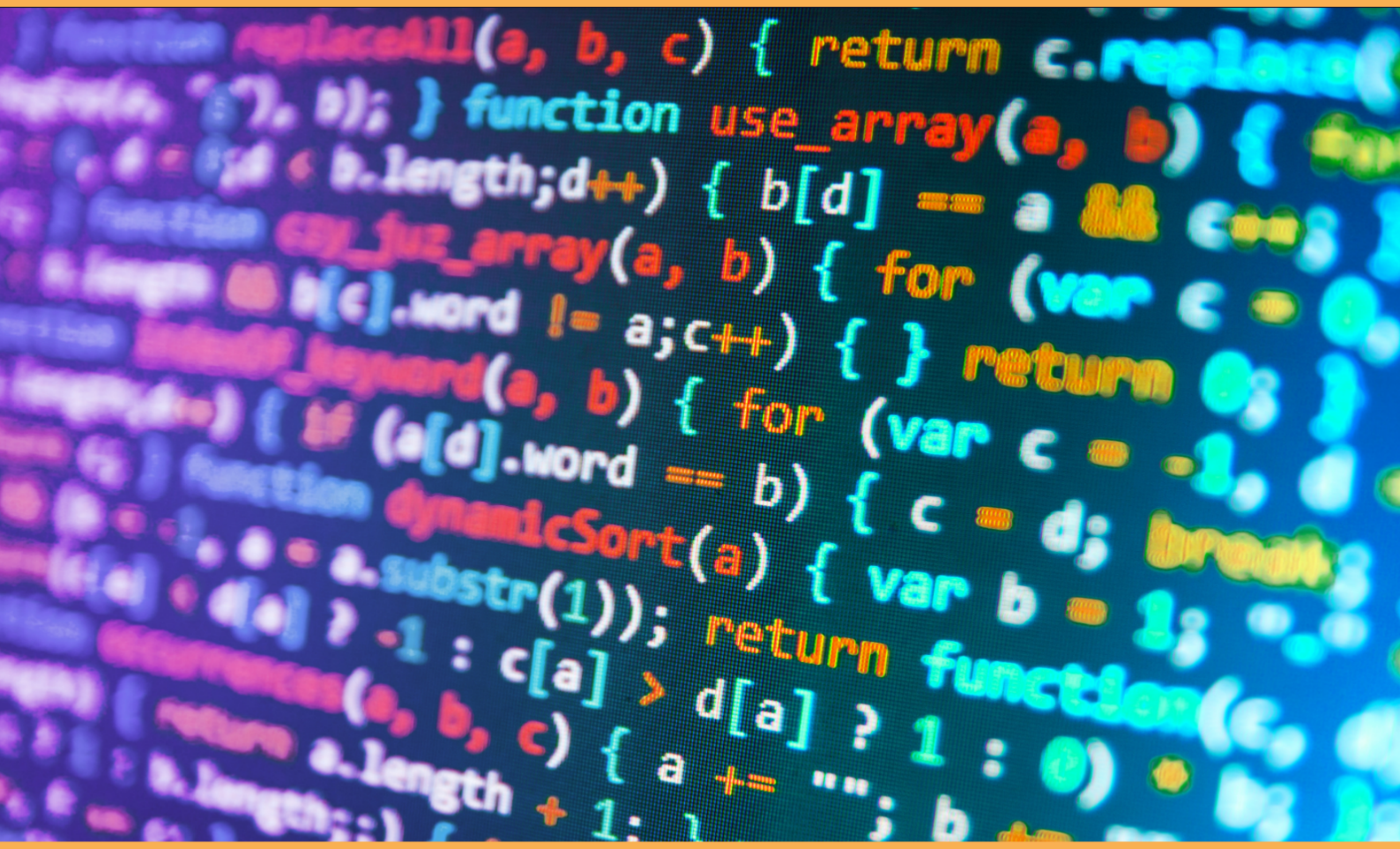

JULIA PARA ECONOMISTAS REGIONAIS

Introdução aos Métodos Regionais em Julia

2ª edição

ALAN LEAL

"Todas as coisas estão relacionadas entre si, mas as coisas mais próximas estão mais relacionadas que as distantes."

Tobler (1970)

Ao meu querido afilhado, José Augusto.

Agradecimentos

Este livro é mais um capítulo no meu aprendizado da Ciência Regional e Julia. Comecei a estudar Economia Regional na minha graduação na UFV, tive contato mais denso e profissional com ela no mestrado na UFMG e na USP isso se expandiu, abarcando várias linguagens de programação, inclusive Julia, que me permitiram elaborar este manuscrito. Assim, estou em dívida com a UFV, UFMG e a USP e seus professores que contribuíram direta ou indiretamente para a elaboração deste texto. Agradeço, portanto, também ao contribuinte brasileiro e paulista que financia essas universidades.

Em especial, gostaria de agradecer aos Professores Eduardo Haddad, André Chagas e Carlos Azzoni, cujos ensinamentos proporcionados em suas disciplinas e no seminário semanal do NEREUS contribuíram sobremaneira para meu enriquecimento enquanto economista especializado em ciências regionais.

Por fim, um *disclaimer* se faz necessário. Este livro foi elaborado utilizando recursos de inteligência artificial (Claude.AI da Anthropic e Grok do X) para criação, edição e revisão de textos e códigos. Enquanto a editorialização do que fazia parte do livro foi determinada unicamente por mim, utilizei diferentes IAs para consolidar textos, apresentações e códigos que eu havia feito previamente, visando consolidar informações de uma forma mais acessível e direta.

Esta é a segunda edição do livro, se diferenciando da primeira pelo seu conteúdo relacionado aos modelos de equilíbrio geral, usando o *exact-hat algebra* como método de solução.

Sumário

Capítulo 1

Introdução

1.1 Introdução

A linguagem de programação Julia (Bezanson et al. (2017)) surgiu como uma resposta inovadora aos desafios enfrentados pela comunidade científica e acadêmica na busca por uma ferramenta que combinasse a facilidade de uso de linguagens dinâmicas como Python e R com o desempenho de linguagens compiladas como C e Fortran. Desenvolvida no MIT por Jeff Bezanson, Stefan Karpinski, Viral Shah e Alan Edelman, Julia foi apresentada ao público em 2012, marcando o início de uma nova era na computação científica.

1.2 História e Desenvolvimento

O desenvolvimento de Julia começou em 2009, quando seus criadores identificaram uma lacuna significativa nas linguagens de programação existentes. A maioria das linguagens forçava os desenvolvedores a escolher entre performance e produtividade - um compromisso co-

nhecido como “the two-language problem”. Cientistas e pesquisadores frequentemente escreviam protótipos em uma linguagem de alto nível como Python ou MATLAB, apenas para terem que reescrever o código em C++ ou Fortran quando precisavam de maior performance.

Julia foi projetada desde o início para resolver este problema, oferecendo:

- Performance comparável a C e Fortran
- Sintaxe clara e acessível similar a Python
- Um sistema de tipos sofisticado e flexível
- Despacho múltiplo (multiple dispatch) como paradigma central
- Gerenciamento automático de memória
- Uma robusta biblioteca padrão
- Um ecossistema crescente de pacotes

A primeira versão estável (1.0) foi lançada em agosto de 2018, marcando um momento crucial para a adoção da linguagem em ambientes de produção.

1.3 Características Fundamentais

1.3.1 Performance Excepcional

Uma das características mais notáveis de Julia é sua performance excepcional. A linguagem utiliza um compilador JIT (Just-In-Time) baseado no LLVM, que permite que o código seja otimizado em tempo

de execução. Isso resulta em performance próxima ou igual a linguagens compiladas tradicionais, sem sacrificar a facilidade de desenvolvimento.

1.3.2 Multiple Dispatch

O sistema de despacho múltiplo de Julia é uma de suas características mais distintivas. Ele permite que diferentes métodos de uma função sejam chamados baseados nos tipos de todos os argumentos, não apenas do primeiro como em muitas outras linguagens orientadas a objetos. Isso proporciona uma flexibilidade extraordinária na implementação de algoritmos e contribui para a expressividade da linguagem.

1.3.3 Ecossistema Científico

Julia possui um rico ecossistema de pacotes científicos, muitos dos quais são desenvolvidos e mantidos por pesquisadores e instituições acadêmicas. Alguns dos pacotes mais notáveis incluem:

- `JuMP`: Para otimização matemática
- `Distributions.jl`: Para análise estatística
- `DifferentialEquations.jl`: Para resolução de equações diferenciais
- `DataFrames.jl`: Para manipulação de dados tabulares

1.4 Julia na Economia

1.4.1 Aplicações em Economia Computacional

A adoção de Julia na economia tem crescido significativamente nos últimos anos, especialmente em áreas que exigem computação intensiva. A linguagem tem se mostrado particularmente útil em:

Modelagem Macroeconômica

Julia é extensivamente utilizada para desenvolver e simular modelos macroeconômicos complexos (Aruoba and Fernández-Villaverde (2015)). Sua capacidade de lidar eficientemente com cálculos matriciais e álgebra linear torna-a ideal para:

- Modelos DSGE (Dynamic Stochastic General Equilibrium)
- Análise de séries temporais
- Simulações de Monte Carlo
- Estimação bayesiana

Microeconomia Computacional

Na microeconomia, Julia tem se destacado em:

- Teoria dos jogos computacional
- Modelagem de escolha discreta
- Análise de equilíbrio de mercado
- Otimização de portfolios

1.4.2 Casos de Uso Proeminentes

Banco Central Federal Reserve

O Federal Reserve adotou Julia para várias aplicações, incluindo:

- Modelagem macroeconômica
- Análise de estabilidade financeira
- Previsão econométrica
- Simulação de cenários de política monetária

Bancos Centrais Internacionais

Outros bancos centrais também têm adotado Julia:

- Banco Central Europeu: Para modelagem macroeconômica
- Banco da Inglaterra: Para simulações financeiras
- Banco do Brasil: Para análise de dados econômicos

1.5 Pacotes Econômicos

1.5.1 QuantEcon.jl

Desenvolvido pela QuantEcon Initiative, este pacote oferece ferramentas para:

- Modelagem econômica dinâmica
- Teoria dos jogos

- Análise de séries temporais
- Otimização econômica

1.5.2 EconModels.jl

Um pacote focado em:

- Modelos de equilíbrio geral
- Análise de bem-estar
- Simulações de política econômica

1.6 Vantagens para Economistas

1.6.1 Performance Superior

A performance superior de Julia é particularmente relevante para economistas que trabalham com:

- Grandes conjuntos de dados
- Simulações complexas
- Estimações intensivas computacionalmente
- Otimização em larga escala

1.6.2 Facilidade de Uso

Julia oferece várias vantagens para economistas:

- Sintaxe matemática intuitiva
- Integração seamless com LaTeX para documentação
- Suporte nativo para álgebra linear
- Ferramentas robustas de visualização

1.7 Desafios e Limitações

Apesar de suas muitas vantagens, Julia ainda enfrenta alguns desafios:

1.7.1 Maturidade do Ecossistema

Embora em rápido crescimento, o ecossistema de Julia ainda é menos maduro que o de linguagens mais estabelecidas como Python e R. Isto pode se manifestar em:

- Menor disponibilidade de pacotes específicos
- Documentação menos abrangente em alguns casos
- Menor quantidade de recursos de aprendizado

1.8 Futuro de Julia na Economia

O futuro de Julia na economia parece promissor, com várias tendências emergentes:

1.8.1 Adoção Crescente

- Mais instituições financeiras adotando Julia
- Crescimento do uso em pesquisa acadêmica
- Expansão do ecossistema de pacotes econômicos

1.9 Instalação e Configuração do Julia

A instalação e configuração adequada do Julia é um passo fundamental para começar a trabalhar com análises regionais. Esta seção apresenta os procedimentos necessários para diferentes sistemas operacionais e configurações específicas para economia regional.

