Lab 5: Sea-Level Rise

Fri., Feb. 23

# Setup

## Load packages

using CSV  
using DataFrames  
using DataFramesMeta  
using Distributions  
using Plots  
using StatsPlots  
using Unitful  
  
Plots.default(; margin=5Plots.mm)

## Local package

using Revise  
using HouseElevation

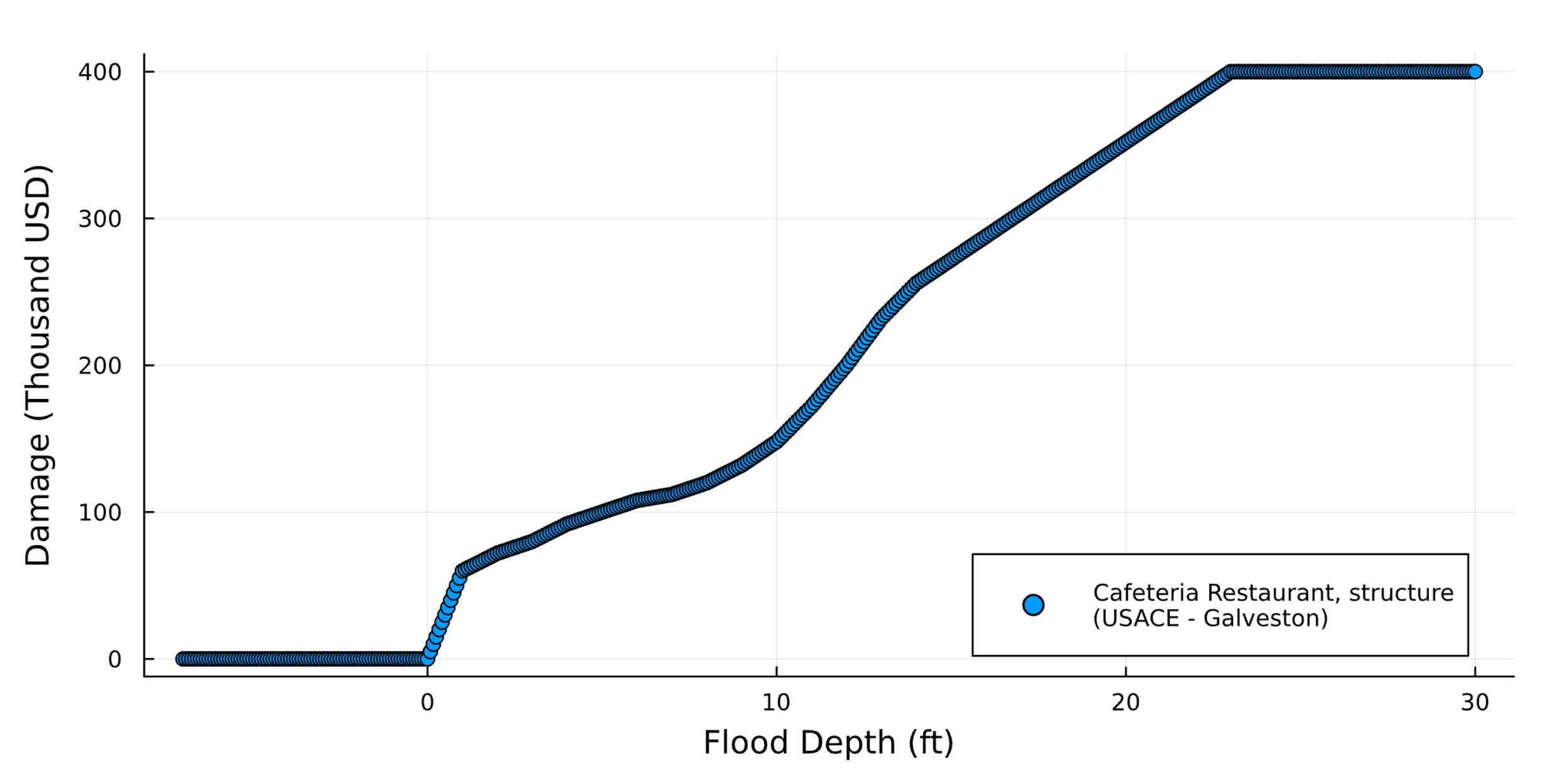
## House Setup and Plot

I will be using the house I chose for Lab 3. The house is the Fisherman’s Wharf which is a restraunt. There is no clear house area and house hight, I will still be using the data from previous labs. The house value should be a little bit more since it is a legacy of great seafood since the 1940s.

