



JULIA PARA ECONOMISTAS REGIONAIS

Introdução aos Métodos Regionais em
Julia

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 primeira edição do livro e novos procedimentos e atualizações aparecerão no futuro. Este trabalho está licenciado sob uma Licença Creative Commons Atribuição 4.0 Internacional (CC BY 4.0).

Sumário

1	Introdução	1
1.1	Introdução	1
1.2	História e Desenvolvimento	1
1.3	Características Fundamentais	2
1.3.1	Performance Excepcional	2
1.3.2	Multiple Dispatch	3
1.3.3	Ecosistema Científico	3
1.4	Julia na Economia	3
1.4.1	Aplicações em Economia Computacional	4
1.4.2	Casos de Uso Proeminentes	5
1.5	Pacotes Econômicos	5
1.5.1	QuantEcon.jl	5
1.5.2	EconModels.jl	6
1.6	Vantagens para Economistas	6
1.6.1	Performance Superior	6
1.6.2	Facilidade de Uso	7
1.7	Desafios e Limitações	7
1.7.1	Maturidade do Ecosistema	7
1.8	Futuro de Julia na Economia	7
1.8.1	Adoção Crescente	8

1.9	Instalação e Configuração do Julia	8
1.9.1	Download e Instalação Básica	8
1.9.2	Configuração do Ambiente	9
1.9.3	Pacotes Específicos para Economia Regional	9
1.9.4	Configuração do Editor	10
1.9.5	Verificação da Instalação	10
1.9.6	Resolução de Problemas Comuns	11
1.9.7	Manutenção e Atualização	11
1.9.8	Boas Práticas de Configuração	11
1.10	Conclusão	12
2	Econometria Espacial	13
2.1	Introdução	13
2.2	Fundamentos Teóricos	13
2.2.1	Modelo SAR (Spatial Autoregressive Model)	14
2.2.2	Modelo SEM (Spatial Error Model)	14
2.2.3	Modelo SAC/SARAR	15
2.3	Efeitos Marginais	15
2.3.1	Efeitos Diretos	16
2.3.2	Efeitos Indiretos	16
2.3.3	Efeitos Totais	16
2.4	Aplicação Prática: Análise da Disseminação do Vírus Zika	16
2.5	Considerações sobre Implementação Computacional	17
2.6	Implementação Prática	17
2.6.1	Códigos	17
2.6.2	Replicando Amaral et al. (2019)	24
2.7	Conclusão	27

3	Insumo-Produto	29
3.1	Introdução	29
3.2	Dados e Estrutura da Matriz de Insumo-Produto	29
3.2.1	Matriz de Coeficientes Técnicos	30
3.2.2	Modelo Básico de Leontief	30
3.2.3	Interpretação da Inversa de Leontief	31
3.3	Multiplicadores de Insumo-Produto	31
3.3.1	Tipos de Multiplicadores	32
3.3.2	Decomposição dos Efeitos	33
3.3.3	Multiplicadores de Emprego	33
3.3.4	Multiplicadores de Renda	34
3.3.5	Interpretação dos Multiplicadores	34
3.3.6	Multiplicadores Tipo I e Tipo II	35
3.4	Índices de Ligação na Análise de Insumo-Produto:	
	Mensuração de Encadeamentos Setoriais	35
3.4.1	Índices de Ligação de Hirschman-Rasmussen	36
3.4.2	Índice de Ligação para Trás	36
3.4.3	Índice de Ligação para Frente	37
3.5	Coeficientes de Variação	37
3.5.1	Coeficiente de Variação das Ligações para Trás	37
3.5.2	Coeficiente de Variação das Ligações para Frente	38
3.6	Identificação de Setores-Chave	38
3.7	Índices de Ligação usando a Matriz de Ghosh	38
3.8	Método de Extração Hipotética em Insumo-Produto	39
3.8.1	Abordagens de Extração	39
3.8.2	Extração da Estrutura de Compras	39
3.8.3	Formulação Matemática	40
3.8.4	Extração da Estrutura de Vendas	40

