os estudos transversais consistem em coletar dados em um único instante do tempo enquanto os longitudinais buscam reunir informações em mais de um momento. Interessante notar que a perspectiva do tempo é o grande diferencial das duas abordagens e de maneira intuitiva se percebe que a variação do tempo pode vir a ser um fator importante da modelagem (Skinner, 2003).

Entretanto, a adição desta nova dimensão traz consigo grandes dificuldades que devem ser cuidadosamente analisadas. Um primeiro ponto poderia ser a respeito da forma de coleta dos dados. Pesquisas transversais repetidas formam um tipo de estudo realizado ao longo do tempo em que as mesmas variáveis de interesse são pesquisadas em diferentes amostras independentes entre si. Existem também as pesquisas retrospectivas que são estudos realizados em um único instante do tempo em que os integrantes da pesquisa devem responder perguntas a respeito do passado. De acordo com Vieira (2005), este tipo de estudo sofre certos tipos de incerteza, a saber: (i) subnotificação: falha ao declarar eventos; (ii) Sobrenotificação: descrever situações que não aconteceram; (iii) Efeito telescópio: suscetibilidade para sobre ou subestimar o tempo desde o evento; (iv) Amontoação: tendência em arredondar o tempo desde o fato para um certo número; e (v) Subestimar a mudança dentro do período de referência.

Finalmente, existem as pesquisas prospectivas em painel que são estudos em que as informações são de fato coletadas ao longo do tempo. Neste caso, se destacam as pesquisas

do tipo painel fixo, em que as mesmas ou outras variáveis são medidas para os mesmos indivíduos, ou seja, a amostra se mantém a mesma; e as do tipo painel rotativo, em que grupos de unidades amostrais podem ser incluídos ou retirados da amostra segundo algum critério (Vieira, 2012). O foco deste presente trabalho é lidar com dados de pesquisas do tipo painel fixo, mas vale ressaltar que o contexto pode ser expandido para pesquisas do tipo painel rotativo contendo subamostras do tipo painel fixo ou também pesquisas transversais envolvendo análise multivariada.

1.2 MÉTODOS DE AMOSTRAGEM

De uma forma geral, é de amplo conhecimento na teoria geral da amostragem que, apesar da simplicidade teórica ao se considerar um esquema de amostragem aleatória simples (AAS), na prática esta metodologia é muito pouco adotada, pois, dadas as mesmas restrições orçamentárias, métodos probabilísticos diferentes de AAS podem fornecer informações mais precisas que esta última. Somado a isso, dificilmente se encontram cadastros adequados para esta forma de amostragem e é comum ocorrer nas pesquisas situações de não resposta, obrigando o pesquisador a considerar observações com pesos desiguais (Vieira, 2005). Além disso, a má especificação nas análises do plano amostral usado para a seleção da amostra pode resultar em estimativas enviesadas, reforçando a necessidade de se estudar metodologias que considerem o esquema de amostragem adotado (Battisti, 2008).

Além das questões intrínsecas aos dados longitudinais apresentadas na seção anterior e também dos problemas gerais da amostragem já descritos, é importante discutir a respeito do método de amostragem adotado nas pesquisas longitudinais, pois, os efeitos da não consideração correta do plano amostral podem vir a ser ainda maiores neste tipo de pesquisa (Vieira, Smith e Salgueiro, 2016). Outro ponto associado ao estudo do tipo painel é que a amostra é selecionada apenas no primeiro instante do tempo e depois é acompanhada nas demais rodadas. Logo, uma questão a se considerar é como fazer com que a amostra continue permitindo a realização de inferência estatística para toda a população, uma vez que ela foi selecionada aleatoriamente na primeira rodada da pesquisa.

Os planos amostrais complexos geralmente são resultantes de uma combinação de AAS com e sem reposição com outro(s) plano(s) amostral(is), como por exemplo: (i) Amostragem estratificada; (ii) Amostragem conglomerada; e (iii) Amostragem com probabilidades desiguais; dentre outros. No primeiro caso, define-se a variável de estratificação que formará os estratos e posteriormente se seleciona dentro de cada estrato um conjunto de unidades de forma aleatória. No segundo caso, define-se a variável de conglomeração que formará os conglomerados. Seleciona-se então um conjunto de conglomerados de forma aleatória e se coleta os dados de todos os indivíduos pertencentes a esses grupos (no caso da seleção em um estágio). No terceiro caso, usam-se probabilidades desiguais quando as

unidades amostrais tem variação de tamanho e este tamanho está fortemente relacionado com as variáveis de interesse. Em todos os casos definem-se também os pesos amostrais que, além de lidarem com as diferentes probabilidades de seleção, podem ser ajustados para correção da não resposta (Battisti, 2008).

Sendo assim, como revisão, consideramos inicialmente o caso da seleção de uma amostra para o caso transversal.

1.2.1 Caso transversal

Seja $\boldsymbol{v} = \{1, \ldots, N\}$ uma população finita de tamanho N e $\boldsymbol{S} = \{i_1, \ldots, i_n\}$ uma amostra tal que $\boldsymbol{S}$ é um subconjunto de $\boldsymbol{v}$ e n é o tamanho da amostra com $1 \leq n \leq N$. Definimos o esquema de amostragem como $p(\boldsymbol{S})\ \forall \boldsymbol{S} \subset \boldsymbol{\zeta}$ em que $\boldsymbol{\zeta}$ é o conjunto de todas as possíveis amostras de $\boldsymbol{v}$.

Seja $\boldsymbol{s}$ a amostra de fato selecionada e $p(\boldsymbol{s}) = P(\boldsymbol{S} = \boldsymbol{s})$ o plano amostral da pesquisa. Vale destacar as seguintes propriedades do esquema de amostragem (Vieira, 2012):

$$0 \leq p(\boldsymbol{S}) \leq 1\ \forall \boldsymbol{S} \subset \boldsymbol{\zeta} \tag{1.1}$$

e

$$\sum_{\boldsymbol{s}} p(\boldsymbol{s}) = 1. \tag{1.2}$$

Considere w_i o peso amostral do indivíduo i na amostra $\boldsymbol{s}$ e π_i a probabilidade de inclusão do i-ésimo indivíduo na amostra $\boldsymbol{s}$. Assim, caso não haja não resposta, temos

$$w_i = 1/\pi_i \tag{1.3}$$

e

$$\pi_i = P(i \in \boldsymbol{s}) = \sum_{i \in \boldsymbol{s}} p(\boldsymbol{s}) \tag{1.4}$$

com $\pi_i > 0\ \forall i \in \boldsymbol{v}$.

Note que no contexto da amostragem aleatória simples temos $p(\boldsymbol{s}) = \dfrac{1}{\binom{N}{n}}$. Dessas $\binom{N}{n}$ possíveis amostras o indivíduo i pertence a $\binom{N-1}{n-1}$. Logo, todos os indivíduos possuem a mesma probabilidade de seleção e, por conseguinte, mesmo peso amostral, isto é, $\pi_i = \dfrac{n}{N}$ e $w_i = \dfrac{N}{n}\ \forall i \in \boldsymbol{s}$.

Finalmente, seja z_i qualquer variável que não dependa da escolha de $\boldsymbol{s}$. Segundo Vieira (2005), temos

$$E_p\left(\sum_{i \in \boldsymbol{s}} w_i z_i\right) = E_p\left(\sum_{i \in \boldsymbol{s}} \pi_i^{-1} z_i\right) = \sum_{i=1}^{N} z_i \tag{1.5}$$

e

$$\operatorname*{plim}_{n\to\infty}\left[\frac{\sum\limits_{i\in\boldsymbol{s}} w_i z_i}{\sum\limits_{i=1}^{N} z_i}\right] = 1, \tag{1.6}$$

que mostra a consistência do estimador de Horvitz-Thompson, definido como

$$\hat{Y}_{HT} = \sum_{i\in\boldsymbol{s}} w_i z_i, \tag{1.7}$$

em relação ao plano amostral (Vieira, 2005) utilizado para estimar o total de uma população. $E_p(.)$ descrito acima representa o valor esperado de uma estatística sob a distribuição de amostragem gerada por $p(\boldsymbol{s})$. Este resultado será muito importante para este trabalho pois garantirá a consistência dos estimadores mesmo sob um desenho amostral complexo.

1.2.2 Caso longitudinal

Conforme dito anteriormente, em pesquisas longitudinais do tipo painel a amostra é selecionada no primeiro instante do tempo e acompanhada nas demais rodadas. Portanto, o processo de seleção da amostra em pesquisas longitudinais passa a ser semelhante ao caso das pesquisas transversais e para garantir que se consiga fazer inferência estatística ao longo do tempo vamos assumir, como Vieira (2012), que a população $\boldsymbol{\upsilon}$ se mantém fixa em todas as rodadas da pesquisa.

Além disso, nas pesquisas longitudinais o peso amostral w_i exige ainda mais cuidados. Além de incorporar as probabilidades de seleção e a não-resposta na rodada conforme ocorre no caso transversal, o peso amostral deve passar a incluir o ajuste para não-resposta também nas rodadas anteriores da pesquisa. Desse modo, w_i deve ser recalculado em todas as ocasiões. Entretanto, em análises que consideram dados de todas as rodadas utiliza-se o peso da última rodada, que será denotado por w_{iT}.

1.3 MODELOS DE SUPERPOPULAÇÃO

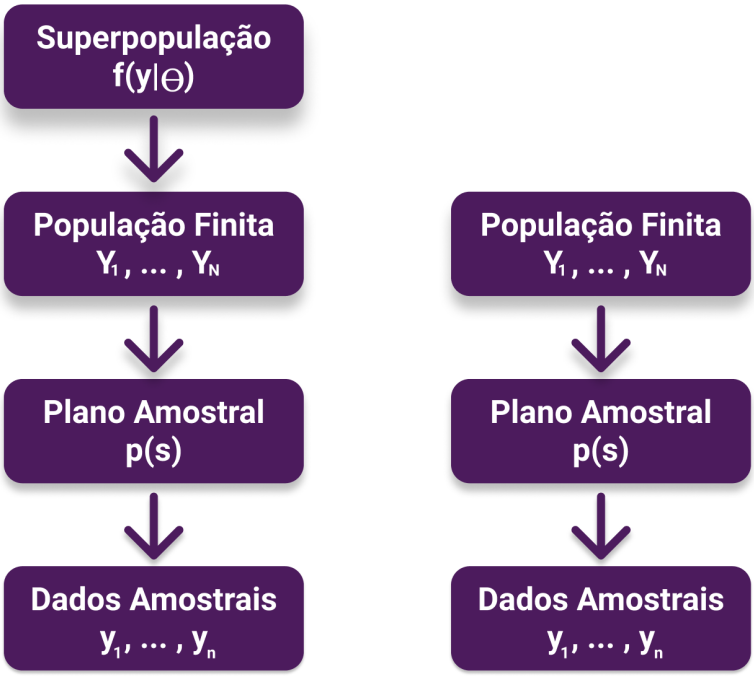
A área de modelagem estatística de regressão recebeu um grande impulso desde a criação dos modelos lineares generalizados (MLGs) no início da década de 1970. Desde então, essa grande área da estatística tem se tornado cada vez mais o principal campo de interesse dos pesquisadores modernos, sendo desenvolvidas novas metodologias cada vez mais complexas a cada ano (James et al., 2013). É também neste contexto que se insere a análise de dados longitudinais com dados amostrais complexos.

Como é de costume acontecer na estatística, a maior parte das metodologias leva em consideração que os dados foram coletados por amostragem aleatória simples, mas conforme dito anteriormente grande parte das pesquisas adotadas atualmente não envolvem este esquema de amostragem. Assim, torna-se necessário adaptar as tradicionais metodologias

de regressão para dados de amostras envolvendo plano amostral complexo. Para isso, surge o conceito de superpopulação.

Na inferência clássica, dada uma população de interesse $\boldsymbol{\upsilon}$, seleciona-se uma amostra aleatória $\boldsymbol{s}$ e a partir desta amostra se conduz inferência sobre algum parâmetro de interesse da população. Na abordagem de superpopulação, assume-se que a própria população é gerada por um mecanismo denominado superpopulação. A partir da população observada, seleciona-se uma amostra aleatória sob algum plano amostral complexo e por meio desta amostra se busca fazer inferência sobre algum parâmetro de interesse da superpopulação. O diagrama a seguir descreve de forma visual a diferença entre as abordagens.

Figura 1 - Diagrama comparativo entre caso clássico e modelos de superpopulação



Fonte: Elaborado pelo autor. (2021).

Nos próximos capítulos esta ideia será melhor sistematizada. Por ora, basta compreender as consequências ao se adotar este tipo de metodologia. A principal delas é a possibilidade de ajustar modelos de regressão para dados amostrais complexos, permitindo que seja feita inferência a respeito das características do mecanismo gerador das observações na população, o que também possibilita, em especial, a inferência causal.

1.3.1 Modelos longitudinais

Conforme descreve Vieira (2005), métodos para modelagem longitudinal devem considerar a variação da variável resposta $\boldsymbol{Y}$ na população, assim como ao longo do tempo. Ou seja, é importante que a metodologia proposta seja capaz de lidar com as variações intra-indivíduos — que são as mudanças que ocorrem no mesmo indivíduo em um intervalo de tempo — e as inter-indivíduos — que são aquelas existentes entre os diferentes indivíduos em um determinado instante. Outra questão importante é a forma como é mensurado o próprio tempo, isto é, de maneira discreta ou contínua. No presente trabalho apenas a opção de tempo discreto será abordada, sendo as rodadas equidistantes.

Feitas estas considerações, alguns exemplos de metodologias são: (i) Modelos Hierárquicos, ou também conhecidos como Modelos Multiníveis, que consistem em agrupar as unidades amostrais em níveis de hierarquia e, assim, sermos capazes de considerar a relação tanto entre os indivíduos dentro do mesmo nível quanto entre os níveis; (ii) Modelos Marginais, que se resumem em modelar a média marginal da variável de interesse; (iii) Modelos de Efeitos Fixos e Aleatórios, que são casos particulares dos modelos hierárquicos, e que partem do princípio de que os coeficientes do modelo variam entre indivíduos (esta abordagem é bastante útil para se estudar traços latentes não observáveis da população); e (iv) Modelos de Transição, que envolvem adicionar como covariáveis do modelo a própria variável resposta e suas covariáveis em tempos anteriores. No presente trabalho iremos abordar a classe de modelos de efeitos fixos e aleatórios.

1.4 SOFTWARE R E PACOTES ESTATÍSTICOS

"O R é um ambiente de 'software' livre para computação estatística e gráficos. Compila e roda em uma ampla variedade de plataformas UNIX, Windows e MacOS" (R Project, 2021). Atualmente, porém, o R se tornou muito mais do que um simples ambiente de *software* estatístico e se estabeleceu como uma das principais linguagens de programação de alto nível com potencial para lidar com problemas de modelagem estatística, contando com uma série de ferramentas modernas que permitem inclusive o desenvolvimento de aplicativos para a internet e manipulação de grandes bases de dados.

O uso do R tem se dado principalmente por estatísticos, matemáticos, programadores e cientistas contemporâneos no geral. As próprias universidades do Brasil e do mundo têm se esforçado bastante para ensinar e incentivar seus alunos a trabalharem com a ferramenta e o motivo desta operação tem sido a necessidade computacional no desenvolvimento de estudos científicos. O resultado deste necessário esforço é que cada vez mais novos pacotes capazes de lidarem com problemas cada vez mais complexos estão sendo desenvolvidos em um intervalo de tempo cada vez menor, possibilitando que os demais usuários da comunidade usufruam deste material e apliquem em suas respectivas áreas de interesse.

O R oferece duas principais formas de disponibilizar um novo pacote, são eles: (*i*) Github: plataforma virtual gratuita para armazenamento de repositórios utilizando o sistema Git; (*ii*) CRAN: repositório oficial do Projeto R contendo os grandes pacotes da linguagem. Normalmente, quando um pacote no R é desenvolvido, primeiro o autor disponibiliza no Github uma versão de desenvolvimento e após ele se mostrar consistente, ele coloca no CRAN a versão estável, contendo inclusive um manual com as principais funcionalidades do pacote. Até o presente momento, não foi possível realizar a submissão de inclusão do pacote fruto deste trabalho para o CRAN, estando, portanto, disponível apenas no Github.

1.5 MOTIVAÇÃO E OBJETIVOS

Entre os estatísticos e matemáticos, principalmente, existe uma grande discussão a respeito do que é estatística, ou melhor, quais são os limites da área no que diz respeito a sua interseção com a matemática. Deixando a profundidade filosófica do assunto de lado, neste trabalho, iremos considerar toda a parte que envolve teorização matemática como simplesmente matemática, a parte que envolve estudo da coleta, organização, análise e interpretação dos dados como sendo estatística e toda a parte que envolve aplicação computacional como sendo computação.

Até o final do século XIX, a profissão do estatístico estava fortemente ligada a uma dualidade matemática e estatística. A grande dificuldade, no entanto, era a inviabilidade de aplicar em larga escala as técnicas desenvolvidas. O advento da era computacional no século XX afetou a sociedade como um todo e, em especial, o trabalho dos estatísticos. Com isso, a profissão pôde sair dessa dualidade e passar a ser composta pela tríplice combinação entre estatística, matemática e computação. A necessidade de se projetar grandes pesquisas e analisar grandes bases de dados considerando todos os aspectos estatísticos e matemáticos tem obrigado o estatístico não somente a compreender a metodologia teórica, mas também saber aplicá-la computacionalmente. A dinâmica envolvendo estas três áreas ficou conhecida como modelagem estatística (James et al., 2013).

Figura 2 - Charge sobre modelagem estatística

Fonte: Elaborado por Gabriela Theotonio dos Santos. (2021).

Apesar do humor na charge acima, a metáfora do escultor representa bem o papel do estatístico contemporâneo e mostra a necessidade de haver um equilíbrio entre as três áreas mencionadas para que se tenha um processo completo de modelagem estatística. A ausência de qualquer um desses elementos inviabiliza a obtenção do produto final, que é o modelo. Por um lado, conforme já apresentamos anteriormente, existe a dificuldade de incorporar às teorias matemáticas toda a metodologia estatística envolvendo a incorporação do plano amostral complexo das pesquisas. Por outro lado, existe uma dificuldade de traduzir para termos práticos computacionais aquilo que são abstrações matemáticas desenvolvidas pela etapa anterior.

É, portanto, importante que o estatístico tenha a habilidade de encarar qualquer problema considerando estes três aspectos. A primeira etapa do processo, que é a integração entre a matemática e a estatística, foi justamente o trabalho desenvolvido por Vieira (2005), que buscou incorporar à classe de modelos longitudinais de efeitos fixos e aleatórios as características do plano amostral complexo, além de realizar o mesmo para modelos de covariância estruturada. Já o presente trabalho busca realizar a segunda etapa do processo, que é uma integração mais eficiente desta metodologia com a computação, e

disponibilizar para a comunidade um pacote no R capaz de lidar com dados longitudinais envolvendo o plano amostral complexo.

A metáfora do escultor é de grande importância para aqueles que desejam compreender os objetivos deste trabalho. Antes do escultor martelar o cinzel sobre a pedra ele precisa ter uma série de cuidados, como escolher a pedra ideal para seu trabalho, verificar se existem rachaduras na mesma, providenciar as ferramentas necessárias como óculos de proteção, máscaras contra poeira, luvas, cinzéis e martelos, fazer um esboço do desenho na pedra e cuidar para que a pedra não rache, para só então começar de fato seu trabalho. Mesmo assim, durante o próprio processo de esculpimento é necessário muita atenção e paciência, tomando cuidado com os detalhes, além de dominar as técnicas necessárias. No fim, vem a etapa dos toques finais, que envolve o uso de diversas lixas para refinar a obra e remover eventuais marcas do cinzel.

Neste sentido, da mesma maneira que o escultor precisa estudar cuidadosamente a melhor forma de esculpir a pedra a fim de obter a escultura mais bela, assim também o programador precisa estudar a melhor forma de estruturar o seu código, de maneira que o modelo seja o mais limpo e eficiente o possível. Assim, espera-se com esse trabalho apresentar técnicas e discussões sobre uma possível abordagem computacional para a metodologia de dados longitudinais amostrais complexos. Além disso, no presente trabalho, o R será abordado não na perspectiva do usuário que irá usufruir do material desenvolvido, mas sim na visão de quem pretende implementar nesta linguagem de programação alguma metodologia estatística e disponibilizar para uso dos demais membros da comunidade o conteúdo desenvolvido.

No próximo capítulo será apresentada uma discussão a respeito da composição de algoritmos, programas de computador, arquitetura de software, princípios para programação funcional, fábrica de funções, as raízes da linguagem R e metaprogramação, além de reflexões sobre o desenvolvimento da programação ao longo do tempo.

No capítulo 3 será feita uma revisão sobre a metodologia desenvolvida por Vieira (2005) para estimação dos efeitos fixos do modelo e também da matriz de covariância, além dos seus respectivos erros-padrão. Serão discutidas formas de se estruturar dados multidimensionais no R considerando as vantagens e desvantagens de cada abordagem e serão organizados fluxogramas contendo os passos a serem seguidos para estimação dos parâmetros, além de discutir questões de generalização e aproveitamento de funções. Por fim, será abordada a comunicação da função principal com o usuário, através dos parâmetros de entrada e de saída.

No capítulo 4 será feito o mesmo processo, porém para o caso de modelos de covariância estruturada. Será feita uma revisão sobre a metodologia desenvolvida por Vieira (2005) para estimação pontual dos efeitos aleatórios permanentes e transitórios e também seus respectivos erros-padrão. Serão apresentados os fluxogramas a serem

seguidos, questões de aproveitamento de funções, processo de otimização e a comunicação com o usuário, além de conter algumas medidas de qualidade de ajuste desenvolvidas por Vieira (2005) que irão servir como forma de complementar a utilidade do pacote desenvolvido.

No quinto capítulo será apresentada a implementação de algumas funcionalidades internas descritas nos capítulos anteriores e também a aplicação das funções a um conjunto de dados mostrando na prática os resultados do pacote desenvolvido. Finalmente, será feito uma conclusão retomando os principais pontos abordados com algumas considerações finais e também apresentando sugestões de implementação para o futuro.

2 LINGUAGEM R E PROGRAMAÇÃO FUNCIONAL

2.1 ALGORITMOS E PROGRAMAS DE COMPUTADOR

Um programa de computador pode ser definido como "o produto resultante da atividade intelectual de um programador. Essa atividade, por sua vez, depende de um treinamento prévio em abstração e modelagem de problemas, bem como do uso da lógica na verificação das soluções" (Souza et al., 2011). Em outras palavras, um *software* é o resultado de um conjunto de instruções bem definidas por um humano de forma lógica a fim de serem executadas por uma máquina de maneira a cumprir algum objetivo. Atualmente, essas máquinas consistem em computadores digitais que funcionam através de processadores digitais operando por meio de sinais elétricos binários. A representação matemática deste processo pode ser feita utilizando uma sequência de zeros e uns. Definimos este conjunto de instruções lógicas passadas a um computador como sendo um algoritmo.

Um dos primeiros grandes problemas enfrentados pelos programadores de computadores digitais na década de 1940 foi a dificuldade de se escrever um algoritmo por meio de uma sequência de zeros e uns, também conhecido como linguagem de máquina, pois exigia um esforço intelectual muito elevado e o número de erros de implementação lógica eram altos. Assim, com o objetivo de tornar a programação mais simples para o ser humano foram surgindo as linguagens de programação, que permitiam que o programador escrevesse algoritmos por meio de códigos que seriam traduzidos posteriormente para a linguagem de máquina. Foi na década de 1950 que surgiram linguagens como *Assembly*, *FORTRAN* e *COBOL*.

Entretanto, note que o processo de construção lógica de um algoritmo não depende da linguagem de programação que será utilizada e portanto pode ser discutido de maneira geral. A construção de um algoritmo é semelhante à escrita de um texto de redação, no sentido de que ambos devem conter um início, meio e fim. No caso de um algoritmo, o início consiste nos parâmetros de entrada, o meio nos passos a serem seguidos e o fim nos parâmetros de saída. Os parâmetros de entrada consistem em um conjunto de dados que devem ser fornecidos por um humano ou uma máquina de forma a serem utilizados pelo corpo do algoritmo no processamento de dados e os parâmetros de saída consistem nos resultados de tais processamentos.

Quanto ao corpo do algoritmo, o matemático e programador Edsger Wybe Dijkstra, em sua tentativa de mostrar a validade de algoritmos por meio de demonstrações matemáticas, descobriu que qualquer algoritmo poderia ser escrito por meio da combinação de execuções sequenciais de procedimentos lógicos, estruturas condicionais e estruturas de repetição. Atualmente, estas duas últimas estruturas são conhecidas principalmente como *if/else*, *for* ou *while*. Para mais informações sobre o trabalho de Dijkstra, ver Martin (2019). Esta forma de programar ficou conhecida como programação estruturada.

Observe que, por causa desta composição, um algoritmo pode ser formado por outros algoritmos mais simples que o primeiro aninhados segundo algum critério. Estes podem ser formados por outros ainda mais simples e, seguindo este processo recursivo, deve-se chegar nas estruturas lógicas mais básicas que são representadas pelas sequências binárias que serão executadas pela máquina. Este mecanismo de aninhamento de algoritmos é fundamental e é denominado decomposição funcional (Martin, 2019). O corpo de um algoritmo, mesmo que decomposto em funções, deve sempre estar condicionado às regras de controle da programação estruturada.

A tentativa de Dijkstra em demonstrar matematicamente a validade de um algoritmo acabou caindo em desuso apesar da a ideia ser bastante interessante. No seu lugar, popularizou-se a ideia de testes por meio do método científico. Neste sentido, o foco dos testes de validação de algoritmos deixou de ser a comprovação de que um algoritmo é verdadeiro, ou seja, está correto e passou a ser a comprovação de que o mesmo é falso, ou seja, está incorreto. Neste sentido, "um teste pode provar que um programa está incorreto mas não pode provar que o mesmo está correto" (Martin, 2019).

A primeira e grande vantagem de se adotar o uso da decomposição funcional de algoritmos está justamente na criação de testes de validação, pois um algoritmo pode ser decomposto em algoritmos cada vez mais simples e independentes entre si, permitindo que testes possam ser aplicados em partes específicas de um código, o que facilita encontrar erros de implementação lógica e ajuda na organização do programa.

2.2 ORGANIZAÇÃO DE FUNÇÕES

Na seção anterior, discutimos brevemente a respeito da construção de algoritmos segundo uma programação estruturada e também funcional. Vimos que é possível aninhar funções de forma que o algoritmo possa ser dividido em partes menores ajudando na organização e também na realização de testes. Nesta seção, iremos apresentar algumas práticas descritas por Martin (2011) relacionadas à escrita de funções de forma eficiente e organizada. O objetivo desta seção é introduzir medidas que serão benéficas nas demais partes deste trabalho e não discutir extensivamente o assunto. Para maiores informações sobre este tema, ver Martin (2011).

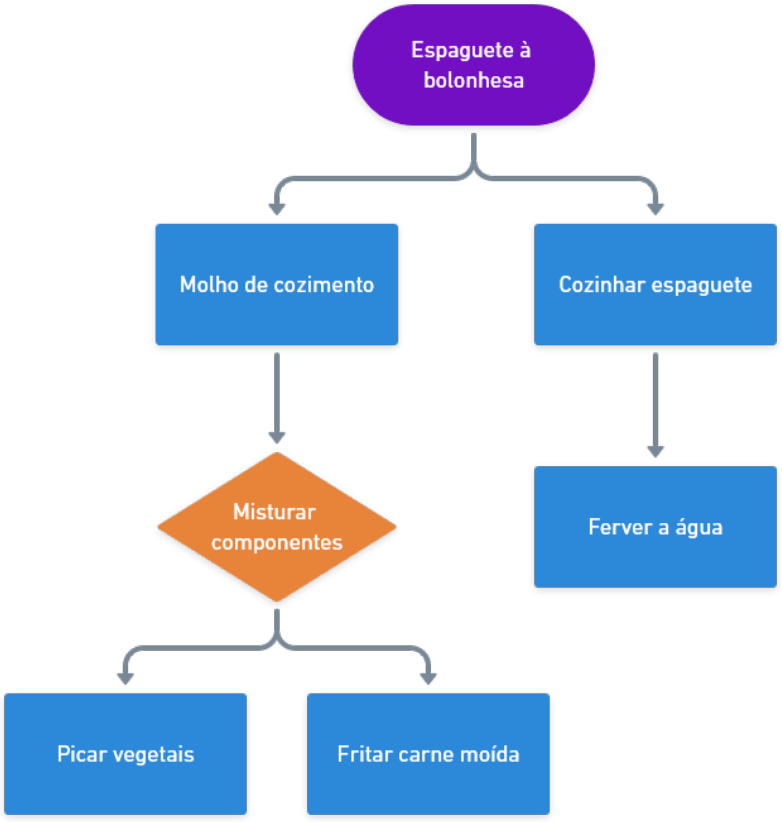
A primeira característica de funções bem escritas é o fato de serem pequenas. Não existe uma regra quanto ao tamanho máximo de linhas que elas devem ocupar, mas cabe ao programador observar se determinada parte do código pode ser particionada de forma a manter a estrutura de um código limpo (Martin, 2011). É justamente a prática de separar o algoritmo em funções cada vez mais simples que naturalmente irá reduzir o tamanho de cada função.