1.9.1 Download e Instalação Básica

O processo de instalação do Julia pode ser realizado através dos seguintes passos:

1. Acesse o site oficial: https://julialang.org/downloads/
2. Escolha a versão apropriada para seu sistema operacional (Windows, macOS, ou Linux)
3. Baixe o instalador da versão estável mais recente
4. Execute o instalador seguindo as instruções específicas do seu sistema

1.9.2 Configuração do Ambiente

Após a instalação básica, é necessário configurar o ambiente de desenvolvimento:

```
# Abra o REPL do Julia e digite:
using Pkg
# Instale os pacotes essenciais
Pkg.add("CSV")
Pkg.add("DataFrames")
Pkg.add("Plots")
Pkg.add("Statistics")
```

1.9.3 Pacotes Específicos para Economia Regional

Para análises regionais, recomenda-se a instalação dos seguintes pacotes:

```
# Pacotes para an lise espacial
Pkg.add("GeoInterface")
Pkg.add("Shapefile")
Pkg.add("GeoStats")

# Pacotes para econometria espacial
Pkg.add("SpatialRegression")
Pkg.add("NearestNeighbors")

# Pacotes para visualiza  o
Pkg.add("Makie")
Pkg.add("GeoMakie")
```

1.9.4 Configuração do Editor

Para uma experiência de desenvolvimento mais produtiva, recomenda-se a configuração de um editor de código adequado:

- **VS Code com Extensão Julia**:
  1. Instale o Visual Studio Code
  2. Instale a extensão Julia
  3. Configure o caminho do executável Julia
  4. Configure o ambiente de trabalho
- **Jupyter Notebook com Kernel Julia**:

```
    # No REPL do Julia:
    using Pkg
    Pkg.add("IJulia")

```

1.9.5 Verificação da Instalação

Para verificar se a instalação foi bem-sucedida, execute o seguinte código:

```
# Teste b sico de funcionalidade
println("Julia vers o: ", VERSION)

# Teste de pacotes
using CSV, DataFrames, Plots
println("Pacotes b sicos carregados com sucesso!")

# Teste de computa  o num rica
x = rand(1000)
@time mean(x)  # Deve mostrar tempo de execu  o
```

1.9.6 Resolução de Problemas Comuns

Alguns problemas comuns e suas soluções:

- **Erro de Path**: Adicione Julia ao PATH do sistema
- **Conflitos de Versão**: Use ambientes isolados com `Project.toml`
- **Problemas de Memória**: Configure os limites de memória apropriados

```
# Configura  o de mem ria (se necess rio)
# Adicione ao arquivo de configura  o:
ENV["JULIA_MAX_NUM_THREADS"] = "auto"
```

1.9.7 Manutenção e Atualização

Para manter o ambiente Julia atualizado:

```
# Atualizar todos os pacotes
using Pkg
Pkg.update()

# Verificar status
Pkg.status()

# Testar ap s atualiza  es
Pkg.test("NomeDoPacote")
```

1.9.8 Boas Práticas de Configuração

Recomendações para uma configuração robusta:

1. Mantenha um arquivo `Project.toml` para cada projeto

2. Use controle de versão (git) para seus códigos
3. Documente as dependências e versões específicas
4. Configure backups automáticos do ambiente

```
# Exemplo de cria  o de novo projeto
using Pkg
Pkg.activate("MeuProjetoRegional")
Pkg.add(["DataFrames", "CSV", "Plots"])
```

1.10 Conclusão

Julia representa uma evolução significativa nas ferramentas disponíveis para economistas e pesquisadores quantitativos. Sua combinação única de performance e facilidade de uso, junto com seu crescente ecossistema de pacotes específicos para economia, a torna uma escolha cada vez mais atraente para análise econômica e pesquisa quantitativa.

A linguagem continua a evoluir e melhorar, com uma comunidade ativa e engajada contribuindo para seu desenvolvimento. Para economistas que trabalham com análises computacionalmente intensivas ou que buscam uma ferramenta mais eficiente para seu trabalho quantitativo, Julia oferece uma alternativa poderosa e promissora às linguagens tradicionais.

Capítulo 2

Econometria Espacial

2.1 Introdução

A econometria espacial tem se destacado como uma ferramenta fundamental para a análise de dados que apresentam dependência espacial. Como destacado por Elhorst (2014), a importância desta área tem crescido significativamente nas últimas décadas, especialmente devido à maior disponibilidade de dados georreferenciados e ao desenvolvimento de ferramentas computacionais mais sofisticadas.

Neste capítulo, abordaremos os principais modelos de econometria espacial, com foco especial nos modelos SAR (Spatial Autoregressive Model), SEM (Spatial Error Model) e SAC/SARAR (Spatial Autoregressive Combined Model). Utilizaremos como exemplo prático a análise de dados sobre a disseminação do vírus Zika no Brasil, seguindo o trabalho de Amaral et al. (2019).

2.2 Fundamentos Teóricos

2.2.1 Modelo SAR (Spatial Autoregressive Model)

O modelo SAR, também conhecido como modelo autorregressivo espacial, é especificado como:

$$y = \rho W y + X\beta + u, \quad u \sim N(0, \sigma^2) \tag{2.1}$$

onde:

- y é o vetor da variável dependente
- ρ é o parâmetro de autocorrelação espacial
- W é a matriz de pesos espaciais
- X é a matriz de variáveis explicativas
- β é o vetor de coeficientes
- u é o termo de erro

A estimação deste modelo é realizada via máxima verossimilhança, com a função de log-verossimilhança dada por:

$$\begin{aligned} logL(\beta, \rho, \sigma^2) = &-\frac{n}{2} log(2\pi\sigma^2) + log|I_n - \rho W| \\ &- \frac{1}{2\sigma^2}(y - X\beta)'(I_n - \rho W')(I_n - \rho W)(y - X\beta) \end{aligned} \tag{2.2}$$

2.2.2 Modelo SEM (Spatial Error Model)

O modelo SEM considera a dependência espacial no termo de erro:

$$y = X\beta + u, \quad u = \lambda W u + e, \quad e \sim N(0, \sigma^2) \tag{2.3}$$

onde λ é o parâmetro de autocorrelação espacial do erro. A função de log-verossimilhança para este modelo é:

$$logL(\beta, \lambda, \sigma^2) = \frac{n}{2} log(2\pi\sigma^2) + log|I_n - \lambda W| - \frac{1}{2\sigma^2}(y - X\beta)'(y - X\beta) \quad (2.4)$$

2.2.3 Modelo SAC/SARAR

O modelo SAC (também conhecido como SARAR) combina os dois tipos de dependência espacial:

$$y = \rho W_1 y + X\beta + u, \quad u = \lambda W_2 u + e, \quad e \sim N(0, \sigma^2) \quad (2.5)$$

A função de log-verossimilhança neste caso é:

$$logL(\beta, \rho, \lambda, \sigma^2) = \frac{n}{2} log(2\pi\sigma^2) + log|I_n - \rho W_1| + log|I_n - \lambda W_2| - \frac{1}{2\sigma^2}(y - \rho W_1 y - X\beta)'(y - \rho W_1 y - X\beta) \quad (2.6)$$

2.3 Efeitos Marginais

Uma característica importante dos modelos espaciais é que os coeficientes β não podem ser interpretados diretamente como efeitos marginais (LeSage and Pace (2009)). É necessário calcular:

2.3.1 Efeitos Diretos

$$\frac{1}{n}\sum_{i}\phi_{ii}\beta_r = \frac{1}{n}tr(I-\rho W)^{-1}\beta_r \tag{2.7}$$

2.3.2 Efeitos Indiretos

$$\frac{1}{n}\sum_{i}\sum_{j}\phi_{ij}\beta_r - \frac{1}{n}\sum_{i}\phi_{ii}\beta_r = \frac{1}{n}tr(I-\rho W)^{-1}\beta_r \tag{2.8}$$

2.3.3 Efeitos Totais

$$\frac{1}{n}\sum_{i}\sum_{j}\phi_{ij}\beta_r \tag{2.9}$$

2.4 Aplicação Prática: Análise da Disseminação do Vírus Zika

Utilizando os dados de Amaral et al. (2019), aplicamos os três modelos para analisar a disseminação do vírus Zika no Brasil. As variáveis explicativas incluem:

- ln_gdp: logaritmo do PIB
- ln_pop: logaritmo da população
- mobility: índice de mobilidade
- environ: índice ambiental
- sanitation: índice de saneamento

Os resultados da estimação do modelo SAR mostram um coeficiente de autocorrelação espacial (ρ) de 0,238, indicando a presença significativa de spillovers espaciais na disseminação do vírus. O modelo SEM apresenta um λ de 0,238, sugerindo dependência espacial nos erros. Já o modelo SAC apresenta $\rho = 0,085$ e $\lambda = 0,171$.

2.5 Considerações sobre Implementação Computacional

A implementação destes modelos requer cuidados especiais, especialmente no cálculo dos determinantes das matrizes $(I - \rho W)$ e na otimização da função de verossimilhança. Como destacado por Kelejian and Piras (2017), é importante utilizar métodos numéricos eficientes para estas operações.