3.9	Medidas Percentuais	41
3.10	Interpretação dos Resultados	41
3.10.1	Vantagens e Limitações	41
3.10.2	Vantagens	42
3.10.3	Conclusão	42
3.11	Implementação na Julia	42
3.11.1	Inversa de Leontief	43
3.11.2	Multiplicadores	43
3.11.3	Multiplicadores de Emprego	43
3.11.4	Multiplicadores de Produção	43
3.11.5	Multiplicadores de Produção Truncado	44
3.11.6	Índices de Ligação para frente e para trás	44
3.11.7	Extração Hipotética	45
4	Operações Vetoriais e Mapas na Julia	47
4.1	Introdução	47
4.2	Manipulação de Dados Espaciais	47
4.2.1	Leitura e Escrita de Dados Espaciais	48
4.2.2	Tipos de Geometrias	48
4.2.3	Operações Espaciais	49
4.2.4	Transformações de Coordenadas	50
4.2.5	Boas Práticas e Considerações	51
4.2.6	Conclusão	51
4.3	Mapas	51
4.4	Visualização de Mapas em Julia	52
4.4.1	Bibliotecas de Visualização	52
4.4.2	Mapas Coropléticos	52
4.4.3	Mapas de Calor (Heatmaps)	53
4.4.4	Mapas de Fluxo	54

4.4.5	Mapas Interativos	55
4.4.6	Elementos Cartográficos	55
4.4.7	Personalização Visual	56
4.4.8	Exportação de Mapas	57
4.4.9	Boas Práticas	57
4.4.10	Conclusão	58
4.4.11	Referências Recomendadas	58
Sobre o Autor		61

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-

nhedido 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
- Estimacão 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
- Estimativas 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:

```
1 # Abra o REPL do Julia e digite:
2 using Pkg
3 # Instale os pacotes essenciais
4 Pkg.add("CSV")
5 Pkg.add("DataFrames")
6 Pkg.add("Plots")
7 Pkg.add("Statistics")
```

1.9.3 Pacotes Específicos para Economia Regional

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

```
1 # Pacotes para análise espacial
2 Pkg.add("GeoInterface")
3 Pkg.add("Shapefile")
4 Pkg.add("GeoStats")
5
6 # Pacotes para econometria espacial
7 Pkg.add("SpatialRegression")
8 Pkg.add("NearestNeighbors")
9
10 # Pacotes para visualização
11 Pkg.add("Makie")
12 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:**

```
1 # No REPL do Julia:  
2 using Pkg  
3 Pkg.add("IJulia")  
4
```

1.9.5 Verificação da Instalação

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

```
1 # Teste básico de funcionalidade  
2 println("Julia versão: ", VERSION)  
3  
4 # Teste de pacotes  
5 using CSV, DataFrames, Plots  
6 println("Pacotes básicos carregados com sucesso!")  
7  
8 # Teste de computação numérica  
9 x = rand(1000)  
10 @time mean(x) # Deve mostrar tempo de execução
```

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

```
1 # Configuração de memória (se necessário)
2 # Adicione ao arquivo de configuração:
3 ENV["JULIA_MAX_NUM_THREADS"] = "auto"
```

1.9.7 Manutenção e Atualização

Para manter o ambiente Julia atualizado:

```
1 # Atualizar todos os pacotes
2 using Pkg
3 Pkg.update()
4
5 # Verificar status
6 Pkg.status()
7
8 # Testar após atualizações
9 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

```
1 # Exemplo de criação de novo projeto
2 using Pkg
3 Pkg.activate("MeuProjetoRegional")
4 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:

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

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:

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

2.2.2 Modelo SEM (Spatial Error Model)

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

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

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

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

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:

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

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

$$\frac{1}{n} \sum_i \phi_{ii} \beta_r = \frac{1}{n} \text{tr}(I - \rho W)^{-1} \beta_r \quad (2.7)$$

2.3.2 Efeitos Indiretos

$$\frac{1}{n} \sum_i \sum_j \phi_{ij} \beta_r - \frac{1}{n} \sum_i \phi_{ii} \beta_r = \frac{1}{n} \text{tr}(I - \rho W)^{-1} \beta_r \quad (2.8)$$

2.3.3 Efeitos Totais

$$\frac{1}{n} \sum_i \sum_j \phi_{ij} \beta_r \quad (2.9)$$

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

