Máquinas de Fluxo - Labs1

Alunos:

- Alessandro Cappellini Portugal
- Rodrigo Kenji Kuroda
- Jean Carlos Reisdorfer
- Laís Tussi
- Stella Colferai


```
using Plots , Distributions , Measurements , StatsPlots , DataFrames , Gadfly , PlutoUI , Statistics , StatsBase , LaTeXStrings , Latexify
```

```
y =
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,  more ,39, 40, 41,
```

```
x =
```

```
[-0.2, 1.0, 2.0, 3.1, 4.0, 5.0, 6.1, 7.3, 8.2, 9.25, 10.3, 11.3, 12.4, 13.45, 14.48, 15.1
```

```
σx =
```

```
[0.2, 0.1, 0.1, 0.1, 0.1, 0.05, 0.2, 0.1, 0.1, 0.05, 0.1, 0.1, 0.05, 0.05, 0.05, 0.1, 0.1,
```

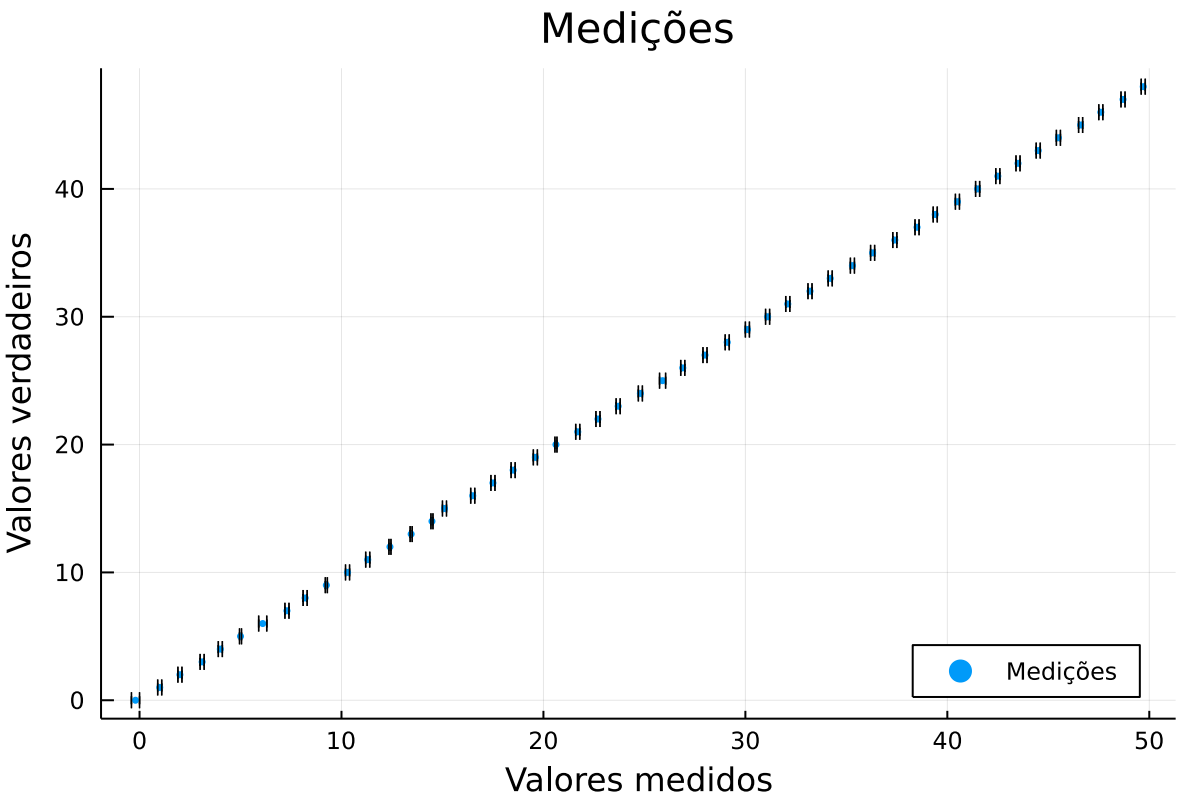
```
([0, 50], [0.163019, 0.163019])
```

```
begin
    #Definição dos limites de aceitação para α = 95%

    limy = [2std(d1), 2*std(d1)]
    limx = [0, 50]
    limx, limy
end
```

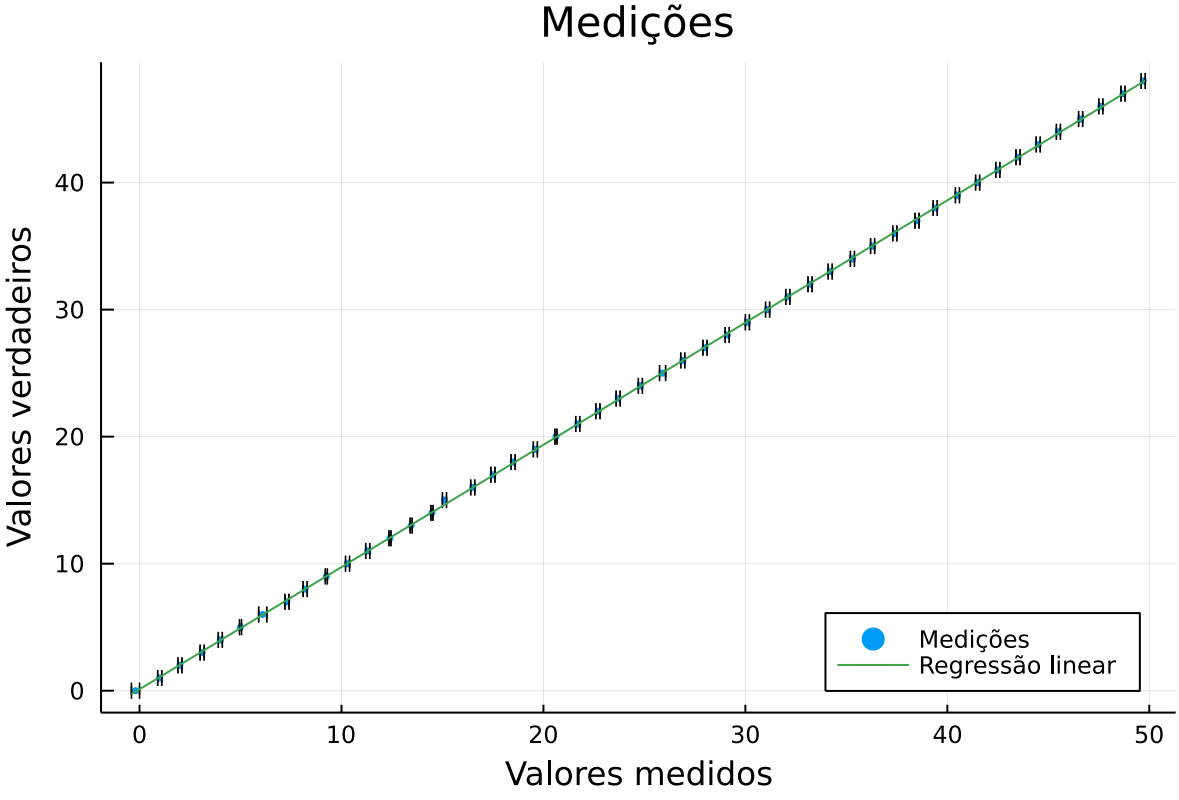
Sem filtrar dados espúrios

Equação 1° grau




```
ŷ (generic function with 1 method)
```

```
begin
    A = [x[i]^j for i in 1:49, j in 0:1]
    x̂ = A\y
    ŷ(x) = x̂' * [1,x]
end
```