2.6 Implementação Prática

2.6.1 Códigos

Os códigos usados para a estimação do SAR, SEM e SARAR são indicados logo a seguir:

- SAR

```
using Optim
using LinearAlgebra
using ForwardDiff
using Statistics
using Distributions
using Printf
```

```
7  using PrettyTables
8  function sar_likelihood(params,n,X,y,W)
9      sigma2,rho,beta = params[1],params[2], params[3:end]
10     epsilon = (I(n) - rho*W)*y-X*beta
11     loglik = -(n/2)*log(2*pi*sigma2) -((1/(2*sigma2))*(
       epsilon'*epsilon))+logdet(I(n)-rho*W)
12     return -loglik
13 end
14 function sar_coef(X,y,W)
15     n_x=size(X)[2]
16     n=size(X)[1]
17     initial_params = vcat(10,0.5,zeros(n_x))
18     lower_bounds = [0;-1;fill(-Inf,n_x)]
19     upper_bounds = [Inf;1;fill(Inf,n_x)]
20     result = optimize(params -> sar_likelihood(params, n,
       X,y,W),
21                       lower_bounds, upper_bounds,
       initial_params,Fminbox())
22     beta = result.minimizer
23     ll=-result.minimum
24     return  beta,ll
25 end
26 function sar_sdev(X,y,W,beta,n)
27     likelihood_beta_only = beta -> sar_likelihood(beta, n
       , X, y, W)
28     H=ForwardDiff.hessian(likelihood_beta_only, beta)
29     sigmak = ForwardDiff.value.(real.(H))\I
30     desvio_padrao=sqrt.(diag(sigmak))
31     return desvio_padrao
32 end
33 function sar_pvalor(y,X,coefs,desvios_padroes)
34     n=size(X)[1]
35     k=length(coefs)
36     dof = n - k
```

```
37      confidence_level = 0.95
38      t_value = quantile(TDist(dof), 1 - (1 -
    confidence_level) / 2)
39      confidence_intervals_lower=[(coef - t_value * std)
40          for (coef, std) in zip(coefs, desvios_padroes)]
41      confidence_intervals_upper=[(coef + t_value * std)
42          for (coef, std) in zip(coefs, desvios_padroes)]
43      p_values = [2 * (1 - cdf(TDist(dof), abs(coef / std))
    )
44          for (coef, std) in zip(coefs, desvios_padroes)]
45      return hcat(confidence_intervals_lower,
    confidence_intervals_upper,p_values)
46 end
47 function sar(X,y,W)
48      n=size(X)[1]
49      coefs,ll=sar_coef(X,y,W)
50      desvios_padroes=sar_sdev(X,y,W,coefs,n)
51      ic_pvalores=sar_pvalor(y,X,coefs,desvios_padroes)
52      sigma2=coefs[1]
53      rho=vcat(coefs[2],desvios_padroes[2])
54      coefs=hcat(coefs,desvios_padroes,ic_pvalores)[3:end
    ,:]
55      nobs=n
56      k=length(coefs[:,1])
57      dof=n-k
58      results=(coefs = coefs, sigma2 = sigma2, rho = rho,
59              nobs=nobs,dof=dof,ll=ll)
60      return results
61 end
```

- SEM

```
1 using Optim
2 using LinearAlgebra
3 using ForwardDiff
```

```
4 using Statistics
5 using Distributions
6 using Printf
7 using PrettyTables
8 function log_likelihood_sem(params,n,X,y,W)
9     log_det = logdet(I(n) - params[2] * W)
10    log_likelihood = -n/2 * log(2 * pi) - n/2 * log(
    params[1]) + log_det - (1 / (2 * params[1])) *(y-X*
    params[3:end])'* ((I(n)-params[2]*W') * (I(n)-params
    [2]*W))*(y-X*params[3:end])
11    return -log_likelihood
12 end
13 function sem_coefs(X,y,W)
14    n_x=size(X)[2]
15    n=size(X)[1]
16    initial_params = vcat(1,0.5,zeros(n_x))
17    lower_bounds = [0;-1;fill(-Inf,n_x)]
18    upper_bounds = [Inf;1;fill(Inf,n_x)]
19    result = optimize(params -> sar_likelihood(params,n,X
    ,y,W),
20                      lower_bounds, upper_bounds,
    initial_params,Fminbox())
21    beta = result.minimizer
22    ll=-result.minimum
23    return beta,ll
24 end
25 function sem_sdev(X,y,W,beta,n)
26    likelihood_beta_only = beta -> sar_likelihood(beta, n
    , X, y, W)
27    hessian_matrix = ForwardDiff.hessian(
    likelihood_beta_only, beta)
28    cov_matrix = inv(hessian_matrix)
29    std_devs = sqrt.(diag(cov_matrix))
30    return std_devs
```

```
31 end
32 function sem_pvalor(y,X,coefs,desvios_padroes)
33     n=size(X)[1]
34     k=length(coefs)
35     dof = n - k
36     confidence_level = 0.95
37     t_value = quantile(TDist(dof), 1 - (1 -
       confidence_level) / 2)
38     confidence_intervals_lower=[(coef - t_value * std)
39         for (coef, std) in zip(coefs, desvios_padroes)]
40     confidence_intervals_upper=[(coef + t_value * std)
41         for (coef, std) in zip(coefs, desvios_padroes)]
42     p_values = [2 * (1 - cdf(TDist(dof), abs(coef / std))
       )
43         for (coef, std) in zip(coefs, desvios_padroes)]
44     return hcat(confidence_intervals_lower,
       confidence_intervals_upper,p_values)
45 end
46 function sem(X,y,W)
47     n=length(y)
48     coefs,ll=sem_coefs(X,y,W)
49     sigma2=coefs[1]
50     desvios_padroes=sem_sdev(X,y,W,coefs,n)
51     ic_pvalores=sar_pvalor(y,X,coefs,desvios_padroes)
52     lambda=hcat(coefs[2],desvios_padroes[2])
53     coefs=hcat(coefs,desvios_padroes,ic_pvalores)[3:end
       ,:]
54     nobs=n=size(X)[1]
55     k=length(coefs[:,1])
56     dof=n-k
57     results=(coefs = coefs, sigma2 = sigma2, lambda =
       lambda,
58              nobs=nobs,dof=dof,ll=ll)
59     return results
```

```
60 end
```

- SARAR

```
1  using Optim
2  using LinearAlgebra
3  using ForwardDiff
4  using Statistics
5  using Distributions
6  using Printf
7  using PrettyTables
8
9  function log_likelihood_sarar(params,n,X,y,W,M)
10     I_n = I(n)
11     u = y - params[2] * W * y - X * params[4:end]
12     tilde_u = (I_n - params[3] * M) * u
13     log_det_W = logdet(I_n - params[2] * W)
14     log_det_M = logdet(I_n - params[3] * M)
15     quad_form = tilde_u' * tilde_u
16     log_likelihood = - (n / 2) * log(2 * pi) - (n / 2) *
       log(params[1]) +
17         log_det_W + log_det_M - (1 / (2 * params[1])) *
       quad_form
18     return -log_likelihood
19 end
20
21 function sarar_coef(y,X,W,M)
22     n_x=size(X)[2]
23     n=length(y)
24     initial_params = vcat(1,0.5,0.5,zeros(n_x))
25     lower_bounds = [0;-1;-1;fill(-Inf,n_x)]
26     upper_bounds = [Inf;1;1;fill(Inf,n_x)]
27     result = optimize(params -> log_likelihood_sarar(
       params, n,X,y,W,M),
28                       lower_bounds, upper_bounds,
```

```
    initial_params,Fminbox())
29    beta = result.minimizer
30    ll=-result.minimum
31    return beta,ll
32 end
33
34 function sarar_std(X,y,W,M,beta,n)
35    likelihood_beta_only = beta -> log_likelihood_sarar(
      beta, n, X, y, W,M)
36    hessian_matrix = ForwardDiff.hessian(
      likelihood_beta_only, beta)
37    cov_matrix = inv(hessian_matrix)
38    std_devs = sqrt.(diag(cov_matrix))
39    return std_devs
40 end
41
42 function sarar_pvalor(y,X,coefs,desvios_padroes)
43    n=size(X)[1]
44    k=length(coefs)
45    dof = n - k
46    confidence_level = 0.95
47    t_value = quantile(TDist(dof), 1 - (1 -
      confidence_level) / 2)
48    confidence_intervals_lower=[(coef - t_value * std)
49        for (coef, std) in zip(coefs, desvios_padroes)]
50    confidence_intervals_upper=[(coef + t_value * std)
51        for (coef, std) in zip(coefs, desvios_padroes)]
52    p_values = [2 * (1 - cdf(TDist(dof), abs(coef / std)))
53        for (coef, std) in zip(coefs, desvios_padroes)]
54    return hcat(confidence_intervals_lower,
      confidence_intervals_upper,p_values)
55 end
56
57 function sarar(X,y,W,M)
```

```
58      n=length(y)
59      coefs,ll=sarar_coef(y,X,W,M)
60      desvios_padroes=sarar_std(X,y,W,M,coefs,n)
61      ic_pvalores=sar_pvalor(y,X,coefs,desvios_padroes)
62      sigma2=coefs[1]
63      rho=hcat(coefs[2],desvios_padroes[2])
64      lambda=hcat(coefs[3],desvios_padroes[3])
65      coefs=hcat(coefs,desvios_padroes,ic_pvalores)[4:end,:]
66      nobs=n=size(X)[1]
67      k=length(coefs[:,1])
68      dof=n-k
69      results=(coefs = coefs, sigma2 = sigma2, rho=rho,
        lambda=lambda,
70              nobs=nobs,dof=dof,ll=ll)
71      return results
72  end
```

2.6.2 Replicando Amaral et al. (2019)

Como forma de consolidar o conteúdo exposto anteriormente e também validar os códigos criados para esse livro, replicam-se os modelos existentes em Amaral et al. (2019). Os resultados são exibidos a seguir:

- SAR

```
1 sar_model=sar(X,y,W)
2 names_col=names(df)
3 sar_summary(sar_model,names_col)
4 .------------.---------.---------.----------.----------.----------.

5 Maximum Likelihood Estimation of SAR Model
6 .------------.---------.---------.----------.----------.----------.
```

```
Log−Likelihood: −737.902
Number of observations: 184
sigma2: 176.214
.−−−−−−−−−−−−.−−−−−−−−−.−−−−−−−−−.−−−−−−−−−−.−−−−−−−−−−.−−−−−−−−−−.
|   Variable   |      beta     | Std Dev | Lower CI | Upper CI | p−value  |
:−−−−−−−−−−−−+−−−−−−−−−+−−−−−−−−−+−−−−−−−−−−+−−−−−−−−−−+−−−−−−−−−−:
|   constant   | 16.627  | 39.880  | −62.078 |   95.332  | 0.677241 |
|    ln_gdp    | −2.600  |   6.923 | −16.263 |   11.063  | 0.707744 |
|    ln_pop    |  1.228  |   8.970 | −16.475 |   18.931  | 0.89124  |
|   mobility   | −14.590 | 35.789  | −85.221 |   56.041  | 0.684016 |
|   environ    |  5.229  | 11.160  | −16.796 |   27.254  | 0.639967 |
| sanitation   |  3.247  | 10.118  | −16.721 |   23.216  | 0.748625 |
'−−−−−−−−−−−−'−−−−−−−−−'−−−−−−−−−'−−−−−−−−−−'−−−−−−−−−−'−−−−−−−−−−'
rho:  0.238, Standard Deviation: 0.113
```

- SEM

```
sem_model=sem(X,y,W)
names_col=names(df)
sar_summary(sem_model,names_col)
.−−−−−−−−−−−−.−−−−−−−−−.−−−−−−−−−.−−−−−−−−−−.−−−−−−−−−−.−−−−−−−−−−.

Maximum Likelihood Estimation of SEM Model
.−−−−−−−−−−−−.−−−−−−−−−.−−−−−−−−−.−−−−−−−−−−.−−−−−−−−−−.−−−−−−−−−−.

Log−Likelihood: −737.902
Number of observations: 184
sigma2: 176.214
.−−−−−−−−−−−−.−−−−−−−−−.−−−−−−−−−.−−−−−−−−−−.−−−−−−−−−−.−−−−−−−−−−.
|   Variable   |      beta     | Std Dev | Lower CI | Upper CI | p−value  |
:−−−−−−−−−−−−+−−−−−−−−−+−−−−−−−−−+−−−−−−−−−−+−−−−−−−−−−+−−−−−−−−−−:
|   constant   | 16.627  | 39.880  | −62.078 |   95.332  | 0.677241 |
|    ln_gdp    | −2.600  |   6.923 | −16.263 |   11.063  | 0.707744 |
```

```
|    ln_pop    |   1.228  |   8.970  | −16.475 |   18.931  | 0.89124  |
|   mobility   | −14.590 | 35.789  | −85.221 |   56.041  | 0.684015 |
|   environ    |   5.229  |  11.160  | −16.796 |   27.254  | 0.639967 |
| sanitation |   3.247  |  10.118  | −16.721 |   23.216  | 0.748625 |
’−−−−−−−−−−−−’−−−−−−−−−’−−−−−−−−−’−−−−−−−−−−’−−−−−−−−−−’−−−−−−−−−−’
lambda: 0.238, Standard Deviation: 0.113
```