Um segundo ponto atrelado ao primeiro é que funções devem possuir uma única finalidade, ou seja, não basta simplesmente dividir uma função por causa do seu tamanho,

ao particionar uma função em outras menores o foco deve ser justamente o parâmetro de saída. Isto não está relacionado com o número de parâmetros que serão retornados pela função, mas sim com o objetivo pelo qual ela foi criada. Se uma função é responsável por implementar duas funcionalidades, a mesma pode ser particionada em outras duas simples que realizam apenas uma tarefa cada (Martin, 2011).

Não existem evidências de que ambas as práticas acima auxiliem na eficiência de um programa de forma direta, mas existe uma influência quanto a correção de erros por causa da organização, uma vez que funções menores são mais fáceis de serem testadas. Além disso, funções menores estão mais sujeitas a reutilização em outras partes do algoritmo, pois a sua simplicidade permite que ela seja versátil. Existe ainda o ganho quanto à leitura do código que faz com que um outro programador tenha menos dificuldade de compreender a finalidade do programa. E também existe o ganho de manutenção, pois uma vez que as funções são independentes umas das outras, modificar o comportamento de uma não exigirá, ou exigirá pouco, modificar a outra, o que possibilita o programa crescer em termos de funcionalidades com mais facilidade (Martin, 2011).

Figura 2 - Algoritmo para fazer espaguete à bolonhesa segundo estilo funcional



Fonte: Elaborado pelo autor. (2021).

Existem alguns indícios que evidenciam situações em que uma função pode ser particionada. O primeiro deles é a existência de seções. Por exemplo, uma parte da função é responsável por inicializar as variáveis que serão utilizadas, uma segunda parte efetua o cálculo e a terceira organiza os resultados. Claramente esta função não possui uma única finalidade. A própria descrição do exemplo mostra que ela está realizando três tarefas, logo, pode ser particionada. Funções simples não permitem a existência de seções. Outro indício é a presença de estruturas condicionais ou de repetição. Normalmente, tais estruturas devem conter apenas chamadas de outras funções que irão executar o procedimento desejado. A única exceção é no caso de instruções *switch*, que naturalmente são estruturas condicionais extensas. Neste caso, o recomendado é criar uma fábrica de funções e por meio do uso de polimorfismo inserir o *switch* nesta estrutura (Martin, 2011). Iremos tratar de fábrica de funções posteriormente ainda neste capítulo.

A próxima prática está relacionada com a organização sequencial do algoritmo, ou seja, a leitura do código, que deve sempre ser feita de cima para baixo, de forma que o nível de complexidade da função cresça ao percorrer as linhas. Isto quer dizer que as estruturas mais simples da função devem aparecer no início e as mais complexas no final. Uma forma de compreender este princípio é comparar o algoritmo com a narrativa de uma história (Martin, 2011). O início de uma história é sempre mais simples que seu clímax. Além disso, não se corta, por exemplo, o meio de uma cena de guerra de um filme para mostrar uma noite tranquila de sono de um personagem pouco influente na história e que não está relacionado com a problemática. O mesmo vale para a organização das funções. Deve-se manter coerência na ordem de execução dos procedimentos e funções menos abstratas devem ser executadas primeiro.

Um detalhe que frequentemente é ignorado por programadores é o uso de nomes descritivos tanto para funções quanto para objetos. O nome do objeto deve ter relação com aquilo que ele é e o nome da função deve descrever aquilo que ela irá realizar (Martin, 2011). É preferível um nome extenso mas descritivo do que uma única letra que não signifique nada. Isto é importante pois contribui bastante para a organização do código e para a narrativa da história conforme apresentado no parágrafo anterior. Note que esta tarefa é simples de se realizar quando se usa funções simples pois esta possui apenas uma finalidade, mas quando o código não está bem organizado, de fato escolher um nome descritivo se torna uma tarefa desafiadora.

Finalmente, devemos tratar dos parâmetros das funções, tanto de entrada quanto de saída. Em relação ao primeiro, Martin (2011) afirma que a quantidade ideal de parâmetros para uma função é zero. Quanto menos parâmetros melhor, pois facilita a criação de testes de validação da função. Segundo o autor, para se ter mais de três parâmetros em uma função deve-se ter um motivo muito especial, porque provavelmente o número de parâmetros pode ser reduzido. Por exemplo, funções com apenas um parâmetro lógico devem ser evitados pois normalmente isto compromete o princípio da única finalidade da

função. Assim, quando este for o caso, é preferível separar em duas funções distintas. No caso de uma função com mais de três parâmetros, é provável que os mesmos possam ser organizados dentro de um objeto que sintetize tal conjunto de parâmetros. Combinar o nome da função com o nome do parâmetro também é considerado uma boa prática.

Já com relação aos parâmetros de saída é importante que eles representem exatamente o nome da função que os caracterizou. Deve-se, portanto, evitar que uma função retorne escondido algo que não foi especificado pelo nome da mesma pois isto pode gerar confusão ao se buscar compreender de onde foi que certo objeto foi originado.

Tudo isto que foi apresentado nesta seção não deve ser entendido como uma regra a ser seguida a todo custo. Dificilmente o programador terá total dimensão dos problemas da modelagem antes de começar a implementar as funcionalidades do programa. Assim, é recomendado que primeiro o programador gaste um tempo organizando fluxogramas contendo os principais passos a serem seguidos. Depois, deve-se começar a implementar da forma como for possível mesmo que não respeite os princípios apresentados. Após implementar um conjunto de funcionalidades o programador deve voltar e reescrever o código, desta vez respeitando as boas práticas da programação funcional, a fim de continuar a implementação das demais funcionalidades. Esta última etapa é conhecida como refatoração de código.

2.3 LINGUAGEM R

Todos princípios que foram discutidos acima podem ser levados em consideração em qualquer linguagem de programação funcional, mas para que uma linguagem de programação seja considerada funcional é necessário que ela utilize funções puras. Uma função é dita ser pura se: (i) dado um conjunto de parâmetros de entrada, os parâmetros de saída são sempre os mesmos; (ii) a função não possui efeitos colaterais, ou seja, não é capaz de alterar internamente o valor de uma variável global ou salvar arquivos em disco, por exemplo Wickham (2014). Estritamente falando, o R não constitui uma linguagem de programação funcional, pois algumas de suas funções mais básicas violam as condições acima.

Em termos de análise de dados, por exemplo, o R não seria, de fato, muito útil se não permitisse a geração de números pseudo-aleatórios. Entretanto, neste trabalho estaremos interessados em trabalhar com a linguagem R, pois além de fornecer todos os pré-requisitos para a modelagem estatística, a própria criação da linguagem nasceu dos princípios da decomposição funcional. Salvo em exceções bem específicas, esta linguagem originou-se com o propósito de resolver problemas por meio do uso de funções, puras, sempre que possível. Apesar desta característica não ser necessária, é altamente desejável conforme discutido na seção anterior.

No R, funções podem ser manipuladas como se fossem objetos quaisquer da

linguagem, ou seja, podemos atribuir uma função a um objeto, podemos passar funções como parâmetros de entrada de outras funções e também podemos retornar funções como parâmetros de saída de outras. Cada uma dessas formas de manipular funções recebeu um nome. Uma função que recebe um objeto como parâmetro de entrada e retorna um objeto como parâmetro de saída é conhecida como função regular. Este é o caso mais conhecido e adotado pelos programadores em geral. Uma função que recebe uma outra função como parâmetro de entrada e retorna um objeto é chamada de funcional. Uma função que recebe um objeto de entrada e retorna uma função é chamada fábrica de funções. Finalmente, uma função que recebe uma função de entrada e retorna uma função como saída é chamada de operador funcional.

Os funcionais, apesar do nome ser incomum, são frequentemente utilizados por programadores em R de nível intermediário. Um dos maiores exemplos de funcionais é a família *apply*, que é um conjunto de funções construídas com o objetivo de substituir o uso da estrutura de repetição *for* por uma estrutura que se adapte melhor ao estilo funcional de programar. Conforme apresenta Wickham (2014), muitos acreditam que o *for* não é recomendado pois possui má performance, mas na realidade o grande lado negativo tem a ver com a programação funcional. Ao utilizar o *for* dificilmente os princípios da seção anterior serão respeitados o que resultará em "poluição" do código. Mas os funcionais não servem simplesmente para substituir estruturas de iteração. Outros exemplos famosos são as funções *integrate* que realiza o cálculo de integrais e *optim* que realiza mecanismos de otimização, que inclusive são utilizados neste trabalho mais adiante.

Em essência, operadores funcionais e fábricas de funções estão relacionadas, sendo o primeiro um caso particular do segundo. Como mencionamos na seção anterior, as fábricas de funções podem ser entendidas como formas de substituir o uso do *switch* por meio do uso de polimorfismos. De fato, as fábricas de funções nada mais são do que funções que criam funções. Assim, em uma situação em que uma mesma estrutura condicional deve ser implementada - porém considerando funções diferentes - uma fábrica de funções pode ser adotada como forma de evitar a repetição desta mesma estrutura em diversas partes do código. Esta ideia é utilizada neste trabalho no capítulo 4, quando estivermos nos referindo a diferentes funções de ajuste dos parâmetros. Iremos considerar tanto operador funcional quanto fábrica de funções como sendo simplesmente fábrica de funções.

2.3.1 Metaprogramação

Além das funcionalidades apresentadas, existe ainda uma que é extremamente poderosa e está implementada no R, não existindo no mesmo nível em outras linguagens de programação. Tal funcionalidade é chamada metaprogramação. Este é um dos assuntos mais complexos existentes na linguagem e compreendê-lo perfeitamente demanda muito esforço e tempo, mas dá ao programador um controle surpreendente sobre o desenvolvimento

do seu programa.

Conforme apresenta Wickham (2014), em R, o código também é um tipo de estrutura de dado. Desta forma, assim como uma função pode ser passada como parâmetros de funções, ou como retorno de outra, ou ainda atribuída a um objeto qualquer, uma linha de código também pode realizar as mesmas operações. Sua estrutura de dado é denominada expressão. Tudo isto é possível pois existe uma diferença na linguagem entre capturar uma expressão e avaliá-la.

Considere, por exemplo, um programa com a seguinte linha de código:

$$x = 1 + 2 \ .$$

Naturalmente, o programador ao executar esta linha esperaria que x fosse igual a 3 e de fato é, pois, de maneira automática o R captura a expressão acima e avalia o seu resultado. Mas não necessariamente precisaria ser, pois o programador poderia forçar o R a, antes de avaliar a expressão, armazená-la em um objeto qualquer. Assim, torna-se possível manipular linhas de código sem de fato executá-las.

A metaprogramação no R é um exemplo claro a respeito da forma como a linguagem foi arquitetada para operar, que é por meio do uso de decomposição funcional. Isto quer dizer que, internamente, toda expressão é decomposta através de uma árvore de sintaxe abstrata, em outras estruturas de dados mais simples, que são as (i) chamadas, (ii) símbolos, (iii) constantes e (iv) *pairlists*. Não tratamos aqui em detalhes sobre estes assuntos para não perdermos o foco do trabalho. Para informações sobre a implementação de metaprogramação, ver Wickham (2014).

O fato é que usar expressões no R facilita a implementação de diversas funcionalidades em um algoritmo mais complexo. As mais conhecidas são as fórmulas, que são comumente utilizadas por funções que realizam ajuste de modelos de regressão. O uso de fórmulas permite que o usuário do programa especifique de forma simples o que se deseja estudar e internamente a função irá traduzir a fórmula fornecida em uma estrutura cuja sintaxe o R consegue avaliar.

Neste trabalho, o uso de expressões tem um papel fundamental no desenvolvimento do algoritmo. Precisaremos ajustar o modelo de regressão longitudinal e fazer isso sem utilizar fórmulas seria um trabalho desnecessário. Além disso, será necessário derivar matrizes cujos valores não estão determinados, impossibilitando o uso de derivadas numéricas. Assim, o uso de expressões possibilitarão derivar matrizes de forma analítica sem necessariamente avaliar o resultado, o que engrandece ainda mais o valor do pacote desenvolvido.

3 MODELOS DE REGRESSÃO PARA DADOS LONGITUDINAIS COMPLEXOS

No presente capítulo será apresentada, de forma geral, a metodologia desenvolvida por Vieira (2005), que é a base para o desenvolvimento do pacote estatístico resultante deste trabalho, denominado 'clm'.

Neste sentido, será abordada, inicialmente, a estimação dos parâmetros da regressão, $\boldsymbol{\beta}$, e seus respectivos erros-padrão. Em seguida, será apresentado o procedimento adotado para estimação da matriz de covariância $\boldsymbol{\Sigma}$, definida a seguir, e também seus erros-padrão. Vale destacar que todos os procedimentos serão vistos sob a ótica da incorporação do plano amostral $p(\boldsymbol{s})$ da pesquisa na estimação dos parâmetros e também na perspectiva computacional.

3.1 CONSIDERAÇÕES INICIAIS

Seja $\boldsymbol{\upsilon} = \{1, 2, \ldots, N\}$ uma população finita de tamanho N, fixa nas ocasiões $1, \ldots, T$.

Seja $\boldsymbol{Y}_i = (Y_{i1}, \ldots, Y_{iT})^{'}$ o vetor aleatório contendo T observações repetidas equidistantes no tempo sobre a variável de interesse para as unidades $i = 1, 2, \ldots, N$ sobre as T rodadas da pesquisa. Considere $\boldsymbol{Y}_i$ como sendo contínuo tal que

$$E(\boldsymbol{Y}_i) = \boldsymbol{\mu}_i(\boldsymbol{\beta}) \tag{3.1}$$

seja o vetor $T \times 1$ com seus respectivos valores esperados, em que

$$\boldsymbol{\mu}_i(\boldsymbol{\beta}) = [\mu(\boldsymbol{x}_{i1}, \boldsymbol{\beta}), \ldots, \mu(\boldsymbol{x}_{iT}\boldsymbol{\beta})]^{'}. \tag{3.2}$$

Na equação anterior, $\boldsymbol{x}_{it}$ é um vetor $1 \times Q$ de Q covariáveis fixas, $\boldsymbol{\beta}$ é um vetor $Q \times 1$ de parâmetros desconhecidos e $\mu(.,.)$ é conhecida. Considere $\boldsymbol{y}_i$ o valor observado de $\boldsymbol{Y}_i$.

Finalmente, seja

$$\boldsymbol{\Sigma} = COV(\boldsymbol{Y}_i) = E\{[\boldsymbol{Y}_i - \boldsymbol{\mu}_i][\boldsymbol{Y}_i - \boldsymbol{\mu}_i]^{'}\} \tag{3.3}$$

a matriz $T \times T$ de covariância da população sobre $\boldsymbol{Y}$. Note que $\boldsymbol{\Sigma}$ definido acima não depende de i.

O modelo de efeitos fixos e aleatórios, também denominado modelo misto conforme Singer, Nobre e Rocha (2018), pode ser definido de forma geral como

$$\boldsymbol{Y}_i = \boldsymbol{X}_i\boldsymbol{\beta} + \boldsymbol{\epsilon}_i, \tag{3.4}$$

em que $\boldsymbol{Y}_i$ é um vetor $T \times 1$ da variável de interesse do indivíduo i nas rodadas $1, \ldots, T$, $\boldsymbol{X}_i$ é uma matriz $T \times Q$ de covariáveis, $\boldsymbol{\beta}$ é um vetor $Q \times 1$ de parâmetros desconhecidos,

que é o vetor de efeitos fixos, e $\boldsymbol{\epsilon}_i$ é um vetor $T \times 1$ de erros aleatórios, que é o vetor de efeitos aleatórios.

De acordo com Vieira (2005), o vetor de efeitos aleatórios pode ser decomposto em duas partes, a saber: os efeitos aleatórios permanentes e os efeitos aleatórios transitórios. Assim, $\boldsymbol{\epsilon}_i$ pode ser reescrito como

$$\epsilon_{it} = u_i + v_{it}, \tag{3.5}$$

em que u_i representa os efeitos individuais permanentes, logo são os mesmos para todo $t = 1, \ldots, T$, e v_{it} os efeitos transitórios. Dado o modelo de regressão anterior, assume-se $E(\boldsymbol{\epsilon}_i) = \mathbf{0}$ e $VAR(\boldsymbol{\epsilon}_i) = \boldsymbol{\Sigma}$.

3.2 AGRUPAMENTO DE DADOS

O modelo considerado é multidimensional, ou seja, existe a dimensão do indivíduo que varia de $i = 1, \ldots, N$, a dimensão do tempo que varia de $t = 1, \ldots, T$ e também a das covariáveis que varia de $q = 1, \ldots, Q$. A abordagem adotada para a apresentação do modelo poderia ser geral, no sentido de considerar matrizes tridimensionais, ou então fixando uma das dimensões e apresentando as demais. Esta última estratégia foi adotada. Observando a forma como a metodologia foi originalmente apresentada é possível perceber que a dimensão considerada como fixa foi a do indivíduo, e em alguns momentos, também a do tempo. Por isso a maior parte das expressões anteriores possuem o subíndice i ou it. Isto se deu pela facilidade de interpretar e visualizar os resultados. Entretanto, é importante destacar que qualquer uma das dimensões poderia ser fixada, mas dependendo da situação, a estratégia poderia trazer complicações para a representação do modelo por causa de dependência entre as observações da dimensão fixada.

A questão a respeito da forma de representar as partes do modelo é importante, pois pode ter implicações diretas no âmbito computacional. Uma das primeiras, e fundamentais, traduções de abstrações matemáticas desenvolvidas nas linguagens de programação foram os vetores unidimensionais, que agrupavam números escalares em uma sequência específica. A partir dela, surgiu o conceito de matriz bidimensional como sendo um conjunto de vetores unidimensionais agrupados sob algum critério. E sucessivamente, foram surgindo as matrizes multidimensionais como conjuntos de vetores unidimensionais agrupados por vários critérios, também denominados dimensões. Até o nível de matriz bidimensional as operações matriciais foram bem definidas mas o mesmo já não acontece no caso multidimensional, cabendo ao programador discernir como operar tais estruturas em termos de vetores ou matrizes bidimensionais.

Existe ainda uma última estrutura para agrupar dados bastante interessante que são as listas[1]. Essa estrutura de dados utiliza a ideia de fixar as dimensões para agrupar não

[1] O nome pode variar entre as linguagens de programação mas é certo que todas as linguagens

somente dados de um mesmo tipo como ocorrem com as matrizes e vetores, mas também qualquer combinação de tipos de dados, inclusive outras listas. Essa estrutura genérica possui um ganho que é a capacidade de trabalhar com entes matemáticos multidimensionais de tipos variados de forma relativamente simples mas, por ser mais complexo que as matrizes e vetores, possui um custo computacional mais elevado conforme será visto posteriormente neste trabalho. A tabela abaixo mostra um exemplo de agrupamento por meio do uso de listas para o caso de dois indivíduos observados em dois momentos no tempo.

Tabela 1 – Exemplo de abordagem para agrupamento de dados com listas

Indivíduo			
1	Estrato	1	
	Conglomerado	1	
	Peso Amostral	1,20	
	Var. Resposta	0,5	
		0,8	
	Covariáveis	10	1
		12	1
2	Estrato	1	
	Conglomerado	1	
	Peso Amostral	1,35	
	Var. Resposta	0,7	
		0,5	
	Covariáveis	7	2
		10	2

Fonte: Elaborado pelo autor (2021).

Logo, da mesma forma que fixar, em termos matemáticos, a dimensão do indivíduo facilita a compreensão da metodologia, isto também auxilia na perspectiva da programação, pois dá as diretrizes a serem seguidas na criação das listas ou matrizes multidimensionais.

Entretanto, pode ser que nem sempre a forma de fixar as dimensões matematicamente seja a melhor maneira em termos computacionais. Uma vez que operações matriciais ocorrem muito mais rápido do que iterações, é razoável pensar que fixar a dimensão do tempo ao invés do indivíduo seja melhor em termos de custo computacional, mas isto obrigaria o programador a repensar a teoria matemática nessa nova perspectiva e isso possui um custo intelectual.

Uma outra possível abordagem seria buscar transformar este ambiente multidimensional em um bidimensional por meio do uso de variáveis indicadoras em uma tabela de dados. A tabela abaixo mostra um exemplo desta forma de agrupamento.

de alto nível modernas possuem essa estrutura implementada.

Tabela 2 – Exemplo de abordagem para agrupamento de dados com tabela de dados

Indivíduo	Tempo	Estrato	Conglomerado	Peso Amostral	Resposta	Cov. 1
1	Antes	1	1	1,20	0,5	10
1	Depois	1	1	1,20	0,8	12
2	Antes	1	1	1,35	0,7	5
2	Depois	1	1	1,35	0,5	20
3	Antes	1	2	1,01	1	7
3	Depois	1	2	1,01	0,3	10

Fonte: Elaborado pelo autor (2021).

A vantagem desta abordagem, que é chamada de forma longa na literatura dos dados longitudinais, é a facilidade de visualização dos dados brutos, que é mais intuitiva que a anterior. Além disso, a importação e exportação de dados neste formato é bastante simples, possibilitando a fácil integração do pacote com softwares externos, como por exemplo, o excel ou o google planilhas. Por outro lado, as informações contidas nas tabelas de dados são redundantes, de forma que uma mesma informação se repete diversas vezes e isso em situações de grandes bases de dados pode vir a ser um problema.

Um detalhe interessante é que as tabelas de dados, em essência, também são listas mas com características específicas. Portanto a conversão de uma abordagem para a outra é relativamente fácil, tornando possível utilizar ambos os métodos na resolução do mesmo problema, buscando apreciar as vantagens de cada um.

3.3 ESTIMAÇÃO DOS EFEITOS FIXOS $\boldsymbol{\beta}$

Suponha que o vetor aleatório $\boldsymbol{Y}_i = (Y_{i1}, \ldots, Y_{iT})^{'}$ tenha distribuição Normal T-variada com média $\boldsymbol{\mu}_i(\boldsymbol{\beta})$ e matriz de covariância $\boldsymbol{\Sigma}$ conforme descrito anteriormente, isto é, $\boldsymbol{Y}_i \sim N_T[\boldsymbol{\mu}_i(\boldsymbol{\beta}), \boldsymbol{\Sigma}]$.

Seja $\boldsymbol{y}_i$ um vetor com os valores observados da variável resposta para cada indivíduo i em cada rodada t. Inicialmente, considere que toda a população finita é observada e que $\boldsymbol{Y}_i, \ldots, \boldsymbol{Y}_N$ são mutuamente independentes. Assim, a função de densidade conjunta de todas as observações, também chamado função de verossimilhança do censo, pode ser definida como o produto das densidades normais marginais, ou seja,

$$
\begin{aligned}
f(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N | \boldsymbol{\beta}, \boldsymbol{\Sigma}) &= \prod_{i=1}^{N} \left\{ \frac{1}{(2\pi)^{\frac{T}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{[\boldsymbol{y}_i - \boldsymbol{\mu}_i(\boldsymbol{\beta})]^{'} \boldsymbol{\Sigma}^{-1} [\boldsymbol{y}_i - \boldsymbol{\mu}_i(\boldsymbol{\beta})]}{2}} \right\} \\
&= (2\pi)^{-\frac{N.T}{2}} |\boldsymbol{\Sigma}|^{-\frac{N}{2}} e^{-\sum_{i=1}^{N} \frac{[\boldsymbol{y}_i - \boldsymbol{\mu}_i(\boldsymbol{\beta})]^{'} \boldsymbol{\Sigma}^{-1} [\boldsymbol{y}_i - \boldsymbol{\mu}_i(\boldsymbol{\beta})]}{2}} . \qquad (3.6)
\end{aligned}
$$

A princípio, considere a matriz $\boldsymbol{\Sigma}$ ser conhecida. O vetor de efeitos fixos $\boldsymbol{\beta}$ pode

então ser estimado maximizando o logaritmo da verossimilhança do censo com relação a $\boldsymbol{\beta}$, que é dado por

$$L_N[\boldsymbol{\beta}] = log\left[\frac{1}{(2\pi)^{\frac{N.T}{2}}|\boldsymbol{\Sigma}|^{\frac{N}{2}}}e^{-\sum_{i=1}^{N}\frac{[\boldsymbol{y}_i-\boldsymbol{\mu}_i(\boldsymbol{\beta})]'\boldsymbol{\Sigma}^{-1}[\boldsymbol{y}_i-\boldsymbol{\mu}_i(\boldsymbol{\beta})]}{2}}\right]$$
$$= -\frac{1}{2}\left[N.T\ log\ 2\pi + N\ log\ |\boldsymbol{\Sigma}| + \sum_{i=1}^{N}[\boldsymbol{y}_i-\boldsymbol{\mu}_i(\boldsymbol{\beta})]'\boldsymbol{\Sigma}^{-1}[\boldsymbol{y}_i-\boldsymbol{\mu}_i(\boldsymbol{\beta})]\right]. \quad (3.7)$$

Considere, agora, que apenas uma amostra $\boldsymbol{s}$ de tamanho n foi selecionada segundo um plano amostral $p(\boldsymbol{s})$ e que dessa amostra foram observadas as respostas $\boldsymbol{y}_i,\ i \in \boldsymbol{s}$.

Assim, derivando (3.7) em relação a $\boldsymbol{\beta}$

$$\frac{\partial L[\boldsymbol{\beta}]}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} w_i\,[\boldsymbol{y}_i-\boldsymbol{\mu}_i(\boldsymbol{\beta})]'\boldsymbol{\Sigma}^{-1}\frac{\partial \boldsymbol{\mu}_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}. \quad (3.8)$$

Considere $\boldsymbol{\mu}_i(\boldsymbol{\beta}) = \boldsymbol{X}_i\boldsymbol{\beta}$. Logo, $\frac{\partial \boldsymbol{\mu}_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \boldsymbol{X}_i$ e portanto

$$\frac{\partial L[\boldsymbol{\beta}]}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} w_i\,[\boldsymbol{y}_i-\boldsymbol{X}_i\boldsymbol{\beta}]'\boldsymbol{\Sigma}^{-1}\boldsymbol{X}_i\ . \quad (3.9)$$

Finalmente, igualando a expressão acima a $\mathbf{0}$, encontra-se o estimador de máxima pseudo-verossimilhança para $\boldsymbol{\beta}$, que é dado por

$$\hat{\boldsymbol{\beta}}(\boldsymbol{\Sigma}) = \left(\sum_{i=1}^{n} w_i \boldsymbol{X}'_i\boldsymbol{\Sigma}^{-1}\boldsymbol{X}_i\right)^{-1}\sum_{i=1}^{n} w_i\boldsymbol{X}'_i\boldsymbol{\Sigma}^{-1}\boldsymbol{y}_i. \quad (3.10)$$

Entretanto, na prática, a matriz de covariância $\boldsymbol{\Sigma}$ é desconhecida. Isto é um problema porque encontrar um estimador de máxima pseudo-verossimilhança para $\boldsymbol{\Sigma}$ pode ser uma tarefa desafiadora. Portanto, são necessárias algumas adaptações na equação anterior. Uma possível solução é substituir a matriz de covariância $\boldsymbol{\Sigma}$ pela matriz de covariância de trabalho $\boldsymbol{V}$, que é uma matriz estruturada pela covariância dos resíduos do modelo ajustado. Segundo Vieira (2005), o estimador $\hat{\boldsymbol{\beta}}(\boldsymbol{V})$ é um estimador aproximadamente não viesado para $\boldsymbol{\beta}$ independente da escolha de $\boldsymbol{V}$ desde que w_i leve em conta o plano amostral e a não resposta.

Seja $\boldsymbol{R}$ uma matriz de correlação de trabalho correspondente a $\boldsymbol{V}$, de modo que

$$\boldsymbol{R} = \boldsymbol{D}_V^{-1}\boldsymbol{V}\boldsymbol{D}_V^{-1}, \quad (3.11)$$
$$\boldsymbol{V} = \boldsymbol{D}_V\boldsymbol{R}\boldsymbol{D}_V \quad (3.12)$$

em que $\boldsymbol{D}_V = [diag(\boldsymbol{V})]^{\frac{1}{2}}$.

Existem diversas formas de escolher a matriz $\boldsymbol{R}$. Na prática, considera-se inicialmente $\boldsymbol{V} = \boldsymbol{I}$, em que $\boldsymbol{I}$ é a matriz identidade de mesmas dimensões de $\boldsymbol{V}$, estima-se $\boldsymbol{\beta}$ por

meio do estimador $\hat{\boldsymbol{\beta}}(\boldsymbol{V})$, estima-se os resíduos $\boldsymbol{\epsilon}_i$ por meio do estimador $\hat{\boldsymbol{\epsilon}}_i$ e finalmente estima-se a correlação entre os indivíduos por meio do estimador $\hat{\rho}$, que segundo Vieira (2005), é dado por

$$\hat{\rho} = \frac{\dfrac{\sum_{i=1}^{n}\left(\sum_{t=1}^{T}\hat{\epsilon}_{it}^{*}\right)^2}{n.T} - \dfrac{\sum_{i=1}^{n}\sum_{t=1}^{T}(\hat{\epsilon}_{it}^{*})^2}{n.T}}{n.(T-1)}, \tag{3.13}$$

em que $\hat{\epsilon}_{it}^{*} = \dfrac{\hat{\epsilon}_{it} - \overline{\epsilon}_i}{\hat{\sigma}_i}$, $\hat{\epsilon}_{it} = y_{it} - \boldsymbol{X}_{it}\hat{\boldsymbol{\beta}}(\boldsymbol{V})$, $\hat{\sigma}_i^2 = \dfrac{\sum_{t=1}^{T}\hat{\epsilon}_{it}^2}{T} - (\overline{\epsilon}_i)^2$ e $\overline{\epsilon}_i = \dfrac{\sum_{t=1}^{T}\hat{\epsilon}_{it}}{T}$.