$$f(x) = 0.1123 + 0.9622x$$

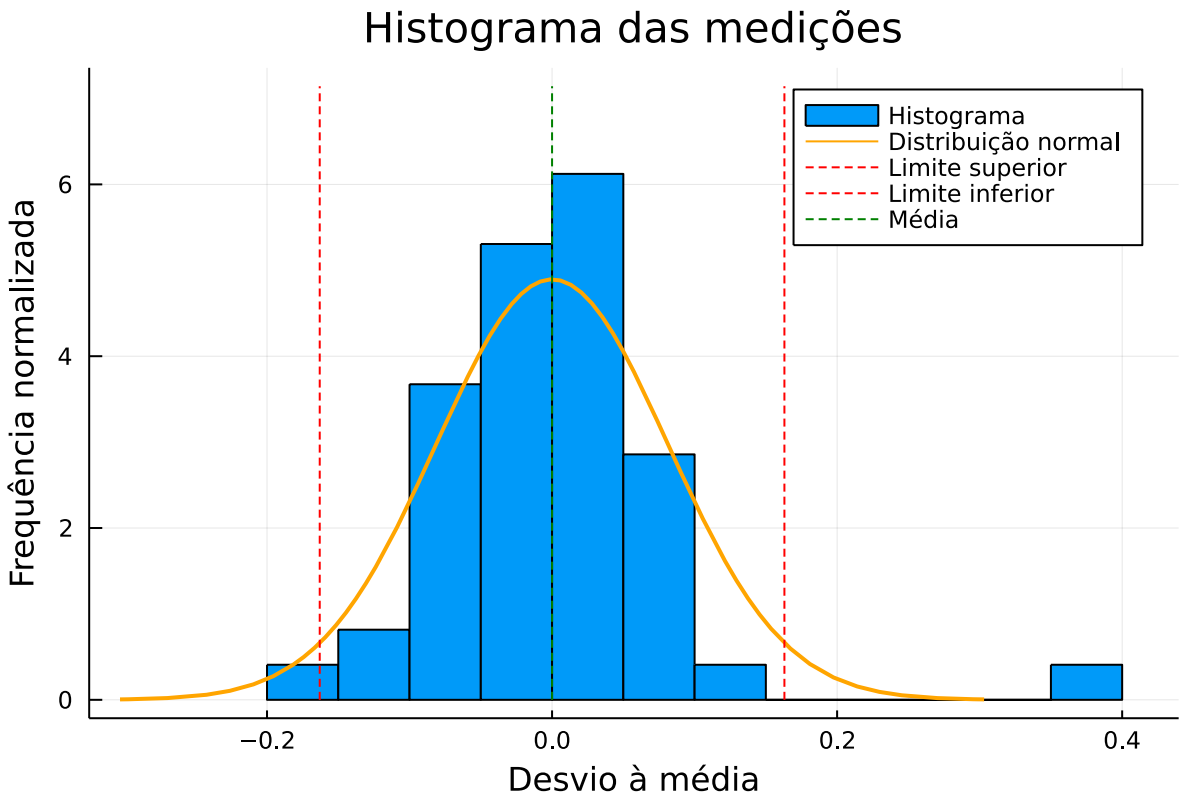
```
[0.112284, 0.962185]
```

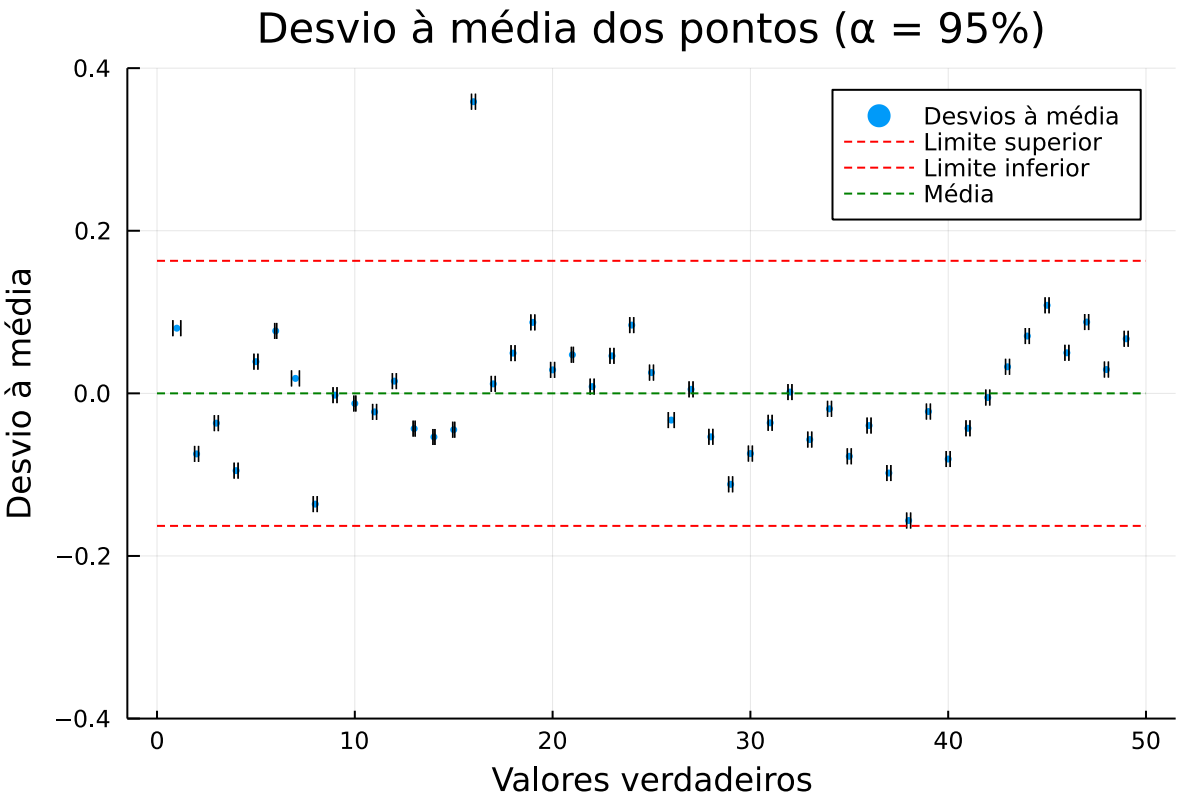
```
x̂
```

```
([-0.0801525, 1.07447, 2.03665, 3.09506, 3.96102, 4.92321, 5.98161, 7.13624, 8.0022,     r
```