- SARAR

```
    M=W
    sarar_model=sarar(X,y,W,M)
    sarar_summary(sarar_model,names_col)
.−−−−−−−−−−−−.−−−−−−−−−.−−−−−−−−−.−−−−−−−−−−.−−−−−−−−−−.−−−−−−−−−−.

Maximum Likelihood Estimation of SAC Model
.−−−−−−−−−−−−.−−−−−−−−−.−−−−−−−−−.−−−−−−−−−−.−−−−−−−−−−.−−−−−−−−−−.

Log−Likelihood: −737.764
Number of observations: 184
sigma2: 176.676
.−−−−−−−−−−−−.−−−−−−−−−.−−−−−−−−−.−−−−−−−−−−.−−−−−−−−−−.−−−−−−−−−−.
|   Variable   |      beta      | Std Dev | Lower CI | Upper CI | p−value  |
:−−−−−−−−−−−−+−−−−−−−−−+−−−−−−−−−+−−−−−−−−−−+−−−−−−−−−−+−−−−−−−−−−:

|   constant   | 15.360  | 40.525  | −64.620 |   95.341  | 0.705119 |
|    ln_gdp    | −2.890 |   7.135  | −16.973 |   11.193  | 0.685943 |
|    ln_pop    |   1.486  |   9.109  | −16.491 |   19.463  | 0.870596 |
|   mobility   | −13.968 | 37.093  | −87.175 |   59.240  | 0.706961 |
|   environ    |   5.964  |  11.523  | −16.777 |   28.706  | 0.605369 |
| sanitation |   4.277  |  10.677  | −16.796 |   25.351  | 0.689199 |
’−−−−−−−−−−−−’−−−−−−−−−’−−−−−−−−−’−−−−−−−−−−’−−−−−−−−−−’−−−−−−−−−−’
rho:  0.085, Standard Deviation: 0.322
lambda: 0.171, Standard Deviation: 0.311
```

Os efeitos diretos e indiretos do SARAR podem ser estimados da

seguinte forma[1]:

```
beta_complet=vcat(sarar_model.sigma2,sarar_model.rho[1],sarar_model.lambda[1],
    sarar_model.coefs[:,1])
SARAR_effects=effects_sarar(y,X,W,M,beta_complet)
effects_summary(SARAR_effects,names_col[2:end])
.------------.----------------.------------------.---------------.
|  Variable  | Direct Effects | Indirect Effects | Total Effects |
:------------+----------------+------------------+---------------:
|   ln_gdp   |     -2.894     |      -0.264      |     -3.159    |
|   ln_pop   |     1.488      |      0.136       |     1.624     |
|  mobility  |    -13.987     |      -1.278      |    -15.265    |
|  environ   |     5.973      |      0.546       |     6.518     |
| sanitation |     4.283      |      0.391       |     4.675     |
'------------'----------------'------------------'---------------'
```

2.7 Conclusão

Os modelos de econometria espacial oferecem uma ferramenta poderosa para a análise de dados com dependência espacial. A escolha entre os modelos SAR, SEM e SAC deve ser baseada na natureza da dependência espacial presente nos dados e nos objetivos específicos da análise.

[1] O efeito total das variáveis do modelo SEM são dados por seus respectivos coeficientes, não há, pois, efeitos indiretos.

Capítulo 3

Insumo-Produto

3.1 Introdução

A análise de insumo-produto constitui um dos principais ramos da economia quantitativa aplicada, tendo sua origem no trabalho seminal de Wassily Leontief publicado em 1936 "Quantitative Input-Output Relations in the Economic System of the United States"e posteriormente expandido em 1941 com "The Structure of American Economy 1919-1939"(Vale and Perobelli (2020)).

3.2 Dados e Estrutura da Matriz de Insumo-Produto

A base fundamental para a análise de insumo-produto é a matriz de transações intersetoriais, que registra os fluxos de bens e serviços entre os diferentes setores da economia em termos monetários. Conforme apresentado em Vale and Perobelli (2020), os principais componentes são:

- Matriz de consumo intermediário ($\mathbf{Z}$): registra as transações entre os setores
- Vetor de demanda final ($\mathbf{y}$): consumo das famílias, governo, investimento e exportações
- Vetor de valor bruto da produção ($\mathbf{x}$): produção total de cada setor
- Vetor de valor adicionado ($\mathbf{v}$): remunerações, impostos e excedente operacional bruto

3.2.1 Matriz de Coeficientes Técnicos

O primeiro passo da análise é a construção da matriz de coeficientes técnicos ($\mathbf{A}$), que representa a tecnologia de produção da economia. Cada elemento a_{ij} é calculado como:

$$a_{ij} = \frac{z_{ij}}{x_j} \tag{3.1}$$

onde z_{ij} representa o fluxo monetário do setor i para o setor j, e x_j é o valor bruto da produção do setor j. O coeficiente a_{ij} indica quanto o setor j necessita do setor i para produzir uma unidade monetária.

3.2.2 Modelo Básico de Leontief

O modelo básico parte da identidade contábil:

$$\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{y} \tag{3.2}$$

que pode ser rearranjada para:

$$\mathbf{x} = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{y} \tag{3.3}$$

onde $(\mathbf{I} - \mathbf{A})^{-1}$ é a matriz inversa de Leontief, também conhecida como matriz de requerimentos totais.

3.2.3 Interpretação da Inversa de Leontief

A matriz inversa de Leontief ($\mathbf{B} = (\mathbf{I} - \mathbf{A})^{-1}$) captura os efeitos diretos e indiretos de mudanças na demanda final sobre a produção setorial. Cada elemento b_{ij} representa o aumento necessário na produção do setor i para atender a um aumento de uma unidade monetária na demanda final do setor j.

Por exemplo, se $b_{ij} = 0.5$, isso significa que para cada R$1,00 de aumento na demanda final do setor j, o setor i precisará aumentar sua produção em R$0,50 para atender aos efeitos diretos e indiretos dessa mudança através da cadeia produtiva.

A matriz inversa de Leontief constitui a base para diversas análises mais avançadas em insumo-produto, como o cálculo de multiplicadores, índices de ligação e análises de impacto. Sua correta interpretação é fundamental para compreender as interdependências setoriais na economia.

3.3 Multiplicadores de Insumo-Produto

Os multiplicadores são ferramentas fundamentais na análise de insumo-produto, permitindo avaliar o impacto total de variações na demanda final sobre variáveis como produção, emprego e renda. Conforme Vale and Perobelli (2020), podemos classificá-los em diferentes tipos, cada

um capturando aspectos específicos das relações intersetoriais.

3.3.1 Tipos de Multiplicadores

Multiplicadores de Produção

O multiplicador de produção é calculado a partir da matriz inversa de Leontief e pode ser dividido em:

Multiplicador Simples de Produção

Para um setor j, é definido como:

$$m_j^{(o)} = \sum_{i=1}^{n} b_{ij} \tag{3.4}$$

onde b_{ij} são os elementos da matriz inversa de Leontief. Este multiplicador representa o aumento total na produção da economia necessário para atender a um aumento de uma unidade monetária na demanda final do setor j.

Multiplicador Total de Produção

Calculado a partir do modelo fechado de Leontief (incluindo o consumo das famílias como endógeno):

$$m_j^{(o)} = \sum_{i=1}^{n+1} b_{ij}^{*} \tag{3.5}$$

onde b_{ij}^{*} são os elementos da matriz inversa de Leontief do modelo fechado.

3.3.2 Decomposição dos Efeitos

Os multiplicadores podem ser decompostos em diferentes efeitos:

- **Efeito Direto**: impacto imediato sobre o próprio setor
- **Efeito Indireto**: impacto sobre os demais setores da cadeia produtiva
- **Efeito Induzido**: impacto adicional gerado pelo aumento da renda e consumo das famílias

3.3.3 Multiplicadores de Emprego

Coeficientes de Emprego

Os multiplicadores de emprego partem dos coeficientes diretos de emprego:

$$c_j^e = \frac{v_j^e}{x_j} \tag{3.6}$$

onde v_j^e é o número de empregos no setor j e x_j é o valor bruto da produção.

Multiplicador Simples de Emprego

É calculado como:

$$ME_j = \sum_{i=1}^{n} c_i^e b_{ij} \tag{3.7}$$

Este multiplicador indica o número total de empregos gerados na economia para cada R$1 milhão de aumento na demanda final do setor j.

3.3.4 Multiplicadores de Renda

Coeficientes de Renda

Similarmente, partem dos coeficientes diretos de renda:

$$c_j^r = \frac{v_j^r}{x_j} \tag{3.8}$$

onde v_j^r representa a renda (remunerações) do setor j.

Multiplicador Simples de Renda

É calculado como:

$$MR_j = \sum_{i=1}^{n} c_i^r b_{ij} \tag{3.9}$$

3.3.5 Interpretação dos Multiplicadores

Vale and Perobelli (2020) enfatizam que os multiplicadores devem ser interpretados como limites superiores dos impactos, pois assumem:

- Oferta perfeitamente elástica dos fatores de produção
- Coeficientes técnicos fixos (retornos constantes de escala)
- Ausência de mudanças tecnológicas no curto prazo

3.3.6 Multiplicadores Tipo I e Tipo II

Multiplicadores Tipo I

Relacionam os efeitos diretos e indiretos com os efeitos diretos:

$$M_j^I = \frac{\text{Efeitos Diretos + Indiretos}}{\text{Efeitos Diretos}} \tag{3.10}$$

Multiplicadores Tipo II

Incluem também os efeitos induzidos:

$$M_j^{II} = \frac{\text{Efeitos Diretos + Indiretos + Induzidos}}{\text{Efeitos Diretos}} \tag{3.11}$$

Os multiplicadores são instrumentos essenciais para quantificar os impactos de variações na demanda final sobre diferentes variáveis econômicas. Sua correta interpretação permite avaliar a importância relativa dos setores e orientar políticas públicas.

3.4 Índices de Ligação na Análise de Insumo-Produto: Mensuração de Encadeamentos Setoriais

Os índices de ligação são métricas fundamentais na análise de insumo-produto que permitem identificar setores-chave na economia através da mensuração de suas interdependências. Baseado em Rasmussen (1952) e Hirschman (1958), estes índices avaliam o poder de encade-

amento dos setores tanto como demandantes quanto como fornecedores de insumos.

3.4.1 Índices de Ligação de Hirschman-Rasmussen

Conforme Vale and Perobelli (2020), os índices são calculados a partir da matriz inversa de Leontief ($\mathbf{B}$), considerando:

- b_{ij}: elementos da matriz inversa de Leontief
- B^*: média de todos os elementos de $\mathbf{B}$
- $b_{.j}$: soma dos elementos da coluna j
- $b_{i.}$: soma dos elementos da linha i

3.4.2 Índice de Ligação para Trás

O índice de ligação para trás (poder de dispersão) é calculado como:

$$U_j = \frac{b_{.j}/n}{B^*} \tag{3.12}$$

onde:

- $U_j > 1$ indica que o setor j tem um poder de encadeamento para trás acima da média
- $U_j < 1$ indica poder de encadeamento para trás abaixo da média

Este índice mede o quanto um setor demanda dos outros setores da economia. Um valor maior que 1 indica que uma mudança unitária na demanda final do setor j gera um aumento acima da média na economia.