```
1 using Optim
2 using LinearAlgebra
3 using ForwardDiff
4 using Statistics
5 using Distributions
6 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     l_n = l(n)
11     u = y - params[2] * W * y - X * params[4:end]
12     tilde_u = (l_n - params[3] * M) * u
13     log_det_W = logdet(l_n - params[2] * W)
14     log_det_M = logdet(l_n - params[3] * M)
15     quad_form = tilde_u' * tilde_u
16     log_likelihood = - (n / 2) * log(2 * pi) - (n / 2) *
17         log(params[1]) +
18         log_det_W + log_det_M - (1 / (2 * params[1])) *
19         quad_form
20     return -log_likelihood
21 end
22
23 function sarar_coef(y,X,W,M)
24     n_x=size(X)[2]
25     n=length(y)
26     initial_params = vcat(1,0.5,0.5,zeros(n_x))
27     lower_bounds = [0;-1;-1;fill(-Inf,n_x)]
28     upper_bounds = [Inf;1;1;fill(Inf,n_x)]
29     result = optimize(params -> log_likelihood_sarar(
30         params, n,X,y,W,M),
31         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 -----

```

```

7 Log-Likelihood: -737.902
8 Number of observations: 184
9 sigma2: 176.214
10 -----
11 | Variable | beta | Std Dev | Lower CI | Upper CI | p-value |
12 :-----+-----+-----+-----+-----+
13 | constant | 16.627 | 39.880 | -62.078 | 95.332 | 0.677241 |
14 | ln_gdp | -2.600 | 6.923 | -16.263 | 11.063 | 0.707744 |
15 | ln_pop | 1.228 | 8.970 | -16.475 | 18.931 | 0.89124 |
16 | mobility | -14.590 | 35.789 | -85.221 | 56.041 | 0.684016 |
17 | environ | 5.229 | 11.160 | -16.796 | 27.254 | 0.639967 |
18 | sanitation | 3.247 | 10.118 | -16.721 | 23.216 | 0.748625 |
19 '-----'-----'-----'-----'-----'
20 rho: 0.238, Standard Deviation: 0.113

```

- SEM

```

1 sem_model=sem(X,y,W)
2 names_col=names(df)
3 sar_summary(sem_model,names_col)
4 -----
5 Maximum Likelihood Estimation of SEM Model
6 -----
7 Log-Likelihood: -737.902
8 Number of observations: 184
9 sigma2: 176.214
10 -----
11 | Variable | beta | Std Dev | Lower CI | Upper CI | p-value |
12 :-----+-----+-----+-----+-----+
13 | constant | 16.627 | 39.880 | -62.078 | 95.332 | 0.677241 |
14 | ln_gdp | -2.600 | 6.923 | -16.263 | 11.063 | 0.707744 |

```

```

15 | ln_pop | 1.228 | 8.970 | -16.475 | 18.931 | 0.89124 |
16 | mobility | -14.590 | 35.789 | -85.221 | 56.041 | 0.684015 |
17 | environ | 5.229 | 11.160 | -16.796 | 27.254 | 0.639967 |
18 | sanitation | 3.247 | 10.118 | -16.721 | 23.216 | 0.748625 |
19 '-----'
20 lambda: 0.238, Standard Deviation: 0.113

```

• SARAR

```

1 M=W
2 sarar_model=sarar(X,y,W,M)
3 sarar_summary(sarar_model,names_col)
4 -----
5 Maximum Likelihood Estimation of SAC Model
6 -----
7 Log-Likelihood: -737.764
8 Number of observations: 184
9 sigma2: 176.676
10 -----
11 | Variable | beta | Std Dev | Lower CI | Upper CI | p-value |
12 :-----+-----+-----+-----+-----+-----:
13 | constant | 15.360 | 40.525 | -64.620 | 95.341 | 0.705119 |
14 | ln_gdp | -2.890 | 7.135 | -16.973 | 11.193 | 0.685943 |
15 | ln_pop | 1.486 | 9.109 | -16.491 | 19.463 | 0.870596 |
16 | mobility | -13.968 | 37.093 | -87.175 | 59.240 | 0.706961 |
17 | environ | 5.964 | 11.523 | -16.777 | 28.706 | 0.605369 |
18 | sanitation | 4.277 | 10.677 | -16.796 | 25.351 | 0.689199 |
19 '-----'
20 rho: 0.085, Standard Deviation: 0.322
21 lambda: 0.171, Standard Deviation: 0.311

```

Os efeitos diretos e indiretos do SARAR podem ser estimados da

seguinte forma¹:

```

1 beta_complet=vcats(sarar_model.sigma2,sarar_model.rho[1],sarar_model.lambda[1],
   sarar_model.coefs[:,1])
2 SARAR_effects=effects_sarar(y,X,W,M,beta_complet)
3 effects_summary(SARAR_effects,names_col[2:end])
4 -----
5 | Variable | Direct Effects | Indirect Effects | Total Effects |
6 :-----+-----+-----+-----:
7 | ln_gdp | -2.894 | -0.264 | -3.159 |
8 | ln_pop | 1.488 | 0.136 | 1.624 |
9 | mobility | -13.987 | -1.278 | -15.265 |
10 | environ | 5.973 | 0.546 | 6.518 |
11 | sanitation | 4.283 | 0.391 | 4.675 |
12 '-----'-----'-----'-----'

```

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.

¹O 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 (Z): registra as transações entre os setores
- Vetor de demanda final (y): consumo das famílias, governo, investimento e exportações
- Vetor de valor bruto da produção (x): produção total de cada setor
- Vetor de valor adicionado (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 (A), que representa a tecnologia de produção da economia. Cada elemento a_{ij} é calculado como:

$$a_{ij} = \frac{z_{ij}}{x_j} \quad (3.1)$$

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:

$$x = Ax + y \quad (3.2)$$

que pode ser rearranjada para:

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

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:

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

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):

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

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:

$$c_j^e = \frac{v_j^e}{x_j} \quad (3.6)$$

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:

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

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:

$$c_j^r = \frac{v_j^r}{x_j} \quad (3.8)$$

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

Multiplicador Simples de Renda

É calculado como:

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

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:

$$M_j^I = \frac{\text{Efeitos Diretos} + \text{Indiretos}}{\text{Efeitos Diretos}} \quad (3.10)$$

Multiplicadores Tipo II

Incluem também os efeitos induzidos:

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

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:

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

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:

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

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

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

3.5.2 Coeficiente de Variação das Ligações para Frente

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

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:

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

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:

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

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

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

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

O impacto total da extração é medido por:

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

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:

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

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

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

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

O impacto total é medido por:

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

3.9 Medidas Percentuais

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

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

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

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 $n \times n$. 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
6     ./ (e./x)
7     EF = Diagonal(e./x)*Bf[1:end-1, 1:end-1]
8     multiplicador_emprego_truncado=sum(EF,dims=1)'
9     multiplicador_emprego_tipoii=
10    multiplicador_emprego_truncado./(e./x)
11    return hcat(Setores,multiplicador_emprego_simples,
12               multiplicador_emprego_tipoi,multiplicador_emprego_truncado,
13               multiplicador_emprego_tipoii)
```

3.11.4 Multiplicadores de Produção

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

```

4     eftfechado=sum(Bf[i,1:n] for i=1:n+1)
5     eftaberto=sum(B[i,:] for i=1:n)
6     eftinduzido=sum(Bf[i,1:n] for i=1:n+1)-sum(B[i,:] for i=1:
n)
7     eftdireto=sum(A[i,:] for i=1:n)
8     eftindireto=sum(B[i,:] for i=1:n)-sum(A[i,:] for i=1:n)
9     return hcat(setores ,eftfechado ,eftaberto ,eftinduzido ,
eftdireto ,eftindireto)
10 end