Segundo Vieira (2012), a partir do cálculo de $\hat{\rho}$ pode-se definir $\boldsymbol{R}$ como

1. Permutável, ou seja, $R_{tt'} = 1$ quando $t = t^{'}$ e $\hat{\rho}$ caso contrário;
2. Estacionária, $R_{tt'} = 1$ quando $t = t^{'}$, $\hat{\rho}$ se $|t - t^{'}| \leq g$ e 0 caso contrário, em que g é a maior distância temporal considerada para o cálculo da covariância;
3. Autorregressiva, $R_{tt'} = 1$ quando $t = t^{'}$ e $\hat{\rho}^{|t-t^{'}|}$ caso contrário;
4. Não estacionária, $R_{tt'} = 1$ quando $t = t^{'}$, $\hat{\rho}_{tt'}$ se $|t - t^{'}| \leq g$ em que $\hat{\rho}_{tt'} = \hat{\rho}_{t't}$ é a correlação entre os resíduos nos instantes t e $t^{'}$;
5. Não estruturada, $R_{tt'} = 1$ quando $t = t^{'}$, $\hat{\rho}_{tt'}$ caso contrário.

Além do método que considera a matriz de covariância de trabalho $\boldsymbol{V}$ uma possível abordagem é considerar um estimador natural de $\boldsymbol{\Sigma}$ que será visto mais adiante.

3.3.1 Concepção inicial do pacote

Tendo em vista o método de estimação dos efeitos fixos, torna-se possível arquitetar as primeiras funcionalidades do pacote a ser criado no R. Por meio da ordem dos somatórios das equações anteriores fica evidente que a estrutura dos dados será armazenada em uma lista fixada na dimensão dos indivíduos de forma que a dimensão do tempo e das covariáveis fiquem livres para serem trabalhadas no nível de operações matriciais conforme mostra a tabela 1.

Entretanto, vale a pena aproveitar a estrutura das tabelas de dados como forma de entrada na função principal do pacote. Assim, o usuário, que provavelmente não tem experiência com o uso de listas, pode importar uma planilha de algum outro software ou arquivo qualquer. Essa capacidade de trabalhar com dados oriundos de lugares diversos traz uma gama de possibilidades de implementação para o futuro, inclusive fazer a integração com bancos de dados, que suportam volumes gigantescos de informação. Internamente, o pacote irá transformar essa tabela em uma lista, a fim de eliminar as eventuais informações redundantes.

Além disso, é necessário que o usuário forneça qual a relação entre as variáveis que será explorada. O R possui implementado nativamente uma estrutura chamada fórmula, que permite que o usuário explicite não somente as variáveis em questão mas também a relação entre elas, evitando que uma combinação das variáveis da tabela de dados seja criada na tabela original. Um exemplo de fórmula no R é

$$y \sim x_1 + x_2 + x_1 \times x_2 + x_1^2$$

que determina que a variável de interesse é denominada y na tabela de dados e as variáveis explicativas são x_1 e x_2. Além disso existem outras duas covariáveis não existentes na tabela original, que são $x_1 \times x_2$ e x_1^2, que devem ser incorporadas ao modelo. A adoção de fórmulas é considerada uma boa prática no R devido a sua simplicidade e organização do código, além de ser intuitivo para o usuário implementar.

A dimensão das covariáveis já está bem especificada, mas resta ainda informar quais são as dimensões dos indivíduos e do tempo. Como o fornecimento dessas dimensões é obrigatório, pode-se acrescentar simplesmente dois parâmetros de entrada à função principal que receberão os nomes das variáveis indicadoras que as representam na tabela de dados. Para que a função não fique sobrecarregada de parâmetros, pode-se agrupar tempo e indivíduo dentro de um único objeto.

Com relação a características adicionais do plano amostral, até o presente momento não foi necessário a incorporação de variáveis de estratificação e conglomeração. Apenas o peso amostral é utilizado na estimação de $\boldsymbol{\beta}$. Logo, a princípio, apenas adicionar um parâmetro de entrada para receber o nome da variável que contém w_i na tabela de dados solucionará o problema. Caso o usuário não forneça uma variável para este parâmetro, o pacote automaticamente deve colocar $w_i = 1 \; \forall i = 1, \ldots, n$.

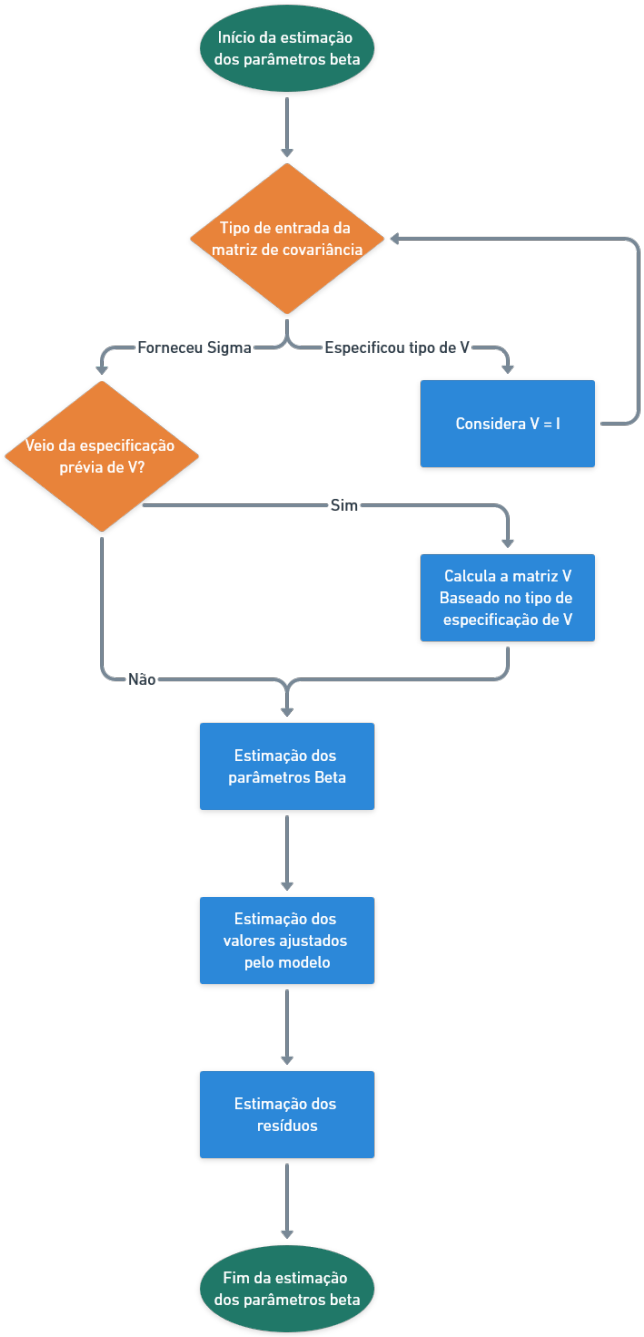
Finalmente, a parte mais importante desta etapa é a incorporação da matriz de covariância $\boldsymbol{\Sigma}$ ao pacote. Até então, ou $\boldsymbol{\Sigma}$ foi considerado conhecido e utilizado na estimação dos efeitos fixos, ou então foi utilizado a matriz de covariância de trabalho $\boldsymbol{V}$ em seu lugar. Sendo assim, pode-se acrescentar um parâmetro de entrada, de forma que o usuário possa fornecer uma matriz de covariância qualquer ou o nome de alguma das formas apresentadas por Vieira (2005) para o cálculo de $\boldsymbol{R}$. No primeiro caso, a função irá implementar diretamente a estimação do $\boldsymbol{\beta}$, e no segundo, irá efetuar o cálculo de $\boldsymbol{V}$ e depois estimar $\boldsymbol{\beta}$.

Essa dualidade de possibilidades com relação ao parâmetro de entrada sigma faz com que a execução desta mesma função seja recursiva e permita o cálculo de $\boldsymbol{V}$ simplesmente chamando-a internamente com $\boldsymbol{V} = \boldsymbol{I}$. São toques sutis como esse que um programador deve desenvolver de forma a organizar e limpar seu código. Isto não foi algo especificado pela metodologia mas simplifica bastante o desenvolvimento da mesma. Inclusive, é possível criar um mecanismo de iteração de forma que o processo vá se repetindo até que

$\boldsymbol{V}$ atinja convergência. Isso o fará generalizar a própria ferramenta desenvolvida e dará ainda mais consistência para o pacote.

O fluxograma abaixo mostra o processo para estimação dos parâmetros da regressão, em especial, o processo recursivo descrito no parágrafo anterior.

Figura 4 - Fluxograma da Estimação dos Efeitos Fixos



Fonte: Elaborado pelo autor. (2021).

3.4 ESTIMAÇÃO DA VARIÂNCIA DE $\hat{\boldsymbol{\beta}}$

Vieira (2005) mostra que $p(\boldsymbol{s})$ possui forte influência na estimação da variância de $\boldsymbol{\beta}$, de forma que a consideração incorreta do desenho amostral pode resultar em resultados enviesados das estimativas.

Sendo assim, uma possível abordagem para o cálculo da variância dos efeitos fixos é considerar o método de linearização de Taylor. Este método consiste em aplicar sobre $\boldsymbol{\beta}$ expansões em séries de Taylor, resultando em aproximações de primeira ordem do estimador da variância. Para isso, dois pressupostos necessitam ser respeitados, a saber: (i) $\hat{\boldsymbol{\beta}}(\boldsymbol{V})$ é um estimador consistente para $\boldsymbol{\beta}$; (ii) a expansão de Taylor do estimador causa contribuições desprezíveis para sua variância.

Considere inicialmente a seguinte extensão da expressão do estimador $\hat{\boldsymbol{\beta}}(\boldsymbol{V})$:

$$\left[\hat{\boldsymbol{\beta}}(\boldsymbol{V}) - \boldsymbol{\beta}\right] = \left(\sum_{i=1}^{n} w_i \boldsymbol{X}_i' \boldsymbol{V}^{-1} \boldsymbol{X}_i\right)^{-1} \sum_{i=1}^{n} w_i \boldsymbol{X}_i' \boldsymbol{V}^{-1} \boldsymbol{\epsilon}_i \tag{3.14}$$

em que $\boldsymbol{\epsilon}_i$ é o termo de erro dado por $\boldsymbol{\epsilon}_i = \boldsymbol{Y}_i - \boldsymbol{X}_i\boldsymbol{\beta}$.

De acordo com Vieira (2005), o estimador da variância de $\hat{\boldsymbol{\beta}}(\boldsymbol{V})$ é dado por:

$$var_L\left[\hat{\boldsymbol{\beta}}(\boldsymbol{V}) - \boldsymbol{\beta}\right] = \left(\sum_{i=1}^{n} w_i \boldsymbol{X}_i' \boldsymbol{V}^{-1} \boldsymbol{X}_i\right)^{-1} . \, var_L\left(\sum_{i=1}^{n} w_i \boldsymbol{X}_i' \boldsymbol{V}^{-1} \boldsymbol{\epsilon}_i\right) . \left(\sum_{i=1}^{n} w_i \boldsymbol{X}_i' \boldsymbol{V}^{-1} \boldsymbol{X}_i\right)^{-1} \tag{3.15}$$

em que L indica o termo linearizado.

Os termos das extremidades da expressão anterior são conhecidos, porém o termo médio que envolve o cálculo da variância ainda precisa ser explorado. Como o parâmetro populacional $\boldsymbol{\beta}$ é desconhecido, é inviável o cálculo do termo de erro. Assim, uma possível solução é considerar o estimador do resíduo, dado por $\hat{\boldsymbol{\epsilon}}_i = \boldsymbol{Y}_i - \boldsymbol{X}_i\hat{\boldsymbol{\beta}}(\boldsymbol{V})$. O termo médio anterior passa a ser então

$$var_L\left(\sum_{i=1}^{n} w_i \boldsymbol{X}_i' \boldsymbol{V}^{-1} \hat{\boldsymbol{\epsilon}}_i\right). \tag{3.16}$$

Seja $\boldsymbol{Z}_i = \boldsymbol{X}_i' \boldsymbol{V}^{-1} \hat{\boldsymbol{\epsilon}}_i$. A expressão anterior pode ser reescrita como

$$var_L\left(\sum_{i=1}^{n} w_i \boldsymbol{Z}_i\right). \tag{3.17}$$

Note que $\hat{\boldsymbol{Z}} = \sum_{i=1}^{n} w_i \boldsymbol{Z}_i$ é um estimador do total populacional de $\boldsymbol{Z}$ conforme atesta a expressão (1.5).

Segue abaixo um exemplo do cálculo de $\hat{\boldsymbol{Z}}$ considerando um plano amostral estratificado em múltiplos estágios com amostragem com reposição de conglomerados no primeiro estágio e amostragem com ou sem reposição nos estágios seguintes. Considere

ainda probabilidades de seleção iguais ou desiguais tanto no primeiro estágio quanto nos demais. Seguindo este desenho de amostragem, a expressão de $\hat{\boldsymbol{Z}}$ pode ser reescrita como

$$\hat{\boldsymbol{Z}} = \sum_{h=1}^{H} \hat{\boldsymbol{Z}}_h = \sum_{h=1}^{H} \sum_{j=1}^{m_h} \hat{\boldsymbol{Z}}_{hj} = \sum_{h=1}^{H} \sum_{j=1}^{m_h} \sum_{i=1}^{n_{hj}} w_{hji} \boldsymbol{Z}_{hji} \ , \tag{3.18}$$

em que H é o número de estratos na amostra, m_h é o número de conglomerados dentro do estrato h, n_{hj} é o número de indivíduos pertencentes ao conglomerado j do estrato h e w_{hji} é o peso amostral do indivíduo i, do conglomerado j e do estrato h. Note que $\hat{\boldsymbol{Z}}$ é um estimador do tipo Horvitz-Thompson. Logo, Vieira (2005) apresenta que

$$var_L[\hat{\boldsymbol{Z}}] = \sum_{h=1}^{H} \frac{m_h}{m_h - 1} \left[\sum_{j=1}^{m_h} (\hat{\boldsymbol{Z}}_{hj} - \overline{\hat{\boldsymbol{Z}}}_h)^2 \right] \ , \tag{3.19}$$

em que $\overline{\hat{\boldsymbol{Z}}}_h$ é a média de $\hat{\boldsymbol{Z}}_{hj}$ no estrato h.

Finalmente, o estimador da variância de $\hat{\boldsymbol{\beta}}(\boldsymbol{V})$ neste caso é dado por:

$$var_L \left[\hat{\boldsymbol{\beta}}(\boldsymbol{V}) - \boldsymbol{\beta} \right] = \left(\sum_{i=1}^{n} w_i \boldsymbol{X}_i' \boldsymbol{V}^{-1} \boldsymbol{X}_i \right)^{-1} . \ var_L[\hat{\boldsymbol{Z}}] . \left(\sum_{i=1}^{n} w_i \boldsymbol{X}_i' \boldsymbol{V}^{-1} \boldsymbol{X}_i \right)^{-1} . \tag{3.20}$$

Vale destacar que o exemplo acima possui como caso particular a situação em que as unidades amostrais foram selecionadas de forma independente.

3.4.1 Incorporação do plano amostral ao pacote

Mediante toda a metodologia apresentada até o momento, este é o ponto mais complexo em termos computacionais, pois não existe uma fórmula geral que possa ser aplicada a qualquer plano amostral $p(\boldsymbol{s})$. Até mesmo o próprio exemplo anterior, apesar de abranger uma gama de possibilidades de especificação do plano amostral que envolvam estratificação, conglomeração e pesos amostrais, é apenas um exemplo e não pode ser utilizado em qualquer situação. Além disso, a própria especificação do plano amostral é uma tarefa complicada justamente pela infinidade de possibilidades de definição do mesmo.

Uma possível solução para o segundo problema, que é a especificação do plano amostral no R, é utilizar outros pacotes do R que já fazem este tipo de controle, como por exemplo o pacote *survey* desenvolvido por Thomas Lumley, que é uma grande referência para se trabalhar com dados amostrais complexos. O pacote permite que o usuário especifique corretamente o plano amostral da pesquisa e o traduza para uma estrutura mais metódica e geral que seja mais simples de ser utilizado por outras funções. A dificuldade deste método é a necessidade de compreender esta estrutura metódica que o pacote *survey* retorna ao usuário para que este objeto possa ser reaproveitado. É importante também estar sempre acompanhando as atualizações do pacote, pois uma mudança na estrutura do objeto contendo a especificação do plano amostral pode vir a comprometer todo o restante.

Quanto ao primeiro problema, o próprio exemplo apresentado na seção anterior indica qual caminho deve ser seguido para solucioná-lo. Conforme mostra a expressão (3.18), o cálculo de $\hat{\boldsymbol{Z}}$ é feito por meio do agrupamento dos indivíduos mediante o estrato e o conglomerado a que pertencem. Assim, considerando que o pacote consegue identificar corretamente a especificação do plano amostral, seja pelo *survey* ou outro qualquer, basta que seja estruturada a forma de agrupar os dados e depois calcular as diversas parcelas de $\hat{\boldsymbol{Z}}_{hj}$ e $\hat{\boldsymbol{Z}}_h$. Importante notar que a função que faz o cálculo das parcelas de $\hat{\boldsymbol{Z}}$ é uma só e depende unicamente do agrupamento fornecido para ela. Isso faz com que o foco fique unicamente sobre as diversas formas de conduzir estes agrupamentos, simplificando o problema.

No que diz respeito ao cálculo da variância de $\hat{\boldsymbol{Z}}$ propriamente dito, o problema também é simplificado quando analisado nesta perspectiva dos agrupamentos. A variância do estimador de Horvitz-Thompson possui uma fórmula geral e pode ser usado sob qualquer plano amostral. Logo, basta que as parcelas de $\hat{\boldsymbol{Z}}$ tenham sido calculadas e agrupadas corretamente para que a expressão possa ser generalizada. Santos (2016) apresenta diversas fórmulas gerais da variância do estimador de Horvitz-Thompson utilizando as probabilidades de seleção π_i mas que podem facilmente ser reescritas em termos do peso amostral w_i. Uma dessas formas que apresentaram bom desempenho é dada por

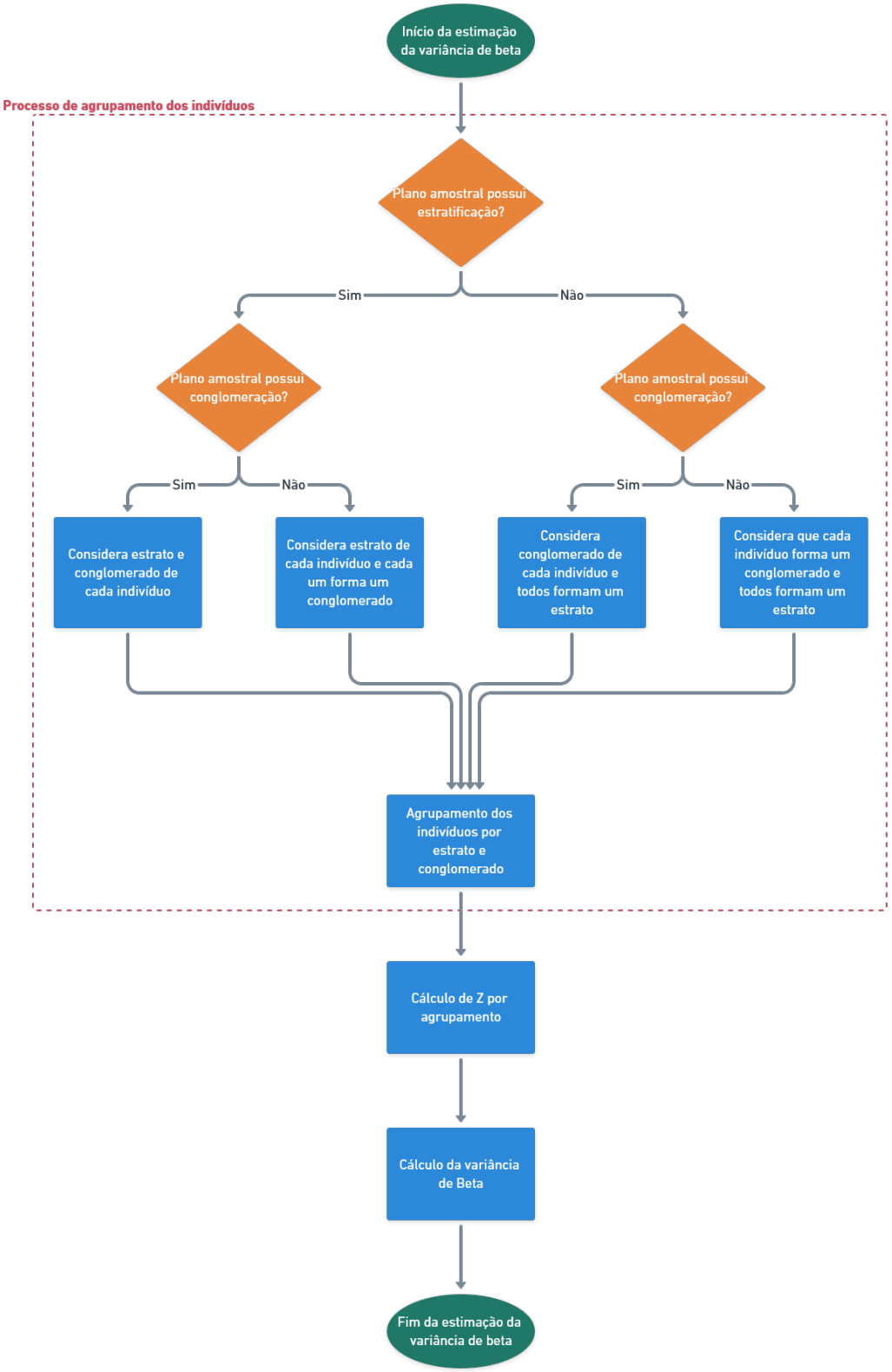
$$var(\hat{Y}_{HT}) = \frac{n}{n-1}\sum_{i\in\boldsymbol{s}}(1-\pi_i)\left(\frac{y_i}{\pi_i} - A\right)^2 \tag{3.21}$$

em que $A = \sum_{i\in\boldsymbol{s}} a_i\frac{y_i}{\pi_i}$ e $a_i = \frac{(1-\pi_i)}{\sum_{j\in\boldsymbol{s}}(1-\pi_j)}$. Faça $\hat{Y}_{HT} = \hat{\boldsymbol{Z}}$ e $y_i = \boldsymbol{Z}_i$ para voltar ao contexto da estimação de $\boldsymbol{Z}$.

Até o presente momento, o pacote desenvolvido não conta com estes níveis de generalização, mas já consideram o caso em que o usuário fornece como parâmetros de entrada uma variável de estratificação, uma de conglomeração ou ambos. O fluxograma abaixo mostra o procedimento considerado até então para o cálculo da variância de $\hat{\boldsymbol{\beta}}(\boldsymbol{V})$. Note que primeiro deve ser resolvido o problema da especificação do plano amostral para só então buscar generalizar as formas de agrupamento dos indivíduos e também do cálculo da variância de $\hat{\boldsymbol{Z}}$. Afim de evitar o fornecimento de muitos parâmetros de entrada na função principal, pode-se criar um único objeto que incorpore tanto estratificação, conglomeração e peso amostral. Esta abordagem facilita que no futuro este objeto seja substituído pelo resultado da especificação do plano amostral pelo pacote *survey*.

Vale ressaltar que outros métodos de estimação da variância podem ser incorporados futuramente ao pacote, como por exemplo Jackknife e Naive apresentados por Vieira (2005). Estes métodos em nada afetariam a parte de agrupamento dos indivíduos, mas apenas as partes posteriores a esta.

Figura 5 - Fluxograma da Estimação da Variância dos Efeitos Fixos



Fonte: Elaborado pelo autor. (2021).

3.5 ESTIMAÇÃO DE $\boldsymbol{\Sigma}$

Conforme discutido anteriormente, encontrar o estimador de máxima pseudo-verossimilhança de $\boldsymbol{\Sigma}$ pode vir a ser uma tarefa bastante complicada. Uma possível solução é utilizar o estimador natural de $\boldsymbol{\Sigma}$, denominado $\hat{\boldsymbol{\Sigma}}$.

De acordo com a expressão (3.3) que define $\boldsymbol{\Sigma}$, considere $\boldsymbol{S}_N$ o estimador censo da matriz de covariância populacional de $\boldsymbol{Y}_i$. Isto é

$$\boldsymbol{S}_N = \begin{bmatrix} var_N(\boldsymbol{Y}_{i1}) & & & \\ cov_N(\boldsymbol{Y}_{i2}, \boldsymbol{Y}_{i1}) & var_N(\boldsymbol{Y}_{i2}) & & \\ \vdots & & \ddots & \\ cov_N(\boldsymbol{Y}_{iT}, \boldsymbol{Y}_{i1}) & cov_N(\boldsymbol{Y}_{iT}, \boldsymbol{Y}_{i2}) & \dots & var_N(\boldsymbol{Y}_{iT}) \end{bmatrix} \tag{3.22}$$

em que $var_N(\boldsymbol{Y}_{it})$ é o estimador censo da variância populacional no instante t, $cov_N(\boldsymbol{Y}_{it}, \boldsymbol{Y}_{it'})$ é o estimador censo da covariância populacional entre t e $t^{'}$, e

$$var_N(\boldsymbol{Y}_{it}) = (N-1)^{-1} . \sum_{i=1}^{N} (y_{it} - \hat{\mu}_{N\ it})^2 \tag{3.23}$$

$$cov_N(\boldsymbol{Y}_{it}, \boldsymbol{Y}_{it'}) = (N-1)^{-1} . \sum_{i=1}^{N} (y_{it} - \hat{\mu}_{N\ it})(y_{it'} - \hat{\mu}_{N\ it'}) \tag{3.24}$$

em que, $\hat{\mu}_{N\ it}$ é o estimador censo da média μ_{it} definida em (3.1) e (3.2), isto é,

$$\hat{\mu}_{N\ it} = \mu_{it}(\hat{\boldsymbol{\beta}}). \tag{3.25}$$

Vieira (2005) mostra que $\boldsymbol{S}_N$ é um estimador não viesado de $\boldsymbol{\Sigma}$, ou seja,

$$E(\boldsymbol{S}_N) = \boldsymbol{\Sigma}. \tag{3.26}$$

até mesmo quando $\hat{\boldsymbol{\mu}}_{N\ it}$ inclui o uso de covariáveis.

Considere agora que apenas n indivíduos foram observados em uma amostra selecionada a partir de um plano amostral complexo $p(\boldsymbol{s})$. Seja $\boldsymbol{S}_w$ a matriz $T \times T$ de covariância amostral considerando o desenho de amostragem, isto é

$$\boldsymbol{S}_w = \begin{bmatrix} var_w(\boldsymbol{Y}_{i1}) & & & \\ cov_w(\boldsymbol{Y}_{i2}, \boldsymbol{Y}_{i1}) & var_w(\boldsymbol{Y}_{i2}) & & \\ \vdots & & \ddots & \\ cov_w(\boldsymbol{Y}_{iT}, \boldsymbol{Y}_{i1}) & cov_w(\boldsymbol{Y}_{iT}, \boldsymbol{Y}_{i2}) & \dots & var_w(\boldsymbol{Y}_{iT}) \end{bmatrix} \tag{3.27}$$

em que $var_w(\boldsymbol{Y}_{it})$ é o estimador da variância populacional no instante t, $cov_w(\boldsymbol{Y}_{it}, \boldsymbol{Y}_{it'})$ é o estimador da covariância populacional entre t e $t^{'}$, e

$$var_w(\boldsymbol{Y}_{it}) = \left(\sum_{i=1}^{n} w_i \right)^{-1} . \sum_{i=1}^{n} w_i (y_{it} - \hat{\mu}_{it})^2 \tag{3.28}$$

$$cov_w(\boldsymbol{Y}_{it}, \boldsymbol{Y}_{it'}) = \left(\sum_{i=1}^{n} w_i \right)^{-1} . \sum_{i=1}^{n} w_i (y_{it} - \hat{\mu}_{it})(y_{it'} - \hat{\mu}_{it'}) \tag{3.29}$$

em que $\hat{\mu}_{it}$ é o estimador da média μ_{it} tal que $\hat{\mu}_{it} = \hat{\mu}(\boldsymbol{x}_{it}, \hat{\boldsymbol{\beta}}(\boldsymbol{V}))$.