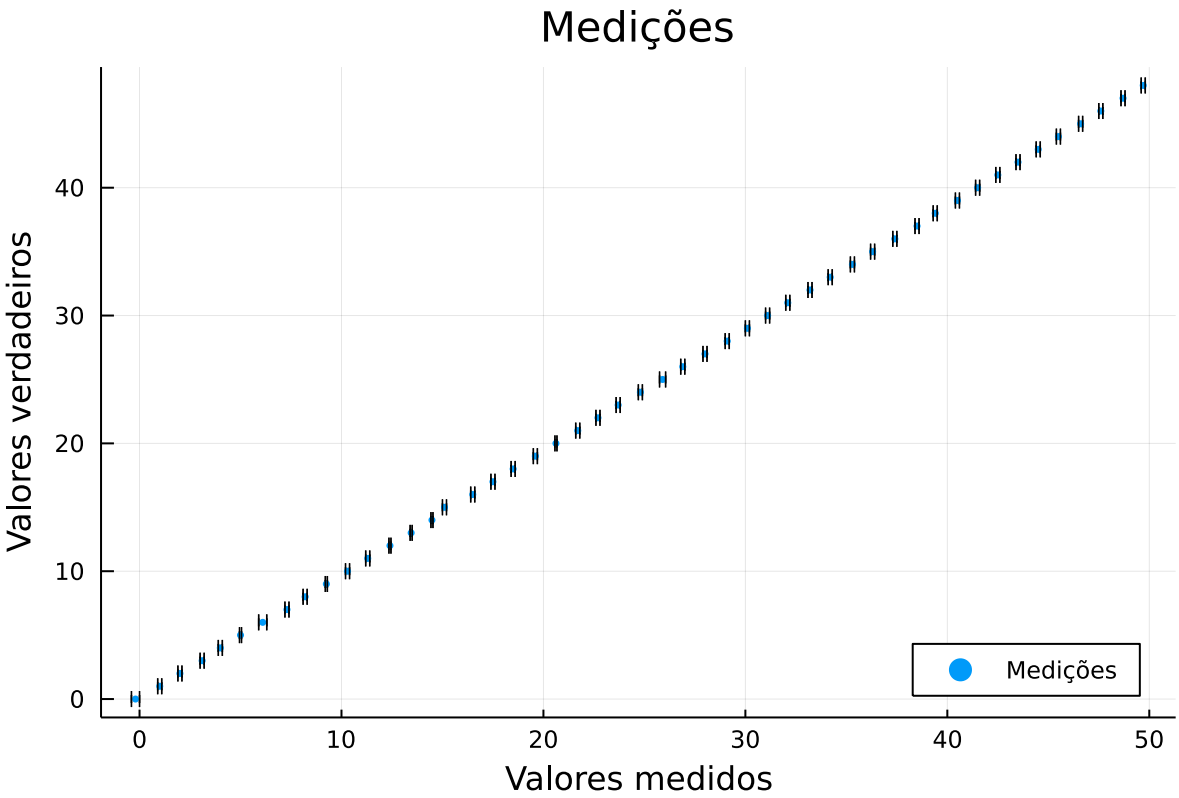
```
begin
    c1 = x̂[1]
    c2 = x̂[2]
    eq1(x) = c1 + c2*x

    r1 = convert(Array{Float64},[])
    for i in 1:49
        append!(r1, eq1(x[i]))
    end
    d1 = y .- r1
    r1, d1
end
```



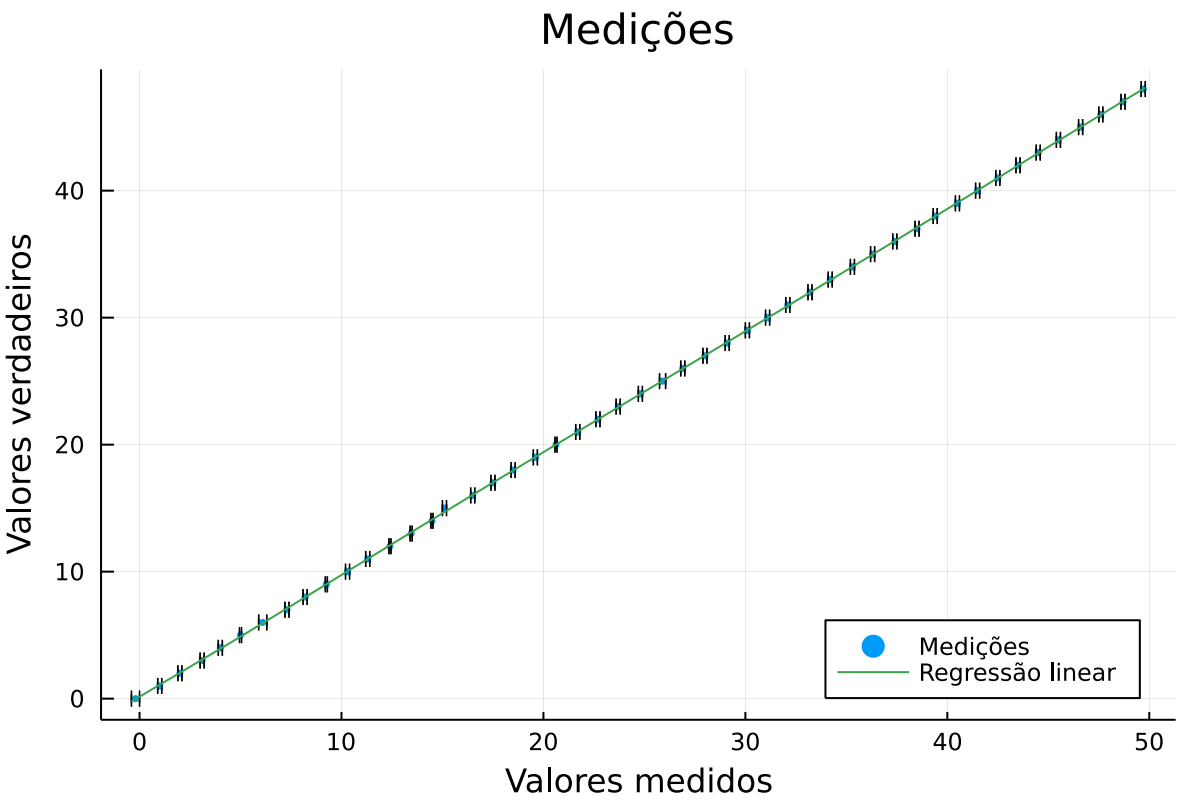


Equação 5° grau



```
ẏ (generic function with 1 method)
```

```
begin
    C = [x[i]^j for i in 1:49, j in 0:5]
    ẋ = C\y
    ẏ(x) = ẋ' * [1,x,x^2,x^3,x^4,x^5]
end
```



$$f(x) = 0.1516 + 0.9184x + 0.0071x^2 - 0.0004x^3 + 0.0x^4 - 0.0x^5$$

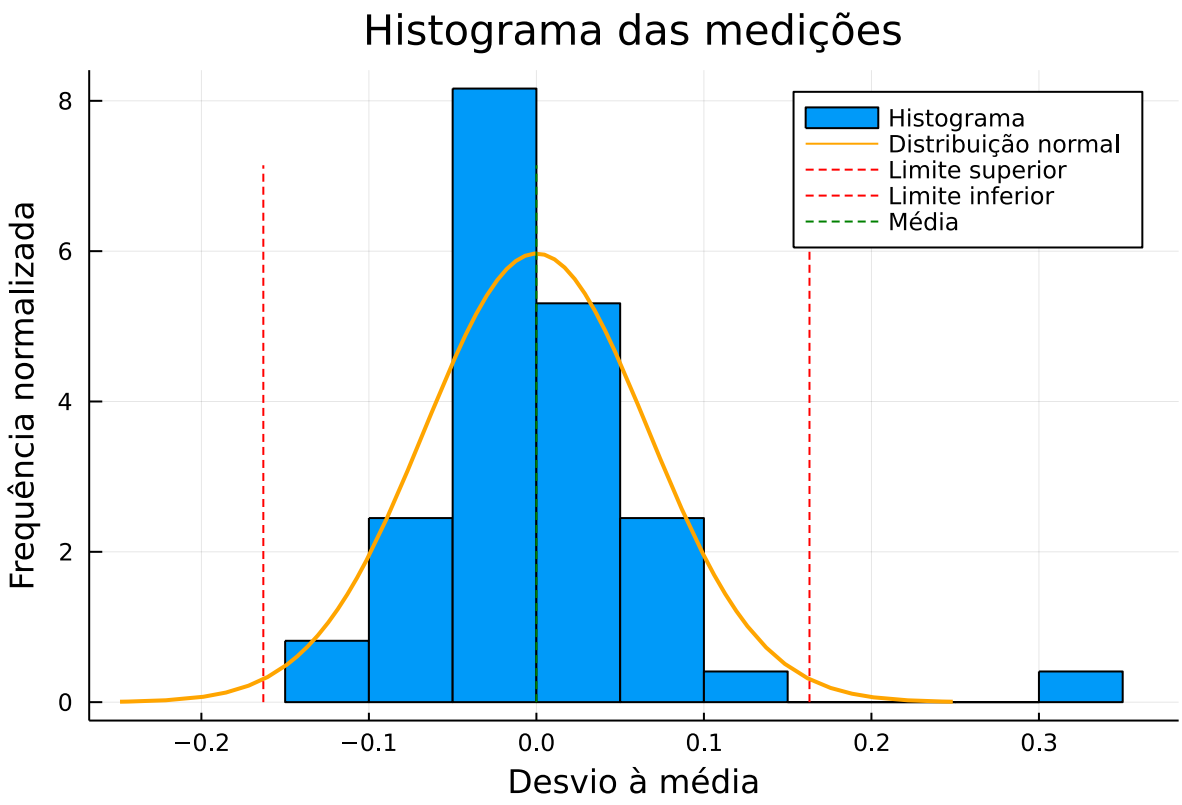
```
[0.151648, 0.918411, 0.0071463, -0.000391679, 8.57422e-6, -6.48792e-8]
```