```

3.11.5 Multiplicadores de Produção Truncado

```

1 using LinearAlgebra
2 function multiplicadores_produ o_truncado (A,B,Bf ,setores)
3     n=size(A)[1]
4     eftfechado=sum(Bf[i,1:n] for i=1:n)
5     eftaberto=sum(B[i,:] for i=1:n)
6     eftinduzido=sum(Bf[i,1:n] for i=1:n)-sum(B[i,:] for i=1:n)
7     eftdireto=sum(A[i,:] for i=1:n)
8     eftindireto=sum(B[i,:] for i=1:n)-sum(A[i,:] for i=1:n)
9     return hcat(setores ,eftfechado ,eftaberto ,eftinduzido ,
eftdireto ,eftindireto)
10 end

```

3.11.6 Índices de Ligação para frente e para trás

Índices de Ligação para trás

```

1 using LinearAlgebra
2 function lig_tras (B ,setores)
3     n=size(B)[1]
4     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)
5     for j=1:n)/(n^2))
6     return hcat(setores , lig)
7 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
9     (size (A) [1]) '*(B_hip*y)))/(ones (size (A) [1]) '*(B*y)))*100
10    impacto_elect_gas_absoluto=ones (size (A) [1]) '*(B*y)-ones (
11    size (A) [1]) '*(B_hip*y)
12    return hcat (impacto_elect_gas_absoluto ,
13    impacto_elect_gas_por_cento)
14 end

```


Capítulo 4

Operações Vetoriais e Mapas na Julia

4.1 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.

4.2 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.

4.2.1 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:

```
1 using Shapefile, GeoInterface
2
3 # Leitura de um arquivo shapefile
4 shp = Shapefile.Handle("municipios.shp")
5 table = Shapefile.Table(shp)
6 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.).

4.2.2 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:

```
1 # Criando um ponto
2 ponto = Point([-43.2, -22.9]) # Rio de Janeiro
3
4 # Criando um poligono
5 coords = [
6     [-44.0, -23.0],
7     [-44.0, -22.0],
8     [-43.0, -22.0],
9     [-43.0, -23.0],
10    [-44.0, -23.0]
11 ]
12 poligono = Polygon([coords])
```

4.2.3 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:

```
1 using GeoStats
2
3 # Calculando o centroide de um poligono
4 centroid = centroid(geometria)
```

Operações Topológicas

As operações topológicas permitem analisar relações espaciais entre geometrias:

```
1 # Verificando interseção entre geometrias
2 intersects(geom1, geom2)
3
4 # Calculando a área de interseção
5 intersection_area = area(intersection(geom1, geom2))
```

Agregação Espacial

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

```
1 # Agregando municípios em microrregiões
2 function agregar_municipios(municipios, micro_ids)
3     agregado = Dict()
4     for (mun, id) in zip(municipios, micro_ids)
5         if haskey(agregado, id)
6             agregado[id] = union(agregado[id], mun)
7         else
8             agregado[id] = mun
9         end
10    end
11    return agregado
12 end
```

4.2.4 Transformações de Coordenadas

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

```
1 using Proj
2
3 # Convertendo de WGS84 para UTM
4 wgs84 = Proj.crs"EPSG:4326" # Sistema global
```

```
5 utm = Proj.crs"EPSG:32723" # UTM zona 23S (comum no
   Brasil)
6
7 coordenadas_utm = transform(wgs84, utm, [-43.2, -22.9])
```

4.2.5 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

4.2.6 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.

4.3 Mapas

4.4 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.

4.4.1 Bibliotecas de Visualização

Para criar mapas em Julia, utilizamos principalmente:

```
1 using Plots
2 using GeoPlots # Para mapas básicos
3 using Makie    # Para visualizações interativas
4 using Colors  # Para esquemas de cores
5 using StatsBase # Para classificação de dados
```

4.4.2 Mapas Coropléticos

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

```
1 function criar_mapa_coropletico(shp, dados;
2     coluna="PIB_PC",
3     titulo="PIB per capita")
4     # Definir esquema de cores
5     colors = cgrad(:viridis, 5, categorical=true)
6
7     # Classificar dados em quintis
8     breaks = quantile(dados[coluna], 0:0.2:1)
9
10    # Criar o mapa
```

```
11 plot = plot()
12 for (i, feature) in enumerate(shp)
13     valor = dados[coluna][i]
14     classe = findfirst(x -> x >= valor, breaks)
15     cor = colors[classe]
16
17     plot!(feature, color=cor, legend=false)
18 end
19
20 title!(titulo)
21 return plot
22 end
```

4.4.3 Mapas de Calor (Heatmaps)

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