Vieira (2005) mostra também que $\boldsymbol{S}_w$ é um estimador consistente de $\boldsymbol{\Sigma}$ independente da escolha da matriz de covariância de trabalho $\boldsymbol{V}$, ou seja,

$$E(E_p(\boldsymbol{S}_w)) = \boldsymbol{\Sigma}, \tag{3.30}$$

em que $E(.)$ representa o tradicional operador que calcula a esperança de variáveis aleatórias.

3.6 ESTIMAÇÃO DA VARIÂNCIA DE $\hat{\boldsymbol{\Sigma}}$

Seja $vech(.)$ o operador que transforma os elementos não duplicados de uma matriz qualquer em um vetor coluna com esses elementos[2]. Por exemplo,

$$vech(\boldsymbol{\Sigma}) = (\Sigma_{11}, \Sigma_{21}, \ldots, \Sigma_{T1}, \Sigma_{22}, \ldots, \Sigma_{T2}, \ldots, \Sigma_{TT})'. \tag{3.31}$$

Dessa forma, $vech(\boldsymbol{S}_w)$ pode ser reescrito em função dos resíduos $\hat{\epsilon}_{it}$, ou seja

$$vech(\boldsymbol{S}_w) = \begin{bmatrix} \left(\sum_{i=1}^{n} w_i\right)^{-1} . \sum_{i=1}^{n} w_i \hat{\epsilon}_{i1}\hat{\epsilon}_{i1} \\ \left(\sum_{i=1}^{n} w_i\right)^{-1} . \sum_{i=1}^{n} w_i \hat{\epsilon}_{i2}\hat{\epsilon}_{i1} \\ \vdots \\ \left(\sum_{i=1}^{n} w_i\right)^{-1} . \sum_{i=1}^{n} w_i \hat{\epsilon}_{iT}\hat{\epsilon}_{i(T-1)} \\ \left(\sum_{i=1}^{n} w_i\right)^{-1} . \sum_{i=1}^{n} w_i \hat{\epsilon}_{iT}\hat{\epsilon}_{iT} \end{bmatrix} = \left(\sum_{i=1}^{n} w_i\right)^{-1} . \sum_{i=1}^{n} w_i \boldsymbol{c}_i \tag{3.32}$$

em que $\boldsymbol{c}_i$ é um vetor coluna $k \times 1$ contendo o produto dos resíduos de $vech(\boldsymbol{S}_w)$.

Note que $vech(\boldsymbol{S}_w)$ pode ser interpretado como a razão de dois totais. Isto é

$$vech(\boldsymbol{S}_w) = \frac{\sum_{i=1}^{n} w_i \boldsymbol{c}_i}{\sum_{i=1}^{n} w_i} = \frac{\overline{\boldsymbol{z}}}{\overline{w}} \tag{3.33}$$

em que $\overline{\boldsymbol{z}} = n^{-1} \sum_{i=1}^{n} \boldsymbol{z}_i$ é a matriz $k \times 1$ de médias de $\boldsymbol{z}_i$ com $\boldsymbol{z}_i = w_i \boldsymbol{c}_i$ e $\overline{w} = n^{-1} \sum_{i=1}^{n} w_i$ é a média dos pesos amostrais.

Conforme apresenta Vieira (2005), uma possível solução para encontrar a variância de $\hat{\boldsymbol{\Sigma}}$ é aplicar uma expansão de Taylor de primeira ordem em $vech(\boldsymbol{S}_w)$. Sendo assim, o estimador da variância pode ser aproximado como

$$var_L(vech(\hat{\boldsymbol{\Sigma}})) = var_L(vech(\boldsymbol{S}_w)) = \frac{1}{n^2} var\left(\sum_{i=1}^{n} \boldsymbol{u}_i\right) = \frac{1}{n^2} var(\boldsymbol{B}) \tag{3.34}$$

[2] Se a matriz for simétrica, este processo equivale a criar um vetor coluna com os elementos da matriz triangular inferior ou superior.

em que

$$\boldsymbol{u}_i = \frac{1}{\mu_w}\left(\boldsymbol{z}_i - \frac{\boldsymbol{\mu}_z}{\mu_w} w_i\right) \tag{3.35}$$

$$\boldsymbol{B} = \sum_{i=1}^{n} \boldsymbol{u}_i \ . \tag{3.36}$$

Como μ_w e $\boldsymbol{\mu}_z$ são desconhecidos, as expressões anteriores podem ser estimadas por

$$\hat{\boldsymbol{u}}_i = \frac{1}{\overline{w}}\left(\boldsymbol{z}_i - \frac{\overline{\boldsymbol{z}}}{\overline{w}} w_i\right) \tag{3.37}$$

$$\hat{\boldsymbol{B}} = \sum_{i=1}^{n} \hat{\boldsymbol{u}}_i \ , \tag{3.38}$$

em que tanto $\hat{\boldsymbol{u}}_i$ quanto $\hat{\boldsymbol{B}}$ são matrizes $k \times 1$.

Assim como na estimação da variância de $\hat{\boldsymbol{\beta}}(\boldsymbol{V})$, $\hat{\boldsymbol{Z}}$ era um estimador de total populacional, $\hat{\boldsymbol{B}}$ também é, ou seja, é um estimador de Horvitz-Thompson. Logo, para o exemplo considerado na seção (3.4) que envolve estratificação e conglomeração, $\hat{\boldsymbol{B}}$ pode ser reescrito como

$$\hat{\boldsymbol{B}} = \sum_{h=1}^{H} \hat{\boldsymbol{B}}_h = \sum_{h=1}^{H}\sum_{j=1}^{m_h} \hat{\boldsymbol{B}}_{hj} = \sum_{h=1}^{H}\sum_{j=1}^{m_h}\sum_{i=1}^{n_{hj}} \hat{\boldsymbol{u}}_{hji} \ , \tag{3.39}$$

em que H é o número de estratos na amostra, m_h é o número de conglomerados dentro do estrato h e n_{hj} é o número de indivíduos pertencentes ao conglomerado j do estrato h.

Logo,

$$var_L[\hat{\boldsymbol{B}}] = \sum_{h=1}^{H} \frac{m_h}{m_h - 1}\left[\sum_{j=1}^{m_h} (\hat{\boldsymbol{B}}_{hj} - \overline{\hat{\boldsymbol{B}}}_h)^2\right] \ , \tag{3.40}$$

em que $\overline{\hat{\boldsymbol{B}}}_h$ é a média de $\hat{\boldsymbol{B}}_{hj}$ no estrato h.

Finalmente, o estimador da variância de $\hat{\boldsymbol{\Sigma}}$ neste caso é dado por:

$$var_L\left[vech(\hat{\boldsymbol{\Sigma}})\right] = \frac{1}{n^2} var(\hat{\boldsymbol{B}}). \tag{3.41}$$

3.6.1 Consideração de $\hat{\boldsymbol{\Sigma}}$ no pacote

A partir da metodologia apresentado na seção anterior, é possível perceber que a incorporação da estimação de $\hat{\boldsymbol{\Sigma}}$ no pacote não é uma tarefa complicada, uma vez que para calcular a matriz $\boldsymbol{S}_w$ é necessário ter apenas os resíduos que já foram calculados anteriormente.

Sendo assim, a única questão a se considerar é a estimação da variância de $\boldsymbol{S}_w$. Neste caso, a própria metodologia mostrou que existe uma semelhança muito grande

entre este cálculo e o da variância de $\hat{\boldsymbol{\beta}}(\boldsymbol{V})$, de forma que a grande problemática está em agrupar os dados corretamente, ou seja, ao resolver o problema do agrupamento dos dados para $\hat{\boldsymbol{\beta}}(\boldsymbol{V})$, o mesmo vale para $\boldsymbol{S}_w$. Além disso, a função que calcula $var_L[\hat{\boldsymbol{B}}]$ pode ser a mesma que calcula $var_L[\hat{\boldsymbol{Z}}]$. Desta forma, basta apenas colocar como parâmetro de entrada da função uma lista contendo os valores de $\hat{\boldsymbol{B}}$, $\hat{\boldsymbol{Z}}$ ou qualquer outro estimador de Horvitz-Thompson agrupado corretamente.

3.7 COMENTÁRIOS FINAIS

A implementação de toda a metodologia apresentada neste capítulo já é suficiente para criar um pacote no R capaz de lidar com ajuste de modelos longitudinais considerando o plano amostral complexo. Até o momento, o pacote consegue estimar os parâmetros $\boldsymbol{\beta}$ e $\boldsymbol{\Sigma}$ da superpopulação considerando estratificação, conglomeração e probabilidades de seleção desiguais, além do ajuste para não resposta dado pela correção do peso amostral. Além disso, o pacote calcula o erro padrão dos estimadores, tornando possível a criação de intervalo de confiança e também teste de hipóteses.

A possibilidade de fornecer $\boldsymbol{\Sigma}$ como parâmetro de entrada da função permite automaticamente a implementação de um algoritmo de convergência dos parâmetros que poderia ser muito útil. Outro aspecto que este pacote proporciona é a facilidade de conduzir estudos de simulação, buscando, por exemplo, mostrar os efeitos da não consideração correta do plano amostral na estimação dos parâmetros, ou qualquer outra característica de interesse do pesquisador.

Em termos de melhoria, o pacote ainda precisa trabalhar na forma de incorporar o desenho de amostragem da pesquisa. Uma meta futura será realizar a integração do pacote 'clm' com o pacote *survey*. Em seguida, é necessário explorar a generalização do agrupamento dos indivíduos, de forma que este possa ser generalizado para qualquer tipo de plano amostral. Finalmente, criar as funções gerais que realizam o cálculo da variância do estimador de Horvitz-Thompson. Este último ponto seria menos complicado para ser implementado, mas só faria sentido se houvesse a consideração de outros planos amostrais, além dos que foram apresentados até agora.

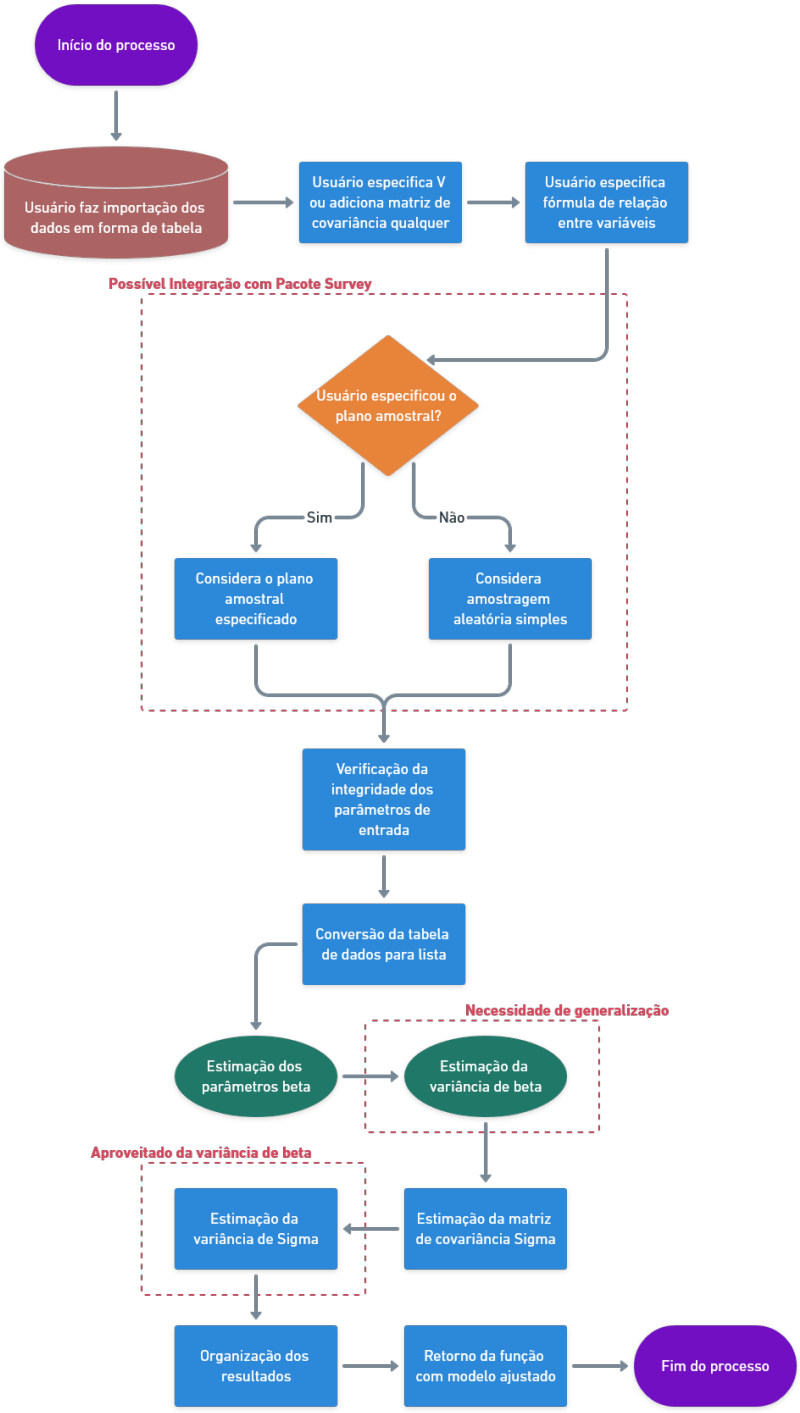
Existe ainda uma segunda parte a ser trabalhada e esta será vista no próximo capítulo, que são modelos de covariância estruturada para dados longitudinais amostrais complexos. Esta outra etapa pode ser trabalhada de forma independente desta primeira, apesar de ser consequência desta. Assim, não seria natural considerar ambas as partes em um único lugar. Mesmo assim, ambas as funções serão vistas como partes integrantes do mesmo pacote denominado 'clm'.

Finalmente, uma última questão a se considerar é a respeito do retorno da função para o usuário. Em linhas gerais, é imprescindível que a função retorne de forma clara e objetiva as estimativas dos betas, seus respectivos erros-padrão e a estimava de sigma. O

erro-padrão de sigma, neste caso, pode vir a poluir a tela com muitas informações, pois a matriz triangular da mesma pode ser bastante grande. Para $T = 5$, por exemplo, são 15 elementos de sigma que precisarão ser apresentados, e caso haja alguns elementos não significativos, isto não possui uma interpretabilidade muito forte como é no caso dos betas. Ainda sim, a variância de sigma deve estar disponível ao usuário caso ele queira realmente vê-la. Como é de costume em regressão, é interessante adicionar o p-valor das estimativas dos betas e também informar se os parâmetros estimados são significativos ou não. Outra questão interessante de fornecer como retorno são as características do plano amostral, uma vez que o pacote possui esta funcionalidade como destaque. Finalmente, deve-se mostrar qual foi a matriz sigma fornecida pelo usuário caso seja este o caso ou então qual foi a matriz $\boldsymbol{V}$ calculada, além também do $\hat{\rho}$.

Neste sentido, a função principal criada no pacote foi denominada 'clm' e o fluxograma geral da função é dado abaixo.

Figura 6 - Fluxograma geral da função 'clm'



Fonte: Elaborado pelo autor. (2021).

4 MODELOS DE COVARIÂNCIA ESTRUTURADA PARA DADOS LONGITUDINAIS COMPLEXOS

4.1 CONSIDERAÇÕES INICIAIS

Neste capítulo será vista uma possível abordagem para a modelagem da matriz de covariância desenvolvida por Vieira (2005), que é considerar que $\boldsymbol{\Sigma}$ é função de um vetor de parâmetros desconhecidos $\boldsymbol{\theta}$ de dimensão $b \times 1$ com $b < k$ em que k é o número de elementos distintos de $\boldsymbol{\Sigma}$, ou seja, $k = \frac{T(T+1)}{2}$. Esta restrição aplicada na matriz de covariância faz com que seja possível encontrar padrões de comportamento de $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ e os ganhos em termos de interpretabilidade do modelo são muito positivos.

Neste sentido, será abordada a estimação dos parâmetros $\boldsymbol{\theta}$ e seus respectivos erros-padrão. Vale destacar que todos os procedimentos serão vistos sob a ótica da incorporação do plano amostral $p(\boldsymbol{s})$ da pesquisa na estimação dos parâmetros e também na perspectiva computacional por meio da criação da função 'cov_clm'. Todas as suposições apresentadas na seção 3.1 do capítulo anterior se mantém também para este capítulo.

Como forma de motivação da metodologia, seguem abaixo dois exemplos de matrizes de covariância estruturada, $\boldsymbol{\Sigma}(\boldsymbol{\theta})$, definidas a partir de modelos alternativos.

4.2 EXEMPLOS

4.2.1 Modelo de correlação uniforme - UCM

Conforme apresentado na seção 3.1 do capítulo anterior, o modelo de efeitos fixos e aleatórios pode ser definido como

$$\boldsymbol{Y}_i = \boldsymbol{X}_i\boldsymbol{\beta} + \boldsymbol{\epsilon}_i, \tag{4.1}$$

sendo que $\boldsymbol{\epsilon}_i$ pode ser reescrito como

$$\epsilon_{it} = u_i + v_{it}, \tag{4.2}$$

em que u_i representa o efeito aleatório permanente, logo é a mesma para todo $t = 1, \ldots, T$, e v_{it} o efeito aleatório transitório. Dado o modelo de regressão anterior, assume-se $E(\boldsymbol{\epsilon}_i) = \mathbf{0}$ e $VAR(\boldsymbol{\epsilon}_i) = \boldsymbol{\Sigma}$.

De acordo com Vieira (2005), um caso especial desta classe de modelos tem como base a consideração de que u_i e v_{it} são mutuamente independentes com $E(u_i) = 0$, $VAR(u_i) = \sigma_u^2$, $E(v_{it}) = 0$, $VAR(v_{it}) = \sigma_v^2$, $COV(v_{it}, v_{it'}) = 0\ \forall t \neq t'$ e $COV(u_i, v_{it}) = 0$.

Assim sendo, a variância e a covariância de Y_{it} são dadas por

$$\begin{aligned} VAR(Y_{it}) &= E[(\epsilon_{it})^2] = E[u_i + v_{it}]^2 = E[u_i^2 + 2u_i v_{it} + v_{it}^2] = \sigma_u^2 + 2E[u_i v_{it}] + \sigma_v^2 \\ &= \sigma_u^2 + \sigma_v^2 \qquad (4.3) \\ COV(Y_{it}, Y_{it'}) &= E[(u_i + v_{it})(u_i + v_{it'})] = E[(u_i)^2] + E[u_i v_{it}] + E[u_i v_{it'}] + E[v_{it} v_{it'}] = E[(u_i)^2] \\ &= \sigma_u^2 \qquad (4.4) \end{aligned}$$

para todo $t \neq t^{'}$.

Portanto, $\mathbf{\Sigma} = \mathbf{\Sigma}(\boldsymbol{\theta})$ pode ser escrita como

$$\mathbf{\Sigma}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_u^2 + \sigma_v^2 & \sigma_u^2 & \dots & \sigma_u^2 \\ \sigma_u^2 & \sigma_u^2 + \sigma_v^2 & \dots & \sigma_u^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_u^2 & \sigma_u^2 & \dots & \sigma_u^2 + \sigma_v^2 \end{bmatrix} \qquad (4.5)$$

em que $\boldsymbol{\theta} = (\sigma_u^2, \sigma_v^2)^{'}$ é um vetor de parâmetros desconhecidos e $\mathbf{\Sigma}(\boldsymbol{\theta})$ é uma matriz $T \times T$.

4.2.2 Modelo de efeitos transitórios autorregressivos de primeira ordem - AR1

Neste outro caso apresentado pelo autor, assume-se que a componente de efeitos transitórios é dada por

$$v_{it} = \gamma v_{it-1} + \delta_{it}, \ t = 2, \dots, T \qquad (4.6)$$

em que $-1 < \gamma < 1$ e δ_{it} é um termo de erro com

$$E(\delta_{it}) = 0 \qquad (4.7)$$

e

$$VAR(\delta_{it}) = \sigma_\delta^2. \qquad (4.8)$$

Segundo Vieira (2005), considerando que v_{it} e δ_{it} são mutuamente independentes e que v_{it} constitui uma série temporal estacionária, tem-se

$$\sigma_v^2 = \frac{\sigma_\delta^2}{1 - \gamma^2}. \qquad (4.9)$$

Trabalhando ainda sob os mesmos pressupostos do exemplo anterior a respeito de u_i e v_{it} com exceção da covariância dos efeitos transitórios ser igual a zero, tem-se

$$VAR(Y_{it}) = \sigma_u^2 + \sigma_v^2 = \sigma_u^2 + \frac{\sigma_\delta^2}{1 - \gamma^2} \qquad (4.10)$$

e

$$COV(Y_{it}, Y_{it'}) = E[(u_i)^2] + E[v_{it}v_{it'}] = \sigma_u^2 + \gamma^{|t-t'|}\sigma_v^2. \tag{4.11}$$

Portanto, $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(\boldsymbol{\theta})$ pode ser escrita como

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_u^2 + \sigma_v^2 & \sigma_u^2 + \gamma^1\sigma_v^2 & \dots & \sigma_u^2 + \gamma^{T-1}\sigma_v^2 \\ \sigma_u^2 + \gamma^1\sigma_v^2 & \sigma_u^2 + \sigma_v^2 & \dots & \sigma_u^2 + \gamma^{T-2}\sigma_v^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_u^2 + \gamma^{T-1}\sigma_v^2 & \sigma_u^2 + \gamma^{T-2}\sigma_v^2 & \dots & \sigma_u^2 + \sigma_v^2 \end{bmatrix} \tag{4.12}$$

em que $\boldsymbol{\theta} = (\sigma_u^2, \sigma_v^2, \gamma)'$ é um vetor de parâmetros desconhecidos e $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ é uma matriz $T \times T$.

Note que diferentes estruturas de covariância surgem a partir da variação das especificações dos efeitos transitórios do modelo. Isto será de fundamental importância posteriormente neste capítulo.

4.3 ESTIMAÇÃO DO PARÂMETRO $\boldsymbol{\theta}$

Assim como no capítulo anterior, suponha que $\boldsymbol{Y}_i \sim N_T[\boldsymbol{\mu}_i(\boldsymbol{\beta}), \boldsymbol{\Sigma}(\boldsymbol{\theta})]$, em que $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ é a matriz de covariância $T \times T$. De acordo com Vieira (2005), a função log-verossimilhança do censo é dada por

$$L_N[\boldsymbol{\beta}, \boldsymbol{\theta}] = -\frac{1}{2}\left[N.T\ log\ 2\pi + N\ log\ |\boldsymbol{\Sigma}(\boldsymbol{\theta})| + \sum_{i=1}^{N} [\boldsymbol{y}_i - \boldsymbol{\mu}_i(\boldsymbol{\beta})]' \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1} [\boldsymbol{y}_i - \boldsymbol{\mu}_i(\boldsymbol{\beta})]\right]. \tag{4.13}$$

em que $\boldsymbol{\theta} = (\theta_1, \dots, \theta_b)'$ é um vetor de dimensão $b \times 1$ de parâmetros desconhecidos.

Note que a adaptação de $\boldsymbol{\Sigma}$ para $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ não afeta a estimação dos parâmetros $\boldsymbol{\beta}$. Assim,

$$\hat{\boldsymbol{\beta}}(\boldsymbol{\Sigma}(\boldsymbol{\theta})) = \left(\sum_{i=1}^{n} w_i \boldsymbol{X}_i' \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1} \boldsymbol{X}_i\right)^{-1} \sum_{i=1}^{n} w_i \boldsymbol{X}_i' \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1} \boldsymbol{y}_i. \tag{4.14}$$

Para $\boldsymbol{\mu}_i(\boldsymbol{\beta}) = \boldsymbol{X}_i\boldsymbol{\beta}$, $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}$ e $\boldsymbol{S}_w = \left(\sum_{i=1}^{n} w_i\right)^{-1} . \sum_{i=1}^{n} w_i [\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta}] [\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta}]'$, Vieira (2005) apresenta que $L_N[\boldsymbol{\beta}, \boldsymbol{\theta}]$ pode ser reescrito como

$$L_N[\boldsymbol{\beta}, \boldsymbol{\theta}] = -\frac{1}{2}\sum_{i=1}^{n} w_i \left[tr(\boldsymbol{S}_w \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}) + log|\boldsymbol{\Sigma}(\boldsymbol{\theta})|\right], \tag{4.15}$$

em que o operador $tr(.)$ representa o traço da matriz em questão.

Considere ainda $F(\boldsymbol{\theta})$ uma função tal que

$$\begin{aligned} F(\boldsymbol{\theta}) &= tr(\boldsymbol{S}_w \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}) + log|\boldsymbol{\Sigma}(\boldsymbol{\theta})| - log|\boldsymbol{S}_w| - T \\ &= tr(\boldsymbol{S}_w \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}) - log|\boldsymbol{S}_w \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}| - T \end{aligned} \tag{4.16}$$

em que T é representa o número total de rodadas.

Observe que $L_N[\boldsymbol{\beta}, \boldsymbol{\theta}]$ pode ser escrito em função de $F(\boldsymbol{\theta})$, pois a adição dos termos $-log|\boldsymbol{S}_w| - T$ afeta em nada o cálculo do estimador de $\boldsymbol{\theta}$. Inclusive, estes termos foram adicionados apenas para que $F(\boldsymbol{\theta})$ seja igual a zero quando $\boldsymbol{S}_w = \boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}})$. Logo,

$$L_N[\boldsymbol{\beta}, \boldsymbol{\theta}] \ \alpha \ -\frac{1}{2}\sum_{i=1}^{n} w_i F(\boldsymbol{\theta}). \tag{4.17}$$

Segundo Vieira (2005), derivando $L_N[\boldsymbol{\beta}, \boldsymbol{\theta}]$ em relação a θ_j tem-se

$$\frac{\partial L_N[\boldsymbol{\beta}, \boldsymbol{\theta}]}{\partial \theta_j} = -\frac{1}{2}\sum_{i=1}^{n} w_i \frac{\partial F(\boldsymbol{\theta})}{\partial \theta_j} = -\frac{1}{2}\sum_{i=1}^{n} w_i tr\left[\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}(\boldsymbol{\Sigma}(\boldsymbol{\theta}) - \boldsymbol{S}_w)\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\frac{\partial \boldsymbol{\Sigma}(\boldsymbol{\theta})}{\partial \theta_j}\right], \tag{4.18}$$

com $j = 1, \ldots, b$.

Dessa forma, $\hat{\theta}_j$ é aquele que soluciona a equação

$$tr\left[\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}(\boldsymbol{\Sigma}(\boldsymbol{\theta}) - \boldsymbol{S}_w)\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\frac{\partial \boldsymbol{\Sigma}(\boldsymbol{\theta})}{\partial \theta_j}\right] = 0. \tag{4.19}$$

Em geral, encontrar uma solução analítica para $\hat{\theta}_j$ não é uma solução fácil, principalmente quando o número de rodadas da pesquisa é grande. Neste caso, o mais indicado é utilizar métodos computacionais de otimização.

4.3.1 Propostas alternativas

A forma descrita anteriormente não é a única maneira de se estimar $\boldsymbol{\theta}$. Vieira (2005) descreve ainda outras diversas possibilidades para $F(\boldsymbol{\theta})$. A ideia é que F seja uma função que mede a distância entre $\boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}})$ e $\boldsymbol{S}_w$. Definido F, basta encontrar $\hat{\theta}_j$ que minimize a função.

Entretanto, é necessário que F tenha as seguintes propriedades: (i) F deve ser uma função escalar; (ii) $F(\boldsymbol{\theta}) \geq 0 \ \forall \boldsymbol{\theta}$; (iii) $F(\hat{\boldsymbol{\theta}}) = 0$ se, e somente se, $\boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}}) = \boldsymbol{S}_w$; (iv) F deve ser contínua e diferenciável de segunda ordem com relação a $\boldsymbol{\theta}$. Vieira (2005) descreve que dada a estrutura da matriz de covariância $\boldsymbol{\Sigma}(\boldsymbol{\theta})$, se $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ for contínua então a minimização de qualquer F que satisfaça as propriedades anteriores irá resultar em um estimador consistente para $\boldsymbol{\theta}$.

No presente estudo, não são analisadas em detalhes as diferentes formas de se definir e minimizar a função F. Para mais informações, ver (Vieira, 2005). Porém, seguem abaixo algumas alternativas:

1. $F(\boldsymbol{\theta})_{ULSC} = \frac{1}{2}tr[(\boldsymbol{S}_w - \boldsymbol{\Sigma}(\boldsymbol{\theta}))^2]$;
2. $F(\boldsymbol{\theta})_{GLSC2} = \frac{1}{2}tr[((\boldsymbol{S}_w - \boldsymbol{\Sigma}(\boldsymbol{\theta}))\boldsymbol{S}_w^{-1})^2]$;

3. $F(\boldsymbol{\theta})_{GLSC3} = \frac{1}{2} tr[(\boldsymbol{S}_w \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1} - I)^2]$;

4. $F(\boldsymbol{\theta})_{GLSC4} = [vech(\boldsymbol{S}_w) - vech(\boldsymbol{\Sigma}(\boldsymbol{\theta}))]' \, \hat{\boldsymbol{C}}_c^{-1} \, [vech(\boldsymbol{S}_w) - vech(\boldsymbol{\Sigma}(\boldsymbol{\theta}))]$;

em que I representa a matriz identidade de mesma dimensão e $\hat{\boldsymbol{C}}_c = n \; . \; var_L \, [vech(\boldsymbol{S}_w)]$ é dado pela expressão (3.41).