3.4.3 Índice de Ligação para Frente

O índice de ligação para frente (sensibilidade da dispersão) é calculado como:

$$U_i = \frac{b_{i.}/n}{B^*} \tag{3.13}$$

onde:

- $U_i > 1$ indica que o setor i tem um poder de encadeamento para frente acima da média
- $U_i < 1$ indica poder de encadeamento para frente abaixo da média

Este índice mede o quanto um setor é demandado pelos outros setores da economia. Um valor maior que 1 indica que o setor i é mais sensível que a média a mudanças na demanda dos outros setores.

3.5 Coeficientes de Variação

Para complementar a análise dos índices de ligação, utilizam-se coeficientes de variação que mostram como os impactos se distribuem entre os setores:

3.5.1 Coeficiente de Variação das Ligações para Trás

$$V_{.j} = \frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}[b_{ij} - (b_{.j}/n)]^2}}{b_{.j}/n} \tag{3.14}$$

3.5.2 Coeficiente de Variação das Ligações para Frente

$$V_{i.} = \frac{\sqrt{\frac{1}{n-1}\sum_{j=1}^{n}[b_{ij} - (b_{i.}/n)]^2}}{b_{i.}/n} \tag{3.15}$$

Quanto menores os coeficientes de variação, mais uniformemente distribuídos são os impactos entre os setores.

3.6 Identificação de Setores-Chave

Um setor é considerado setor-chave na economia quando apresenta simultaneamente:

- $U_j > 1$ (forte encadeamento para trás)
- $U_i > 1$ (forte encadeamento para frente)

Vale and Perobelli (2020) destacam que estes setores contribuem acima da média para o crescimento da economia por possuírem fortes efeitos de encadeamento em termos do fluxo de bens e serviços.

3.7 Índices de Ligação usando a Matriz de Ghosh

Uma abordagem alternativa para calcular os índices de ligação para frente utiliza o modelo de oferta de Ghosh. Neste caso, o índice é calculado como:

$$U_i^G = \frac{g_{i.}/n}{G^*} \tag{3.16}$$

onde $g_{i.}$ representa a soma dos elementos das linhas da matriz inversa de Ghosh e G^* é a média dos elementos desta matriz.

Os índices de ligação são ferramentas essenciais para identificar setores estratégicos na economia e compreender as relações de interdependência setorial. A análise conjunta dos índices de ligação para frente e para trás, complementada pelos coeficientes de variação, permite uma avaliação abrangente da estrutura produtiva.

3.8 Método de Extração Hipotética em Insumo-Produto

O método de extração hipotética, conforme apresentado por Vale and Perobelli (2020), é uma técnica que permite avaliar a importância de um setor na estrutura econômica através da simulação de sua remoção do sistema produtivo. A magnitude da redução no nível de atividade econômica após a extração indica o grau de interdependência do setor com o resto da economia.

3.8.1 Abordagens de Extração

Vale e Perobelli (2020) destacam três possíveis formas de extração:

- Extração total do setor (colunas e linhas)
- Extração da estrutura de compras (ligação para trás - colunas)
- Extração da estrutura de vendas (ligação para frente - linhas)

3.8.2 Extração da Estrutura de Compras

3.8.3 Formulação Matemática

Partindo do modelo básico de Leontief:

$$\mathbf{x} = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{y} \tag{3.17}$$

A extração da estrutura de compras do setor j implica em:

$$\mathbf{x}^{(j)*} = (\mathbf{I} - \mathbf{A}^{(j)*})^{-1}\mathbf{y} \tag{3.18}$$

onde $\mathbf{A}^{(j)*}$ é a matriz com a j-ésima coluna zerada.

O impacto total da extração é medido por:

$$t^{(j)} = \mathbf{i}'\mathbf{x} - \mathbf{i}'\mathbf{x}^{(j)*} \tag{3.19}$$

onde $t^{(j)}$ representa a redução total na produção da economia devido à extração do setor j.

3.8.4 Extração da Estrutura de Vendas

Formulação pelo Lado da Oferta

Utilizando o modelo de Ghosh:

$$\mathbf{x}' = \mathbf{v}(\mathbf{I} - \mathbf{F})^{-1} \tag{3.20}$$

A extração da estrutura de vendas do setor i resulta em:

$$\mathbf{x}^{(i)*} = \mathbf{v}(\mathbf{I} - \mathbf{F}^{(i)*})^{-1} \tag{3.21}$$

onde $\mathbf{F}^{(i)*}$ é a matriz com a i-ésima linha zerada.

O impacto total é medido por:

$$t^{(i)} = \mathbf{xi} - \mathbf{x}^{(i)*}\mathbf{i} \tag{3.22}$$

3.9 Medidas Percentuais

Para facilitar comparações, os impactos podem ser normalizados como percentual da produção total:

$$T^{(j)} = \frac{\mathbf{i}'\mathbf{x} - \mathbf{i}'\mathbf{x}^{(j)*}}{\mathbf{i}'\mathbf{x}} \times 100 \tag{3.23}$$

$$T^{(i)} = \frac{\mathbf{xi} - \mathbf{x}^{(i)*}\mathbf{i}}{\mathbf{i}'\mathbf{x}} \times 100 \tag{3.24}$$

3.10 Interpretação dos Resultados

Vale e Perobelli (2020) enfatizam que os resultados devem ser interpretados como:

- **Extração para trás**: mede a dependência dos outros setores em relação aos insumos do setor extraído
- **Extração para frente**: mede a dependência do setor extraído em relação à demanda dos outros setores
- Quanto maior o impacto da extração, maior é a importância do setor para a economia

3.10.1 Vantagens e Limitações

3.10.2 Vantagens

- Permite quantificar a importância relativa dos setores
- Captura tanto efeitos para frente quanto para trás
- Resultados de fácil interpretação

Limitações

- Assume que a extração não altera os coeficientes técnicos dos demais setores
- Não considera possíveis substituições entre insumos
- Representa um caso extremo de cessação completa das atividades setoriais

3.10.3 Conclusão

O método de extração hipotética é uma ferramenta poderosa para avaliar a importância relativa dos setores na economia. Sua aplicação permite identificar setores estratégicos cujo funcionamento é crucial para o desempenho econômico global, fornecendo subsídios importantes para a formulação de políticas públicas.

3.11 Implementação na Julia

As próximas páginas detalham a implementação dos métodos de insumo-produto anteriormente descritos na linguagem de programação Julia.

3.11.1 Inversa de Leontief

Considere a matriz de coeficientes técnicos A, de tamanho nxn. Então, em Julia, sua inversa pode ser calculada com o comando inv:

```
1     inv(I(n)-A)
```

Em que $I(n)$ é a matriz identidade de ordem $n \times n$.

3.11.2 Multiplicadores

3.11.3 Multiplicadores de Emprego

```
1  using LinearAlgebra
2  function multiplicadores_emprego(e,x,B,Bf,Setores)
3      E=Diagonal(e./x)*B
4      multiplicador_emprego_simples=sum(E,dims=1)'
5      multiplicador_emprego_tipoi=multiplicador_emprego_simples
      ./(e./x)
6      EF = Diagonal(e./x)*Bf[1:end-1, 1:end-1]
7      multiplicador_emprego_truncado=sum(EF,dims=1)'
8      multiplicador_emprego_tipoii=
      multiplicador_emprego_truncado./(e./x)
9      return hcat(Setores,multiplicador_emprego_simples,
      multiplicador_emprego_tipoi,multiplicador_emprego_truncado,
      multiplicador_emprego_tipoii)
10 end
```

3.11.4 Multiplicadores de Produção

```
1  using LinearAlgebra
2  function multiplicadores_produ  o(A,B,Bf,setores)
3      n=size(A)[1]
```

```
    eftfechado=sum(Bf[i,1:n] for i=1:n+1)
    eftaberto=sum(B[i,:] for i=1:n)
    eftinduzido=sum(Bf[i,1:n] for i=1:n+1)-sum(B[i,:] for i=1:
   n)
    eftdireto=sum(A[i,:] for i=1:n)
    eftindireto=sum(B[i,:] for i=1:n)-sum(A[i,:] for i=1:n)
    return hcat(setores,eftfechado,eftaberto,eftinduzido,
   eftdireto,eftindireto)
end
```

3.11.5 Multiplicadores de Produção Truncado

```
using LinearAlgebra
function multiplicadores_produ o_truncado(A,B,Bf,setores)
    n=size(A)[1]
    eftfechado=sum(Bf[i,1:n] for i=1:n)
    eftaberto=sum(B[i,:] for i=1:n)
    eftinduzido=sum(Bf[i,1:n] for i=1:n)-sum(B[i,:] for i=1:n)
    eftdireto=sum(A[i,:] for i=1:n)
    eftindireto=sum(B[i,:] for i=1:n)-sum(A[i,:] for i=1:n)
    return hcat(setores,eftfechado,eftaberto,eftinduzido,
   eftdireto,eftindireto)
end
```

3.11.6 Índices de Ligação para frente e para trás

Índices de Ligação para trás

```
using LinearAlgebra
function lig_tras(B,setores)
    n=size(B)[1]
    lig=(sum(B[i,:] for i=1:n)/n)/(sum(sum(B[i,j] for i=1:n)
   for j=1:n)/(n^2))
```

```
5     return hcat(setores,lig)
6 end
```

Índices de Ligação para frente

```
1 using LinearAlgebra
2 function lig_frente(B,setores)
3     n=size(B)[1]
4     lig=(sum(B[:,j] for j=1:n)/n)/(sum(sum(B[i,j] for i=1:n)
    for j=1:n)/(n^2))
5     return hcat(setores,lig)
6 end
```

3.11.7 Extração Hipotética

```
1 using LinearAlgebra
2 function extracao_hipotetica(A,setor,y)
3     B=inversa_leontief(I(size(A)[1])-A)
4     A_hip=copy(A)
5     A_hip[setor,:].=0
6     A_hip[:,setor].=0
7     B_hip=inversa_leontief(I(size(A)[1])-A_hip)
8     impacto_elect_gas_por_cento=((ones(size(A)[1])'*(B*y)-ones
    (size(A)[1])'*(B_hip*y))/(ones(size(A)[1])'*(B*y)))*100
9     impacto_elect_gas_absoluto=ones(size(A)[1])'*(B*y)-ones(
    size(A)[1])'*(B_hip*y)
10    return hcat(impacto_elect_gas_absoluto,
    impacto_elect_gas_por_cento)
11 end
```

Capítulo 4

Equilíbrio Geral

4.1 Introdução

O *exact hat algebra* representa uma das mais elegantes e poderosas ferramentas para análise de políticas comerciais em modelos de equilíbrio geral. Desenvolvida originalmente por Dekle, Eaton e Kortum (2007) e posteriormente refinada por Caliendo and Parro (2015), esta metodologia permite que pesquisadores analisem os efeitos de choques econômicos sem a necessidade de calibrar todos os parâmetros do modelo - uma limitação significativa dos modelos CGE (*Computable General Equilibrium*) tradicionais.