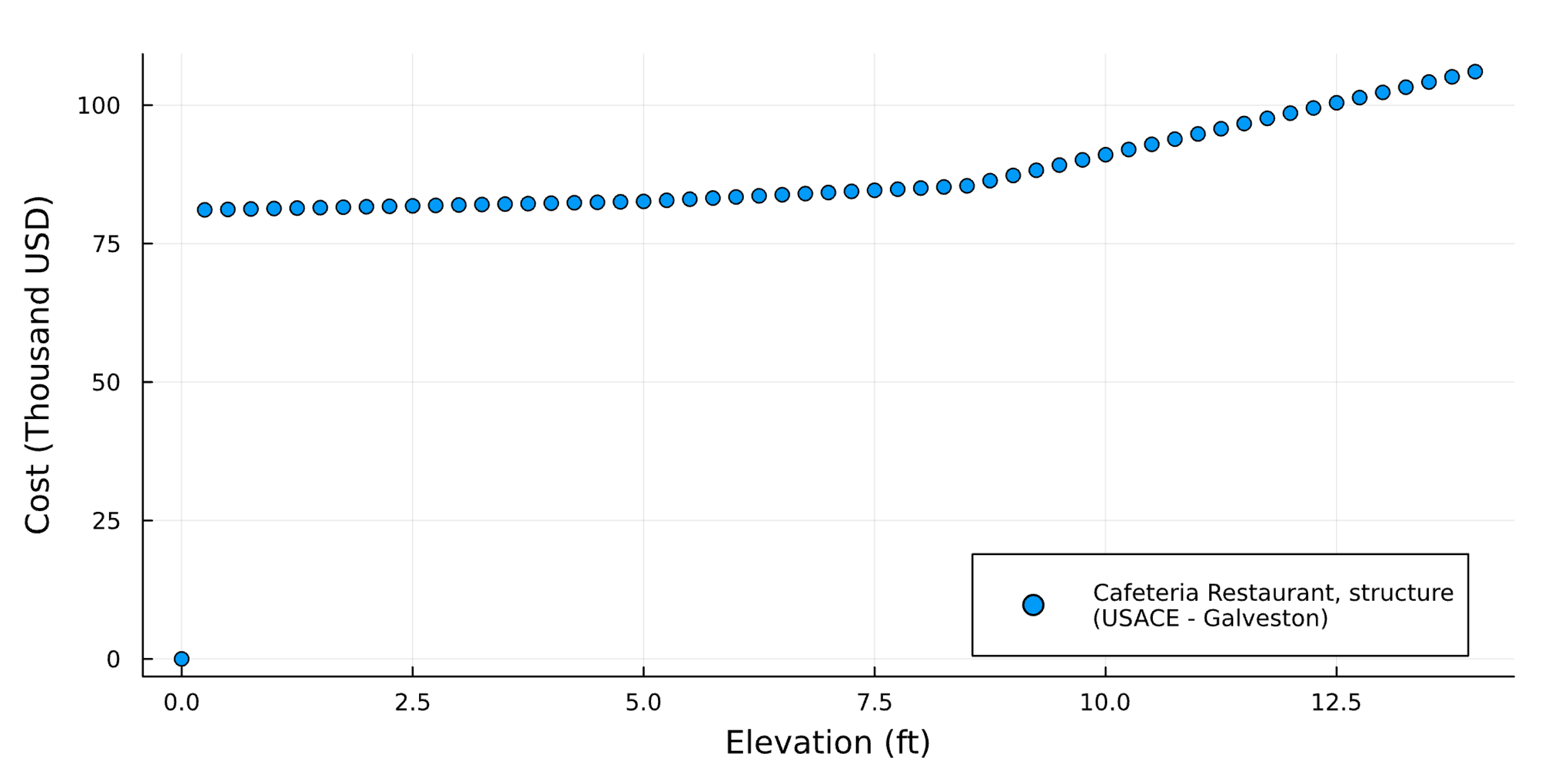
house = let  
 haz\_fl\_dept = CSV.read("data/haz\_fl\_dept.csv", DataFrame)   
 desc = "Cafeteria Restaurant, structure"  
 row = @rsubset(haz\_fl\_dept, :Description == desc)[1, :]   
 area = 750u"ft^2"  
 height\_above\_gauge = 4u"ft"  
 House(  
 row;  
 area=area,  
 height\_above\_gauge=height\_above\_gauge,  
 value\_usd=400\_000,  
 )  
end