4.3.2 Incorporação ao pacote

O capítulo anterior marcou a conclusão da primeira grande função do pacote, que é a função 'clm', responsável por calcular as estimativas de $\hat{\boldsymbol{\beta}}$, $\hat{\boldsymbol{\Sigma}}$ e seus respectivos erros-padrão. Logo, parte-se do princípio que $\boldsymbol{S}_w$ já foi calculado e a única questão a se preocupar é com relação a $\boldsymbol{\Sigma}(\boldsymbol{\theta})$.

Neste sentido, é necessário começar pela própria especificação da matriz de covariância. Conforme visto no início deste capítulo, a estrutura de $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ não é única, diferentes modelos e suposições resultam em diferentes estruturas. A missão de controlar isto não é fácil.

Por um lado, pode-se restringir a escolha da matriz de covariância a um número limitado de opções, como por exemplo, permitir que o usuário faça a escolha de ajuste considerando um modelo UCM ou AR1. Esta abordagem facilita o controle interno da função, pois os casos já estão bem definidos, não exigindo lidarmos com generalização. Além disso, esta estratégia simplifica o processo para o usuário, pois este precisa apenas informar qual o tipo de ajuste desejado, não sendo necessário fornecer explicitamente a forma da matriz indicada. Entretanto, a função fica bastante restrita pois abre mão de uma gama de possibilidades de ajuste e limita o uso do pacote a casos bastante específicos e longe da realidade de complexidade das pesquisas atuais.

Por outro lado, pode-se buscar deixar a especificação de $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ livre para o usuário como um parâmetro de entrada, de maneira que o próprio usuário seja capaz de fornecer a estrutura desejada por ele. A vantagem desta abordagem é o fato de tornar a função bastante genérica e permitir que o usuário tenha o poder de discernir qual a melhor forma da matriz de covariância para o seu caso, deixando-o livre, inclusive, para fornecer estruturas extremamente complexas fruto de outros trabalhos desenvolvidos. A dificuldade nesse caso é justamente generalizar a função de forma a receber qualquer estrutura para $\boldsymbol{\Sigma}(\boldsymbol{\theta})$. Além disso, conforme se vê pela expressão (4.19), torna-se necessário que o usuário também forneça as estruturas das matrizes contendo as derivadas parciais de $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ em relação a cada um dos parâmetros de interesse e isto pode ser trabalhoso.

Considerando estes dois pontos de vista, fica evidente que cada um tem algo de positivo a se considerar. O primeiro permite que o usuário faça ajustes de modelos sem muito esforço e o segundo o torna livre para escolher de maneira ilimitada a estrutura da matriz de covariância desejada. Sendo assim, o almejado é permitir que o parâmetro

de entrada da estrutura da matriz de covariância receba ambos os tipos de argumento: uma cadeia de caracteres especificando a forma desejada ou uma matriz de expressões não avaliadas contendo a estrutura da $\boldsymbol{\Sigma}(\boldsymbol{\theta})$. O R felizmente possui pacotes que realizam derivadas simbólicas de expressões não avaliadas. Sendo assim, é possível facilitar o processo para o usuário e exigir apenas que ele forneça a expressão da $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ e deixe o cálculo das derivadas parciais como responsabilidade do pacote.

Outro ponto que deve-se ter em mente é que o cálculo das estimativas de $\hat{\boldsymbol{\theta}}$ irão resultar de um processo de otimização e se a função que realiza este procedimento não for bem configurada, a mesma pode não vir a funcionar bem recebendo uma estrutura de covariância qualquer e existe ainda o problema de encontrar máximos e mínimos locais na otimização, o que coloca sobre as mãos do usuário a responsabilidade de testar o ajuste com diferentes valores iniciais a fim de garantir que de fato as estimativas de $\hat{\boldsymbol{\theta}}$ são confiáveis.

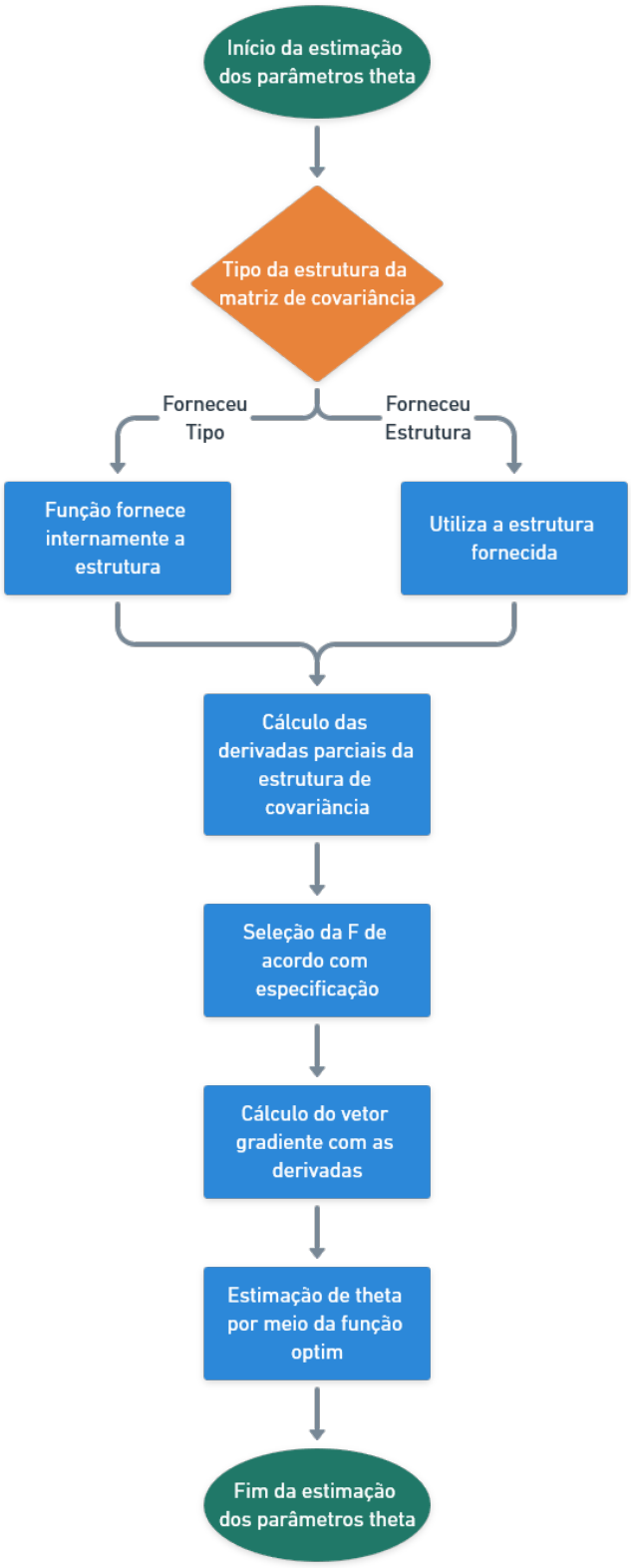
Considerando que o primeiro problema está solucionado, falta ainda realizar o processo de minimização da F a fim de encontrar as estimativas para $\hat{\boldsymbol{\theta}}$. O R possui também diversas funções que realizam processos de otimização, como por exemplo, a função *optim*, que será adotada neste pacote. A própria função *optim* possui diversos métodos de optimização com configurações alternativas. É interessante que tal diversidade seja aproveitada. Assim, ficam como possibilidade de parâmetros de entrada as configurações da *optim*, permitindo que o usuário tenha o poder de controlar o ajuste de $\hat{\boldsymbol{\theta}}$.

Finalmente, existe também o problema da própria especificação da função F. Conforme apresentado anteriormente, diversas funções podem ser definidas e é interessante que o pacote consiga lidar com essa diversidade. Entretanto, como a minimização de qualquer uma daquelas Fs irá fornecer estimadores consistentes para $\boldsymbol{\theta}$, não é necessário que o usuário tenha o poder de fornecer a sua própria função. A implementação das quatro alternativas apresentadas na seção 4.3.1 são suficientes. Entretanto, a fim de garantir o processo de otimização, é necessário que sejam calculadas manualmente as derivadas parciais de cada uma das Fs, assim como foi feito para o caso da máxima pseudo-verossimilhança. Estas derivadas são dadas abaixo.

1. $\frac{\partial F(\boldsymbol{\theta})_{ULSC}}{\partial \theta_j} = tr\left[(\boldsymbol{\Sigma}(\boldsymbol{\theta}) - \boldsymbol{S}_w)\frac{\partial \boldsymbol{\Sigma}(\boldsymbol{\theta})}{\partial \theta_j}\right];$
2. $\frac{\partial F(\boldsymbol{\theta})_{GLSC2}}{\partial \theta_j} = tr\left[(\boldsymbol{S}_w^{-1}\boldsymbol{\Sigma}(\boldsymbol{\theta}) - I)\boldsymbol{S}_w^{-1}\frac{\partial \boldsymbol{\Sigma}(\boldsymbol{\theta})}{\partial \theta_j}\right];$
3. $\frac{\partial F(\boldsymbol{\theta})_{GLSC3}}{\partial \theta_j} = tr\left[\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\boldsymbol{S}_w(I - \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\boldsymbol{S}_w)\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\frac{\partial \boldsymbol{\Sigma}(\boldsymbol{\theta})}{\partial \theta_j}\right];$
4. $\frac{\partial F(\boldsymbol{\theta})_{GLSC4}}{\partial \theta_j} = tr\left[2\hat{\boldsymbol{C}}_c^{-1}(vech(\boldsymbol{\Sigma}(\boldsymbol{\theta})) - vech(\boldsymbol{S}_w))\frac{\partial vech(\boldsymbol{\Sigma}(\boldsymbol{\theta}))}{\partial \theta_j'}\right].$

O diagrama abaixo representa este processo.

Figura 7 - Fluxograma Estimação Theta da função 'cov_clm'



Fonte: Elaborado pelo autor. (2021).

4.4 ESTIMAÇÃO DA VARIÂNCIA DE $\hat{\boldsymbol{\theta}}$

Seguindo o raciocínio apresentado pela expressão (4.18) para estimação de $\boldsymbol{\theta}$, considere

$$\phi_j(\boldsymbol{\theta}) = \frac{\partial F(\boldsymbol{\theta})}{\partial \theta_j} = tr\left[\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}(\boldsymbol{\Sigma}(\boldsymbol{\theta}) - \boldsymbol{S}_w)\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\frac{\partial \boldsymbol{\Sigma}(\boldsymbol{\theta})}{\partial \theta_j}\right] \tag{4.20}$$

com $j = 1, \ldots, b$, em que b representa o número de parâmetros que compõem $\boldsymbol{\theta}$.

Vieira (2005) apresenta que para o cálculo da variância de $\hat{\boldsymbol{\theta}}$ uma possível abordagem é considerar uma expansão de Taylor de primeira ordem em $\boldsymbol{\phi}(\boldsymbol{\theta})$ com $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}$, ou seja, quando $\boldsymbol{\phi}(\boldsymbol{\theta}) = \mathbf{0}$. Em outras palavras,

$$\mathbf{0} = \boldsymbol{\phi}(\hat{\boldsymbol{\theta}}) \cong \boldsymbol{\phi}(\boldsymbol{\theta}) + \frac{\partial \boldsymbol{\phi}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}) \implies \tag{4.21}$$

$$\boldsymbol{\phi}(\boldsymbol{\theta}) \cong -\frac{\partial \boldsymbol{\phi}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}) \tag{4.22}$$

Note que $\frac{\partial \boldsymbol{\phi}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ representa a matriz hessiana de segundas derivadas de $F(\boldsymbol{\theta})$. Logo,

$$I(\boldsymbol{\theta}) = -\frac{\partial \boldsymbol{\phi}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{4.23}$$

é a pseudo matriz $b \times b$ de informação observada.

Assim, a matriz de covariância de $\hat{\boldsymbol{\theta}}$ pode ser escrita como

$$cov(\hat{\boldsymbol{\theta}}) = I(\hat{\boldsymbol{\theta}})^{-1} cov(\boldsymbol{\phi}(\hat{\boldsymbol{\theta}}))I(\hat{\boldsymbol{\theta}})^{-1} \tag{4.24}$$

em que $cov(\boldsymbol{\phi}(\boldsymbol{\theta}))$ é a matriz de covariância $b \times b$ de $\boldsymbol{\phi}(\boldsymbol{\theta})$ para dado $\boldsymbol{\theta}$ e $cov(\boldsymbol{\phi}(\hat{\boldsymbol{\theta}}))$ é o valor de $cov(\boldsymbol{\phi}(\boldsymbol{\theta}))$ aplicado em $\hat{\boldsymbol{\theta}}$.

Vieira (2005) mostra que $\phi_j(\boldsymbol{\theta})$ pode ser escrito como a razão de dois totais mais uma constante, que é dado por

$$\phi_j(\boldsymbol{\theta}) = tr\left[\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\frac{\partial \boldsymbol{\Sigma}(\boldsymbol{\theta})}{\partial \theta_j}\right] + \frac{\sum_{i=1}^{n} w_i z_{ji}}{\sum_{i=1}^{n} w_i} \tag{4.25}$$

em que

$$z_{ji} = -(\boldsymbol{y}_i - \hat{\boldsymbol{\mu}}_i)' \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\frac{\partial \boldsymbol{\Sigma}(\boldsymbol{\theta})}{\partial \theta_j}\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}(\boldsymbol{y}_i - \hat{\boldsymbol{\mu}}_i) \tag{4.26}$$

com $\hat{\boldsymbol{\mu}}_i = \boldsymbol{X}_i\hat{\boldsymbol{\beta}}(\boldsymbol{V})$.

Seja $\boldsymbol{a}_i = w_i\boldsymbol{z}_i$, $i = 1, \ldots, n$ um vetor $b \times 1$. Então, ignorando as constantes,

$$R_j = \frac{\mu_a}{\mu_w} = \frac{E(\overline{a}_j)}{E(\overline{w})} \tag{4.27}$$

em que

$$\overline{a}_j = \frac{1}{n}\sum_{i=1}^{n} a_{ji} \tag{4.28}$$

$$\overline{w} = \frac{1}{n}\sum_{i=1}^{n} w_i. \tag{4.29}$$

Note que é possível realizar uma expansão de Taylor em μ_a e μ_w. Assim, Vieira (2005) apresenta que

$$R_j \cong \frac{\mu_a}{\mu_w} + \frac{1}{n}\sum_{i=1}^{n}\left(a_{ji} - \frac{\mu_a}{\mu_w}w_i\right)\frac{1}{\mu_w}. \tag{4.30}$$

Logo, a variância de R_j pode ser calculada aproximadamente como

$$var(R_j) \cong var\left(\frac{1}{n}\sum_{i=1}^{n} u_{ji}\right) = \frac{1}{n^2}var\left(\sum_{i=1}^{n} u_{ji}\right) = \frac{1}{n^2}var(B_j) \tag{4.31}$$

em que

$$u_{ji} = \left(a_{ji} - \frac{\mu_a}{\mu_w}w_i\right)\frac{1}{\mu_w} \tag{4.32}$$

e

$$B_j = \sum_{i=1}^{n} u_{ji}. \tag{4.33}$$

Como μ_a e μ_w são parâmetros desconhecidos, substitui-se esses termos por seus respectivos estimadores, $\overline{a}_j$ e $\overline{w}$. Assim,

$$\hat{u}_{ji} = \left(a_{ji} - \frac{\overline{a}_j}{\overline{w}}w_i\right)\frac{1}{\overline{w}} \tag{4.34}$$

$$\hat{B}_j = \sum_{i=1}^{n} \hat{u}_{ji}. \tag{4.35}$$

Finalmente, seja $\hat{\boldsymbol{B}} = (\hat{B}_1, \ldots, \hat{B}_b)'$ um vetor $b \times 1$ de totais. Assim,

$$cov(\boldsymbol{\phi}(\hat{\boldsymbol{\theta}})) = \frac{1}{n^2}cov(\hat{\boldsymbol{B}}). \tag{4.36}$$

Substituindo na expressão (4.24) tem-se

$$cov(\hat{\boldsymbol{\theta}}) = I(\hat{\boldsymbol{\theta}})^{-1}\frac{1}{n^2}cov(\hat{\boldsymbol{B}})I(\hat{\boldsymbol{\theta}})^{-1}. \tag{4.37}$$

para qualquer plano amostral $p(\boldsymbol{s})$.

Para o exemplo que considera amostragem estratificada em múltiplos estágios e conglomeração apresentado nos capítulos anteriores, tem-se

$$cov_L[\hat{\boldsymbol{B}}] = \sum_{h=1}^{H}\frac{m_h}{m_h - 1}\left[\sum_{j=1}^{m_h}(\hat{\boldsymbol{B}}_{hj,j} - \overline{\hat{\boldsymbol{B}}}_{h,j})'(\hat{\boldsymbol{B}}_{hj,k} - \overline{\hat{\boldsymbol{B}}}_{h,k})\right]. \tag{4.38}$$

4.4.1 Proposta alternativa

Não será apresentado em detalhes neste trabalho outras formas de estimar a variância de $\hat{\boldsymbol{\theta}}$. Entretanto, Vieira (2005) apresenta ainda uma segunda alternativa que pode ser utilizada no caso em que F escolhida não foi a de máxima pseudo-verossimilhança. Neste caso,

$$acov(\hat{\boldsymbol{\theta}}) = \frac{1}{n}(\boldsymbol{\Delta}'\boldsymbol{U}^{-1}\boldsymbol{\Delta})^{-1}\boldsymbol{\Delta}'\boldsymbol{U}^{-1}\hat{\boldsymbol{C}}_c\boldsymbol{U}^{-1}\boldsymbol{\Delta}(\boldsymbol{\Delta}'\boldsymbol{U}^{-1}\boldsymbol{\Delta})^{-1} \tag{4.39}$$

em que $\hat{\boldsymbol{C}}_c = n \; . \; var(vech(\boldsymbol{S}_w))$,

$$\boldsymbol{\Delta} = \frac{\partial vech(\boldsymbol{\Sigma}(\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}} \tag{4.40}$$

e $\boldsymbol{U}$ é uma matriz $k \times k$ tal que

$$u_{tt',t''t'''} = \omega_{tt''}\omega_{t't'''} + \omega_{tt'''}\omega_{t't''} \tag{4.41}$$

em que $\omega_{tt'}$ representa um elemento qualquer de $\boldsymbol{W}$, que é qualquer estimador consistente de $\boldsymbol{\Sigma}$. Na prática, usa-se $\hat{\boldsymbol{\Sigma}} = \boldsymbol{W}$.

4.4.2 Incorporação da variância de $\hat{\boldsymbol{\theta}}$ ao pacote

Finalmente, o ajuste de $cov(\hat{\boldsymbol{\theta}})$ encerra o processo de estimação do pacote. Neste ponto, há apenas algumas questões a serem discutidas. Observe que a estimação de $cov(\hat{\boldsymbol{B}})$ é análoga ao que já foi realizado nas seções anteriores, ou seja, a função pode ser reaproveitada.

Entretanto, existe o problema do cálculo da matriz de informação observada. Isto significa dizer que é necessário calcular as segundas derivadas de $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ em relação a $\boldsymbol{\theta}$. Vieira (2005) sugere o uso da matriz de informação esperada, pois dessa maneira é possível simplificar as contas de forma que seja necessário derivar $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ apenas uma vez. Esta seria uma boa solução, mas o R possui uma função de calcula não somente a primeira derivada de uma expressão não avaliada mas também as segundas derivadas. Assim, dada qualquer estrutura de covariância pelo usuário, a função fornece todas as componentes necessárias para a estimação dos parâmetros.

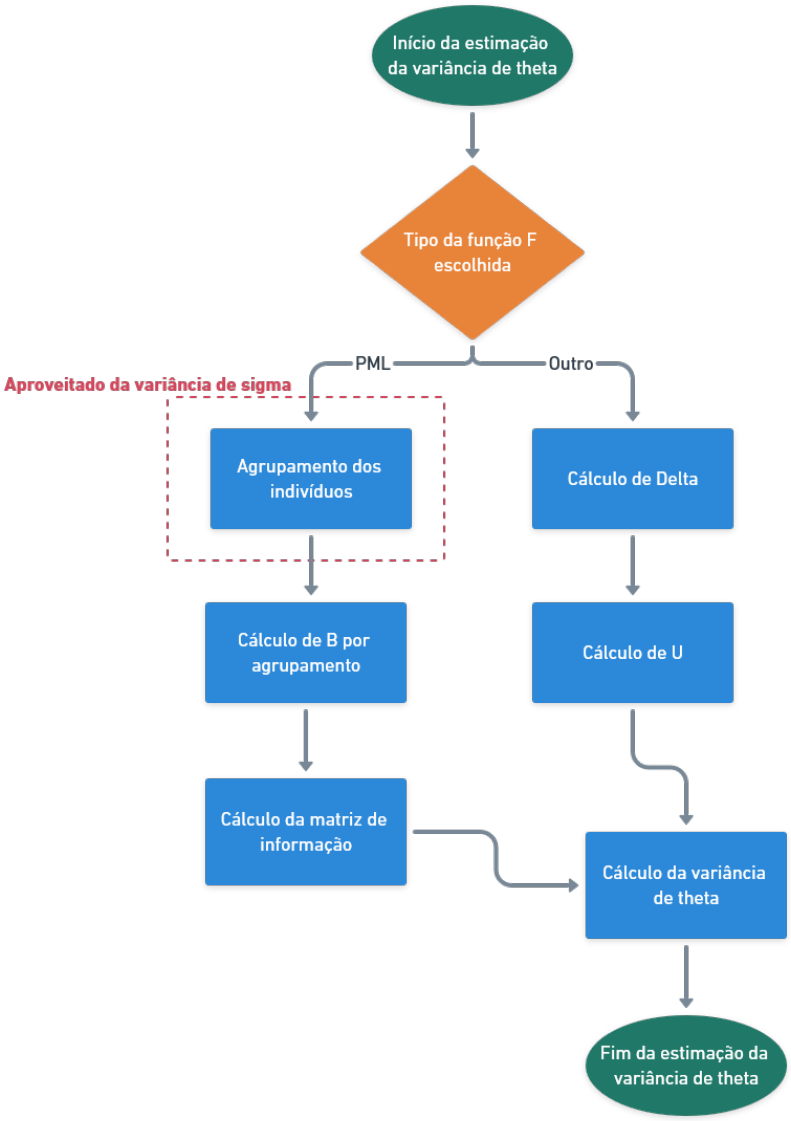
Já no caso da proposta alternativa, torna-se necessário estimar $\boldsymbol{\Delta}$ e $\boldsymbol{U}$. O primeiro também é facilmente calculado pelo mesmo argumento apresentado no parágrafo anterior e o segundo não resulta em grandes dificuldades, uma vez que o seu cálculo depende unicamente da matriz $\boldsymbol{S}_w$. Cabe ao programador apenas identificar qual a melhor forma de iterar de maneira eficiente os elementos que compõem esta matriz.

Quanto à apresentação dos resultados, é importante que a função "cov_clm" retorne de maneira visual para o usuário os valores estimados de $\hat{\boldsymbol{\theta}}$ com seus respectivos erros-padrão. Tendo esses valores, avalia-se a estrutura de covariância fornecida e também

apresenta a matriz $\boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}})$ resultante. Por último, como foi utilizado um processo de otimização, é interessante apresentar alguns resultados desta etapa, como por exemplo, se houve ou não convergência, qual o número de iterações, os métodos utilizados, valores iniciais, se houve ou não restrições no intervalo de busca de $\hat{\boldsymbol{\theta}}$ e também qual o valor da F calculado, pois quanto mais próximo de zero melhor é a estimativa. Todos os demais elementos utilizados pelas seções anteriores podem ser retornados internamente para caso o usuário queira utilizá-los posteriormente.

O diagrama abaixo representa o processo de estimação desta etapa.

Figura 8 - Fluxograma Estimação Variância de $\hat{\boldsymbol{\theta}}$

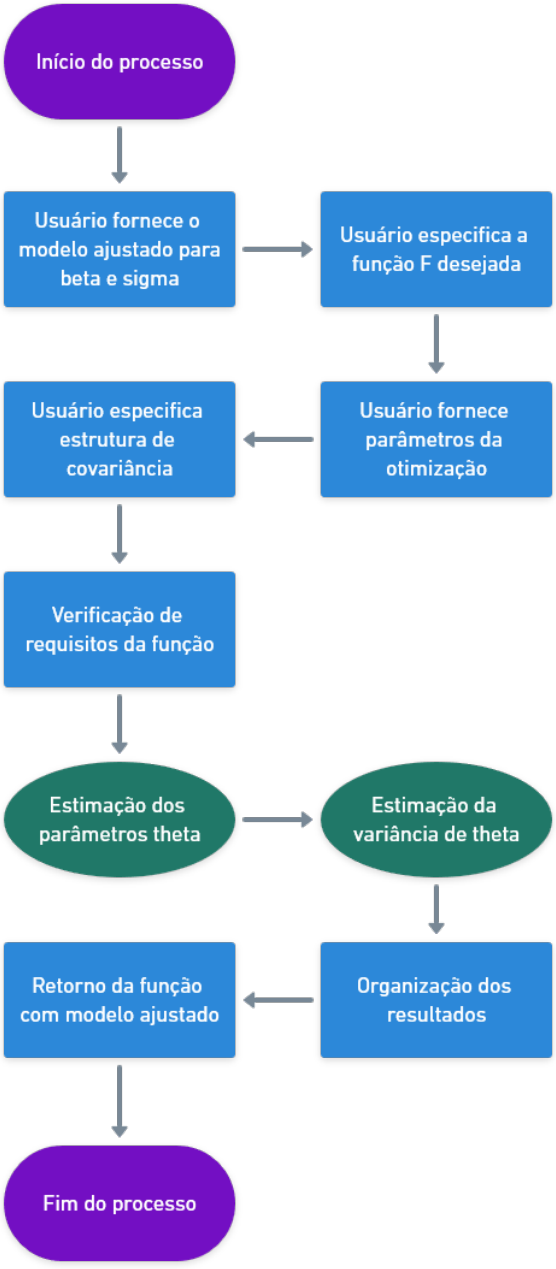


Fonte: Elaborado pelo autor. (2021).

Finalmente, o diagrama abaixo representa o processo completo da modelagem

considerando estruturas de covariância.

Figura 9 - Fluxograma Geral da função 'cov_clm'



Fonte: Elaborado pelo autor. (2021).

4.5 MEDIDAS DE QUALIDADE DE AJUSTE

O último conjunto de métodos desenvolvidos por Vieira (2005) inclui medidas de qualidade de ajuste para a classe de modelos apresentados no capítulo anterior. Diversos métodos foram apresentados, considerando ou não o plano amostral da pesquisa. Neste

sentido, serão apresentadas aqui duas dessas estatísticas neste trabalho levando em conta o desenho de amostragem. A incorporação das mesmas ao pacote será feita juntamente com a estimação dos parâmetros da covariância para que se consiga avaliar o ajuste de diferentes modelos. Pelo fato da implementação destas medidas de qualidade de ajuste serem relativamente simples de serem realizadas, não será detalhado aqui o processo computacional necessário para incorporar estas funções ao pacote.

4.5.1 Estatística RMR

De maneira geral, os testes desenvolvidos desejam avaliar o quão próximo $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ é de $\boldsymbol{\Sigma}$. Assim, considere o seguinte teste de hipóteses:

$$H_0 : \boldsymbol{\Sigma} = \boldsymbol{\Sigma}(\boldsymbol{\theta})$$
$$H_1 : \boldsymbol{\Sigma} \text{ é } \textit{uma matriz de covariância irrestrita.}$$

Seja $\boldsymbol{E}_p = \boldsymbol{\Sigma} - \boldsymbol{\Sigma}(\boldsymbol{\theta})$ a matriz $T \times T$ de resíduos da covariância populacional. Sob H_0, $\boldsymbol{E}_p = \mathbf{0}$. Considere ainda, a matriz de resíduos da covariância amostral, $\hat{\boldsymbol{E}}$, dada por

$$\hat{\boldsymbol{E}} = \boldsymbol{S}_w - \boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}}). \tag{4.42}$$

Note que $\hat{\boldsymbol{E}}$ pode ser utilizada para verificar quais elementos de $\boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}})$ não estão bem ajustados.

A primeira estatística apresentada por Vieira (2005) é chamada RMR, que diz respeito à raiz quadrada do erro quadrático médio. Assim,

$$RMR = \sqrt{2\sum_{t=1}^{T}\sum_{t'=1}^{t}\frac{\hat{E}_{tt'}^2}{T(T+1)}} \tag{4.43}$$

pode ser utilizada para comparar o ajuste de dois modelos diferentes para os mesmos dados. Observe que quanto mais próximo de 0 melhor é o ajuste.