Este capítulo apresenta uma aplicação prática do *exact hat algebra* para analisar os impactos de acordos comerciais sobre o uso da terra no Brasil, baseando-se no trabalho de Leal (2025). A metodologia será explicada de forma didática, desde seus fundamentos teóricos até sua implementação computacional em Julia, proporcionando ao leitor ferramentas práticas para desenvolver suas próprias aplicações.

4.2 O Que É *Exact Hat Algebra*?

4.2.1 Conceito Fundamental

Imagine que você quer estudar como uma redução nas tarifas de importação afeta o PIB, o bem-estar e o uso da terra em diferentes regiões de um país. Uma abordagem tradicional exigiria que você:

1. Calibrasse completamente um modelo da economia, estimando dezenas de parâmetros estruturais
2. Resolvesse o equilíbrio inicial (baseline) do modelo
3. Implementasse o choque de política
4. Resolvesse o novo equilíbrio
5. Comparasse os dois estados

O *exact hat algebra* revoluciona essa abordagem ao trabalhar diretamente com **mudanças relativas** (ou "hats") das variáveis endógenas. Ao invés de resolver $x' = f(p', w', ...)$ para o novo equilíbrio, resolvemos $\hat{x} = g(\hat{p}, \hat{w}, ...)$, onde $\hat{x} = x'/x$ representa a mudança proporcional da variável x.

4.2.2 A Magia dos "Hats"

A beleza desta abordagem reside no fato de que muitos parâmetros estruturais que são difíceis de estimar individualmente desaparecem quando trabalhamos com mudanças relativas. Por exemplo, considere uma função de produção Cobb-Douglas:

$$Y = AK^{\alpha}L^{1-\alpha}$$

Em níveis, precisaríamos conhecer A, K, L, e α. Mas trabalhando com mudanças relativas:

$$\hat{Y} = \hat{A}\hat{K}^{\alpha}\hat{L}^{1-\alpha}$$

Aqui, o parâmetro de produtividade A desaparece completamente da equação! Precisamos apenas do share α (que pode ser observado diretamente nos dados) e das mudanças relativas dos insumos.

4.2.3 Vantagens Metodológicas

O *exact hat algebra* oferece várias vantagens sobre métodos alternativos:

- **Parcimônia paramétrica**: Reduz drasticamente o número de parâmetros que precisam ser estimados externamente
- **Robustez**: Os resultados são menos sensíveis a erros de especificação em parâmetros não-identificados
- **Transparência**: A estrutura do modelo fica mais clara ao focar nas elasticidades e shares observáveis
- **Escalabilidade**: Permite trabalhar com modelos de alta dimensionalidade (muitas regiões e setores)
- **Flexibilidade**: Facilita a implementação de diferentes tipos de choques e fechamentos do modelo

4.3 Estrutura Teórica do Modelo

4.3.1 Economia Multi-Regional e Multi-Setorial

O modelo desenvolvido por Leal (2025) estende a abordagem de Caliendo et al. (2018) para analisar especificamente os efeitos de acordos comerciais sobre o uso da terra. A economia é composta por N regiões (incluindo 27 estados brasileiros e 43 países) e J setores produtivos.

A inovação metodológica principal reside no **fechamento do modelo**: enquanto a literatura tradicional assume mobilidade de trabalho entre regiões, este modelo endogeniza o uso da terra e estruturas (H_n) mantendo o trabalho (L_n) fixo. Esta escolha reflete uma perspectiva de curto prazo mais realista para o contexto brasileiro, onde:

- Os custos de migração inter-regional são substanciais
- O mercado de trabalho apresenta rigidezes institucionais
- O uso da terra (especialmente na Amazônia Legal) pode se expandir rapidamente através do desmatamento

4.3.2 Consumidores

Os consumidores derivam utilidade de uma função Cobb-Douglas sobre bens finais:

$U_n = \prod_{j=1}^{J} (C_n^j)^{\alpha_n^j}$

onde α_n^j representa o share do setor j no consumo da região n, com $\sum_{j=1}^{J} \alpha_n^j = 1$.

A renda per capita é dada por: $I_n = w_n + \frac{r_n H_n - S_n}{L_n}$

onde w_n é o salário, r_n é a taxa de aluguel da terra, H_n é o estoque de terra, S_n é o superávit comercial, e L_n é a população.

4.3.3 Tecnologia de Produção

A tecnologia apresenta uma estrutura aninhada Cobb-Douglas. Bens intermediários são produzidos usando terra/estruturas, trabalho, e insumos intermediários:

$q_n^j(z_n^j) = z_n^j[T_n^j h_n^j(z_n^j)^{\beta_n} l_n^j(z_n^j)^{1-\beta_n}]^{\gamma_n^j} \prod_{k=1}^{J} M_n^{jk}(z_n^k)^{\gamma_n^{jk}}$

onde:

- z_n^j é um choque de produtividade específico à região e setor
- T_n^j é o parâmetro de produtividade fundamental
- β_n é o share da terra na produção
- γ_n^j é o share do valor adicionado
- γ_n^{jk} é o share do insumo intermediário do setor k na produção do setor j

4.3.4 Comércio Internacional e Inter-regional

O comércio segue a estrutura gravitacional padrão com custos iceberg. O preço final do bem j produzido na região i e consumido na região n é:

$p_n^j(z^j) = \min_i \left\{ \frac{\kappa_{ni}^j x_i^j}{z_i^j (T_i^j)^{\gamma_i^j}} \right\}$

onde $\kappa_{ni}^j \geq 1$ representa o custo iceberg de transporte/tarifa do setor j da região i para n.

Assumindo que a produtividade segue uma distribuição Fréchet, as shares de importação são dadas por:

$\pi_{ni}^j = \frac{(\kappa_{ni}^j x_i^j)^{-\theta_j} (T_i^j)^{\gamma_i^j \theta_j}}{\sum_{m=1}^{N} (\kappa_{nm}^j x_m^j)^{-\theta_j} (T_m^j)^{\gamma_m^j \theta_j}}$

onde θ_j é o parâmetro de dispersão da distribuição Fréchet (elasticidade de substituição).

4.4 Implementação do *Exact Hat Algebra*

4.4.1 Sistema de Equações em Mudanças Relativas

O equilíbrio do modelo é caracterizado por um sistema de seis equações principais, todas expressas em termos de mudanças relativas:

1. **Índice de preços**: Mudanças nos preços finais
2. **Custos dos insumos**: Mudanças nos custos de produção
3. **Shares de comércio**: Evolução dos padrões comerciais
4. **Mobilidade de fatores**: Realocação de terra entre regiões
5. **Equilíbrio no mercado de bens**: Oferta e demanda de bens finais
6. **Equilíbrio comercial**: Balança comercial regional

4.4.2 Algoritmo de Solução

O algoritmo iterativo procede em etapas:

1. **Inicialização**: Parte de um *guess* inicial para $\hat{\omega}_n$ (mudança relativa no custo dos fatores)
2. **Loop interno**: Resolve simultaneamente o sistema de preços e custos até convergência:

- Calcula novos índices de preços $\hat{P}_n^j$
- Atualiza custos dos insumos $\hat{x}_n^j$
- Repete até $|\hat{P}_n^j - \hat{P}_{n,anterior}^j| < \epsilon$

3. **Atualização das shares**: Calcula novas shares de comércio $\pi_{ni}^{j'}$
4. **Realocação de terra**: Resolve para $\hat{H}_n$ usando as condições de equilíbrio do mercado de fatores
5. **Mercado de bens**: Calcula produção setorial $X_n^{j'}$ via inversão matricial
6. **Novo guess**: Obtém $\hat{\omega}_n$ atualizado das condições de equilíbrio
7. **Convergência**: Repete até $|\hat{\omega}_n - \hat{\omega}_{n,anterior}| < \epsilon$

4.4.3 Desafios Computacionais

A implementação prática enfrenta vários desafios:

- **Dimensionalidade**: Com 70 regiões e 26 setores, o sistema envolve milhares de variáveis
- **Não-linearidade**: As equações são altamente não-lineares, exigindo métodos robustos
- **Múltiplos equilíbrios**: Podem existir equilíbrios múltiplos[1], requerendo análise de sensibilidade
- **Estabilidade numérica**: Pequenas perturbações podem levar a resultados divergentes

[1] Situação na qual um equilíbrio de uma ou mais equações não se encontra unicamente caracterizado. Este é o caso, pro exemplo, quando há algum tipo de externalidade.

4.5 Aplicação: Acordo Comercial UE-Mercosul

4.5.1 Cenários de Política

Leal (2025) analisa dois cenários principais:

Cenário I - Livre Comércio Irrestrito:

- Redução uniforme de 23,54% nos custos de comércio ($\hat{\kappa}_{ni}^{j} = 0,7646$)
- Aplicável a todos os setores comercializáveis
- Representa um acordo de livre comércio "tradicional"

Cenário II - Acordo Comercial Contingente:

- Redução de 23,54% nos setores não-primários
- Aumento de 50% nos custos para setores primários ($\hat{\kappa}_{ni}^{j} = 1,5$)
- Reflete preocupações ambientais da UE (similar ao EUDR)

4.5.2 Resultados Principais

Os resultados revelam dinâmicas complexas e contra-intuitivas:

- **Bem-estar**: Aumenta em todos os estados da Amazônia Legal nos dois cenários
- **Uso da terra**: Paradoxalmente, pode diminuir mesmo com livre comércio
- **Heterogeneidade regional**: Estados do interior (Amazonas, Mato Grosso) respondem diferentemente da fronteira agrícola

- **Efeitos de segunda ordem**: Linkages setoriais e regionais geram resultados não-óbvios

4.5.3 Mecanismos Econômicos

O modelo revela três mecanismos principais:

1. **Realocação produtiva**: Livre comércio altera o mix de produtos, potencialmente reduzindo a intensidade de terra
2. **Linkages regionais**: Estados amazônicos podem "importar"insumos de outras regiões ao invés de expandir produção local
3. **Variedade e bem-estar**: Maior diversidade de produtos importados aumenta bem-estar mesmo com possível redução na produção agrícola

4.6 Extensões Metodológicas

4.6.1 Especificações Paramétricas Alternativas

Para isolar os mecanismos driving os resultados, Leal (2025) implementa especificações alternativas:

- **No Sectoral Linkages (NS)**: $\gamma_n^j = 1$ e $\gamma_n^{jk} = 0$ - elimina comércio de intermediários
- **No Brazilian Geography (NB)**: Mantém linkages internacionais mas elimina comércio inter-regional no Brasil
- **Modelo Completo**: Permite todos os linkages

Esta abordagem permite decompor os efeitos totais em componentes:

- Efeitos diretos de preços relativos
- Efeitos indiretos via linkages setoriais
- Efeitos indiretos via linkages espaciais

4.6.2 Medidas de Bem-Estar

O modelo calcula mudanças no bem-estar através de:

$\ln(\hat{U}_n) = \sum_{j=1}^{J} \alpha_n^j \left[\ln(\hat{A}_n^j) + \ln\left(\frac{\hat{\omega}_n}{\hat{x}_n^j}\right)\right]$

onde $\hat{A}_n^j$ representa mudanças na Produtividade Total dos Fatores:

$\ln(\hat{A}_n^j) = \ln\left(\frac{(\hat{T}_n^j)^{\gamma_n^j}}{(\hat{\pi}_{nn}^j)^{1/\theta_j}}\right)$

4.7 Implementação em Julia

4.7.1 Estrutura do Código

O código em Julia para implementar o *exact hat algebra* segue uma estrutura modular, com cada equação implementada como uma função separada. Esta abordagem facilita debugging, testes, e modificações futuras.

