```
using Pkg ✓
```

```
begin
    using Optim ✓

    using StanSample ✓
    using StanOptimize ✓

    using GLMakie ✓

    using RegressionAndOtherStories ✓
end
```

```
stan_a7 = "
parameters {
    real x;
}
model {
    target += 15 + 10*x - 2*x^2;
}
";
```

▸ Dict("lp__" ⇒ [27.5, 27.5, 27.5, 27.5], :stan_version ⇒ "2.30.0", "x" ⇒ [2.5, 2.5,

```
begin
    a7 = OptimizeModel("A.7", stan_a7)
    a7o = stan_optimize(a7)
    optim, cnames = read_optimize(a7)
    optim
end
```

```
/var/folders/l7/pr04h0650q5dvqttnvs8s2c00000gn/T/jl_r1krKN/A.7.stan updated.
```

▸ ["lp__", "x"]

```
cnames
```

x̄ = ▸ [2.5, 2.5, 2.5, 2.5]

```
x̄ = optim["x"]
```

ȳ = ▸ [27.5, 27.5, 27.5, 27.5]

```
ȳ = optim["lp__"]
```

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

```
function fun(x)
    return 15 + 10 * x - 2 * x^2
end
```

Using Julia's Optim.jl package

```
* Status: success

* Candidate solution
   Final objective value:     -2.750000e+01

* Found with
   Algorithm:     Nelder-Mead

* Convergence measures
   √(Σ(yᵢ-ȳ)²)/n ≤ 1.0e-08

* Work counters
   Seconds run:   0  (vs limit Inf)
   Iterations:    11
   f(x) calls:    25
```

```
begin
    f1(x::Vector) = -(15 .+ 10 .* x[1] .- 2 .* x[1].^2)
    res = optimize(f1, [5.0])
end
```

▸ [2.5]

```
Optim.minimizer(res)
```

Failed to show value:

```
MethodError: no method matching gl_convert(::Vector{Float64})
Closest candidates are:
gl_convert(!Matched::T) where T<:GeometryBasics.Mesh
@ GLMakie ~/.julia/packages/GLMakie/IhyZ5/src/GLAbstraction/GLUniforms.jl:194
gl_convert(!Matched::T) where T<:Number
@ GLMakie ~/.julia/packages/GLMakie/IhyZ5/src/GLAbstraction/GLUniforms.jl:191
gl_convert(!Matched::T) where T<:ColorTypes.Colorant
@ GLMakie ~/.julia/packages/GLMakie/IhyZ5/src/GLAbstraction/GLUniforms.jl:192
...
```

