ENG1467 - OTIMIZAÇÃO - PUC-Rio, 2022.1

Lista 1: Previsão de série de geração eólica

31/03/2022

Professor: Alexandre Street *Aluno: Thiago Novaes*

O objetivo deste trabalho é o estudo de uma série temporal referente à produção eólica de energia na região nordeste do Brasil. O código completo desenvolvido durante essa atividade pode ser encontrado em `https://github.com/Thiago-NovaesB/MestradoPuc.jl/tree/main/Programa%C3%A7%C3%A3o%20Linear/Lista%201`

1 Análise exploratória e modelagem

1.1 Visualização dos dados

Para facilitar a geração de diversos gráficos de dispersão de y_t em função de y_{t-k}, as duas funções abaixo foram escritas:

```
function L(data::Vector{Float64}, k::Int = 0)
    x = data[1:end-k]
    y = data[k+1:end]

    return x, y
end
function plot_shift(data::Vector{Float64}, k::Int = 0)
    x, y = L(data, k)
    p = plot(x, y, seriestype = :scatter, title = "k = $k")

    return p
end
```

A primeira função faz o atraso k e a segunda plota o gráfico. Com isso, foi possível gerar os seguintes gráficos:

Por meio das visualizações dos gráficos, pode-se verificar que existe uma correlação ao se considerar pequenos atrasos. Já em atrasos médios, a nuvem de pontos fica muito difusa, não indicando uma correção. Em grandes atrasos, temos indícios de baixa correções, em comparação com pequenos atrasos. Sendo assim, as horas candidatas para o modelo auto-regressivo são as próximas a amostra e as com um dia de defasagem da amostra.

1.2 Modelo auto-regressivo

Para ajustar um modelo através de regressão quantílica, é necessário criar uma variável para fazer o papel do erro do estimador. Usando restrições é possível garantir que essa variável será maior ou igual ao módulo do erro. Usando a soma dos erros como função objetivo, teremos finalmente a

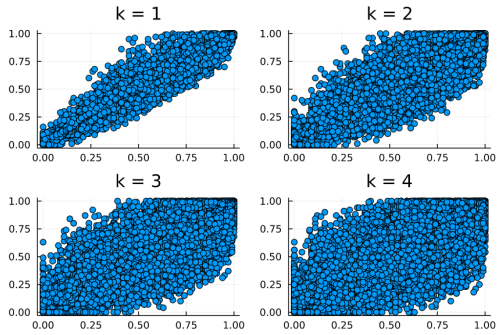

Figure 1: Gráficos com pequenos atrasos.

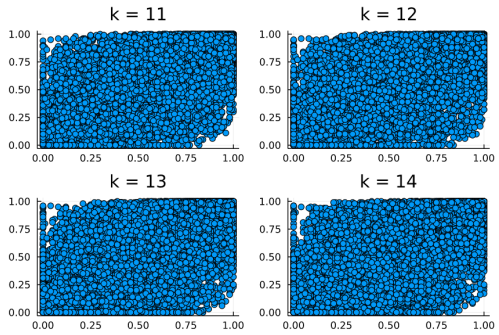

Figure 2: Gráficos com atrasos médios.

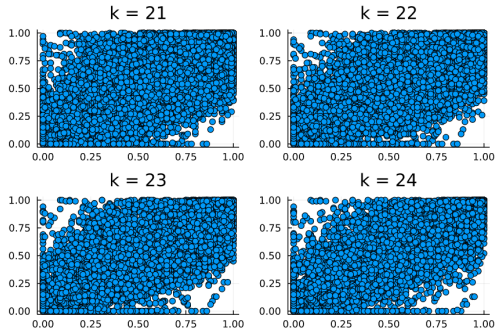

Figure 3: Gráficos com grandes atrasos.