4.5.2 Estatística AGFI

Uma outra estatística apresentada por Vieira (2005) foi a AGFI, que pode ser utilizada para comparar a qualidade do ajuste entre um modelo inicial e um modelo restrito. Esta estatística se baseia no trabalho de Joreskog e Sorbom (1989), levando em conta o plano amostral complexo. A ideia é propor inicialmente uma estatística GFI que calcula a magnitude relativa entre $\boldsymbol{S}_w$ e $\boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}})$. Como o segundo termo depende da função de ajuste adotada, a estatística GFI também dependerá.

$$GFI_{PML} = 1 - \left[\frac{tr((\boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}})^{-1}\boldsymbol{S}_w - \boldsymbol{I})^2)}{tr((\boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}})^{-1}\boldsymbol{S}_w)^2)}\right] \tag{4.44}$$

$$GFI_{ULSC} = 1 - \left[\frac{tr((\boldsymbol{S}_w - \boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}}))^2)}{tr((\boldsymbol{S}_w)^2)}\right] \tag{4.45}$$

$$GFI_{GLSC2} = 1 - \left[\frac{tr((\boldsymbol{I} - \boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}})\boldsymbol{S}_w^{-1})^2)}{T}\right] \tag{4.46}$$

$$GFI_{GLSC4} = 1 - \left[\frac{[vech(\boldsymbol{S}_w) - vech(\boldsymbol{\Sigma}(\boldsymbol{\theta}))]' \boldsymbol{U}^{-1} [vech(\boldsymbol{S}_w) - vech(\boldsymbol{\Sigma}(\boldsymbol{\theta}))]}{vech(\boldsymbol{S}_w)'\boldsymbol{U}^{-1}vech(\boldsymbol{S}_w)}\right] \tag{4.47}$$

$$\tag{4.48}$$

Em seguida, calcula-se a AGFI que penaliza a GFI com base no número de parâmetros do modelo.

$$AGFI = 1 - \frac{k}{df}(1 - GFI) \tag{4.49}$$

em que k é o tamanho da matriz $\boldsymbol{U}$ e df são os graus de liberdade do modelo. Note que quanto mais próximo $AGFI$ for de 1 melhor é o ajuste, pois quando $\boldsymbol{S}_w = \boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}})$ então $GFI = 0$.

5 RESULTADOS E APLICAÇÃO

Nos capítulos anteriores estivemos focados em apresentar não somente a metodologia proposta mas também realizar discussões a respeito da forma como isto poderia ser traduzido para termos computacionais. Neste sentido, desenvolvemos fluxogramas que serviram como guia quanto ao processo de construção do algoritmo necessário para a criação do pacote estatístico e realizamos alguns comentários sobre o processo de implementação do mesmo.

Apesar de tais discussões serem a base para se atingir o objetivo final, é necessário ainda apresentar os resultados obtidos a fim de concluir o processo de criação do pacote estatístico. Assim, neste capítulo, são apresentados alguns tópicos já destacados anteriormente, tendo em vista o resultado final do trabalho desenvolvido.

Vale destacar que até o presente momento o pacote se encontra disponível apenas na plataforma online github, mas espera-se em breve enviar uma submissão para o repositório oficial do Projeto R, denominado CRAN. Na seção 5.2 iremos mostrar como o usuário pode instalar e fazer uso do pacote. Além disso, sabendo que o processo de desenvolvimento de software é dinâmico, as soluções aqui apresentadas não devem ser vistas como definitivas mas sim como uma versão de um produto, de forma que esperamos desenvolver no futuro outras versões, que não somente corrijam eventuais erros de implementação mas também incorporem cada vez mais funcionalidades ao produto.

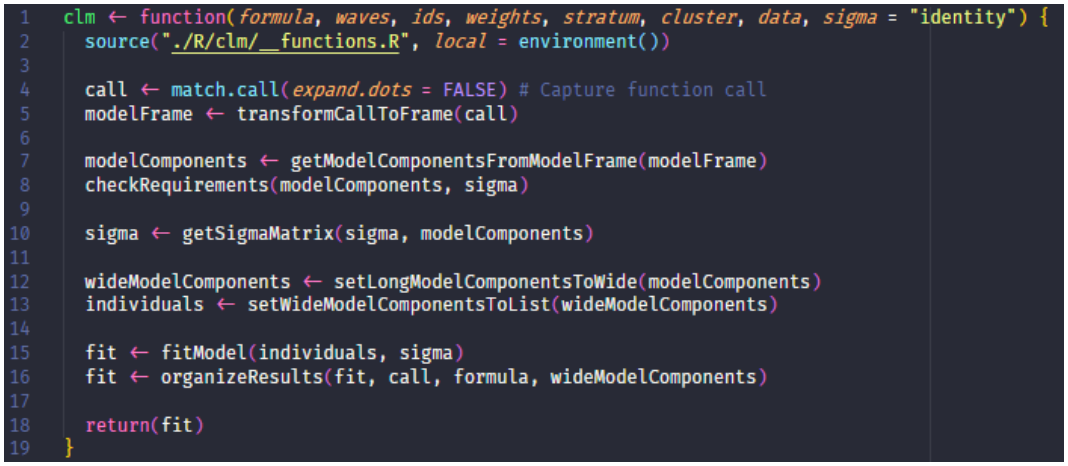
Ao final, realizaremos uma aplicação considerando uma pesquisa real como forma de exemplificação do uso do pacote 'clm', tendo em vista a verificação concreta das vantagens de se desenvolver um pacote para se conduzir um processo de modelagem estatística. A pesquisa é denominada *British Household Panel Survey* - BHPS (Vieira, 2012).

5.1 RESULTADOS DE IMPLEMENTAÇÃO

5.1.1 Função clm

A primeira grande função fruto deste pacote estatístico é a função 'clm', que busca estimar os efeitos fixos do modelo, também conhecidos como parâmetros $\boldsymbol{\beta}$ da regressão, levando em conta o plano amostral da pesquisa, e também estimar a matriz de covariância do modelo considerando as estimativas de $\hat{\boldsymbol{\beta}}$. A função 'clm' foi escrita buscando colocar em prática os princípios da programação funcional, apresentados no capítulo 2, sempre que possível.

Figura 10 - Código fonte da função 'clm'



```
1  clm ← function(formula, waves, ids, weights, stratum, cluster, data, sigma = "identity") {
2    source("./R/clm/__functions.R", local = environment())
3
4    call ← match.call(expand.dots = FALSE) # Capture function call
5    modelFrame ← transformCallToFrame(call)
6
7    modelComponents ← getModelComponentsFromModelFrame(modelFrame)
8    checkRequirements(modelComponents, sigma)
9
10   sigma ← getSigmaMatrix(sigma, modelComponents)
11
12   wideModelComponents ← setLongModelComponentsToWide(modelComponents)
13   individuals ← setWideModelComponentsToList(wideModelComponents)
14
15   fit ← fitModel(individuals, sigma)
16   fit ← organizeResults(fit, call, formula, wideModelComponents)
17
18   return(fit)
19 }
```

Fonte: Elaborado pelo autor. (2021).

Na primeira linha do código encontram-se os parâmetros de entrada e na linha 18 os parâmetros de saída da 'clm'. Quanto ao primeiro, note que o princípio de buscar não exceder três, no máximo 4, parâmetros por função foi respeitado pelas funções internas mas não foi completamente respeitado pela função principal. Conforme mostra a imagem acima, foram necessários 8 parâmetros de entrada, evidenciando que a programação da 'clm' pode melhorar. Por exemplo, os parâmetros *waves*, *ids*, *weights*, *stratum*, *cluster* e *data* podem ser agrupados como um único parâmetro que constituem as características da pesquisa, em especial do plano amostral.

Além desta possível melhoria, a função conseguiu incorporar alguns dos princípios do código limpo. Observe como é possível ler este código como um passo-a-passo a ser seguido de forma que o todo constitui uma história que reflete aos fluxogramas criados no capítulo 3:

> Dados os parâmetros de entrada, carregamos as funções que serão utilizadas de forma local, capturamos a chamada da função e transformamos esta chamada em uma moldura, que será a moldura do modelo. Por meio desta moldura, obtemos as componentes do modelo e verificamos se todos os pré-requisitos necessários estão sendo respeitados. Em seguida, obtemos a matriz de covariância sigma, transformamos as componentes do modelo, que estão no formato *long*, para o formato *wide* e transformamos em uma lista de indivíduos o resultado desta operação. Ajustamos o modelo tendo em vista esta lista de indivíduos e a matriz de covariância sigma. Finalmente, organizamos os resultados e retornamos o modelo ajustado.

Para tornar a análise mais clara, considere abaixo o código da mesma função antes

de ser refatorada, ou seja, antes de ter sido revisada e reescrita.

Figura 11 - Função 'clm' sem refatoração de código

```r
clm <- function(formula, waves, ids, weights, stratum, cluster, data, sigma = "identity", var_method = "linear", ...)
  require(tidyverse)
  require(rlist)

  call <- match.call()

  # REQUIREMENTS
  # ------------------------------------------------------------------------ #
  if (missing(formula)) stop("Invalid 'formula' argument. Please provide a valid formula!")
  if (missing(waves)) stop("Invalid 'waves' argument. Please provide a valid waves array!")
  if (missing(ids)) stop("Invalid 'ids' argument. Please provide a valid ids array!")

  if (is.character(sigma)) {
    if (sigma != "identity" & sigma != "exchangeable" & sigma != "autoregressive") {
      stop("Invalid 'sigma' argument. Please provide a valid sigma array!")
    }
  }

  if (var_method != "naive" & var_method != "linear" & var_method != "jackknife") {
    stop("Invalid 'var_method' argument. It should be naive, linear or jackknife!")
  }

  if (missing(data)) {
    data <- environment(formula)
  } else {
    data <- as_tibble(data)
  }

  mf <- match.call(expand.dots = FALSE)

  m <- match(c("formula", "waves", "ids", "weights", "stratum", "cluster", "data"), names(mf), 0L)

  mf <- mf[c(1L, m)]
  mf$drop.unused.levels <- TRUE
  mf$na.action <- "na.pass"

  mf[[1L]] <- quote(stats::model.frame)
  mf <- eval(mf, parent.frame())

  mt <- attr(mf, "terms")

  req <- requirements(formula, data, mf, mt, sigma)
  # ------------------------------------------------------------------------ #

  individual <- i_structure(req$Y, req$X, req$W, req$H, req$J)

  fit <- list()
  fit$aux <- list(
    N = attr(individual, "N"),
    T = attr(individual, "T"),
    q = attr(individual, "q")
  )
  fit$coefficients <- beta_pml(individual, req$sigma)
  fit$fitted <- do.call(rbind, lapply(individual, function(ind) t(ind$xi %*% fit$coefficients)))
  fit$residuals <- as.matrix(req$Y) - fit$fitted
  rownames(fit$fitted) <- rownames(fit$residuals) <- paste0("ind", seq(fit$aux$N))
  fit$sigma <- req$sigma
  rownames(fit$sigma) <- colnames(fit$sigma) <- paste0("T", seq(fit$aux$T))

  individual <- mapply(function(i, resi) within(i, resi <- as.matrix(unlist(resi))),
    individual, as.list(data.frame(t(fit$residuals))),
    SIMPLIFY = FALSE
  )
  attr(individual, "N") <- fit$aux$N
  attr(individual, "T") <- fit$aux$T
  attr(individual, "q") <- fit$aux$q

  if (var_method == "naive") {
    fit$se <- beta_se_naive(individual, fit, req$sigma)
  } else if (var_method == "linear") {
    fit$se <- beta_se_linear(individual, fit, req$sigma)
  } else if (var_method == "jackknife") {
    fit$se <- beta_se_jackknife(individual, fit, req$sigma)
  }

  result <- sigma_hat(individual, fit)
  individual <- result$individual

  fit$sigma_var <- sigma_var(individual, fit)
  fit$sigma_hat <- result$sigma_hat

  fit$call <- call
  fit$formula <- formula
  fit$model <- list(Y = req$Y, X = req$X, W = req$W, H = req$H, J = req$J)
  fit$individual <- individual

  class(fit) <- "clm"
  return(fit)
}
```

Fonte: Elaborado pelo autor. (2021).

Claramente não é possível realizar a mesma construção de uma história com esta segunda versão do código de forma simples como foi feito anteriormente. Nesta versão, vários princípios não estão sendo respeitados, a começar pelos nomes dos objetos. Nomes como 'mf', 'mt' e 'm' não fornecem informação alguma a respeito da sua constituição e isto dificulta a compreensão do código. Existe ainda o problema da existência de seções. Este é um problema tão evidente que para tentarmos dividir melhor o seu código, adicionamos comentários nas linhas 8 e 43 do código para tentar separar as diferentes seções da função, mas a solução não foi tão elegante, nem eficiente.

Observe ainda que em alguns momentos a função usou o princípio de decomposição funcional mas em outros não. Na seção de requerimentos, por exemplo, primeiramente são feitos vários testes verificando se alguns parâmetros de entrada existem dentro da função, mas depois, na linha 42, ainda é chamada uma função que faz a checagem de requerimentos. Ora, em termos de leitura do código isto fere o princípio de leitura sequencial começando de cima para baixo por meio da construção de uma história.

Além disso, a função *requirements* retorna um objeto que posteriormente é bastante utilizado pelo restante do código e isto é bastante confuso, pois o seu nome não deixa clara a sua finalidade. Se o objetivo é checar se os requisitos estão sendo respeitados, então, ela não deveria retornar nada além de verdadeiro ou falso. Se o seu objetivo é retornar os requisitos da função, então resta uma dúvida sobre quais são esses requisitos, se não os próprios parâmetros de entrada da função.

A segunda parte do código é ainda mais confusa. São linhas e mais linhas de código com nomes pouco intuitivos e sem identificar o que cada parte realiza, o que torna a compreensão do código bastante complexa. Vale ressaltar que ambos os códigos apresentados funcionam e retornam o modelo ajustado mas a primeira opção apresentada é muito superior à segunda no que diz respeito ao cumprimento dos princípios apresentados no capítulo 2.

5.1.2 Função cov_clm

Apresentamos agora os resultados para a função 'cov_clm', que constitui a segunda grande função do pacote desenvolvido. Esta função é responsável por realizar a modelagem da matriz de covariância do modelo por meio do uso de estruturas de covariância.

Figura 12 - Código fonte da função 'cov_clm'

```r
cov_clm <- function(fit, fittingType, sigmaThetaExpr, optimParams) {
  source("./R/cov_clm/__functions.R", local = environment())

  call <- match.call() # Capture function call
  checkRequirements(fit, fittingType, sigmaThetaExpr, optimParams)

  sigmaThetaExprList <- getSigmaThetaExprList(sigmaThetaExpr, fit)
  fittingFunction <- getFittingFunction(fittingType, sigmaThetaExprList, fit)

  derivativesExprList <- getDerivativesExprList(sigmaThetaExprList)
  gradientFunction <- getGradientFunction(derivativesExprList, sigmaThetaExprList, fit, fittingType)

  optimParams <- addConfigToOptimParams(optimParams, fittingFunction, gradientFunction, sigmaThetaExprList)

  fitTheta <- fitThetaModel(optimParams, sigmaThetaExprList, derivativesExprList, fittingType, fit)
  gofMeasures <- getGoodnessOfFit(fitTheta, fit, fittingType)

  fitTheta <- organizeFitTheta(fitTheta, gofMeasures, call, sigmaThetaExpr)

  return(fitTheta)
}
```

Fonte: Elaborado pelo autor. (2021).

Como a própria metodologia para modelagem da matriz de covariância é mais complexa que a modelagem dos efeitos fixos do modelo, naturalmente a programação também será. Assim, desenvolver a função respeitando os princípios do código limpo torna-se um trabalho ainda mais fundamental. O passo-a-passo da função pode ser expresso como:

> Recebemos como parâmetros de entrada um modelo ajustado, um tipo de ajuste a ser utilizado, uma expressão de sigma theta e os parâmetros da optim. Carregamos as funções que serão utilizadas de forma local, capturamos a chamada da função e verificamos se todos os pré-requisitos necessários estão sendo respeitados. Por meio da expressão de sigma theta fornecida obtemos uma lista de expressões de sigma theta e com ela, o modelo ajustado e o tipo de ajuste, obtemos a função de ajuste a ser utilizada. Por meio da lista de expressões de sigma theta, obtemos uma lista de expressões de derivadas e através de um processo análogo ao anterior, obtemos a função gradiente. Em seguida, adicionamos estes objetos aos parâmetros da optim como forma de configuração e realizamos o ajuste do modelo de theta. Calculamos algumas medidas de qualidade de ajuste. Finalmente, organizamos os resultados e retornamos o modelo ajustado.

Analise agora a mesma função antes de ser refatorada.

Figura 13 - Função 'cov_clm' sem refatoração de código

```r
cov_clm <- function(fit, start_value, cov_type = "UCM", fitting_type = "PML", calc_type = "PML",
  method = "L-BFGS-B", ...) {
  call <- match.call()

  if (!is(fit, "clm")) stop("Invalid 'fit' argument. Please provide a valid fit argument!")

  COV_FUN <- cov_fun_check_eval(cov_type)
  params_names <- formalArgs(COV_FUN)

  fitting_optim <- readRDS(paste0("simulation_pedro/theta_fun_t_", fit$aux$T, "/", cov_type, "_",
    fitting_type, ".RDS"))

  Sw_values <- list()
  for (i in seq(fit$aux$T)) {
    for (j in seq(fit$aux$T)) {
      Sw_values[[paste0("a", i, j)]] <- fit$sigma_hat[i, j]
    }
  }

  const_values <- Sw_values

  if (fitting_type == "GLSC4") {
    C_hat_solve_matrix <- solve(fit$aux$N * fit$sigma_var)

    C_hat_solve_values <- list()
    for (i in seq((fit$aux$T^2 + fit$aux$T) / 2)) {
      for (j in seq((fit$aux$T^2 + fit$aux$T) / 2)) {
        C_hat_solve_values[[paste0("b", i, j)]] <- C_hat_solve_matrix[i, j]
      }
    }

    const_values <- c(const_values, C_hat_solve_values)
  }

  theta <- optim(start_value, theta_expr_eval, params_names = params_names, const_values = const_values,
    expr = fitting_optim, method = method, ...)

  names(theta$par) <- params_names

  environment(COV_FUN) <- environment()
  sigma_theta <- do.call(COV_FUN, as.list(theta$par))

  sigma_theta <- list(
    theta = theta, matrix = sigma_theta,
    fitting_type = fitting_type, cov_type = cov_type, params = params_names, start_value = start_value
  )

  sigma_theta$deriv <- deriv_sigma_theta(sigma_theta)

  sigma_theta$U <- U_estim(fit, sigma_theta)

  sigma_theta$theta_var <- theta_var(sigma_theta, fit, calc_type)
  sigma_theta$call <- call
  sigma_theta$calc_type <- calc_type
  sigma_theta$method <- method

  dots <- list(...)

  if (!is.null(dots[["lower"]])) {
    sigma_theta$lower <- dots$lower
  }
  if (!is.null(dots[["upper"]])) {
    sigma_theta$upper <- dots$upper
  }

  if (fitting_type == "PML") {
    GFI_val <- GFI_PML(fit, sigma_theta)
  } else if (fitting_type == "ULSC") {
    GFI_val <- GFI_ULSC(fit, sigma_theta)
  } else if (fitting_type == "GLSC2" | fitting_type == "GLSC3" | fitting_type == "GLSC4") {
    GFI_val <- GFI_GLSC(fit, sigma_theta)
  }

  sigma_theta$gof <- list(
    RMR = RMR(fit, sigma_theta),
    AGFI = AGFI(GFI_val, fit$aux$T * (fit$aux$T + 1) / 2, length(sigma_theta$theta$par))
  )

  class(sigma_theta) <- "clm_sigma_theta"
  return(sigma_theta)
}
```

Fonte: Elaborado pelo autor. (2021).

Compreender o que esta função realiza considerando o código sem refatorar é uma tarefa bastante complicada. Além de todos os problemas apresentados na seção anterior,

neste caso, por causa da complexidade do ajuste do modelo ser muito maior, desenvolver métodos eficientes e generalizados para estimação dos parâmetros considerando o código anterior é desafiador.

O código tem um papel tão importante no desenvolvimento do pacote que foi apenas depois de refatorá-lo que conseguimos implementar uma maneira de estimar os parâmetros $\boldsymbol{\theta}$ de forma eficiente e generalizada por meio do uso de métodos de otimização.

5.1.3 Função sigmaThetaExpr_viewer

Utilizando expressões no R para calcular as primeiras e segundas derivadas analíticas de uma dada estrutura da matriz de covariância e também por meio do conceito de fábrica de funções para criar a função de ajuste e também a função gradiente, conseguimos construir uma forma generalizada de especificação da estrutura de covariância, permitindo que o usuário seja capaz de utilizar qualquer método já implementado no pacote ou então especificar uma estrutura de interesse qualquer e ainda sim a função 'cov_clm' será capaz de estimar os parâmetros, cabendo ao usuário apenas tomar os devidos cuidados para não encontrar mínimos locais por causa dos valores iniciais escolhidos.

Entretanto, a própria visualização da estrutura de covariância pode ser uma tarefa complicada, principalmente quando o usuário escolhe determinar a sua própria estrutura. Assim, desenvolvemos a função 'sigmaThetaExpr_viewer' que permite o usuário visualizar a estrutura fornecida antes mesmo dela ser avaliada, ou seja, por meio de matemática simbólica. A seção 5.3 trará exemplos mais concretos a respeito dessa função e as vantagens ao usá-la, mas como forma de dar uma noção geral, segue abaixo o resultado dessa função para uma estrutura de covariância AR1 para a situação de um estudo longitudinal com 5 rodadas.

Figura 14 - Exemplo função 'sigmaThetaExpr_viewer' AR1 com 5 rodadas

```
> # AR1
  sigmaThetaExpr_viewer("AR1", 5)


Table: sigma2u sigma2v gamma

|   |                T1                |                T2                |                T3                |                T4                |                T5                |
|:--|:--------------------------------:|:--------------------------------:|:--------------------------------:|:--------------------------------:|:--------------------------------:|
|T1 |          sigma2u + sigma2v       | sigma2u + gamma^1 * sigma2v      | sigma2u + gamma^2 * sigma2v      | sigma2u + gamma^3 * sigma2v      | sigma2u + gamma^4 * sigma2v      |
|T2 | sigma2u + gamma^1 * sigma2v      |          sigma2u + sigma2v       | sigma2u + gamma^1 * sigma2v      | sigma2u + gamma^2 * sigma2v      | sigma2u + gamma^3 * sigma2v      |
|T3 | sigma2u + gamma^2 * sigma2v      | sigma2u + gamma^1 * sigma2v      |          sigma2u + sigma2v       | sigma2u + gamma^1 * sigma2v      | sigma2u + gamma^2 * sigma2v      |
|T4 | sigma2u + gamma^3 * sigma2v      | sigma2u + gamma^2 * sigma2v      | sigma2u + gamma^1 * sigma2v      |          sigma2u + sigma2v       | sigma2u + gamma^1 * sigma2v      |
|T5 | sigma2u + gamma^4 * sigma2v      | sigma2u + gamma^3 * sigma2v      | sigma2u + gamma^2 * sigma2v      | sigma2u + gamma^1 * sigma2v      |          sigma2u + sigma2v       |
```

Fonte: Elaborado pelo autor. (2021).

Pelo exemplo acima, vemos que existem três parâmetros a serem estimados nesta estrutura, são eles: 'sigma2u', 'sigma2v' e 'gamma'.

5.2 INSTALAÇÃO DO PACOTE

Existem duas formas de se instalar um pacote no R. A primeira e mais comumente utilizada é por meio do comando 'install.packages()' que utiliza o repositório oficial do Projeto R, CRAN, como sendo a fonte de busca para a instalação. Preferencialmente esta é a melhor opção pois é mais simples e também conta com a aprovação da equipe de gerenciamento do CRAN. Entretanto, adicionar o pacote a este repositório envolve tempo e dedicação, uma vez que a equipe de revisão do Projeto R faz uma série de exigências quanto ao formato que o pacote deve estar e também deve incluir manual explicando o uso de cada função, entre outras questões.

Na hipótese de não ser possível finalizar todos estes procedimentos, existe uma segunda forma de se instalar um pacote no R que é com o pacote 'devtools'. Por meio dele, o usuário deve utilizar o comando 'install_github()' para baixar e instalar o pacote que está armazenado na plataforma github. Para o caso específico deste pacote que está sendo desenvolvido, o usuário deve fornecer como parâmetro da função anterior o nome do repositório em que está armazenado o pacote 'clm' que é 'phmpacheco-ufjf/clm'.

Feito isto, basta utilizar o comando 'library(clm)' para começar a utilizar as funções.

5.3 APLICAÇÃO

5.3.1 Banco de dados

Como forma de exemplificar o funcionamento do pacote, iremos analisar dados da pesquisa domiciliar inglesa *British Household Panel Survey* - BHPS. A BHPS é uma pesquisa domiciliar longitudinal do tipo painel e teve sua primeira rodada em 1991. A amostra foi selecionada por um desenho amostral estratificado em dois estágios com conglomeração por setores postais. Esses setores postais foram selecionados de forma sistemática e considerando probabilidades de seleção proporcionais ao seu tamanho, ou seja, número de domicílios em um mesmo setor postal. Em seguida, em cada um dos setores postais selecionados, foram escolhidos três domicílios de forma aleatória e dentro de cada domicílio todos os indivíduos foram pesquisados (Vieira, 2012).

O banco de dados a ser explorado possui um subconjunto, o qual estamos interessados em analisar, que diz respeito a mudanças nas atitudes em relação ao papel do gênero. As cinco rodadas da pesquisa analisadas são 1991, 1993, 1995, 1997 e 1999. Além disso, o número de indivíduos respondentes constituem um total de 1340 pessoas Vieira (2012).

5.3.2 Estimação dos efeitos fixos e da matriz de covariância

Iremos estimar os efeitos fixos do modelo considerando a variável 'score', que quantifica a perspectiva das mulheres entrevistadas quanto a mudanças nas atitudes em relação ao papel do gênero, como sendo a variável resposta e as variáveis 'wave',

'ageg', 'ecacg' e 'qualifg' como sendo as variáveis explicativas. A tabela abaixo descreve o significado de cada uma destas variáveis.

Tabela 3 – Tabela de descrição das variáveis do banco de dados

Variável	Nome	Valores	Descrição
score	Escore	Numérico	Escore de atitude
wave	Rodada	1	Rodada 1
		3	Rodada 3
		5	Rodada 5
		7	Rodada 7
		9	Rodada 9
ageg	Idade	1	16-21 anos
		2	22-27 anos
		3	28-33 anos
		4	Mais de 34 anos
ecacg	Nível Educacional	1	Curso superior ou mais
		2	Magistério
		3	Ensino Médio
		4	Ensino Fundamental
		5	Outras
qualifg	Atividade Econômica	1	Emprego em tempo integral
		2	Emprego em tempo parcial
		3	Inativa
		4	Estudante em tempo integral
		5	Dona de casa

Fonte: Elaborado pelo autor (2021).

Estaremos considerando os pesos amostrais da pesquisa, uma variável de estratificação e também uma de conglomeração. A matriz de correlação de trabalho escolhida foi a permutável. Segue abaixo o resultado do ajuste do modelo.

Figura 15 - Ajuste do modelo pela função 'clm'

```
> summary(fit)
Call:
clm(formula = score ~ wave + ageg + ecacg + qualifg, waves = wave,
    ids = id, weights = weight, stratum = strata, cluster = cluster,
    data = data0, sigma = "exchangeable")

Coefficients:

             Estimate   Std. Error    t value   P(t>|t|)
------------  ---------  -----------  ---------  ---------  ----
(Intercept)      22.29         0.30    74.0021     0.0000  ***
wave             -0.04         0.02    -2.7355     0.0062  **
ageg2            -0.65         0.21    -3.0434     0.0023  **
ageg3            -0.82         0.24    -3.4073     0.0007  ***
ageg4            -1.01         0.28    -3.6672     0.0002  ***
ecacg2           -0.87         0.11    -8.0011     0.0000  ***
ecacg3           -0.76         0.17    -4.4817     0.0000  ***
ecacg4            0.21         0.24     0.8678     0.3855
ecacg5           -2.03         0.14   -14.2889     0.0000  ***
qualifg2         -0.52         0.25    -2.0632     0.0391  *
qualifg3         -0.66         0.26    -2.5131     0.0120  *
qualifg4         -0.50         0.25    -2.0176     0.0436  *
qualifg5         -1.22         0.26    -4.6979     0.0000  ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of individuals: 1340
Number of waves: 5

Sampling Design:
   Sample Weights: Yes
   Stratification: Yes
   Clusterization: Yes

Covariance Matrix Provided:
   Type: exchangeable
   rho: 0.567

        T1     T2     T3     T4     T5
---  ------  ------  ------  ------  ------
T1    1.000   0.567   0.567   0.567   0.567
T2    0.567   1.000   0.567   0.567   0.567
T3    0.567   0.567   1.000   0.567   0.567
T4    0.567   0.567   0.567   1.000   0.567
T5    0.567   0.567   0.567   0.567   1.000
---

Sigma Estimated:

          T1       T3       T5       T7       T9
---  --------  --------  --------  --------  --------
T1    13.4627   7.6690   7.1577   6.3445   6.0808
T3     7.6690  11.5988   7.8561   6.5722   6.5063
T5     7.1577   7.8561  12.5644   7.9730   7.4928
T7     6.3445   6.5722   7.9730  11.9087   7.9353
T9     6.0808   6.5063   7.4928   7.9353  11.9837
---
```

Fonte: Elaborado pelo autor. (2021).