```
1 function criar_mapa_calor(pontos, valores;
2                             resolucao=100)
3     # Criar grade regular
4     x = range(minimum(first.(pontos)),
5               maximum(first.(pontos)),
6               length=resolucao)
7     y = range(minimum(last.(pontos)),
8               maximum(last.(pontos)),
9               length=resolucao)
10
11     # Calcular densidade
12     densidade = zeros(resolucao, resolucao)
13     for (p, v) in zip(pontos, valores)
14         i = searchsortedfirst(x, first(p))
15         j = searchsortedfirst(y, last(p))
16         densidade[i,j] += v
```



```
17     end
18
19     # Plotar heatmap
20     heatmap(x, y, densidade,
21             color=:thermal,
22             title="Mapa de Calor")
23 end
```

4.4.4 Mapas de Fluxo

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

```
1 function criar_mapa_fluxo(origem, destino, valores;
2                           max_width=5)
3     # Normalizar valores para espessura das linhas
4     max_valor = maximum(valores)
5     espessuras = valores .* (max_width/max_valor)
6
7     # Criar mapa base
8     plot = plot()
9
10    # Adicionar linhas de fluxo
11    for (o, d, e) in zip(origem, destino, espessuras)
12        plot!([o[1], d[1]], [o[2], d[2]],
13              linewidth=e,
14              color=:blue,
15              alpha=0.5)
16    end
17
18    return plot
19 end
```

4.4.5 Mapas Interativos

O `Makie.jl` permite criar visualizações interativas:

```
1 function criar_mapa_interativo(shp, dados)
2     fig = Figure()
3     ax = Axis(fig[1,1])
4
5     # Plotar poligonos
6     poly!(ax, shp, color=dados)
7
8     # Adicionar interatividade
9     on(events(fig).mouseposition) do pos
10         # Atualizar tooltip com informa es
11         # do poligono sob o cursor
12     end
13
14     return fig
15 end
```

4.4.6 Elementos Cartográficos

Para mapas profissionais, é importante adicionar elementos cartográficos:

```
1 function adicionar_elementos_cartograficos!(plot;
2     titulo="Mapa",
3     fonte="IBGE",
4     escala=true,
5     norte=true)
6
7     # Adicionar titulo
8     title!(plot, titulo)
9
10    # Adicionar fonte
```

```
11     annotate!(plot, 0, 0, text(fonte, :bottom))
12
13     if escala
14         # Adicionar barra de escala
15         adicionar_escala!(plot)
16     end
17
18     if norte
19         # Adicionar rosa dos ventos
20         adicionar_norte!(plot)
21     end
22
23     return plot
24 end
25
26 function adicionar_escala!(plot)
27     # Implementa o da barra de escala
28 end
29
30 function adicionar_norte!(plot)
31     # Implementa o da rosa dos ventos
32 end
```

4.4.7 Personalização Visual

A personalização é importante para comunicação efetiva:

```
1 function personalizar_mapa!(plot;
2     tema=:light,
3     paleta=:viridis,
4     fonte="Arial",
5     tamanho_fonte=10)
6
7     # Aplicar tema
```

```
8     theme!(tema)
9
10    # Configurar fonte
11    font!(fonte)
12
13    # Ajustar tamanhos
14    plot!(fontsize=tamanho_fonte)
15
16    return plot
17 end
```

4.4.8 Exportação de Mapas

Para salvar as visualizações em diferentes formatos:

```
1 function exportar_mapa(plot, arquivo;
2     formato=:png,
3     dpi=300,
4     qualidade=90)
5
6     savefig(plot, "$arquivo.$formato",
7         dpi=dpi,
8         quality=qualidade)
9 end
```

4.4.9 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

4.4.10 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.

4.4.11 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.
- 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.
- Kelejian, H. H. and Piras, G. (2017). *Spatial Econometrics: Methods and Applications*. Academic Press, London, UK.
- 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 doutorando 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, além de sua pesquisa acadêmica, atua 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