minimização do erros, ou seja, a variável erro será exatamente o erro. A função abaixo foi escrita para facilitar o ajuste de diversos modelos auto-regressivos diferentes:

```
1
2 function auto_regression_model(data::Vector{Float64}, K::Vector{Int} = [1])
3     n = length(K)
```

```
    N = length(data)
    k_max = maximum(K)

    model = Model(GLPK.Optimizer)
    @variable(model, beta[1:n+1])
    @variable(model, error[1:N-k_max])

    @expression(model, AR[i = 1:N-k_max],
        beta[1] +
        sum(beta[j+1]*data[i + k_max - K[j]] for j = 1:n))

    @constraint(model, [i = 1:N-k_max], error[i] >= + data[i + k_max] - AR[i]
    )
    @constraint(model, [i = 1:N-k_max], error[i] >= - data[i + k_max] + AR[i]
    )

    @objective(model, Min, sum(error))

    optimize!(model)

    return model
end
```

Nela temos como entrada o vetor de dados e um vetor com os atrasos que queremos incluir na regressão. Sendo assim, algumas regressões foram obtidas:

$$y_t^1 = 0.016 + 0.9884 y_{t-1}$$

$$y_t^2 = -0.0007 + 0.92766 y_{t-1} + 0.07175 y_{t-24}$$

$$y_t^3 = 0.00657 + 1.4693 y_{t-1} - 0.54189 y_{t-2} + 0.27898 y_{t-23} - 0.223038 y_{t-24}$$

$$y_t^4 = -0.003401 + 0.924916 y_{t-1} + 0.010679_{t-12} + 0.068019 y_{t-24}$$

Os valores de k escolhidos para as regressões foram escolhidos com base na análise dos gráficos de dispersão. Valores que aparentavam ter grande correlação foram escolhidos. Além disso, foi incluído $k = 12$, para analisar o efeito na auto-regressão de um termo sem correlação.

1.3 Agregação

Para o estudo de sazonalidade, inicialmente serão visualizadas as agregações temporais dos dados, para isso, a seguinte função foi criada:

```
function agregate(data::Vector{Float64}, s::Int)
    agregated = Float64[]
    location = 1
    size = length(data)
```

```
5
6       while true
7           if location+s-1 <= size
8               interval = data[location:location+s-1]
9               value =  sum(interval) / s
10              push!(agregated, value)
11              location = location+s
12          else
13              break
14          end
15      end
16
17      return agregated
18  end
```

Usando essa função, foi possível visualizar os efeitos da agregação:

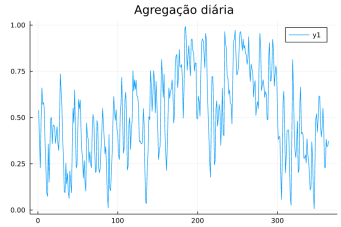

Figure 4: Agregação diária.

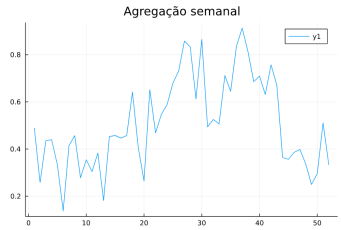

Figure 5: Agregação semanal.

Examinando os gráficos agregados, percebe-se que a curva se suaviza quando maior a agregação, o que era esperado. Na frequência mensal parece existir uma periodicidade, o que é condizente com os conhecimentos básicos de meteorologia. Porém, a serie não parece apresentar um úncia frequência, sendo necessário assim usar harmônicos.

1.4 Sazonalidade

Para facilitar o estudo do ajuste da série para um modelo puramente sazonal, a função abaixo foi escrita. Ela recebe como entrada os dados, um vetor de períodos e um vetor com a quantidade de

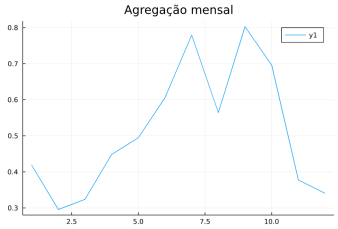

Figure 6: Agregação mensal.