Como o objetivo deste trabalho é discutir a respeito da construção do pacote, não estaremos interessados em fornecer uma interpretação detalhada para o ajuste do modelo no que diz respeito ao banco de dados em questão. Para maiores informações sobre interpretações dos modelos ajustados neste trabalho, sugerimos consultar Vieira

(2005). Neste sentido, limitaremos nossa análise a afirmar que, com exceção da variável 'ecacg4', todas as demais estimativas foram significativas ao nível de 5%.

Ao observar os resultados do ajuste, é natural considerar que o analista possua o interesse em fazer uma seleção de variáveis explicativas do modelo. Critérios como AIC e BIC são frequentemente utilizados para realizar esta tarefa. No R, por exemplo, a função 'stepAIC' aplica seleção *backward*, *forward* ou *stepwise* para modelos oriundos do ajuste de modelos lineares generalizados.

Uma possível implementação para o pacote no futuro é adaptar os tradicionais métodos de seleção de variáveis apresentados no parágrafo anterior para os modelos longitudinais considerando o plano amostral complexo e também características da inferência causal.

Mediante os comentários apresentados no capítulo 3, vemos que a organização visual dos resultados foi pensada de forma que fosse não somente completa, mas, elegante esteticamente. A clareza é importante pois facilita para o usuário interpretar os resultados do ajuste sem se perder em tantos números e a completude é essencial pois sem ela a interpretação fica comprometida.

5.3.3 Estimação dos parâmetros dos efeitos aleatórios

Tendo estimado os parâmetros $\boldsymbol{\beta}$ e $\boldsymbol{\Sigma}$, resta agora estimar os parâmetros $\boldsymbol{\theta}$ de $\boldsymbol{\Sigma}(\boldsymbol{\theta})$. Vimos nos capítulos anteriores que a estimação de $\boldsymbol{\theta}$ depende diretamente a escolha da estrutura da matriz de covariância $\boldsymbol{\Sigma}(\boldsymbol{\theta})$. Até a conclusão deste trabalho, o pacote contava com as seguintes estruturas implementadas internamente: 'UCM', 'AR1', 'ARMA11', 'ARH1', 'Toeplitz', 'Heterogeneous UCM', 'Heterogeneous Toeplitz', 'Predependence Order 1', 'Unstructured'. Isto significa dizer que para o usuário utilizá-las, basta especificar o seu nome como valor do parâmetro de entrada 'sigmaThetaExpr'. Estas estruturas de covariância foram implementadas tendo como referência o trabalho realizado por Singer, Nobre e Rocha (2018) a respeito de modelos mistos considerando amostragem aleatória simples.

Como o objetivo aqui é apenas apresentar as funcionalidades, iremos realizar o ajuste de algumas destas estruturas, estando o ajuste das demais no apêndice A. Além disso, não iremos nos preocupar com o valor inicial fornecido ao pacote para estimação dos parâmetros, destacando que para uma real interpretação dos resultados é necessário uma análise mais minuciosa a respeito do ajuste. Apesar disso, veremos que a maior parte dos ajustes obtiveram bons resultados, indicando que a função desenvolvida possui bom desempenho. Vale destacar que os resultados obtidos foram compatíveis com os obtidos por Vieira (2005).

Em todas as estruturas consideradas iremos primeiro apresentar a estrutura por meio da função 'sigmaThetaExpr_viewer' e posteriormente iremos apresentar os resultados

do ajuste.

5.3.3.1 UCM

Começamos pelo ajuste mais simples.

Figura 16 - 'sigmaThetaExpr_viewer' para UCM com 5 rodadas

```
> # UCM
  sigmaThetaExpr_viewer("UCM", 5)


Table: sigma2u sigma2v

|    |        T1        |        T2        |        T3        |        T4        |        T5        |
|:---|:----------------:|:----------------:|:----------------:|:----------------:|:----------------:|
|T1  | sigma2u + sigma2v |      sigma2u     |      sigma2u     |      sigma2u     |      sigma2u     |
|T2  |      sigma2u     | sigma2u + sigma2v |      sigma2u     |      sigma2u     |      sigma2u     |
|T3  |      sigma2u     |      sigma2u     | sigma2u + sigma2v |      sigma2u     |      sigma2u     |
|T4  |      sigma2u     |      sigma2u     |      sigma2u     | sigma2u + sigma2v |      sigma2u     |
|T5  |      sigma2u     |      sigma2u     |      sigma2u     |      sigma2u     | sigma2u + sigma2v |
```

Fonte: Elaborado pelo autor. (2021).

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_u^2 + \sigma_v^2 & \sigma_u^2 & \dots & \sigma_u^2 \\ \sigma_u^2 & \sigma_u^2 + \sigma_v^2 & \dots & \sigma_u^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_u^2 & \sigma_u^2 & \dots & \sigma_u^2 + \sigma_v^2 \end{bmatrix} \tag{5.1}$$

Figura 17 - Ajuste modelo para UCM com 5 rodadas

```
> summary(fitTheta)
Call:
cov_clm(fit = fit, fittingType = "PML", sigmaThetaExpr = "UCM",
    optimParams = list(par = c(7, 5)))

Fitting Function Type: PML
Sigma Theta Type: UCM

Covariance Params: sigma2u sigma2v

Coefficients:

          Estimate   Std. Error
--------  ---------  -----------
sigma2u      7.1588       0.3880
sigma2v      5.1449       0.1532
---

Optimization Details:

Converged: TRUE
Optim Method: L-BFGS-B
Iteractions Number - Function: 8
Iteractions Number - Gradient: 8
Optim Function Value: 0.1641

          Start Value  Lower   Upper
--------  -----------  ------  ------
sigma2u             7  -       -
sigma2v             5  -       -


Sigma Theta Matrix:

--------  --------  --------  --------  --------
 12.3037    7.1588    7.1588    7.1588    7.1588
  7.1588   12.3037    7.1588    7.1588    7.1588
  7.1588    7.1588   12.3037    7.1588    7.1588
  7.1588    7.1588    7.1588   12.3037    7.1588
  7.1588    7.1588    7.1588    7.1588   12.3037
--------  --------  --------  --------  --------


Goodness of Fit Measures:

RMR: 0.8796
AGFI: 0.9225
```

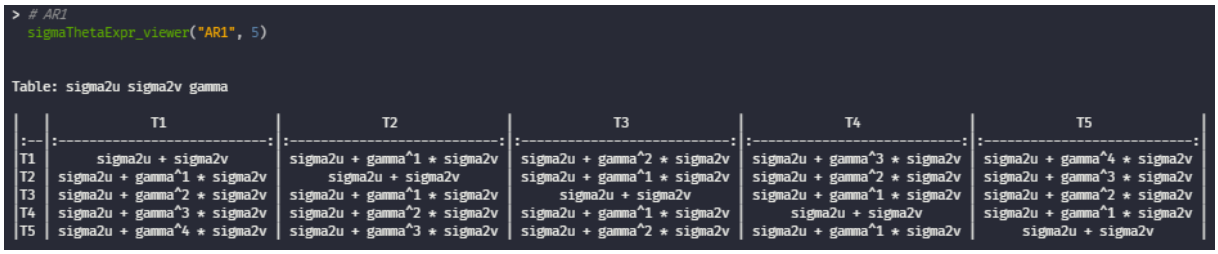
Fonte: Elaborado pelo autor. (2021).

No caso acima, a função de ajuste foi do tipo PML, o processo de otimização atingiu convergência com 8 iterações. O valor da $F(\hat{\boldsymbol{\theta}})$ foi 0.1641, o que é um valor um pouco distante de zero, indicando que possivelmente essa estrutura não é a mais adequada para estes dados. As medidas de qualidade de ajuste serão utilizadas como critério de comparação entre os modelos. RMR próximo 0 e AGFI próximo de 1 indicam modelos com melhores ajuste, lembrando que a RMR considera apenas o resíduo enquanto que a AGFI penaliza o modelo pelo seu número de parâmetros.

5.3.3.2 AR1

Consideramos a estrutura autorregressiva de primeira ordem.

Figura 18 - 'sigmaThetaExpr_viewer' para AR1 com 5 rodadas



```
> # AR1
  sigmaThetaExpr_viewer("AR1", 5)


Table: sigma2u sigma2v gamma
```

	T1	T2	T3	T4	T5
T1	sigma2u + sigma2v	sigma2u + gamma^1 * sigma2v	sigma2u + gamma^2 * sigma2v	sigma2u + gamma^3 * sigma2v	sigma2u + gamma^4 * sigma2v
T2	sigma2u + gamma^1 * sigma2v	sigma2u + sigma2v	sigma2u + gamma^1 * sigma2v	sigma2u + gamma^2 * sigma2v	sigma2u + gamma^3 * sigma2v
T3	sigma2u + gamma^2 * sigma2v	sigma2u + gamma^1 * sigma2v	sigma2u + sigma2v	sigma2u + gamma^1 * sigma2v	sigma2u + gamma^2 * sigma2v
T4	sigma2u + gamma^3 * sigma2v	sigma2u + gamma^2 * sigma2v	sigma2u + gamma^1 * sigma2v	sigma2u + sigma2v	sigma2u + gamma^1 * sigma2v
T5	sigma2u + gamma^4 * sigma2v	sigma2u + gamma^3 * sigma2v	sigma2u + gamma^2 * sigma2v	sigma2u + gamma^1 * sigma2v	sigma2u + sigma2v

Fonte: Elaborado pelo autor. (2021).

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_u^2 + \sigma_v^2 & \sigma_u^2 + \gamma^1\sigma_v^2 & \dots & \sigma_u^2 + \gamma^4\sigma_v^2 \\ \sigma_u^2 + \gamma^1\sigma_v^2 & \sigma_u^2 + \sigma_v^2 & \dots & \sigma_u^2 + \gamma^3\sigma_v^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_u^2 + \gamma^4\sigma_v^2 & \sigma_u^2 + \gamma^3\sigma_v^2 & \dots & \sigma_u^2 + \sigma_v^2 \end{bmatrix} \tag{5.2}$$

Figura 19 - Ajuste modelo para AR1 com 5 rodadas

```
> summary(fitTheta)
Call:
cov_clm(fit = fit, fittingType = "PML", sigmaThetaExpr = "AR1",
    optimParams = list(par = c(7, 5, 0.5)))

Fitting Function Type: PML
Sigma Theta Type: AR1

Covariance Params: sigma2u sigma2v gamma

Coefficients:

          Estimate   Std. Error
--------  ---------  -----------
sigma2u      6.4384       0.4054
sigma2v      5.8851       0.2181
gamma        0.2544       0.0247
---

Optimization Details:

Converged: TRUE
Optim Method: L-BFGS-B
Iteractions Number - Function: 25
Iteractions Number - Gradient: 25
Optim Function Value: 0.0411

          Start Value  Lower   Upper
--------  -----------  ------  ------
sigma2u           7.0  -       -
sigma2v           5.0  -       -
gamma             0.5  -       -


Sigma Theta Matrix:

--------  --------  --------  --------  --------
 12.3234    7.9354    6.8192    6.5352    6.4630
  7.9354   12.3234    7.9354    6.8192    6.5352
  6.8192    7.9354   12.3234    7.9354    6.8192
  6.5352    6.8192    7.9354   12.3234    7.9354
  6.4630    6.5352    6.8192    7.9354   12.3234
--------  --------  --------  --------  --------


Goodness of Fit Measures:

RMR: 0.5132
AGFI: 0.9791
```

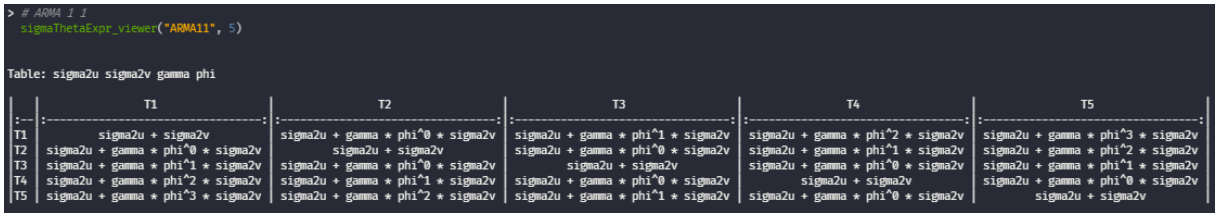
Fonte: Elaborado pelo autor. (2021).

Claramente o ajuste já foi melhor que o UCM.

5.3.3.3 ARMA11

Consideramos a estrutura ARMA 1 1.

Figura 20 - 'sigmaThetaExpr_viewer' para ARMA11 com 5 rodadas



```
> # ARMA 1 1
  sigmaThetaExpr_viewer("ARMA11", 5)


Table: sigma2u sigma2v gamma phi

|    | T1                                | T2                                | T3                                | T4                                | T5                                |
|:---|:----------------------------------|:----------------------------------|:----------------------------------|:----------------------------------|:----------------------------------|
| T1 |          sigma2u + sigma2v        | sigma2u + gamma * phi^0 * sigma2v | sigma2u + gamma * phi^1 * sigma2v | sigma2u + gamma * phi^2 * sigma2v | sigma2u + gamma * phi^3 * sigma2v |
| T2 | sigma2u + gamma * phi^0 * sigma2v |          sigma2u + sigma2v        | sigma2u + gamma * phi^0 * sigma2v | sigma2u + gamma * phi^1 * sigma2v | sigma2u + gamma * phi^2 * sigma2v |
| T3 | sigma2u + gamma * phi^1 * sigma2v | sigma2u + gamma * phi^0 * sigma2v |          sigma2u + sigma2v        | sigma2u + gamma * phi^0 * sigma2v | sigma2u + gamma * phi^1 * sigma2v |
| T4 | sigma2u + gamma * phi^2 * sigma2v | sigma2u + gamma * phi^1 * sigma2v | sigma2u + gamma * phi^0 * sigma2v |          sigma2u + sigma2v        | sigma2u + gamma * phi^0 * sigma2v |
| T5 | sigma2u + gamma * phi^3 * sigma2v | sigma2u + gamma * phi^2 * sigma2v | sigma2u + gamma * phi^1 * sigma2v | sigma2u + gamma * phi^0 * sigma2v |          sigma2u + sigma2v        |
```

Fonte: Elaborado pelo autor. (2021).

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_u^2 + \sigma_v^2 & \sigma_u^2 + \gamma\sigma_v^2 & \dots & \sigma_u^2 + \gamma\phi^3\sigma_v^2 \\ \sigma_u^2 + \gamma\sigma_v^2 & \sigma_u^2 + \sigma_v^2 & \dots & \sigma_u^2 + \gamma\phi^2\sigma_v^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_u^2 + \gamma\phi^3\sigma_v^2 & \sigma_u^2 + \gamma\phi^2\sigma_v^2 & \dots & \sigma_u^2 + \sigma_v^2 \end{bmatrix} \tag{5.3}$$

Figura 21 - Ajuste modelo para ARMA11 com 5 rodadas

```
> summary(fitTheta)
Call:
cov_clm(fit = fit, fittingType = "PML", sigmaThetaExpr = "ARMA11",
    optimParams = list(par = c(7, 5, 0, 0)))

Fitting Function Type: PML
Sigma Theta Type: ARMA11

Covariance Params: sigma2u sigma2v gamma phi

Coefficients:

          Estimate   Std. Error
--------  ---------  -----------
sigma2u      4.5275       1.8514
sigma2v      7.8152       1.8757
gamma        0.4406       0.1285
phi          0.7468       0.1582
---

Optimization Details:

Converged: TRUE
Optim Method: L-BFGS-B
Iteractions Number - Function: 30
Iteractions Number - Gradient: 30
Optim Function Value: 0.0286

          Start Value  Lower   Upper
--------  -----------  ------  ------
sigma2u             7  -       -
sigma2v             5  -       -
gamma               0  -       -
phi                 0  -       -


Sigma Theta Matrix:

--------  --------  --------  --------  --------
 12.3427    7.9711    7.0990    6.4478    5.9615
  7.9711   12.3427    7.9711    7.0990    6.4478
  7.0990    7.9711   12.3427    7.9711    7.0990
  6.4478    7.0990    7.9711   12.3427    7.9711
  5.9615    6.4478    7.0990    7.9711   12.3427
--------  --------  --------  --------  --------


Goodness of Fit Measures:

RMR: 0.4701
AGFI: 0.9842
```

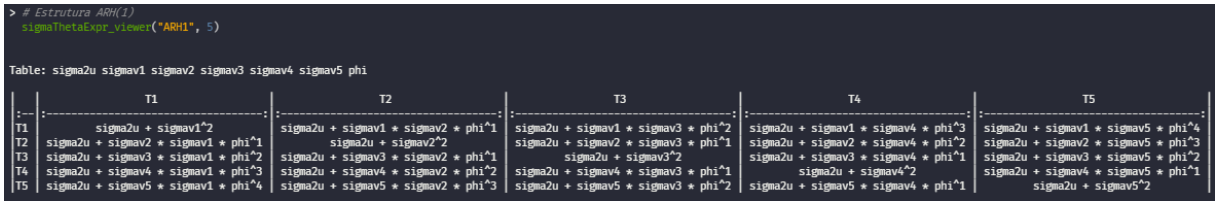
Fonte: Elaborado pelo autor. (2021).

Ajuste ainda melhor que o AR1.

5.3.3.4 ARH1

Consideramos a estrutura ARH 1.

Figura 22 - 'sigmaThetaExpr_viewer' para ARH1 com 5 rodadas



```
> # Estrutura ARH(1)
  sigmaThetaExpr_viewer("ARH1", 5)


Table: sigma2u sigmav1 sigmav2 sigmav3 sigmav4 sigmav5 phi

|  | T1 | T2 | T3 | T4 | T5 |
|:--|:--|:--|:--|:--|:--|
| T1 | sigma2u + sigmav1^2 | sigma2u + sigmav1 * sigmav2 * phi^1 | sigma2u + sigmav1 * sigmav3 * phi^2 | sigma2u + sigmav1 * sigmav4 * phi^3 | sigma2u + sigmav1 * sigmav5 * phi^4 |
| T2 | sigma2u + sigmav2 * sigmav1 * phi^1 | sigma2u + sigmav2^2 | sigma2u + sigmav2 * sigmav3 * phi^1 | sigma2u + sigmav2 * sigmav4 * phi^2 | sigma2u + sigmav2 * sigmav5 * phi^3 |
| T3 | sigma2u + sigmav3 * sigmav1 * phi^2 | sigma2u + sigmav3 * sigmav2 * phi^1 | sigma2u + sigmav3^2 | sigma2u + sigmav3 * sigmav4 * phi^1 | sigma2u + sigmav3 * sigmav5 * phi^2 |
| T4 | sigma2u + sigmav4 * sigmav1 * phi^3 | sigma2u + sigmav4 * sigmav2 * phi^2 | sigma2u + sigmav4 * sigmav3 * phi^1 | sigma2u + sigmav4^2 | sigma2u + sigmav4 * sigmav5 * phi^1 |
| T5 | sigma2u + sigmav5 * sigmav1 * phi^4 | sigma2u + sigmav5 * sigmav2 * phi^3 | sigma2u + sigmav5 * sigmav3 * phi^2 | sigma2u + sigmav5 * sigmav4 * phi^1 | sigma2u + sigmav5^2 |
```

Fonte: Elaborado pelo autor. (2021).

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_u^2 + \sigma_{v1}^2 & \sigma_u^2 + \sigma_{v2}\sigma_{v1}\phi & \dots & \sigma_u^2 + \sigma_{v5}\sigma_{v1}\phi^4 \\ \sigma_u^2 + \sigma_{v2}\sigma_{v1}\phi & \sigma_u^2 + \sigma_{v2}^2 & \dots & \sigma_u^2 + \sigma_{v5}\sigma_{v2}\phi^3 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_u^2 + \sigma_{v5}\sigma_{v1}\phi^4 & \sigma_u^2 + \sigma_{v5}\sigma_{v2}\phi^3 & \dots & \sigma_u^2 + \sigma_{v5}^2 \end{bmatrix} \tag{5.4}$$

Figura 23 - Ajuste modelo para ARH1 com 5 rodadas

```
> summary(fitTheta)
Call:
cov_clm(fit = fit, fittingType = "PML", sigmaThetaExpr = "ARH1",
    optimParams = list(par = c(10, 2, 1, 1, 1, 1, 0.2)))

Fitting Function Type: PML
Sigma Theta Type: ARH1

Covariance Params: sigma2u sigmav1 sigmav2 sigmav3 sigmav4 sigmav5 phi

Coefficients:

          Estimate   Std. Error
--------  ---------  -----------
sigma2u      6.5516       0.3996
sigmav1      2.7053       0.0803
sigmav2      2.3123       0.0628
sigmav3      2.3397       0.0814
sigmav4      2.3264       0.0818
sigmav5      2.2941       0.0739
phi          0.2294       0.0263
---

Optimization Details:

Converged: TRUE
Optim Method: L-BFGS-B
Iteractions Number - Function: 29
Iteractions Number - Gradient: 29
Optim Function Value: 0.0143

          Start Value  Lower   Upper
--------  -----------  ------  ------
sigma2u          10.0  -       -
sigmav1           2.0  -       -
sigmav2           1.0  -       -
sigmav3           1.0  -       -
sigmav4           1.0  -       -
sigmav5           1.0  -       -
phi               0.2  -       -


Sigma Theta Matrix:

--------  --------  --------  --------  --------
 13.8702    7.9867    6.8848    6.6276    6.5688
  7.9867   11.8984    7.7928    6.8348    6.6157
  6.8848    7.7928   12.0259    7.8004    6.8341
  6.6276    6.8348    7.8004   11.9640    7.7760
  6.5688    6.6157    6.8341    7.7760   11.8144
--------  --------  --------  --------  --------


Goodness of Fit Measures:

RMR: 0.4253
AGFI: 0.9895
```

Fonte: Elaborado pelo autor. (2021).

Pouco ganho em relação ao ARMA11, mas ainda melhor que o anterior.

5.3.3.5 Heterogeneous UCM

Consideramos a estrutura UCM heterogênea.

Figura 24 - 'sigmaThetaExpr_viewer' para 'Heterogeneous UCM' com 5 rodadas

```
> # Estrutura uniforme heterogênea
  sigmaThetaExpr_viewer("Heterogeneous UCM", 5)


Table: sigma2u sigmav1 sigmav2 sigmav3 sigmav4 sigmav5 rho
```

	T1	T2	T3	T4	T5
T1	sigma2u + sigmav1^2	sigma2u + sigmav1 * sigmav2 * rho	sigma2u + sigmav1 * sigmav3 * rho	sigma2u + sigmav1 * sigmav4 * rho	sigma2u + sigmav1 * sigmav5 * rho
T2	sigma2u + sigmav2 * sigmav1 * rho	sigma2u + sigmav2^2	sigma2u + sigmav2 * sigmav3 * rho	sigma2u + sigmav2 * sigmav4 * rho	sigma2u + sigmav2 * sigmav5 * rho
T3	sigma2u + sigmav3 * sigmav1 * rho	sigma2u + sigmav3 * sigmav2 * rho	sigma2u + sigmav3^2	sigma2u + sigmav3 * sigmav4 * rho	sigma2u + sigmav3 * sigmav5 * rho
T4	sigma2u + sigmav4 * sigmav1 * rho	sigma2u + sigmav4 * sigmav2 * rho	sigma2u + sigmav4 * sigmav3 * rho	sigma2u + sigmav4^2	sigma2u + sigmav4 * sigmav5 * rho
T5	sigma2u + sigmav5 * sigmav1 * rho	sigma2u + sigmav5 * sigmav2 * rho	sigma2u + sigmav5 * sigmav3 * rho	sigma2u + sigmav5 * sigmav4 * rho	sigma2u + sigmav5^2

Fonte: Elaborado pelo autor. (2021).

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_u^2 + \sigma_{v1}^2 & \sigma_u^2 + \sigma_{v2}\sigma_{v1}\rho & \dots & \sigma_u^2 + \sigma_{v5}\sigma_{v1}\rho \\ \sigma_u^2 + \sigma_{v2}\sigma_{v1}\rho & \sigma_u^2 + \sigma_{v2}^2 & \dots & \sigma_u^2 + \sigma_{v5}\sigma_{v2}\rho \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_u^2 + \sigma_{v5}\sigma_{v1}\rho & \sigma_u^2 + \sigma_{v5}\sigma_{v2}\rho & \dots & \sigma_u^2 + \sigma_{v5}^2 \end{bmatrix} \tag{5.5}$$

Figura 25 - Ajuste modelo para 'Heterogeneous UCM' com 5 rodadas

```
> summary(fitTheta)
Call:
cov_clm(fit = fit, fittingType = "PML", sigmaThetaExpr = "Heterogeneous UCM",
    optimParams = list(par = c(10, 1, 1, 1, 1, 1, 0.2)))

Fitting Function Type: PML
Sigma Theta Type: Heterogeneous UCM

Covariance Params: sigma2u sigmav1 sigmav2 sigmav3 sigmav4 sigmav5 rho

Coefficients:

          Estimate   Std. Error
--------  ---------  -----------
sigma2u     10.4656       1.0622
sigmav1      1.6189       0.4166
sigmav2      1.2710       0.3332
sigmav3      1.2073       0.3155
sigmav4      1.2806       0.3232
sigmav5      1.3618       0.3396
rho         -1.8259       1.4522
---

Optimization Details:

Converged: TRUE
Optim Method: L-BFGS-B
Iteractions Number - Function: 40
Iteractions Number - Gradient: 40
Optim Function Value: 0.1004

          Start Value  Lower   Upper
--------  -----------  ------  ------
sigma2u          10.0  -       -
sigmav1           1.0  -       -
sigmav2           1.0  -       -
sigmav3           1.0  -       -
sigmav4           1.0  -       -
sigmav5           1.0  -       -
rho               0.2  -       -


Sigma Theta Matrix:

--------  --------  --------  --------  --------
 13.0863    6.7085    6.8970    6.6800    6.4403
  6.7085   12.0811    7.6638    7.4934    7.3052
  6.8970    7.6638   11.9230    7.6425    7.4637
  6.6800    7.4934    7.6425   12.1056    7.2813
  6.4403    7.3052    7.4637    7.2813   12.3199
--------  --------  --------  --------  --------


Goodness of Fit Measures:

RMR: 0.7083
AGFI: 0.9241
```

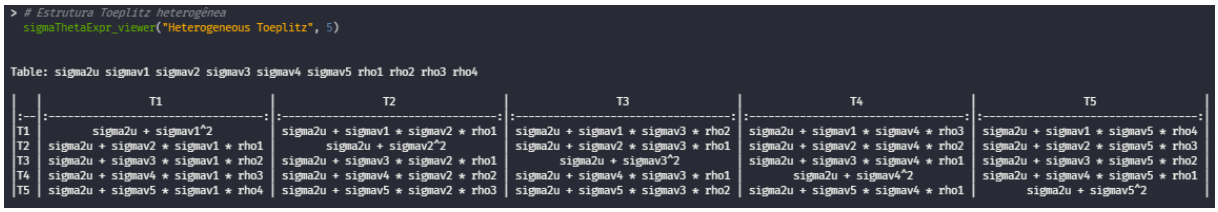
Fonte: Elaborado pelo autor. (2021).

Ajuste não muito satisfatório, inclusive muito próximo do apresentado pela estrutura UCM.

5.3.3.6 Heterogeneous Toeplitz

Finalmente, consideramos a estrutura Toeplitz heterogênea.

Figura 26 - 'sigmaThetaExpr_viewer' para 'Heterogeneous Toeplitz' com 5 rodadas



```
> # Estrutura Toeplitz heterogênea
  sigmaThetaExpr_viewer("Heterogeneous Toeplitz", 5)

Table: sigma2u sigmav1 sigmav2 sigmav3 sigmav4 sigmav5 rho1 rho2 rho3 rho4
```

	T1	T2	T3	T4	T5
T1	sigma2u + sigmav1^2	sigma2u + sigmav1 * sigmav2 * rho1	sigma2u + sigmav1 * sigmav3 * rho2	sigma2u + sigmav1 * sigmav4 * rho3	sigma2u + sigmav1 * sigmav5 * rho4
T2	sigma2u + sigmav2 * sigmav1 * rho1	sigma2u + sigmav2^2	sigma2u + sigmav2 * sigmav3 * rho1	sigma2u + sigmav2 * sigmav4 * rho2	sigma2u + sigmav2 * sigmav5 * rho3
T3	sigma2u + sigmav3 * sigmav1 * rho2	sigma2u + sigmav3 * sigmav2 * rho1	sigma2u + sigmav3^2	sigma2u + sigmav3 * sigmav4 * rho1	sigma2u + sigmav3 * sigmav5 * rho2
T4	sigma2u + sigmav4 * sigmav1 * rho3	sigma2u + sigmav4 * sigmav2 * rho2	sigma2u + sigmav4 * sigmav3 * rho1	sigma2u + sigmav4^2	sigma2u + sigmav4 * sigmav5 * rho1
T5	sigma2u + sigmav5 * sigmav1 * rho4	sigma2u + sigmav5 * sigmav2 * rho3	sigma2u + sigmav5 * sigmav3 * rho2	sigma2u + sigmav5 * sigmav4 * rho1	sigma2u + sigmav5^2

Fonte: Elaborado pelo autor. (2021).