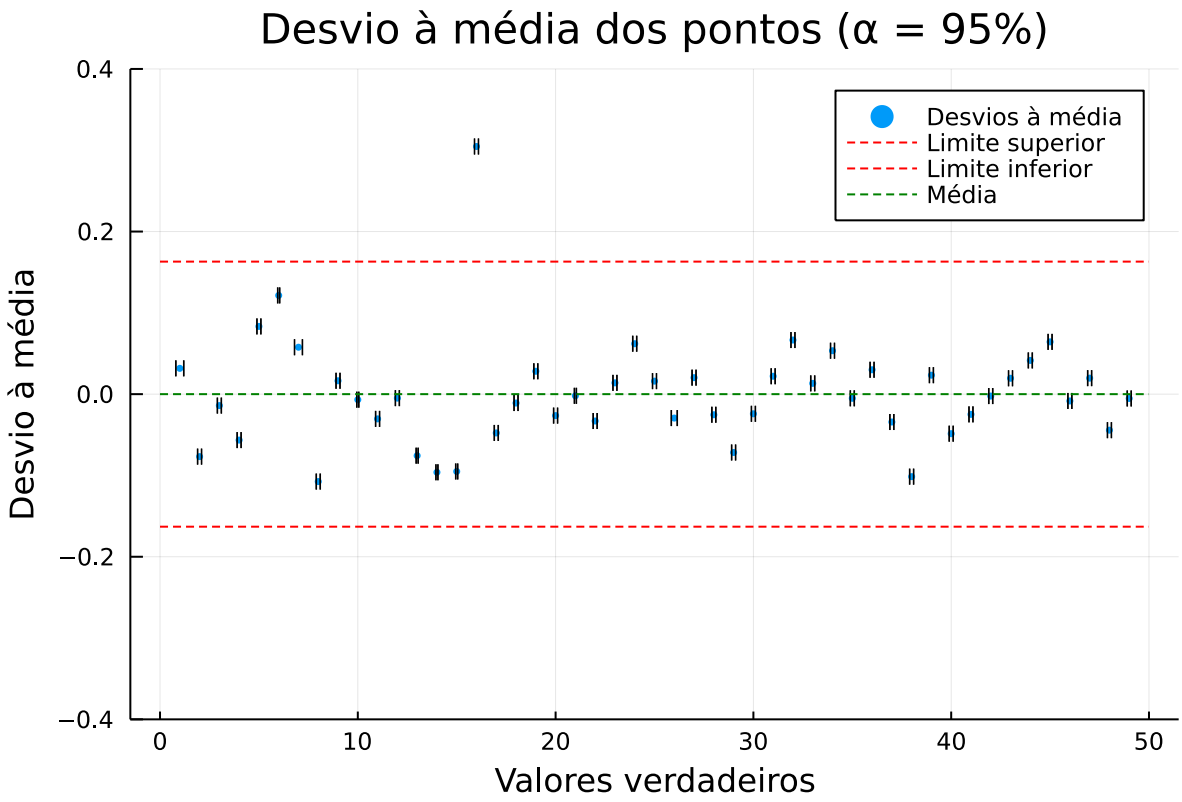
```
ẋ
```

```
([-0.0317451, 1.07682, 2.01406, 3.0565, 3.91669, 4.87856, 5.94229, 7.10751, 7.98354,    r
```

```
begin
    ṫ0 = ẋ[1]
    ṫ1 = ẋ[2]
    ṫ2 = ẋ[3]
    ṫ3 = ẋ[4]
    ṫ4 = ẋ[5]
    ṫ5 = ẋ[6]
    eq3(x) =ṫ0 + ṫ1*x + ṫ2*x^2 + ṫ3*x^3 + ṫ4*x^4 + ṫ5*x^5

    r3 = convert(Array{Float64},[])
    for i in 1:49
        append!(r3, eq3(x[i]))
    end
    d3 = y .- r3
    r3, d3
end
```





Filtrando os dados espúrios

Encontrar Outliers

A filtragem dos dados espúrios funciona pelo princípio de quartis , onde se aplicam as seguintes equações e assim descobre-se os pontos a serem descartados:

$$Lim_{mín} = Q_1 - 1.5(Q_3 - Q_1)$$

$$Lim_{máx} = Q_3 + 1.5(Q_3 - Q_1)$$

```
([15.1], [16])
```

```
begin
    outliers = []
    valoresd1 = []
    indexd1 = []

    Q = quantile(d1)
    Q1 = Q[2]
    Q3 = Q[4]

    out_min = Q1-1.5*(Q3-Q1)
    out_max = Q3+1.5*(Q3 -Q1)

    for i in d1
        if i < out_min
            append!(valoresd1,i)
        elseif i > out_max
            append!(valoresd1,i)
        end
    end

    for i in 1:49
        for j in valoresd1
            if j == d1[i]
                append!(indexd1, i)
            end
        end
    end

    for i in indexd1
        append!(outliers,x[i])
    end

    outliers, indexd1
end
```

Filtrar Outliers

```
1 ponto foi removido!
```

```
begin
    xx = convert(Array{Float64}, [])
    yy = convert(Array{Float64}, [])
    σxx = convert(Array{Float64}, [])
    for i in 1:49
        for j in outliers
            if x[i] != j
                append!(xx, x[i])
                append!(yy, y[i])
                append!(σxx, σx[i])
            end
        end
    end

    if size(outliers,1) == 0
        Print(size(outliers,1)," ponto foi removido!")
    elseif size(outliers,1) == 1
        Print(size(outliers,1)," ponto foi removido!")
    else
        Print(size(outliers,1)," pontos foram removidos!")
    end
end
```

Dados filtrados

Equação 1° grau

```
[0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 16.0, 17
```