harmônicos de cada período (M). Além disso, foi adicionado uma constante na regressão, essa constante pode ser interpretada como um cosseno de frequência zero, assim evitar as propriedades ruins das funções trigonométricas, como assumir valores negativos e ter média 0.

```
function season_model(data::Vector{Float64}, S::Vector{Int} = [8760], M::
    Vector{Int} = [1])

    N = length(data)
    m = length(S)

    model = Model(GLPK.Optimizer)
    @variable(model, error[1:N])
    @variable(model, theta[1:sum(M)])
    @variable(model, phi[1:sum(M)])
    @variable(model, level)

    @expression(model, ST[i = 1:N], level + sum(sum(
        theta[(j != 1 ? sum(M[l] for l = 1:j-1) : 0) + k]*cos(2*pi*k*i/S[j]) +
        phi[(j != 1 ? sum(M[l] for l = 1:j-1) : 0) + k]*sin(2*pi*k*i/S[j])
        for k in 1:M[j]) for j = 1:m))

    @constraint(model, [i = 1:N], error[i] >= + data[i] - ST[i])
    @constraint(model, [i = 1:N], error[i] >= - data[i] + ST[i])

    @objective(model, Min, sum(error))

    optimize!(model)

    return model
end
```

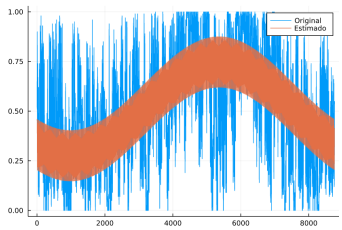

Figure 7: S = 24, 8760 e M= 1, 1

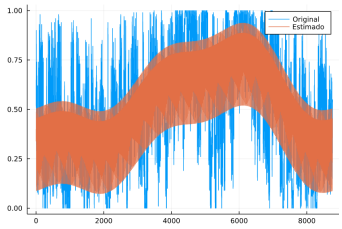

Figure 8: S = 24, 8760 e M= 3, 3

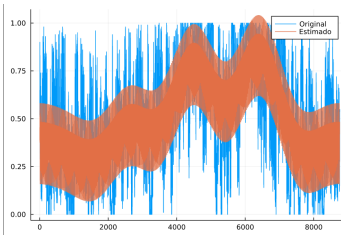

Figure 9: S = 24, 8760 e M= 5, 5

1.5 Modelo completo

Por fim, o modelo completo apresenta componentes auto-regressivas e sazonais, sendo assim, a entrada desta função é a união das entradas das últimas 2 funções apresentadas:

```
function complete_model(data::Vector{Float64}, K::Vector{Int} = [1], S::Vector
    {Int} = [8760], M::Vector{Int} = [1])

    n = length(K)
    N = length(data)
    k_max = maximum(K)
    m = length(S)

    model = Model(GLPK.Optimizer)
    @variable(model, beta[1:n+1])
    @variable(model, error[1:N-k_max])
```

```
    @variable(model, theta[1:sum(M)])
    @variable(model, phi[1:sum(M)])

    @expression(model, AR[i = 1:N-k_max],
        beta[1] +
        sum(beta[j+1]*data[i + k_max - K[j]] for j = 1:n))

    @expression(model, ST[i = 1:N-k_max], sum(sum(
        theta[(j != 1 ? sum(M[l] for l = 1:j-1) : 0) + k]*cos(2*pi*k*i/S[j]) +
        phi[(j != 1 ? sum(M[l] for l = 1:j-1) : 0) + k]*sin(2*pi*k*i/S[j])
        for k in 1:M[j]) for j = 1:m))

    @expression(model, estimate[i = 1:N-k_max], AR[i] + ST[i])

    @constraint(model, [i = 1:N-k_max], error[i] >= + data[i + k_max] -
    estimate[i])
    @constraint(model, [i = 1:N-k_max], error[i] >= - data[i + k_max] +
    estimate[i])

    @objective(model, Min, sum(error))

    optimize!(model)

    return model
end
```

2 Avaliação dos modelos

2.1 *In-sample* e *out-of-sample*

Para fazer uma analise da qualidade dos ajustes dos modelos propostos, deve-se separar o conjunto total de dados em 2 partes, a primeira é chamada de *In-sample* e será usada para treinar o modelo. A segunda parte é chamada de *out-of-sample*, ela será usada como padrão para comparar as previsões geradas pelos modelos.