1. **#map#19** @ *Observables.jl:444* [inlined]
2. **map** @ *Observables.jl:442* [inlined]
3. **const_lift** @ *GLUtils.jl:107* [inlined]
4. **gl_convert**(::Observables.Observable{Vector{Float64}}) @ *GLUniforms.jl:226*
5. **GLMakie.GLAbstraction.RenderObject**(::Dict{Symbol, Any}, ::GLMakie.GLVisualizeShader, ::GLMakie.GLAbstraction.StandardPrerender, ::Nothing, ::GeometryBasics.HyperRectangle{3, Float32}, ::Nothing) @ *GLTypes.jl:336*
6. **assemble_robj**(::Dict{Symbol, Any}, ::GLMakie.GLVisualizeShader, ::GeometryBasics.HyperRectangle{3, Float32}, ::UInt32, ::Nothing, ::Nothing) @ *visualize_interface.jl:100*
7. **assemble_shader**(::Dict{Symbol, Any}) @ *visualize_interface.jl:118*
8. **draw_linesegments**(::Any, ::Observables.Observable{Vector{GeometryBasics.Point{2, Float32}}}, ::Dict) @ *lines.jl:138*
9. (::GLMakie.var"#197#198"{GLMakie.Screen, Makie.Scene, MakieCore.LineSegments{Tuple{Vector{GeometryBasics.Point{2, Float32}}}}}) (::Dict{Symbol, Any}) @ *drawing_primitives.jl:282*
10. (::GLMakie.var"#173#176"{GLMakie.var"#197#198"{GLMakie.Screen, Makie.Scene, MakieCore.LineSegments{Tuple{Vector{GeometryBasics.Point{2, Float32}}}}}, GLMakie.Screen, Makie.Scene, MakieCore.LineSegments{Tuple{Vector{GeometryBasics.Point{2, Float32}}}}}) () @ *drawing_primitives.jl:105*
11. **get!**(::GLMakie.var"#173#176"{GLMakie.var"#197#198"{GLMakie.Screen, Makie.Scene, MakieCore.LineSegments{Tuple{Vector{GeometryBasics.Point{2, Float32}}}}}, GLMakie.Screen, Makie.Scene, MakieCore.LineSegments{Tuple{Vector{GeometryBasics.Point{2, Float32}}}}}, ::Dict{UInt64, GLMakie.GLAbstraction.RenderObject}, ::UInt64) @ *dict.jl:468*
12. **cached_robj!**(::GLMakie.var"#197#198"{GLMakie.Screen, Makie.Scene, MakieCore.LineSegments{Tuple{Vector{GeometryBasics.Point{2, Float32}}}}}, ::GLMakie.Screen, ::Makie.Scene, ::MakieCore.LineSegments{Tuple{Vector{GeometryBasics.Point{2, Float32}}}}) @ *drawing_primitives.jl:83*
13. **draw_atomic** @ *drawing_primitives.jl:261* [inlined]
14. **insert!**(::GLMakie.Screen, ::Makie.Scene, ::MakieCore.LineSegments{Tuple{Vector{GeometryBasics.Point{2, Float32}}}}) @ *drawing_primitives.jl:120*
15. (::GLMakie.var"#179#180"{GLMakie.Screen, Makie.Scene}) (::MakieCore.LineSegments{Tuple{Vector{GeometryBasics.Point{2, Float32}}}}) @ *drawing_primitives.jl:125*
16. **foreach**(::GLMakie.var"#179#180"{GLMakie.Screen, Makie.Scene}, ::Vector{MakieCore.AbstractPlot}) @ *abstractarray.jl:2798*

```
17. insert!(::GLMakie.Screen, ::Makie.Scene, ::MakieCore.Combined{Makie.hlines,
    Tuple{Vector{Float64}}}) @ drawing_primitives.jl:122
18. insertplots!(::GLMakie.Screen, ::Makie.Scene) @ screen.jl:65
19. (::GLMakie.var"#104#106"{GLMakie.Screen})(::Makie.Scene) @ screen.jl:67
20. foreach(::GLMakie.var"#104#106"{GLMakie.Screen},
    ::Vector{Makie.Scene}) @ abstractarray.jl:2798
21. insertplots!(::GLMakie.Screen, ::Makie.Scene) @ screen.jl:67
22. (::GLMakie.var"#104#106"{GLMakie.Screen})(::Makie.Scene) @ screen.jl:67
23. foreach(::GLMakie.var"#104#106"{GLMakie.Screen},
    ::Vector{Makie.Scene}) @ abstractarray.jl:2798
24. insertplots!(::GLMakie.Screen, ::Makie.Scene) @ screen.jl:67
25. scene2image(::Makie.Scene) @ display.jl:41
26. backend_show(::GLMakie.GLBackend, ::IOContext{IOBuffer},
    ::MIME{Symbol("image/png")}, ::Makie.Scene) @ display.jl:46
27. show(::IOContext{IOBuffer}, ::MIME{Symbol("image/png")},
    ::Makie.Figure) @ display.jl:117
28. show_richest(::IOContext{IOBuffer}, ::Any) @ PlutoRunner.jl:1116
29. show_richest_withreturned @ PlutoRunner.jl:1049 [inlined]
30. format_output_default(::Any, ::Any) @ PlutoRunner.jl:957
31. #format_output#50 @ PlutoRunner.jl:974 [inlined]
32. formatted_result_of(::Base.UUID, ::Base.UUID, ::Bool, ::Vector{String},
    ::Nothing, ::Module) @ PlutoRunner.jl:867
33. top-level scope @ none:1
```

```
begin
    f = Figure()
    ax = Axis(f[1, 1]; title="Apendix A.7 optimization")
    x = collect(LinRange(-2.0, 5.0, 100))
    lines!(x, [fun(v) for v in x])
    vlines!(ax, [optim["x"][1]]; color=:red)
    hlines!(ax, [optim["lp__"][1]]; color=:grey, xmin=[0.50], xmax=[0.8])
    annotations!("x̄ = $(optim["x"])", position=(-2,24), textsize=15)
    annotations!("fun(x̄) = $(optim["lp__"])", position=(-2, 22), textsize=15)
    f
end
```