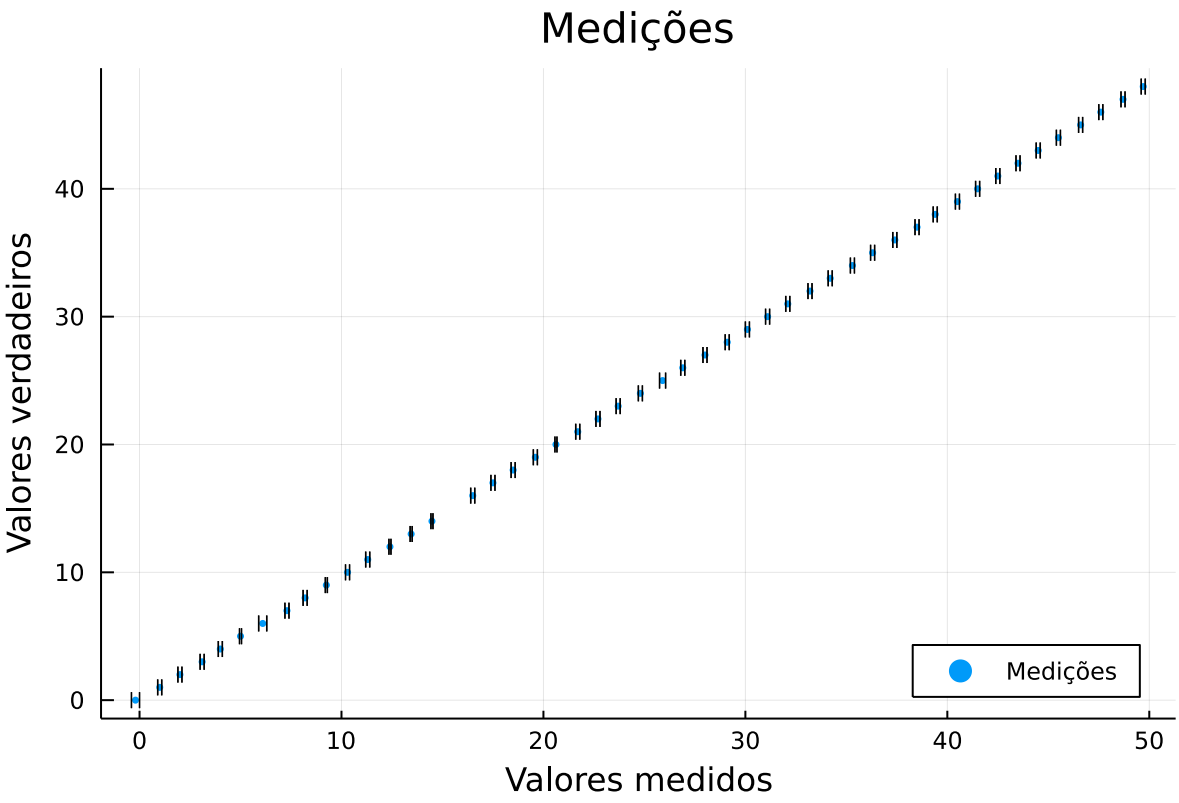
```
yy
```

```
[-0.2, 1.0, 2.0, 3.1, 4.0, 5.0, 6.1, 7.3, 8.2, 9.25, 10.3, 11.3, 12.4, 13.45, 14.48, 16.5
```

```
xx
```

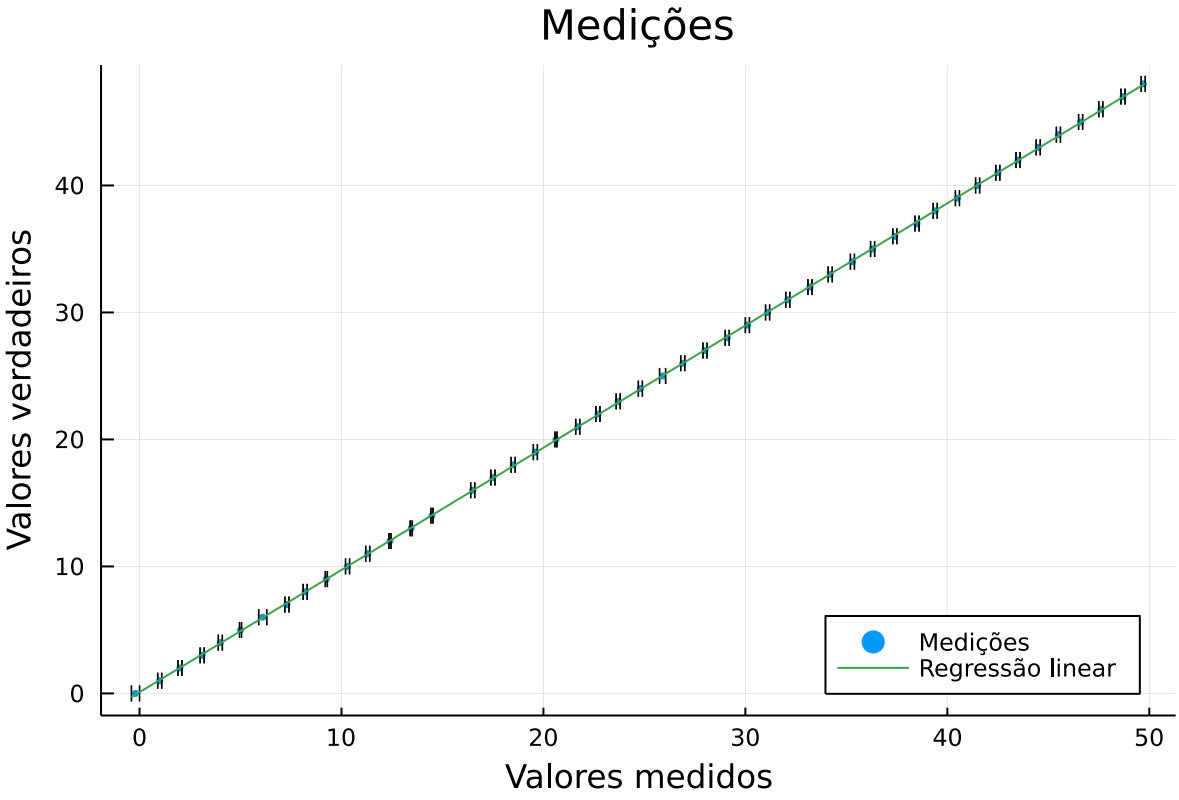
```
[0.2, 0.1, 0.1, 0.1, 0.1, 0.05, 0.2, 0.1, 0.1, 0.05, 0.1, 0.1, 0.05, 0.05, 0.05, 0.1, 0.1
```

```
σxx
```




```
ŷf (generic function with 1 method)
```

```
begin
    B = [xx[i]^j for i in 1:48, j in 0:1]
    x̂f = B\yy
    ŷf(xx) = x̂f' * [1,xx]
end
```