Existe um certo *trade-off* nessa separação. De forma simples, quando maior o *in-sample* melhor, melhor será nosso modelo, pois teve acesso à mais informação da série. Quando maior *out-of-sample*, mais é possível avaliar a qualidade do modelo. Neste trabalho foi Sugerido usar apenas 24 valores no *out-of-sample*, em séries temporais, geralmente a qualidade da previsão piora muito com a quantidade de previsões devido o acúmulo de erro. Além disso, usar o histórico para prever o vento no dia seguinte já é uma aplicação interessante para problemas de operação de curto prazo.

2.2 Avaliação do modelo *in-sample*

A métrica proposta para a análise da qualidade da estimação *in-sample* foi R^2, implementada abaixo:

```
function r_square(data::Vector{Float64}, estimative::Vector{Float64})
    output = 1.0
    n = length(data)
    average = sum(data) / n

    output = 1.0 - sum((data[i] - estimative[i])^2 for i in 1:n) / sum((data[i] - average)^2 for i in 1:n)

    return output
end
```

A tabela abaixo resume o valor de R^2 para cada um dos modelos apresentados anteriormente.

Table 1: R^2 para o modelo auto-regressivo

k	R^2
1	0.9167064580172875
1, 24	0.9216678882602514
1, 2, 23, 24	0.951514603191624
1, 12, 24	0.9216328729344673

Os altos valores encontrados, i.e. próximos de 1, indicam que o modelo ajusta bem os valores *in-sample*. Porém essa métrica não pode ser usada sozinha, é possível perceber pela tabela que ao se adicionar mais termos na regressão, R^2 aumenta. De fato esse é um resultado teórico. Tal fenômeno pode ser interpretado como o ajuste aprendendo particularidades do *in-sample*, mas perdendo poder de previsão. A próxima métrica ajudará na avaliação do poder de previsão do modelo.

Table 2: R^2 para o modelo sazonal

S	M	R^2
24, 8760	1, 1	0.3236818886119991
24, 8760	3, 3	0.3958137622054149
24, 8760	5, 5	0.4439674857712843

O modelo sazonal puro não teve uma boa performance no conjunto *in-sample*, tendo uma métrica R^2 muito baixa. Isso já era esperado por uma inspeção visual nos gráficos apresentados.

2.3 Avaliação do modelo *out-of-sample*

A métrica proposta para a análise da qualidade do modelo foi MAE, implementada abaixo:

```
function mae(data::Vector{Float64}, estimative::Vector{Float64}, T::Int)
    output = 0.0
    K = length(data) - T
    for i in (T+1):(T+K)
        output += abs(data[i] - estimative[i]) / K
```

```
    end
    return output
end
```

Foi implementada também uma função para fazer a previsões:

```
function forecast(data::Vector{Float64}, K::Vector{Int} = [1], S::Vector{Int}
    = [8760], M::Vector{Int} = [1], T::Int = 0)
    model = complete_model(data[1:T], K, S, M)

    beta = value.(model[:beta])
    theta = value.(model[:theta])
    phi = value.(model[:phi])

    n_forecast = length(data) - T

    output = copy(data)
    n = length(beta) - 1
    m = length(S)

    for i in 1:n_forecast #T+1:length(data)
        AR = beta[1] + sum(beta[j+1]*output[T + i - K[j]] for j = 1:n)
        ST = sum(sum(theta[(j != 1 ? sum(M[l] for l = 1:j-1) : 0) + k]*cos(2*
    pi*k*(T+i)/S[j]) +
            phi[(j != 1 ? sum(M[l] for l = 1:j-1) : 0) + k]*sin(2*pi*k*(T+i)/S
    [j])
            for k in 1:M[j]) for j = 1:m)
        output[T+i] = AR + ST
    end

    return data, output, model
end
```

k	S	M	MAE
1	24, 8760	1, 1	0.14295381577654284
1, 24	24, 8760	1, 1	0.1237327839655492
1, 2, 23, 24	24, 8760	1, 1	0.078032376280466
1, 12, 24	24, 8760	1, 1	0.1253964348718293
1	24, 8760	3, 3	0.1339350457594182
1, 24	24, 8760	3, 3	0.11615823397344886
1, 2, 23, 24	24, 8760	3, 3	0.06839543906410278
1, 12, 24	24, 8760	3, 3	0.1190322322096669
1	24, 8760	5, 5	0.1714161598949759
1, 24	24, 8760	5, 5	0.15192924499296068
1, 2, 23, 24	24, 8760	5, 5	0.07970313173920848
1, 12, 24	24, 8760	5, 5	0.1550513045516485

2.4 Conclusão

Pela análise dos erros, percebemos que aumentando o número de termos em nosso modelo, a métrica R^2 tende a melhorar. Isso por que temos mais liberdade para aproximar bem o conjunto *in-sample*. Porém, quanto mais parâmetros existem no modelo, maior a chance dele aprender as particularidades da amostra de dados e não o comportamento da serie temporal.