House{Int64}(750, 400000, 4, DepthDamageFunction{Interpolations.Extrapolation{Float64, 1, Interpolations.GriddedInterpolation{Float64, 1, Vector{Float64}, Interpolations.Gridded{Interpolations.Linear{Interpolations.Throw{Interpolations.OnGrid}}}, Tuple{Vector{Float64}}}, Interpolations.Gridded{Interpolations.Linear{Interpolations.Throw{Interpolations.OnGrid}}}, Interpolations.Flat{Nothing}}}(29-element extrapolate(interpolate((::Vector{Float64},), ::Vector{Float64}, Gridded(Linear())), Flat()) with element type Float64:  
 0.0  
 0.0  
 0.0  
 0.0  
 0.0  
 15.0  
 18.0  
 20.0  
 23.0  
 25.0  
 27.0  
 28.0  
 30.0  
 ⋮  
 58.0  
 64.0  
 68.0  
 72.0  
 76.0  
 80.0  
 84.0  
 88.0  
 92.0  
 96.0  
 100.0  
 100.0), String7("COM8"), "504", String31("USACE - Galveston"), "Cafeteria Restaurant, structure", "missing")

let  
 depths = uconvert.(u"ft", (-7.0u"ft"):(1.0u"inch"):(30.0u"ft"))  
 damages = house.ddf.(depths) ./ 100  
 damages\_1000\_usd = damages .\* house.value\_usd ./ 1000  
 scatter(  
 depths,  
 damages\_1000\_usd;  
 xlabel="Flood Depth",  
 ylabel="Damage (Thousand USD)",  
 label="$(house.description)\n($(house.source))",  
 legend=:bottomright,  
 size=(800, 400),  
 yformatter=:plain, # prevents scientific notation  
 )  
end