$$f(x) = 0.0963 + 0.9625x$$

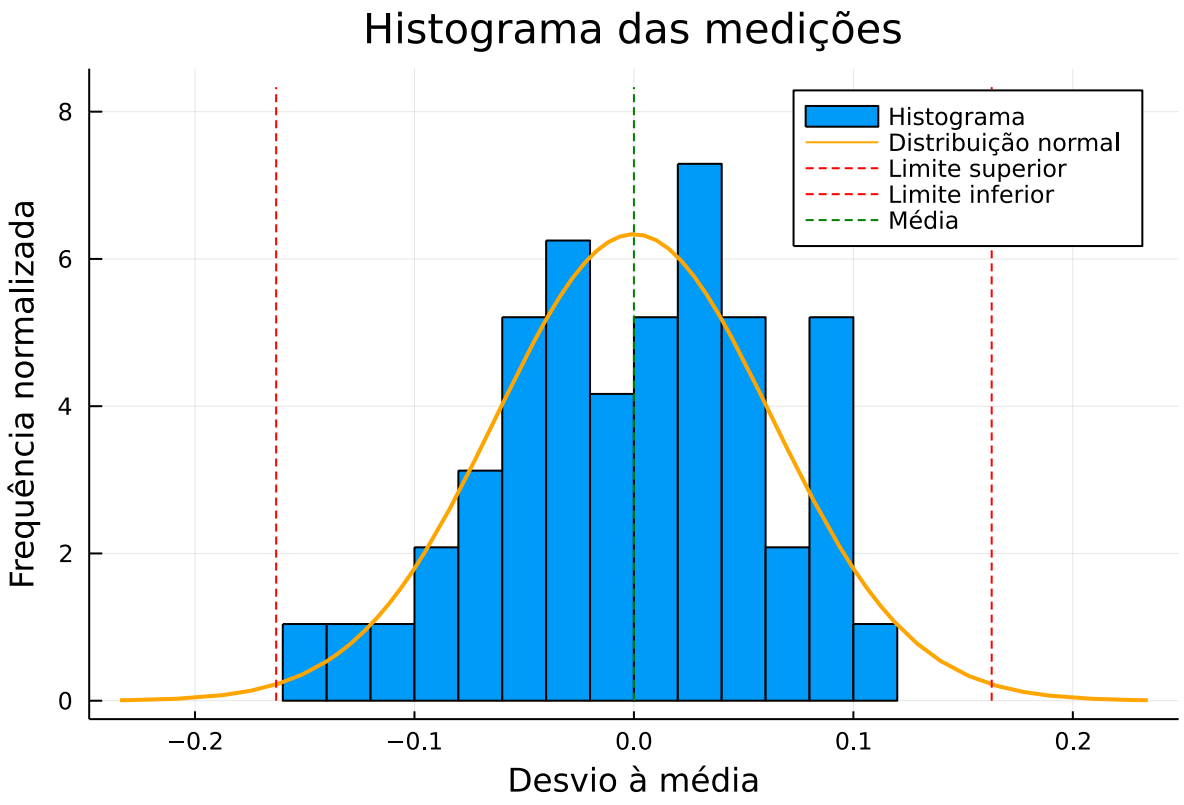
```
[0.0963114, 0.962525]
```

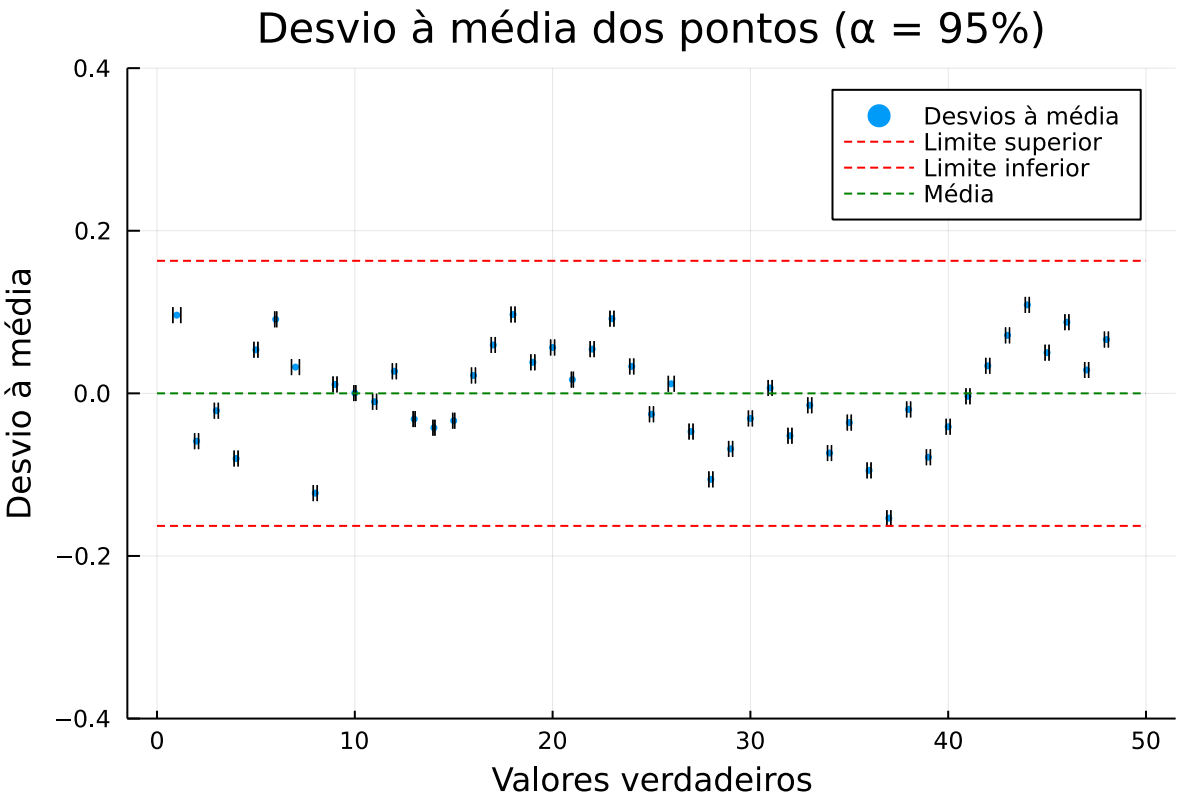
```
x̂f
```

```
([-0.0961935, 1.05884, 2.02136, 3.08014, 3.94641, 4.90893, 5.96771, 7.12274, 7.98901,
```

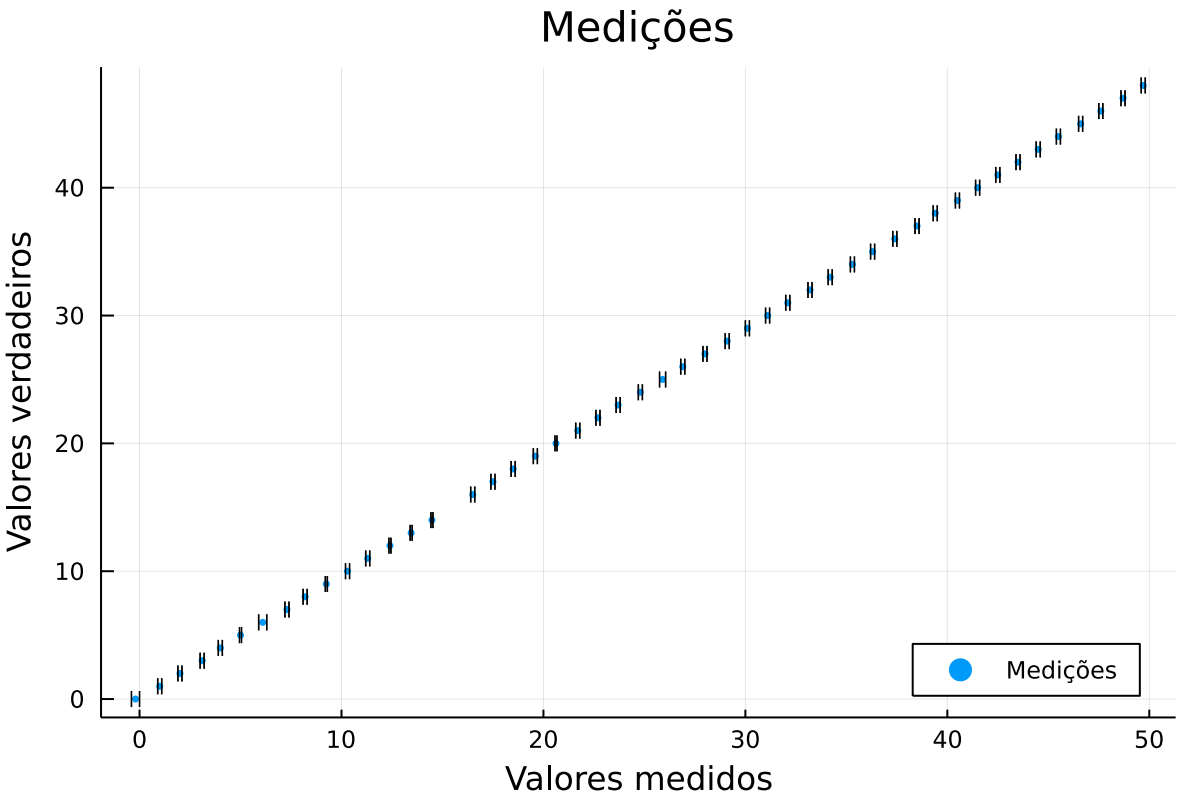
```
begin
    c3 = x̂f[1]
    c4 = x̂f[2]
    eq2(xx) = c3 + c4*xx

    r2 = convert(Array{Float64},[])
    for i in 1:48
        append!(r2, eq2(xx[i]))
    end
    d2 = yy .- r2
    r2, d2
end
```