Ao observar os valores da tabela 2, os erros MAE aumentaram quando usamos 5 harmônicos, isso é um forte indício, que a serie em si, não possui essas frequências, o ajuste do modelo deve ter captado ruído que atrapalhou a previsão da série.

Sendo assim, os melhores modelos foram os que usaram 3 harmônicos, eles obtiveram o menor erro nas previsões. Em particular o modelo $K = 1, 2, 23, 24, S = 24, 8760, M = 3, 3$. Este modelo possui termos regressivos e sazonais o suficiente para captar o comportamento da série, sem exagerar a ponto de guardar particularidades do conjunto de dados.

2.5 Previsões

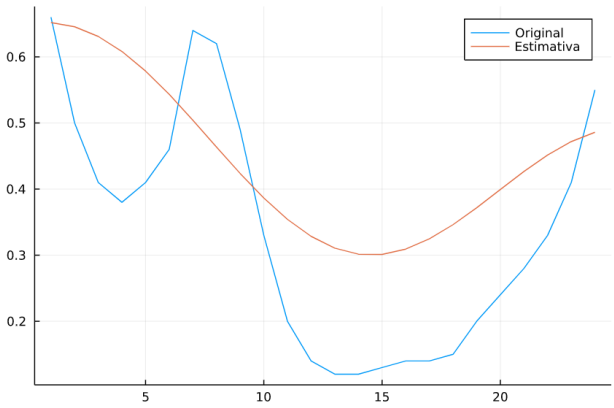

Figure 10: K = 1, S = 24, 8760 e M= 1, 1

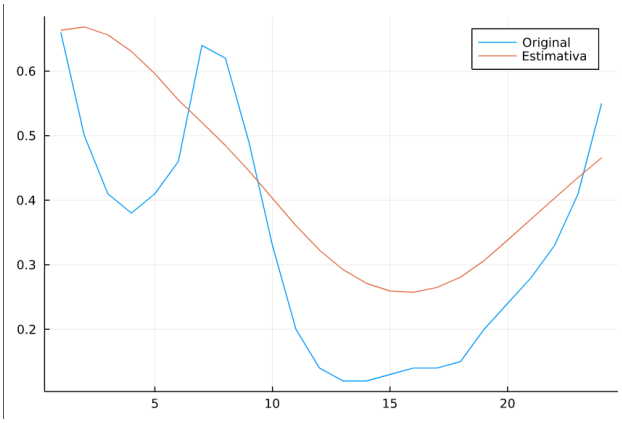

Figure 11: K = 1, 24, S = 24, 8760 e M= 1, 1

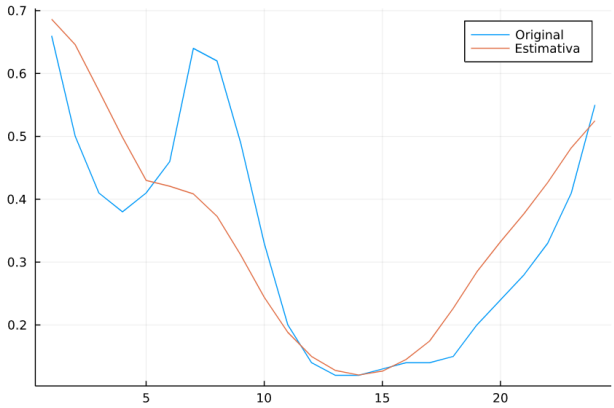

Figure 12: K = 1, 2, 23, 24, S = 24, 8760 e M= 1, 1

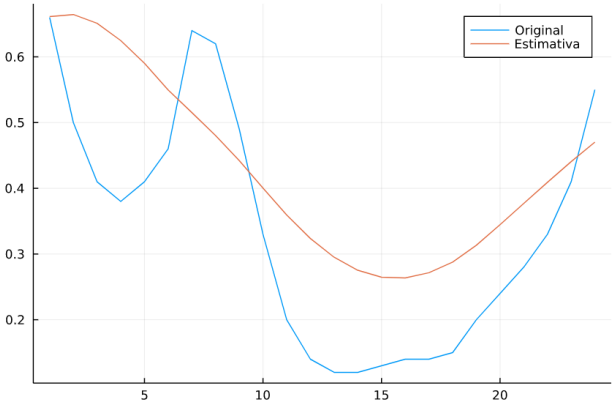

Figure 13: K = 1, 12, 24, S = 24, 8760 e M= 1, 1

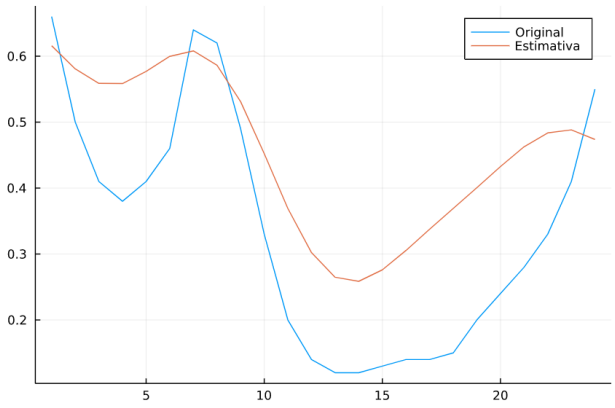

Figure 14: K = 1, S = 24, 8760 e M= 3, 3

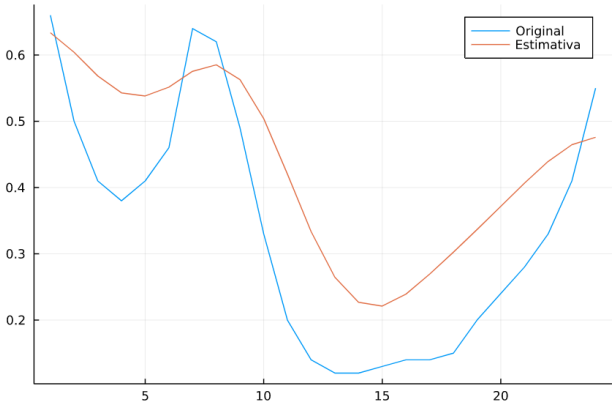

Figure 15: K = 1, 24, S = 24, 8760 e M= 3, 3