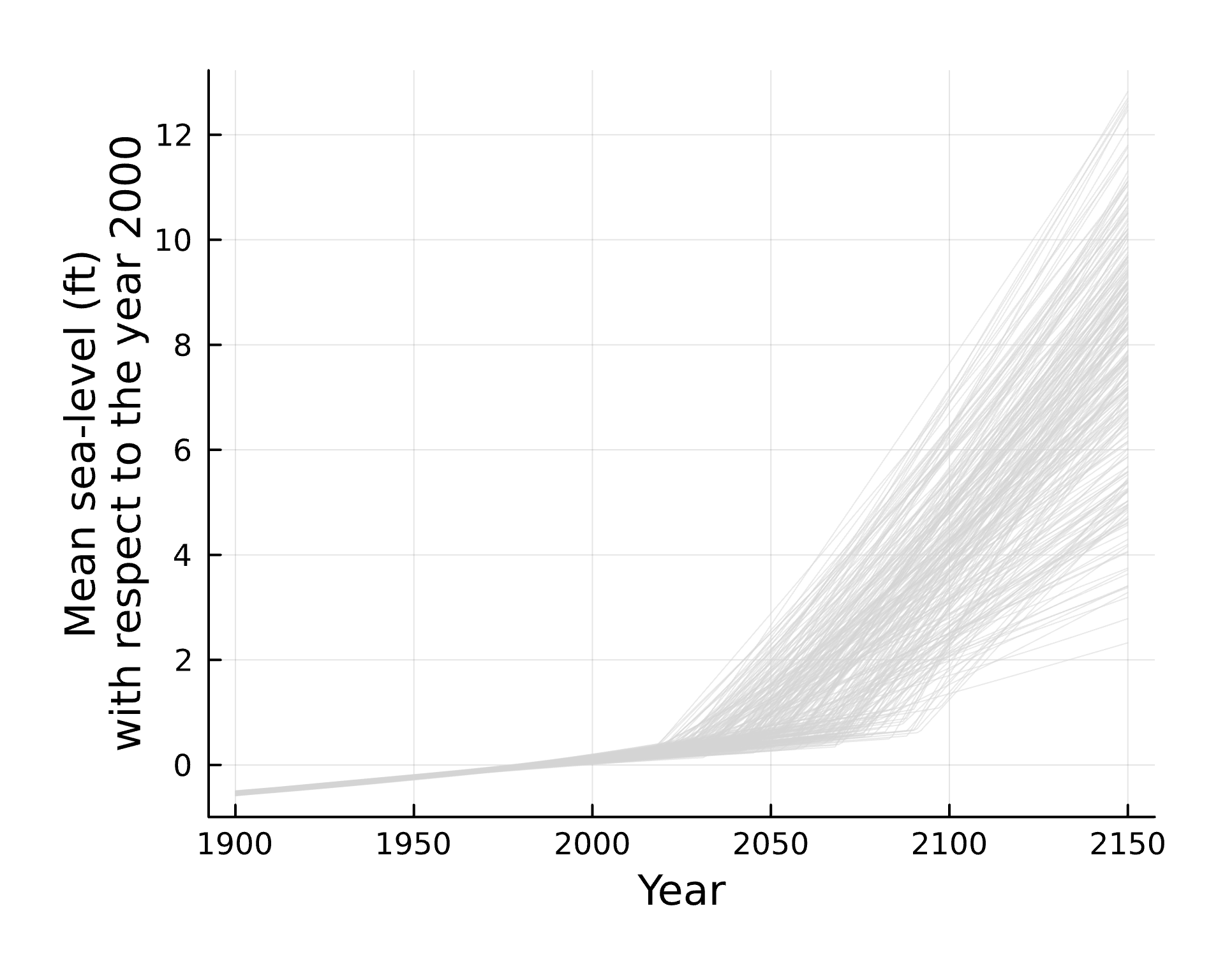
elevation\_cost(house, 10u"ft")  
let  
 elevations = 0u"ft":0.25u"ft":14u"ft"  
 costs = [elevation\_cost(house, eᵢ) for eᵢ in elevations]  
 scatter(  
 elevations,  
 costs ./ 1\_000;  
 xlabel="Elevation",  
 ylabel="Cost (Thousand USD)",  
 label="$(house.description)\n($(house.source))",  
 legend=:bottomright,  
 size=(800, 400),  
 yformatter=:plain, # prevents scientific notation  
 )  
end



## Sea-level rise

slr\_scenarios = let  
 df = CSV.read("data/slr\_oddo.csv", DataFrame)  
 [Oddo17SLR(a, b, c, tstar, cstar) for (a, b, c, tstar, cstar) in eachrow(df)]  
end  
println("There are $(length(slr\_scenarios)) parameter sets")  
  
let  
 years = 1900:2150  
 p = plot(;  
 xlabel="Year",  
 ylabel="Mean sea-level (ft)\nwith respect to the year 2000",  
 label="Oddo et al. (2017)",  
 legend=false  
 )  
 for s in rand(slr\_scenarios, 250)  
 plot!(p, years, s.(years); color=:lightgrey, alpha=0.5, linewidth=0.5)  
 end  
 p  
end

There are 34895 parameter sets



## Storm surge

function draw\_surge\_distribution()  
 μ = rand(Normal(5, 1))  
 σ = rand(Exponential(1.5))  
 ξ = rand(Normal(0.1, 0.05))  
 GeneralizedExtremeValue(μ, σ, ξ)  
end  
  
[draw\_surge\_distribution() for \_ in 1:1000]