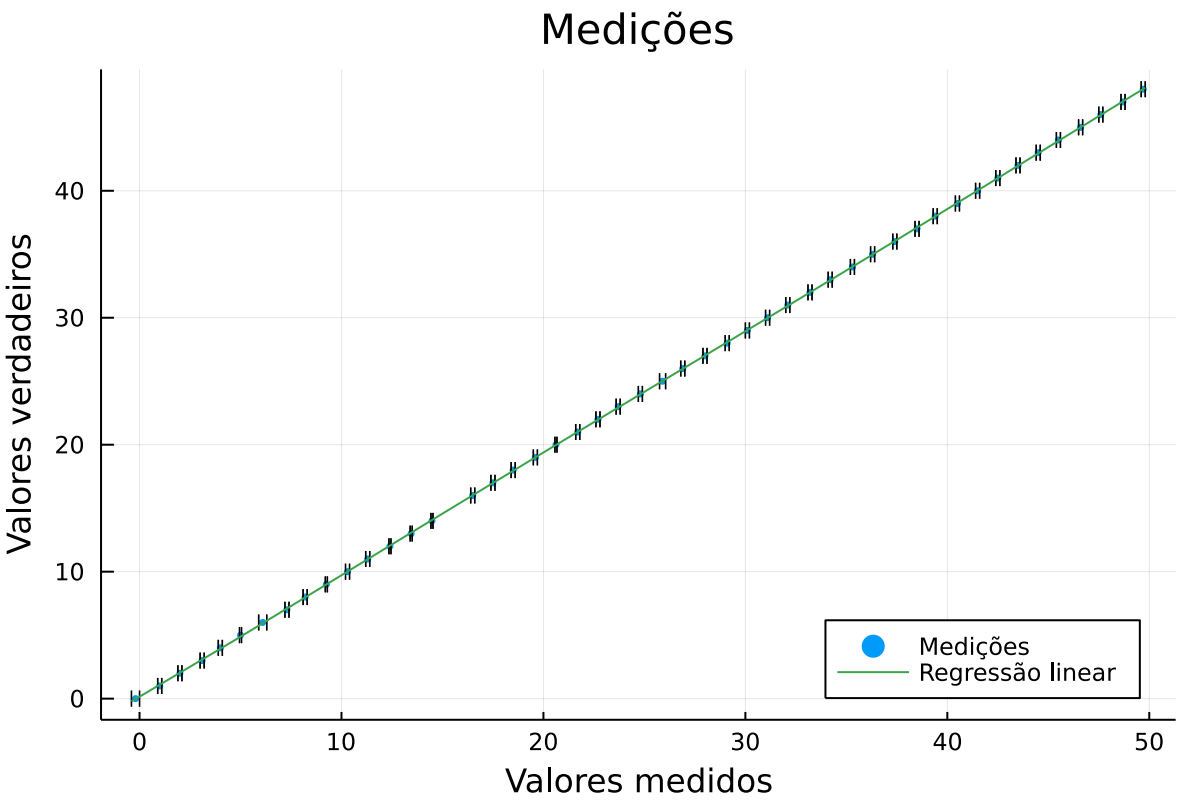


Equação 5° grau



```
ẏf (generic function with 1 method)
```

```
begin
    D = [xx[i]^j for i in 1:48, j in 0:5]
    ẋf = D\yy
    ẏf(xx) = ẋf' * [1,xx,xx^2,xx^3,xx^4,xx^5]
end
```



$$f(x) = 0.1529 + 0.9223x + 0.0059x^2 - 0.0003x^3 + 0.0x^4 - 0.0x^5$$

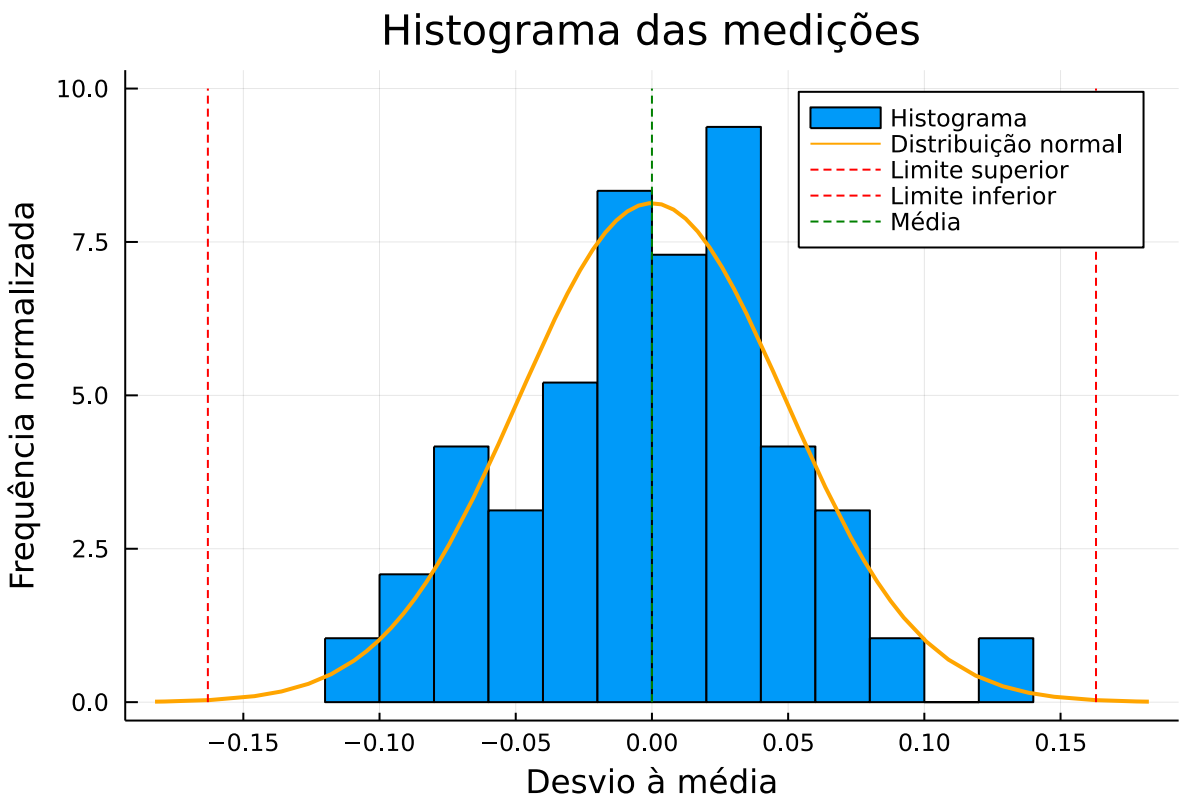
```
[0.152861, 0.922253, 0.00594186, -0.000311264, 6.6053e-6, -4.8604e-8]
```