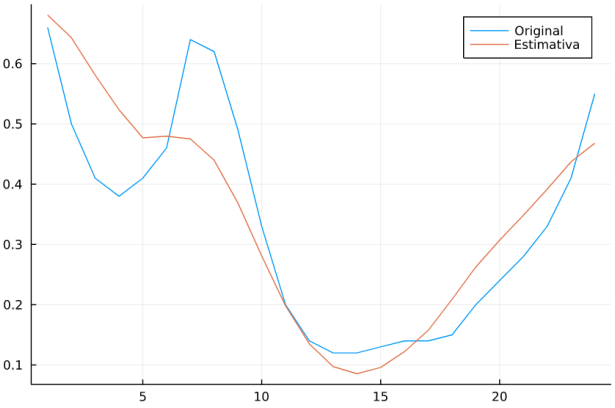

Figure 16: K = 1, 2, 23, 24, S = 24, 8760 e M= 3, 3

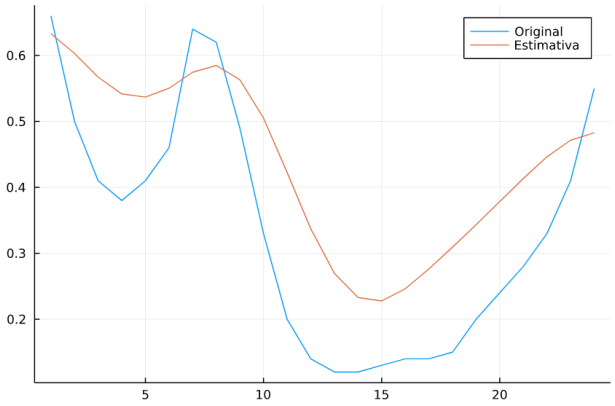

Figure 17: K = 1, 12, 24, S = 24, 8760 e M= 3, 3

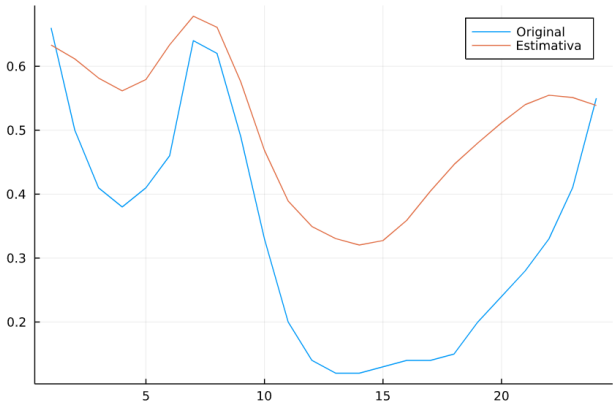

Figure 18: K = 1, S = 24, 8760 e M= 5, 5

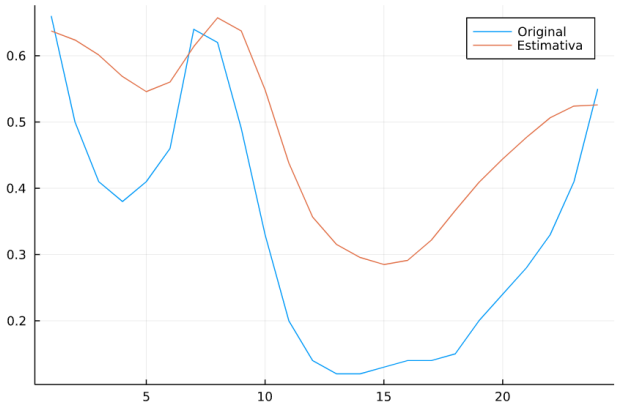

Figure 19: K = 1, 24, S = 24, 8760 e M= 5, 5

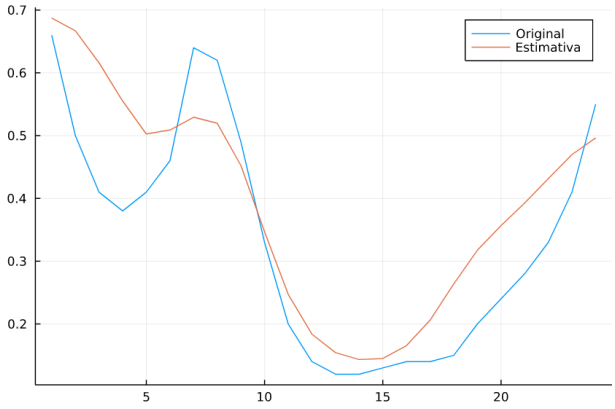

Figure 20: K = 1, 2, 23, 24, S = 24, 8760 e M= 5, 5

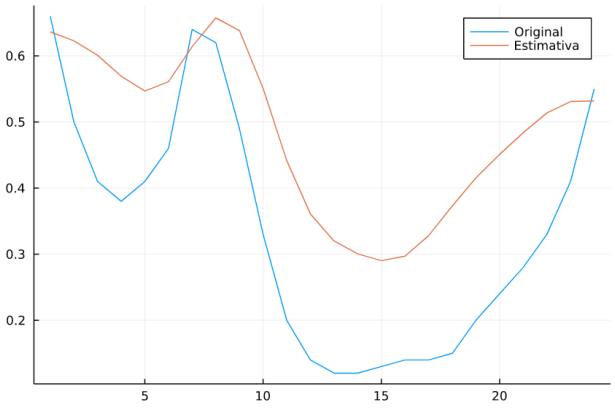

Figure 21: K = 1, 12, 24, S = 24, 8760 e M= 5, 5