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_u^2 + \sigma_{v1}^2 & \sigma_u^2 + \sigma_{v2}\sigma_{v1}\rho_1 & \dots & \sigma_u^2 + \sigma_{v5}\sigma_{v1}\rho_4 \\ \sigma_u^2 + \sigma_{v2}\sigma_{v1}\rho_1 & \sigma_u^2 + \sigma_{v2}^2 & \dots & \sigma_u^2 + \sigma_{v5}\sigma_{v2}\rho_3 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_u^2 + \sigma_{v5}\sigma_{v1}\rho_4 & \sigma_u^2 + \sigma_{v5}\sigma_{v2}\rho_3 & \dots & \sigma_u^2 + \sigma_{v5}^2 \end{bmatrix} \tag{5.6}$$

Figura 27 - Ajuste modelo para 'Heterogeneous Toeplitz' com 5 rodadas

```
> summary(fitTheta)
Call:
cov_clm(fit = fit, fittingType = "PML", sigmaThetaExpr = "Heterogeneous Toeplitz",
    optimParams = list(par = c(10, 2, 1, 1, 1, 1, 0.2, 0.2, 0.2,
        0.2)))

Fitting Function Type: PML
Sigma Theta Type: Heterogeneous Toeplitz

Covariance Params: sigma2u sigmav1 sigmav2 sigmav3 sigmav4 sigmav5 rho1 rho2 rho3 rho4

Coefficients:

          Estimate   Std. Error
--------  ---------  -----------
sigma2u      7.5111       3.2867
sigmav1      2.4491       0.7939
sigmav2      2.1400       0.6907
sigmav3      2.1583       0.8070
sigmav4      2.1491       0.7130
sigmav5      2.0498       0.7629
rho1         0.0839       0.6321
rho2        -0.0996       0.7598
rho3        -0.1982       0.8206
rho4        -0.3047       0.8909
---

Optimization Details:

Converged: TRUE
Optim Method: L-BFGS-B
Iteractions Number - Function: 92
Iteractions Number - Gradient: 92
Optim Function Value: 0.0057

          Start Value  Lower   Upper
--------  -----------  ------  ------
sigma2u          10.0  -       -
sigmav1           2.0  -       -
sigmav2           1.0  -       -
sigmav3           1.0  -       -
sigmav4           1.0  -       -
sigmav5           1.0  -       -
rho1              0.2  -       -
rho2              0.2  -       -
rho3              0.2  -       -
rho4              0.2  -       -


Sigma Theta Matrix:

--------  --------  --------  --------  --------
 13.5091    7.9511    6.9845    6.4678    5.9817
  7.9511   12.0909    7.8988    7.0529    6.6416
  6.9845    7.8988   12.1693    7.9005    7.0704
  6.4678    7.0529    7.9005   12.1298    7.8809
  5.9817    6.6416    7.0704    7.8809   11.7127
--------  --------  --------  --------  --------


Goodness of Fit Measures:

RMR: 0.3335
AGFI: 0.9932
```

Fonte: Elaborado pelo autor. (2021).

De todas as estruturas consideradas, esta foi a que obteve melhor resultado.

5.3.4 Observações adicionais

Apesar de não termos apresentado na subseção anterior, a função 'cov_clm' realiza o ajuste do modelo considerando também outras funções de ajuste, que são aquelas apresentadas no capítulo 4, a saber: ULSC, GLSC2, GLSC3, GLSC4, permitindo que o usuário realize ainda mais combinações de ajuste a fim de encontrar a estrutura que mais se adéque aos dados.

Além disso, existe ainda a possibilidade do usuário fornecer a sua própria estrutura

de interesse de forma totalmente personalizável, inclusive mudando o nome dos parâmetros e combinando estruturas. Como forma de exemplo, considere o caso abaixo em que a estrutura da matriz de covariância foi definida de maneira a combinar as estruturas AR1 e ARMA11, para os efeitos aleatórios permanentes e transitórios, respectivamente, tendo o parâmetro 'gamma' comum a ambos.

Figura 28 - Código utilizado para gerar estrutura AR1 + ARMA11

```
#### Custom AR1 (sigma2u) + ARMA11 (sigma2v) - common gamma

wavesCombination ← expand.grid(t = seq(fit$info$T), tlinha = seq(fit$info$T))

sigmaThetaExpr ← mapply(function(t, tlinha) {
  if (t == tlinha) {
    parse(text = paste0("sigma2u + sigma2v"))
  } else {
    parse(text = paste0("gamma^", abs(t - tlinha), " * sigma2u + gamma * phi^", abs(t - tlinha) - 1, "* sigma2v"))
  }
}, wavesCombination$t, wavesCombination$tlinha, SIMPLIFY = FALSE)

sigmaThetaExpr ← list(
  expr = sigmaThetaExpr,
  paramsNames = c("sigma2u", "sigma2v", "gamma", "phi")
)
```

Fonte: Elaborado pelo autor. (2021).

A grande vantagem ao se utilizar a função 'sigmaThetaExpr_viewer' é que ela é capaz de gerar a visualização para estruturas personalizadas também, possibilitando uma análise exploratória prévia e que o analista verifique se a estrutura desenvolvida por ele está realmente de acordo com o planejado.

Figura 29 - 'sigmaThetaExpr_viewer' para AR1 + ARMA11 com 5 rodadas

```
> # Custom AR1 (sigma2u) + ARMA11 (sigma2v) - common gamma
  sigmaThetaExpr_viewer(sigmaThetaExpr)

Table: sigma2u sigma2v gamma phi

|  | T1 | T2 | T3 | T4 | T5 |
|:--|:---|:---|:---|:---|:---|
| T1 | sigma2u + sigma2v | gamma^1 * sigma2u + gamma * phi^0 * sigma2v | gamma^2 * sigma2u + gamma * phi^1 * sigma2v | gamma^3 * sigma2u + gamma * phi^2 * sigma2v | gamma^4 * sigma2u + gamma * phi^3 * sigma2v |
| T2 | gamma^1 * sigma2u + gamma * phi^0 * sigma2v | sigma2u + sigma2v | gamma^1 * sigma2u + gamma * phi^0 * sigma2v | gamma^2 * sigma2u + gamma * phi^1 * sigma2v | gamma^3 * sigma2u + gamma * phi^2 * sigma2v |
| T3 | gamma^2 * sigma2u + gamma * phi^1 * sigma2v | gamma^1 * sigma2u + gamma * phi^0 * sigma2v | sigma2u + sigma2v | gamma^1 * sigma2u + gamma * phi^0 * sigma2v | gamma^2 * sigma2u + gamma * phi^1 * sigma2v |
| T4 | gamma^3 * sigma2u + gamma * phi^2 * sigma2v | gamma^2 * sigma2u + gamma * phi^1 * sigma2v | gamma^1 * sigma2u + gamma * phi^0 * sigma2v | sigma2u + sigma2v | gamma^1 * sigma2u + gamma * phi^0 * sigma2v |
| T5 | gamma^4 * sigma2u + gamma * phi^3 * sigma2v | gamma^3 * sigma2u + gamma * phi^2 * sigma2v | gamma^2 * sigma2u + gamma * phi^1 * sigma2v | gamma^1 * sigma2u + gamma * phi^0 * sigma2v | sigma2u + sigma2v |
```

Fonte: Elaborado pelo autor. (2021).

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_u^2 + \sigma_v^2 & \gamma^1\sigma_u^2 + \gamma\sigma_v^2 & \dots & \gamma^4\sigma_u^2 + \gamma\phi^3\sigma_v^2 \\ \gamma^1\sigma_u^2 + \gamma\sigma_v^2 & \sigma_u^2 + \sigma_v^2 & \dots & \gamma^3\sigma_u^2 + \gamma\phi^2\sigma_v^2 \\ \vdots & \vdots & \ddots & \vdots \\ \gamma^4\sigma_u^2 + \gamma\phi^3\sigma_v^2 & \gamma^3\sigma_u^2 + \gamma\phi^2\sigma_v^2 & \dots & \sigma_u^2 + \sigma_v^2 \end{bmatrix} \tag{5.7}$$

Finalmente, realizamos o ajuste do modelo.

Figura 30 - Ajuste modelo para AR1 + ARMA11 com 5 rodadas

```
> summary(fitTheta)
Call:
cov_clm(fit = fit, fittingType = "PML", sigmaThetaExpr = sigmaThetaExpr,
    optimParams = list(par = c(7, 5, 0.5, 0)))

Fitting Function Type: PML
Sigma Theta Type: Customized

Covariance Params: sigma2u sigma2v gamma phi

Coefficients:

          Estimate   Std. Error
--------  ---------  -----------
sigma2u      2.6584       2.1238
sigma2v      9.6832       2.1872
gamma        0.6458       0.0132
phi          0.9576       0.0553
---

Optimization Details:

Converged: TRUE
Optim Method: L-BFGS-B
Iteractions Number - Function: 44
Iteractions Number - Gradient: 44
Optim Function Value: 0.0285

          Start Value  Lower   Upper
--------  -----------  ------  ------
sigma2u           7.0  -       -
sigma2v           5.0  -       -
gamma             0.5  -       -
phi               0.0  -       -


Sigma Theta Matrix:

--------  --------  --------  --------  --------
 12.3416    7.9700    7.0969    6.4504    5.9538
  7.9700   12.3416    7.9700    7.0969    6.4504
  7.0969    7.9700   12.3416    7.9700    7.0969
  6.4504    7.0969    7.9700   12.3416    7.9700
  5.9538    6.4504    7.0969    7.9700   12.3416
--------  --------  --------  --------  --------


Goodness of Fit Measures:

RMR: 0.4702
AGFI: 0.9842
```

Fonte: Elaborado pelo autor. (2021).

6 CONCLUSÃO

O advento da era computacional no século XX afetou a sociedade como um todo, em especial, os estatísticos, fazendo com que a profissão passasse a ser composta por uma tríplice combinação entre estatística, matemática e computação. Sob esta nova perspectiva profissional, para se conduzir um processo de modelagem estatística é necessário que as três componentes deste processo sejam exploradas. Por exemplo, uma metodologia que não considera as restrições impostas pela realidade das pesquisas não é factível. Uma pesquisa que não possui formas de computar os dados a fim estimar os parâmetros não produz resultados. Um software estatístico que não possui fundamentação teórica não possui validade.

Neste sentido, enquanto Vieira (2005) buscou tratar questões relacionadas com a dificuldade de incorporar às teorias matemáticas toda a metodologia estatística para pesquisas longitudinais sob amostragem complexa, o presente trabalho buscou traduzir para termos práticos computacionais aquilo que são abstrações matemáticas desenvolvidas pela metodologia do autor.

No capítulo 2 vimos alguns princípios relacionados à criação de algoritmos e as boas práticas da programação que influenciam diretamente na eficiência do programador e consequentemente do software desenvolvido. Além disso, apresentamos algumas particularidades da linguagem de programação R, como fábrica de funções e expressões, que foram questões fundamentais para o desenvolvimento do pacote estatístico 'clm'.

Nos capítulos 3 e 4 buscamos apresentar os fundamentos essenciais da metodologia de Vieira (2005) como a estimação do efeitos fixos e aleatórios do modelo, a matriz de covariância, os parâmetros das estruturas de covariância e algumas medidas de qualidade de ajuste, tudo isto tendo vista a tradução destas abstrações matemáticas em termos computacionais, sem negligenciar a experiência do usuário que irá usufruir do pacote, questões de eficiência e generalidade dos métodos apresentados. Tudo isto foi realizado por meio da construção de fluxogramas e discussões envolvendo possíveis implementações para o futuro.

No capítulo 5 apresentamos primeiramente os ganhos que obtivemos ao considerar os princípios do código limpo, comparando as funções finais do pacote com as mesmas antes de terem sido revisadas e reescritas. Além disso, mostramos como essa perspectiva das boas práticas da computação auxiliaram na visualização de possíveis estratégias para a resolução de certos problemas, o que possibilitou que o pacote atingisse seus objetivos com ainda mais precisão. Em segundo lugar, mostramos como instalar o 'clm' no R tanto por meio do repositório CRAN quanto por meio do github. Finalmente, apresentamos uma aplicação, buscando mostrar na prática os resultados que um usuário geral teria ao utilizar as funções do pacote.

Buscamos ressaltar em diversos momentos que ao tratar do processo de desenvolvimento de software dificilmente estaremos lidando com resultados definitivos de um produto. Pelo contrário, um software é composto por versões, de forma que uma versão posterior a outra tenha corrigido eventuais problemas de implementação da anterior, além de incorporar novas funcionalidades.

Neste sentido, finalizamos este trabalho apresentando algumas possíveis abordagens de implementação para o futuro. Em primeiro lugar, iremos buscar incorporar o pacote *survey* de maneira que a especificação do plano amostral no ajuste dos modelos seja mais padronizada entre os estatísticos que lidam com questões de amostragem e expandir assim o processo de incorporação do plano amostral complexo ao pacote.

Em segundo lugar, buscaremos implementar funcionalidades auxiliares que atuam principalmente de maneira posterior ao ajuste dos modelos. Por exemplo, outras medidas de qualidade de ajuste também desenvolvidas por Vieira (2005) para o modelo em questão, tais como: (i) teste qui-quadrado para testar a igualdade entre $\boldsymbol{S}_w$ e $\boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}})$; (ii) teste de significância de Wald para modelos encaixados a fim comparar um modelo com outro restrito; (iii) medidas de influência global e local, que compõem metodologias de análise de diagnóstico e são bastante úteis na análise do modelo; (iv) métodos de seleção de variáveis para os parâmetros da regressão; entre muitas outras.

Já tratando de uma implementação futura de porte mais robusto, uma possível abordagem interessante é adicionar o uso de covariáveis diferentes do intercepto também no ajuste de $\boldsymbol{\theta}$ de maneira que o pacote se expanda cada vez mais na perspectiva dos modelos mistos gerais, tendo como diferencial a incorporação do plano amostral da pesquisa. Este passo é mais complexo que os demais pois envolve fundamentação teórica que vai além do escopo do trabalho adotado como referência para criação deste pacote, mas cria horizontes para o futuro, uma vez que o processo de modelagem estatística se desenvolve na interação entre as áreas da matemática, estatística e computação.

REFERÊNCIAS

BATTISTI, Iara Denise Endruweit. **Análise de Dados Epidemiológicos Incorporando Planos Amostrais Complexos**: tese de doutorado. 2008. 198 f. Tese (Doutorado) - Curso de Medicina, Faculdade de Medicina, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2008. Cap. 3.

COURANT, Richard; ROBBINS, Herbert. **What is Mathematics?**: an elementary approach to ideas and methods. 5. ed. Nova York: Oxford University Press, 1941.

JAMES, Gareth; WITTEN, Daniela; HASTIE, Trevor; TIBSHIRANI, Robert. **An Introduction to Statistical Learning**: with applications in r. New York: Springer, 2013.

JORESKOG, Karl G.; SORBOM, Dag. **Lisrel 7: a guide to the program and applications.** 2. ed. Chicago: Spss Publications, 1989.

MARTIN, Robert C.. **Arquitetura Limpa**: o guia do artesão para estrutura e design de software. Rio de Janeiro: Alta Books Editora, 2019. Título original: Clean Architecture.

MARTIN, Robert C.. **Código limpo**: habilidades práticas do agile software. 2. ed. Rio de Janeiro: Alta Books Editora, 2011. Título original: Clean Code.

The R Project for Statistical Computing: Getting Started. Getting Started. Disponível em: https://www.r-project.org/. Acesso em: 19 ago. 2021.

SANTOS, Thuany de Aguiar. **Intervalo de Confiança para o parâmetro estimado pelo estimador de Horvitz-Thompson**. 2016. 67 f. Dissertação (Mestrado) - Curso de Estatística, Departamento de Estatística, Universidade de Brasília, Brasília, 2016. Cap. 1.

SINGER, Julio M.; NOBRE, Juvêncio S.; ROCHA, Francisco Marcelo M.. **Análise de dados longitudinais**. São Paulo: Universidade de São Paulo, 2018. 312 p. Versão preliminar.

SKINNER, C. J.. Longitudinal Data: introduction to part d. In: CHAMBERS, R.L.; SKINNER, C. J.. **Analysis of Survey Data**. Southampton: Wiley, 2003. p. 199-204. (Wiley Series In Survey Methodology).

SOUZA, Marco A. Furlan de et al. **Algoritmos e lógica de programação**: um texto introdutório para engenharia. 2. ed. São Paulo: Cengage Learning, 2011.

VIEIRA, Marcel de Toledo. **Modelling Complex Longitudinal Survey Data**: thesis for the degree of doctor of philosophy. 2005. 263 f. Tese (Doutorado) - Curso de Doctor Of Philosophy, School Of Social Sciences, University Of Southampton, Southampton, 2005.

VIEIRA, Marcel de Toledo. **A Consideração da Amostragem Complexa na Análise de Dados Longitudinais**. Piracicaba: 57ª Reunião Anual da Rbras, 2012.

VIEIRA, Marcel de Toledo; SMITH, Peter W.F.; SALGUEIRO, Maria de Fátima. Misspecification Effects in the Analysis of Panel Data. **Journal Of Official Statistics**, [S.L.], v. 32, n. 2, p. 487-505, 28 maio 2016. Walter de Gruyter GmbH. http://dx.doi.org/10.1515/jos-2016-0025.

WICKHAM, Hadley. **Advanced R**. 2. ed. Boca Raton: Chapman And Hall, 2014. (CRC The R Series).

APÊNDICE A – Ajuste das demais estruturas de covariância implementadas internamente no pacote

.1 APLICAÇÃO

.1.1 Estimação dos parâmetros dos efeitos aleatórios

Nesta seção, será apresentado o ajuste para as demais estruturas implementadas internamente na função cov_clm.

.1.1.1 Toeplitz

Figura 31 - 'sigmaThetaExpr_viewer' para Toeplitz com 5 rodadas

```
> # Estrutura Toeplitz
  sigmaThetaExpr_viewer("Toeplitz", 5)


Table: sigma2u sigma2v sigma1 sigma2 sigma3 sigma4

|    | T1                | T2                | T3                | T4                | T5                |
|:---|:------------------|:------------------|:------------------|:------------------|:------------------|
| T1 | sigma2u + sigma2v | sigma2u + sigma1  | sigma2u + sigma2  | sigma2u + sigma3  | sigma2u + sigma4  |
| T2 | sigma2u + sigma1  | sigma2u + sigma2v | sigma2u + sigma1  | sigma2u + sigma2  | sigma2u + sigma3  |
| T3 | sigma2u + sigma2  | sigma2u + sigma1  | sigma2u + sigma2v | sigma2u + sigma1  | sigma2u + sigma2  |
| T4 | sigma2u + sigma3  | sigma2u + sigma2  | sigma2u + sigma1  | sigma2u + sigma2v | sigma2u + sigma1  |
| T5 | sigma2u + sigma4  | sigma2u + sigma3  | sigma2u + sigma2  | sigma2u + sigma1  | sigma2u + sigma2v |
```

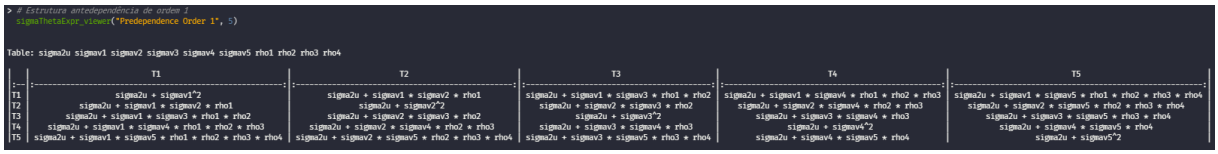
Fonte: Elaborado pelo autor. (2021).

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_u^2 + \sigma_v^2 & \sigma_u^2 + \sigma_1 & \dots & \sigma_u^2 + \sigma_4 \\ \sigma_u^2 + \sigma_1 & \sigma_u^2 + \sigma_v^2 & \dots & \sigma_u^2 + \sigma_3 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_u^2 + \sigma_4 & \sigma_u^2 + \sigma_3 & \dots & \sigma_u^2 + \sigma_v^2 \end{bmatrix} \tag{.1}$$

Para este exemplo específico, não foi possível fazer o ajuste considerando a estrutura Toeplitz seguindo esta parametrização que utiliza a decomposição dos parâmetros em σ_u^2 e σ_v^2 pois o ajuste resultou em uma matriz singular e portanto não inversível. Ressaltamos que considerando a mesma estrutura porém sem a decomposição anterior o modelo foi ajustado corretamente.

.1.1.2 Predependence Order 1

Figura 32 - 'sigmaThetaExpr_viewer' para estrutura antedependência de ordem 1 com 5 rodadas



```
> # Estrutura antedependência de ordem 1
  sigmaThetaExpr_viewer("Predependence Order 1", 5)

Table: sigma2u sigmav1 sigmav2 sigmav3 sigmav4 sigmav5 rho1 rho2 rho3 rho4
```

	T1	T2	T3	T4	T5
T1	sigma2u + sigmav1^2	sigma2u + sigmav1 * sigmav2 * rho1	sigma2u + sigmav1 * sigmav3 * rho1 * rho2	sigma2u + sigmav1 * sigmav4 * rho1 * rho2 * rho3	sigma2u + sigmav1 * sigmav5 * rho1 * rho2 * rho3 * rho4
T2	sigma2u + sigmav1 * sigmav2 * rho1	sigma2u + sigmav2^2	sigma2u + sigmav2 * sigmav3 * rho2	sigma2u + sigmav2 * sigmav4 * rho2 * rho3	sigma2u + sigmav2 * sigmav5 * rho2 * rho3 * rho4
T3	sigma2u + sigmav1 * sigmav3 * rho1 * rho2	sigma2u + sigmav2 * sigmav3 * rho2	sigma2u + sigmav3^2	sigma2u + sigmav3 * sigmav4 * rho3	sigma2u + sigmav3 * sigmav5 * rho3 * rho4
T4	sigma2u + sigmav1 * sigmav4 * rho1 * rho2 * rho3	sigma2u + sigmav2 * sigmav4 * rho2 * rho3	sigma2u + sigmav3 * sigmav4 * rho3	sigma2u + sigmav4^2	sigma2u + sigmav4 * sigmav5 * rho4
T5	sigma2u + sigmav1 * sigmav5 * rho1 * rho2 * rho3 * rho4	sigma2u + sigmav2 * sigmav5 * rho2 * rho3 * rho4	sigma2u + sigmav3 * sigmav5 * rho3 * rho4	sigma2u + sigmav4 * sigmav5 * rho4	sigma2u + sigmav5^2

Fonte: Elaborado pelo autor. (2021).

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_u^2 + \sigma_{v1}^2 & \sigma_u^2 + \sigma_{v2}\sigma_{v1}\rho_1 & \dots & \sigma_u^2 + \sigma_{v5}\sigma_{v1}\rho_1\rho_2\rho_3\rho_4 \\ \sigma_u^2 + \sigma_{v2}\sigma_{v1}\rho_1 & \sigma_u^2 + \sigma_{v2}^2 & \dots & \sigma_u^2 + \sigma_{v5}\sigma_{v1}\rho_2\rho_3\rho_4 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_u^2 + \sigma_{v5}\sigma_{v1}\rho_1\rho_2\rho_3\rho_4 & \sigma_u^2 + \sigma_{v5}\sigma_{v1}\rho_2\rho_3\rho_4 & \dots & \sigma_u^2 + \sigma_{v5}^2 \end{bmatrix} \tag{.2}$$

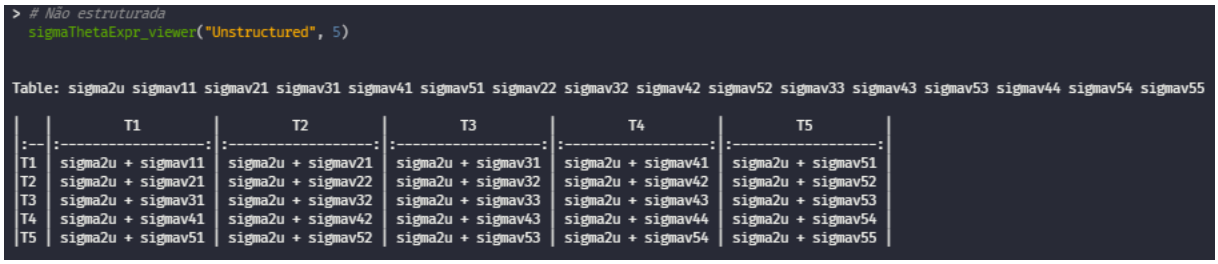
Figura 33 - Ajuste modelo para estrutura antedependência de ordem 1 com 5 rodadas

```
> summary(fitTheta)
Call:
cov_clm(fit = fit, fittingType = "PML", sigmaThetaExpr = "Predependence Order 1",
    optimParams = list(par = c(10, 2.5, 2.5, 2.5, 2.5, 2.5, 0.5,
        0.5, 0.5, 0.5)))

Fitting Function Type: PML
Sigma Theta Type: Predependence Order 1

Covariance Params: sigma2u sigmav1 sigmav2 sigmav3 sigmav4 sigmav5 rho1 rho2 rho3 rho4

Coefficients:

          Estimate   Std. Error
--------  ---------  -----------
sigma2u      6.6031       0.4076
sigmav1      2.6949       0.0874
sigmav2      2.2738       0.0762
sigmav3      2.2901       0.0915
sigmav4      2.3382       0.0832
sigmav5      2.3302       0.0746
rho1         0.2211       0.0433
rho2         0.1886       0.0512
rho3         0.2040       0.0504
rho4         0.2639       0.0477
---

Optimization Details:

Converged: TRUE
Optim Method: L-BFGS-B
Iteractions Number - Function: 41
Iteractions Number - Gradient: 41
Optim Function Value: 0.0125

          Start Value  Lower   Upper
--------  -----------  ------  ------
sigma2u          10.0  -       -
sigmav1           2.5  -       -
sigmav2           2.5  -       -
sigmav3           2.5  -       -
sigmav4           2.5  -       -
sigmav5           2.5  -       -
rho1              0.5  -       -
rho2              0.5  -       -
rho3              0.5  -       -
rho4              0.5  -       -


Sigma Theta Matrix:

--------  --------  --------  --------  --------
 13.8656    7.9577    6.8604    6.6567    6.6172
  7.9577   11.7734    7.5851    6.8076    6.6569
  6.8604    7.5851   11.8478    7.6956    6.8904
  6.6567    6.8076    7.6956   12.0705    8.0408
  6.6172    6.6569    6.8904    8.0408   12.0331
--------  --------  --------  --------  --------


Goodness of Fit Measures:

RMR: 0.4512
AGFI: 0.9854
```

Fonte: Elaborado pelo autor. (2021).

.1.1.3 Unstructured

Figura 34 - 'sigmaThetaExpr_viewer' para matriz não estruturada com 5 rodadas



```
> # Não estruturada
  sigmaThetaExpr_viewer("Unstructured", 5)

Table: sigma2u sigmav11 sigmav21 sigmav31 sigmav41 sigmav51 sigmav22 sigmav32 sigmav42 sigmav52 sigmav33 sigmav43 sigmav53 sigmav44 sigmav54 sigmav55
```

	T1	T2	T3	T4	T5
T1	sigma2u + sigmav11	sigma2u + sigmav21	sigma2u + sigmav31	sigma2u + sigmav41	sigma2u + sigmav51
T2	sigma2u + sigmav21	sigma2u + sigmav22	sigma2u + sigmav32	sigma2u + sigmav42	sigma2u + sigmav52
T3	sigma2u + sigmav31	sigma2u + sigmav32	sigma2u + sigmav33	sigma2u + sigmav43	sigma2u + sigmav53
T4	sigma2u + sigmav41	sigma2u + sigmav42	sigma2u + sigmav43	sigma2u + sigmav44	sigma2u + sigmav54
T5	sigma2u + sigmav51	sigma2u + sigmav52	sigma2u + sigmav53	sigma2u + sigmav54	sigma2u + sigmav55

Fonte: Elaborado pelo autor. (2021).

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_u^2 + \sigma_{v11} & \sigma_u^2 + \sigma_{v21} & \dots & \sigma_u^2 + \sigma_{v51} \\ \sigma_u^2 + \sigma_{v21} & \sigma_u^2 + \sigma_{v22} & \dots & \sigma_u^2 + \sigma_{v52} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_u^2 + \sigma_{v51} & \sigma_u^2 + \sigma_{v52} & \dots & \sigma_u^2 + \sigma_{v55} \end{bmatrix} \tag{.3}$$

Assim como ocorreu com a estrutura Toeplitz, não foi possível fazer o ajuste considerando a matriz não estruturada seguindo esta parametrização que utiliza a decomposição dos parâmetros em σ_u^2 e σ_v^2 pois o ajuste resultou em uma matriz singular e portanto não inversível. Ressaltamos que considerando a mesma estrutura porém sem a decomposição anterior o modelo foi ajustado corretamente e o ajuste resultante foi exatamente igual aos valores da matriz $\boldsymbol{S}_w$, indicando coerência quanto ao desempenho da função.