A estrutura principal inclui:

- **Funções das equações**: Uma função para cada equação do sistema
- **Loop principal**: Algoritmo iterativo de solução

- **Manipulação de dados**: Leitura e preparação dos dados
- **Pós-processamento**: Cálculo de métricas de interesse (bem-estar, PTF, etc.)

4.7.2 Considerações de Performance

Dadas as dimensões do problema (70×26×70 = 127,400 elementos apenas para π_{nij}), otimizações são cruciais:

- **Pré-alocação**: Arrays são pré-alocados para evitar garbage collection
- **Broadcasting**: Operações vetorizadas onde possível
- **Tratamento de casos especiais**: Valores infinitos, NaN, e zeros são tratados explicitamente
- **Convergência adaptativa**: Critérios de convergência ajustáveis por loop

4.7.3 Robustez Numérica

O algoritmo implementa várias verificações de robustez:

- **Bounds checking**: Variáveis são mantidas dentro de intervalos economicamente sensíveis
- **Fallback procedures**: Métodos alternativos quando a convergência falha
- **Random perturbations**: Pequenas perturbações aleatórias ajudam a escapar de pontos de sela

- **Damping**: Fatores de amortecimento (0.05 no código) estabilizam a convergência

4.8 Limitações e Extensões Futuras

4.8.1 Limitações Metodológicas

Apesar de sua elegância, o *exact hat algebra* apresenta algumas limitações:

- **Análise estática**: Não captura dinâmicas de transição
- **Fechamento específico**: Resultados dependem criticamente das escolhas de mobilidade de fatores
- **Linearização local**: Válido apenas para pequenos choques
- **Estrutura paramétrica**: Assume formas funcionais específicas (Cobb-Douglas, CES)

4.8.2 Extensões Promissoras

Várias extensões metodológicas são possíveis:

- **Dinâmica**: Incorporar ajustamentos temporais e formação de capital
- **Incerteza**: Modelar choques estocásticos e análise de risco
- **Heterogeneidade**: Incluir firmas e trabalhadores heterogêneos
- **Inovação endógena**: Modelar mudanças tecnológicas induzidas por políticas

- **Economia ambiental**: Incorporar explicitamente externalidades ambientais

4.9 Conclusões

O *exact hat algebra* representa uma ferramenta poderosa para análise de políticas em modelos de equilíbrio geral, oferecendo um meio-termo elegante entre a parcimônia de modelos teóricos e o realismo de modelos aplicados. Sua implementação no contexto de comércio e uso da terra ilustra tanto o potencial quanto os desafios desta abordagem.

A aplicação de Leal (2025) demonstra que questões aparentemente simples - como "o livre comércio aumenta o desmatamento?- podem ter respostas complexas e contra-intuitivas quando analisadas através de modelos de equilíbrio geral rigorosos. Os resultados sugerem que a relação entre comércio e meio ambiente é mediada por linkages econômicos que só podem ser capturados através de modelagem estrutural.

Para economistas regionais, o *exact hat algebra* oferece uma metodologia escalável e transparente para analisar políticas públicas em contextos multi-regionais. Sua implementação em Julia, com as funções apresentadas a seguir, torna esta ferramenta acessível para pesquisadores interessados em desenvolver suas próprias aplicações.

4.10 Anexo: Implementação das Equações em Julia

As seis equações principais do modelo são implementadas nas seguintes funções Julia:

1. Equação 1:

```
function eq1(N::Int, J::Int, pi_nij::Array{Float64, 3},
   hat_kappa_nij::Array{Float64, 3}, hat_x_ij::Array{
   Float64, 2}, theta_j::Array{Float64, 1}, hat_T_nj::
   Array{Float64, 2}, gamma_nj::Array{Float64, 2})
    a = zeros(N, J)
    for n in 1:N
        for j in 1:J
            a[n, j] = sum(pi_nij[n, :, j] .* (
   hat_kappa_nij[n, :, j] .* hat_x_ij[:, j]).^(-theta_j[j
   ]) .* (hat_T_nj[:, j]).^(gamma_nj[n, j] .* theta_j[j])
   ).^(-1/theta_j[j])
        end
    end
    return a
end
```

2. Equação 2:

```
function eq2(N::Int, J::Int, hat_omega_n::Vector{Float64
   }, gamma_nj::Matrix{T}, hat_P_nj::Matrix{T}, gamma_njk
   ::Array{T,3}) where T<:AbstractFloat
    a = zeros(T, N, J)
    for n in 1:N
        for j in 1:J
            a[n, j] = hat_omega_n[n]^gamma_nj[n, j] *
   prod(hat_P_nj[n, :] .^ gamma_njk[n, j, :])
        end
    end
    return a
end
```

3. Equação 3:

```
function eq3(N::Int, J::Int, pi_nij::Array{Float64,3},
   hat_P_nj::Matrix{Float64},
```

```
    hat_kappa_nij::Array{Float64,3}, hat_x_ij::Matrix{Float64},
    theta_j::Vector{Float64}, hat_T_nj::Matrix{Float64},
    gamma_nj::Matrix{Float64})
    a = zeros(N, N, J)
    for n in 1:N, i in 1:N, j in 1:J
        a[n, i, j] = pi_nij[n, i, j] * (hat_P_nj[n, j] / (hat_kappa_nij[n, i, j] * hat_x_ij[i, j]))^theta_j[j] * hat_T_nj[n, j]^(gamma_nj[i, j] * theta_j[j])
    end
    return a
end
```

4. Equação 4:

```
function eq4_h_endog(N,hat_H_n, hat_omega_n, phi_n, hat_P_n, hat_S_n, hat_L_n, beta_n, Lb)
    num = zeros(27, 1)
    den = zeros(27, 1)
    hat_Ub=zeros(N-26)
    hat_Ub[1] = sum(Ln[1:27]./Lb[1].*(1 ./phi_n[1:27]).*(hat_omega_n[1:27].*hat_H_n[1:27].^(beta_n[1:27]).*hat_L_n[1:27].^(1 .-beta_n[1:27])./hat_P_n[1:27].-((1 .-phi_n[1:27])./phi_n[1:27]).*hat_S_n[1:27]./hat_P_n[1:27])
    #hat_Ub[1]=1
    function equation_system!(F, hat_H_n)
    for n=1:27
        den[n] = (Lb[1]) * (hat_omega_n[n] / (phi_n[n] * hat_P_n[n] * hat_Ub[1] + (1 - phi_n[n]) * (hat_S_n[n] / hat_L_n[n]))) ^ (1 / beta_n[n])
        num[n] = hat_L_n[n] * sum(Ln[1:27] .*hat_H_n[1:27] .* ((hat_omega_n[1:27]) ./ (phi_n[1:27] .* hat_P_n[1:27] .* hat_Ub[1] .+ (1 .- phi_n[1:27]) .* (hat_S_n[1:27] ./ hat_L_n[1:27]))) .^ (1 ./ beta_n
```

```
    [1:27]))
11          F[n]=hat_H_n[n]-num[n]/den[n]
12      end
13      end
14
15      # Create an initial guess for the logarithm of the
    variables
16      x0 = ones(27)
17
18      # Solve the system of equations
19      results = nlsolve(equation_system!, x0, ftol=1e-6)
20      a=results.zero
21      return a
22  end
```

5. Equação 5:

```
1  function eq5_H_var(N, J, alpha_nj, hat_omega_n,hat_L_n,
    hat_H_n, beta_n, In,Ln, Sn, Sn_l,gamma_njk, pi_nij)
2      coefs_A = zeros(N, N, J, J)
3      for n in 1:N,j in 1:J,k in 1:J,i in 1:N
4                      coefs_A[n, i, k, j] = gamma_njk[n, k,
    j]*pi_nij[i, n, k]
5      end
6      # Create the A matrix
7      A = I(N*J) - reshape(permutedims(coefs_A, (1, 3, 2,
    4)), (N * J, N * J))
8      A[isinf.(A)].=0
9      A[isnan.(A)].=0
10     # Create the c vector
11     c = zeros(N, J)
12     for n in 1:N
13         for j in 1:J
14             c[n, j] = alpha_nj[n, j] * (hat_omega_n[n] *
    hat_H_n[n]^(beta_n[n]) * hat_L_n[n]^(1 - beta_n[n]) *
```

```
    (In[n] * Ln[n] + Sn[n] - Sn_l[n]))
15          end
16      end
17      c = reshape(c, (N * J, 1))
18      # Solving the system:
19      return reshape(A\c,(N,J))
20  end
```

6. Equação 6:

```
1  function eq6_H_var(N, J, gamma_nj, pi_nij, X_nj, hat_L_n,
       hat_H_n,beta_n, In, Ln, Sn)
2      num=zeros(N)
3      den=zeros(N)
4      a=zeros(N)
5      for n in 1:N
6          num[n] = sum(gamma_nj[n,j]*sum(pi_nij[i,n,j]*X_nj
       [i,j] for i=1:N) for j=1:J)
7          den[n] = hat_H_n[n]^(beta_n[n])*hat_L_n[n]^(1 -
       beta_n[n]) * (In[n] * Ln[n] + Sn[n])
8          a[n]=num[n]/den[n]
9      end
10     return a
11 end
```

Capítulo 5

Operações Vetoriais e Mapas na Julia

5.1 Introdução

Operações vetoriais e mapas são uma parte crucial do dia-a-dia do economista regional. Dessa forma, esse capítulo almeja exibir alguns dos principais códigos usados para realizar operações entre diferentes vetores e a criação de mapas.

5.2 Manipulação de Dados Espaciais

A análise econômica regional frequentemente requer o processamento e análise de dados espaciais. Julia oferece um rico ecossistema de pacotes para manipulação de dados geográficos, com destaque para `GeoInterface.jl` e `Shapefile.jl`. Nesta seção, abordaremos os conceitos fundamentais e técnicas práticas para trabalhar com dados espaciais em Julia.

5.2.1 Leitura e Escrita de Dados Espaciais

O primeiro passo na análise espacial é a leitura dos dados geográficos. Em Julia, podemos trabalhar com diversos formatos, sendo o Shapefile um dos mais comuns:

```
using Shapefile, GeoInterface

# Leitura de um arquivo shapefile
shp = Shapefile.Handle("municipios.shp")
table = Shapefile.Table(shp)
geometries = Shapefile.shapes(table)
```

O pacote `Shapefile.jl` permite acessar tanto as geometrias quanto os atributos associados. Cada registro no shapefile contém uma geometria (como um polígono representando um município) e dados tabulares associados (como população, PIB, etc.).