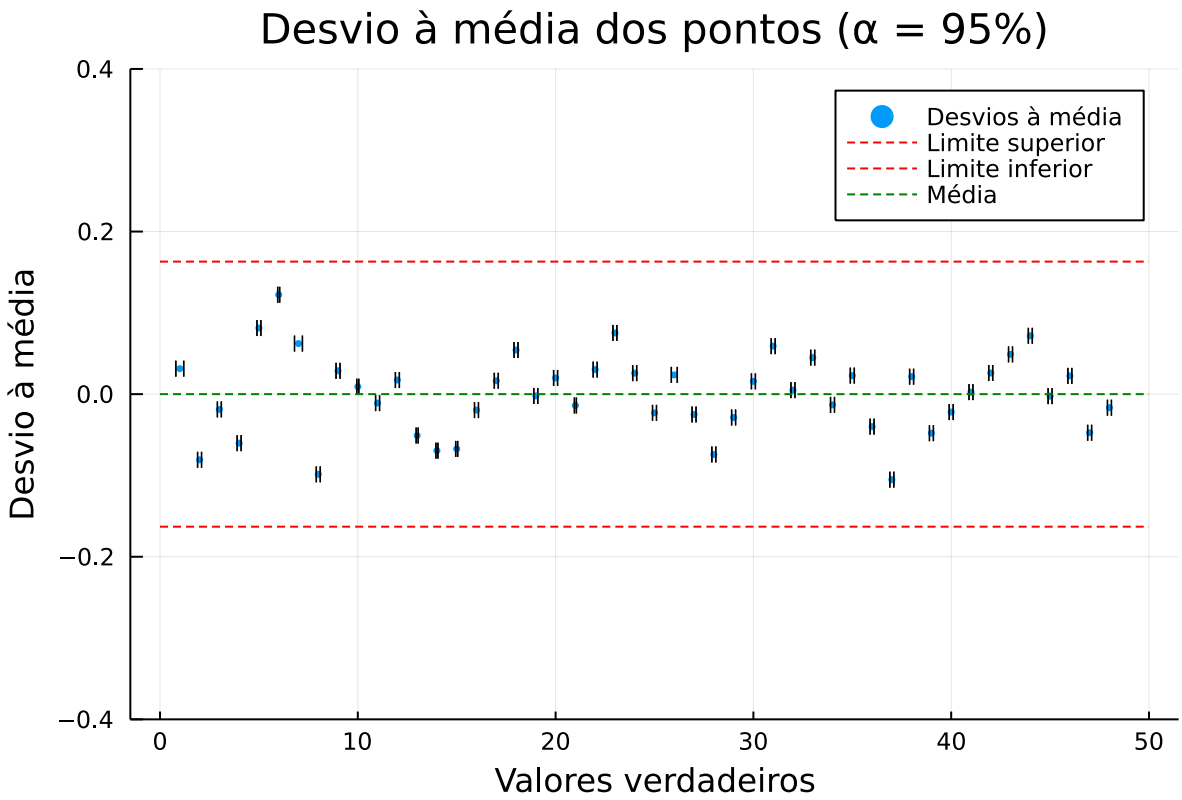
```
ẋf
```

```
([-0.031349, 1.08075, 2.01875, 3.06027, 3.91866, 4.87774, 5.93779, 7.09861, 7.97131,    r
```

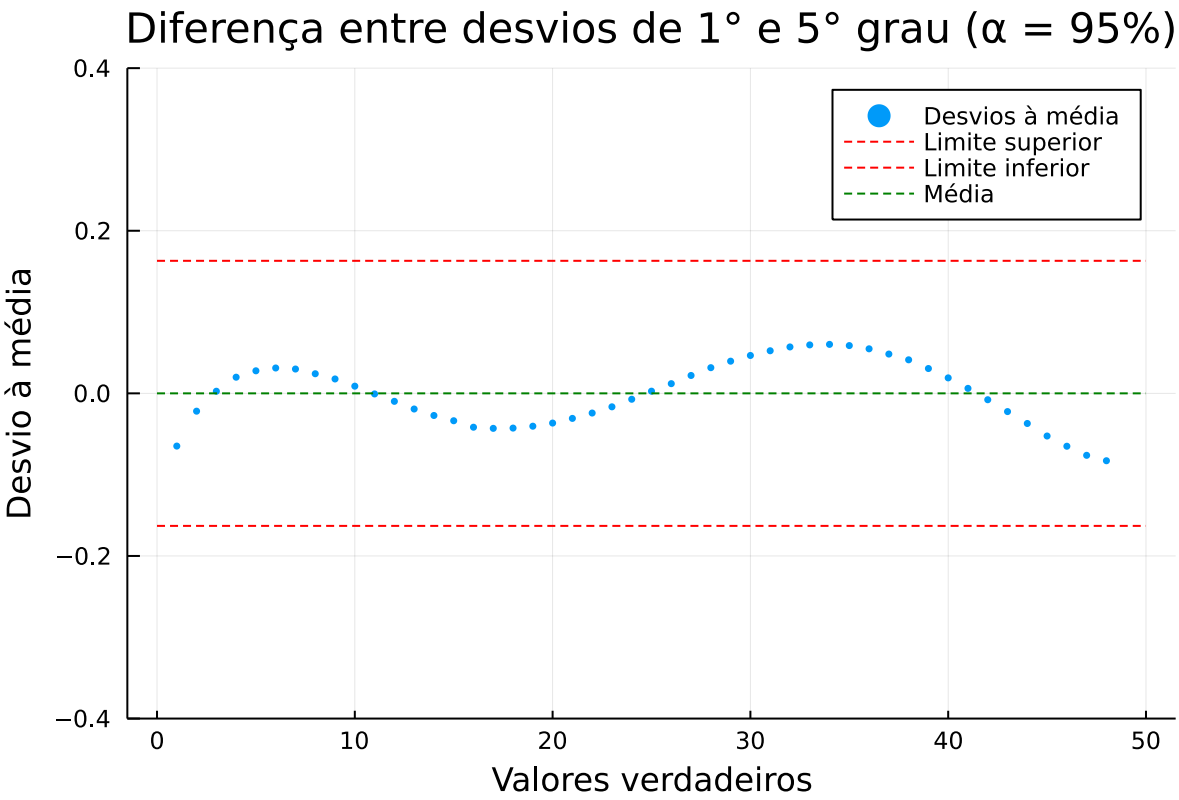
```
begin
    t0 = ẋf[1]
    t1 = ẋf[2]
    t2 = ẋf[3]
    t3 = ẋf[4]
    t4 = ẋf[5]
    t5 = ẋf[6]
    eq4(xx) =t0 + t1*xx + t2*xx^2 + t3*xx^3 + t4*xx^4 + t5*xx^5

    r4 = convert(Array{Float64},[])
    for i in 1:48
        append!(r4, eq4(xx[i]))
    end
    d4 = yy .- r4
    r4, d4
end
```





Comparação de desvios à média entre 5° grau e 1° grau


Diferença entre desvios de 1° e 5° grau (α = 95%)
Desvios à média
Limite superior
Limite inferior
Média
Desvio à média
Valores verdadeiros
0.4
0.2
0.0
−0.2
−0.4
0
10
20
30
40
50