5.2.2 Tipos de Geometrias

Em Julia, diferentes tipos de geometrias são representadas através do protocolo `GeoInterface.jl`. As principais geometrias incluem:

- **Point**: Representa uma localização única no espaço, útil para representar cidades ou centróides
- **LineString**: Sequência de pontos conectados, como estradas ou rios
- **Polygon**: Área fechada, comumente usada para representar limites administrativos
- **MultiPolygon**: Coleção de polígonos, útil para regiões com múltiplas partes ou ilhas

Exemplo de trabalho com diferentes geometrias:

```
# Criando um ponto
ponto = Point([-43.2, -22.9])  # Rio de Janeiro

# Criando um pol gono
coords = [
    [-44.0, -23.0],
    [-44.0, -22.0],
    [-43.0, -22.0],
    [-43.0, -23.0],
    [-44.0, -23.0]
]
poligono = Polygon([coords])
```

5.2.3 Operações Espaciais

Julia oferece diversas operações espaciais fundamentais para análise regional:

Cálculo de Centroides

O centroide é um ponto que representa o centro geométrico de uma região, útil para simplificar análises espaciais:

```
using GeoStats

# Calculando o centroide de um pol gono
centroid = centroid(geometria)
```

Operações Topológicas

As operações topológicas permitem analisar relações espaciais entre geometrias:

```
# Verificando interse  o entre geometrias
intersects(geom1, geom2)

# Calculando a  rea  de interse  o
intersection_area = area(intersection(geom1, geom2))
```

Agregação Espacial

A agregação espacial é fundamental para análises em diferentes níveis geográficos:

```
# Agregando munic pios em microrregi es
function agregar_municipios(municipios, micro_ids)
    agregado = Dict()
    for (mun, id) in zip(municipios, micro_ids)
        if haskey(agregado, id)
            agregado[id] = union(agregado[id], mun)
        else
            agregado[id] = mun
        end
    end
    return agregado
end
```

5.2.4 Transformações de Coordenadas

Em análises regionais, frequentemente precisamos trabalhar com diferentes sistemas de coordenadas. Julia oferece funcionalidades para conversão entre sistemas:

```
using Proj

# Convertendo de WGS84 para UTM
wgs84 = Proj.crs"EPSG:4326"  # Sistema global
```

```
utm = Proj.crs"EPSG:32723"    # UTM zona 23S (comum no Brasil)

coordenadas_utm = transform(wgs84, utm, [-43.2, -22.9])
```

5.2.5 Boas Práticas e Considerações

Ao trabalhar com dados espaciais em Julia, considere:

- **Validação de Geometrias**: Sempre verifique a validade das geometrias antes de operações espaciais
- **Sistema de Coordenadas**: Documente e verifique o sistema de coordenadas dos dados
- **Performance**: Para grandes conjuntos de dados, utilize índices espaciais
- **Precisão Numérica**: Considere a precisão necessária para suas análises

5.2.6 Conclusão

A manipulação de dados espaciais em Julia oferece um conjunto robusto de ferramentas para análise econômica regional. Com o entendimento adequado das geometrias e operações disponíveis, é possível realizar análises espaciais sofisticadas de forma eficiente.

5.3 Mapas

5.4 Visualização de Mapas em Julia

A visualização de dados espaciais é uma parte fundamental da análise econômica regional. Julia oferece diversas bibliotecas para criação de mapas, com destaque para `Plots.jl` com `GeoPlots.jl` e `Makie.jl`. Esta seção apresenta as principais técnicas de visualização de mapas em Julia.

5.4.1 Bibliotecas de Visualização

Para criar mapas em Julia, utilizamos principalmente:

```
using Plots
using GeoPlots  # Para mapas b sicos
using Makie     # Para visualiza  es interativas
using Colors    # Para esquemas de cores
using StatsBase # Para classifica  o de dados
```

5.4.2 Mapas Coropléticos

Os mapas coropléticos são fundamentais para visualizar variáveis socioeconômicas por região. Vamos explorar sua construção:

```
function criar_mapa_coropletico(shp, dados;
                                coluna="PIB_PC",
                                titulo="PIB per capita")
    # Definir esquema de cores
    colors = cgrad(:viridis, 5, categorical=true)

    # Classificar dados em quintis
    breaks = quantile(dados[coluna], 0:0.2:1)

    # Criar o mapa
```

```
    plot = plot()
    for (i, feature) in enumerate(shp)
        valor = dados[coluna][i]
        classe = findfirst(x -> x >= valor, breaks)
        cor = colors[classe]

        plot!(feature, color=cor, legend=false)
    end

    title!(titulo)
    return plot
end
```

5.4.3 Mapas de Calor (Heatmaps)

Os mapas de calor são úteis para visualizar densidade ou intensidade de fenômenos:

```
function criar_mapa_calor(pontos, valores;
                          resolucao=100)
    # Criar grade regular
    x = range(minimum(first.(pontos)),
              maximum(first.(pontos)),
              length=resolucao)
    y = range(minimum(last.(pontos)),
              maximum(last.(pontos)),
              length=resolucao)

    # Calcular densidade
    densidade = zeros(resolucao, resolucao)
    for (p, v) in zip(pontos, valores)
        i = searchsortedfirst(x, first(p))
        j = searchsortedfirst(y, last(p))
        densidade[i,j] += v
```

```
    end

    # Plotar heatmap
    heatmap(x, y, densidade,
            color=:thermal,
            title="Mapa de Calor")
end
```

5.4.4 Mapas de Fluxo

Para visualizar fluxos entre regiões (como migração ou comércio):

```
function criar_mapa_fluxo(origem, destino, valores;
                          max_width=5)
    # Normalizar valores para espessura das linhas
    max_valor = maximum(valores)
    espessuras = valores .* (max_width/max_valor)

    # Criar mapa base
    plot = plot()

    # Adicionar linhas de fluxo
    for (o, d, e) in zip(origem, destino, espessuras)
        plot!([o[1], d[1]], [o[2], d[2]],
              linewidth=e,
              color=:blue,
              alpha=0.5)
    end

    return plot
end
```

5.4.5 Mapas Interativos

O `Makie.jl` permite criar visualizações interativas:

```
function criar_mapa_interativo(shp, dados)
    fig = Figure()
    ax = Axis(fig[1,1])

    # Plotar pol gonos
    poly!(ax, shp, color=dados)

    # Adicionar interatividade
    on(events(fig).mouseposition) do pos
        # Atualizar tooltip com informa  es
        # do pol gono sob o cursor
    end

    return fig
end
```

5.4.6 Elementos Cartográficos

Para mapas profissionais, é importante adicionar elementos cartográficos:

```
function adicionar_elementos_cartograficos!(plot;
    titulo="Mapa",
    fonte="IBGE",
    escala=true,
    norte=true)

    # Adicionar t tulo
    title!(plot, titulo)

    # Adicionar fonte
```

```
    annotate!(plot, 0, 0, text(fonte, :bottom))

    if escala
        # Adicionar barra de escala
        adicionar_escala!(plot)
    end

    if norte
        # Adicionar rosa dos ventos
        adicionar_norte!(plot)
    end

    return plot
end

function adicionar_escala!(plot)
    # Implementa  o da barra de escala
end

function adicionar_norte!(plot)
    # Implementa  o da rosa dos ventos
end
```

5.4.7 Personalização Visual

A personalização é importante para comunicação efetiva:

```
function personalizar_mapa!(plot;
    tema=:light,
    paleta=:viridis,
    fonte="Arial",
    tamanho_fonte=10)

    # Aplicar tema
```

```
    theme!(tema)

    # Configurar fonte
    font!(fonte)

    # Ajustar tamanhos
    plot!(fontsize=tamanho_fonte)

    return plot
end
```

5.4.8 Exportação de Mapas

Para salvar as visualizações em diferentes formatos:

```
function exportar_mapa(plot, arquivo;
    formato=:png,
    dpi=300,
    qualidade=90)

    savefig(plot, "$arquivo.$formato",
        dpi=dpi,
        quality=qualidade)
end
```

5.4.9 Boas Práticas

Ao criar visualizações cartográficas, considere:

- **Escolha de Cores**: Use paletas apropriadas para o tipo de dado
- **Legibilidade**: Garanta que textos e símbolos sejam claros
- **Projeção**: Use projeções apropriadas para a região de interesse

- **Simplificação**: Simplifique geometrias quando apropriado para melhor performance

5.4.10 Conclusão

A visualização de mapas em Julia oferece um conjunto robusto de ferramentas para comunicação efetiva de análises espaciais. Com as técnicas apresentadas, é possível criar desde mapas simples até visualizações interativas sofisticadas.

5.4.11 Referências Recomendadas

- Documentação do Plots.jl
- Manual do GeoPlots.jl
- Tutoriais do Makie.jl
- Guias de cartografia do IBGE

Referências Bibliográficas

Amaral, P., Resende de Carvalho, L., Hernandes Rocha, T. A., da Silva, N. C., and Vissoci, J. R. N. (2019). Geospatial modeling of microcephaly and zika virus spread patterns in brazil. *PLoS ONE*, 14(9):e0222668.

Aruoba, S. B. and Fernández-Villaverde, J. (2015). A comparison of programming languages in macroeconomics. *Journal of Economic Dynamics and Control*, 58:265–273.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98.

Caliendo, L. and Parro, F. (2015). Estimates of the trade and welfare effects of nafta. *The Review of Economic Studies*, 82(1):1–44.

Caliendo, L., Parro, F., Rossi-Hansberg, E., and Sarte, P.-D. (2018). The impact of regional and sectoral productivity changes on the us economy. *The Review of economic studies*, 85(4):2042–2096.

Elhorst, J. P. (2014). *Spatial Econometrics: From Cross-Sectional Data to Spatial Panels*. Springer, Berlin, Heidelberg.

Hirschman, A. O. (1958). *The Strategy of Economic Development*. Yale University Press, New Haven.

Kelejian, H. H. and Piras, G. (2017). *Spatial Econometrics: Methods and Applications*. Academic Press, London, UK.

Leal, A. (2025). *International Trade and Deforestation: the Political Economy behind Brazilian Land-Intensive Exports*. PhD thesis, Universidade de São Paulo, São Paulo, Brasil.

LeSage, J. P. and Pace, R. K. (2009). *Introduction to Spatial Econometrics*. Chapman and Hall/CRC, Boca Raton, FL.

Rasmussen, P. N. (1952). Studies on inter-sectoral relations.

Tobler, W. R. (1970). A computer movie simulating urban growth in the detroit region. *Economic geography*, 46(sup1):234–240.

Vale, V. A. and Perobelli, F. S. (2020). Análise de insumo-produto: teoria e aplicações no r. *Curitiba:[Edição Independente]*.

Sobre o Autor

Alan Marques Miranda Leal é economista e doutor em Teoria Econômica pela Universidade de São Paulo (USP). Com mestrado pela Universidade Federal de Minas Gerais (UFMG) e graduação pela Universidade Federal de Viçosa (UFV), possui sólida formação acadêmica e vasta experiência em análise de dados e pesquisa econômica.

Especialista em economia regional, comércio internacional e meio ambiente, tem contribuído com importantes instituições como o Banco Interamericano de Desenvolvimento e a FIPE. Seus trabalhos de pesquisa abrangem temas diversos, desde políticas ambientais e conservação até questões de desenvolvimento regional e desigualdade de gênero, com publicações em periódicos científicos de relevância internacional.

Atualmente, é pesquisador sênior no Climate Policy Initiative/PUC-Rio.

Além de sua pesquisa acadêmica, já atuou como consultor em projetos que envolvem análise quantitativa, econometria e avaliação de políticas públicas, combinando rigor metodológico com aplicações práticas para o desenvolvimento econômico e social.

www.alanleal-econ.com