1000-element Vector{GeneralizedExtremeValue{Float64}}:  
 GeneralizedExtremeValue{Float64}(μ=3.970046393582958, σ=2.2064662669989707, ξ=0.05414452850117545)  
 GeneralizedExtremeValue{Float64}(μ=6.689950479691339, σ=1.768925374215486, ξ=0.03715284706083437)  
 GeneralizedExtremeValue{Float64}(μ=5.106066660238726, σ=0.4518658652641706, ξ=0.09797323024349153)  
 GeneralizedExtremeValue{Float64}(μ=4.692650373686673, σ=12.022485890089278, ξ=0.0625427807571446)  
 GeneralizedExtremeValue{Float64}(μ=5.646422806076911, σ=0.022361302137148008, ξ=0.16836108630613245)  
 GeneralizedExtremeValue{Float64}(μ=5.849883457318091, σ=1.0309045500825682, ξ=0.14300532811920952)  
 GeneralizedExtremeValue{Float64}(μ=4.885417553347066, σ=1.239993647251837, ξ=0.15035163136450852)  
 GeneralizedExtremeValue{Float64}(μ=6.528959401546641, σ=0.9110392699570631, ξ=0.18420880122390218)  
 GeneralizedExtremeValue{Float64}(μ=5.897869341194189, σ=0.12776505356561324, ξ=0.08735783883046672)  
 GeneralizedExtremeValue{Float64}(μ=5.511191044587064, σ=0.7663849935065847, ξ=0.03632264180695344)  
 GeneralizedExtremeValue{Float64}(μ=5.360831238822873, σ=0.542528193712743, ξ=0.11594677402615453)  
 GeneralizedExtremeValue{Float64}(μ=3.7532094550051456, σ=2.9325920481878986, ξ=0.12182813966965725)  
 GeneralizedExtremeValue{Float64}(μ=5.445477833040628, σ=0.4193757820922991, ξ=0.13309045722562723)  
 ⋮  
 GeneralizedExtremeValue{Float64}(μ=4.3523573123717405, σ=0.6305979850104442, ξ=0.06984909824269585)  
 GeneralizedExtremeValue{Float64}(μ=6.88473969133136, σ=0.18506537857103242, ξ=0.14745980906610143)  
 GeneralizedExtremeValue{Float64}(μ=4.2196700274289825, σ=4.134532144117822, ξ=0.13782513377231168)  
 GeneralizedExtremeValue{Float64}(μ=4.988904820989116, σ=0.09202673028697149, ξ=0.15201144776156353)  
 GeneralizedExtremeValue{Float64}(μ=5.4954265062662015, σ=0.7760186000207325, ξ=0.09428649780673887)  
 GeneralizedExtremeValue{Float64}(μ=5.668630962642458, σ=1.4259776542747051, ξ=0.00018840727226243636)  
 GeneralizedExtremeValue{Float64}(μ=5.442085669844899, σ=0.06510715867337696, ξ=0.12405280515555056)  
 GeneralizedExtremeValue{Float64}(μ=6.049741531921293, σ=1.5819307646986118, ξ=0.14655452050957998)  
 GeneralizedExtremeValue{Float64}(μ=4.268897154546579, σ=0.5983253773951467, ξ=0.13298690867487314)  
 GeneralizedExtremeValue{Float64}(μ=5.994565093692193, σ=0.18817152622515304, ξ=0.05761624816223003)  
 GeneralizedExtremeValue{Float64}(μ=6.170240253523316, σ=1.718881544505454, ξ=0.08999207143282917)  
 GeneralizedExtremeValue{Float64}(μ=5.551331928434862, σ=0.6743921255963987, ξ=0.17235876692959165)

## Discount rate

function draw\_discount\_rate()  
 return rand(Normal(0.04, 0.02))  
end

draw\_discount\_rate (generic function with 1 method)

## SOW

p = ModelParams(  
 house=house,  
 years=2024:2083  
)  
  
sow = SOW(  
 rand(slr\_scenarios),  
 draw\_surge\_distribution(),  
 draw\_discount\_rate()  
)  
  
a = Action(3.0u"ft")  
  
res = run\_sim(a, sow, p)

-264266.6402943388

## large ensamble

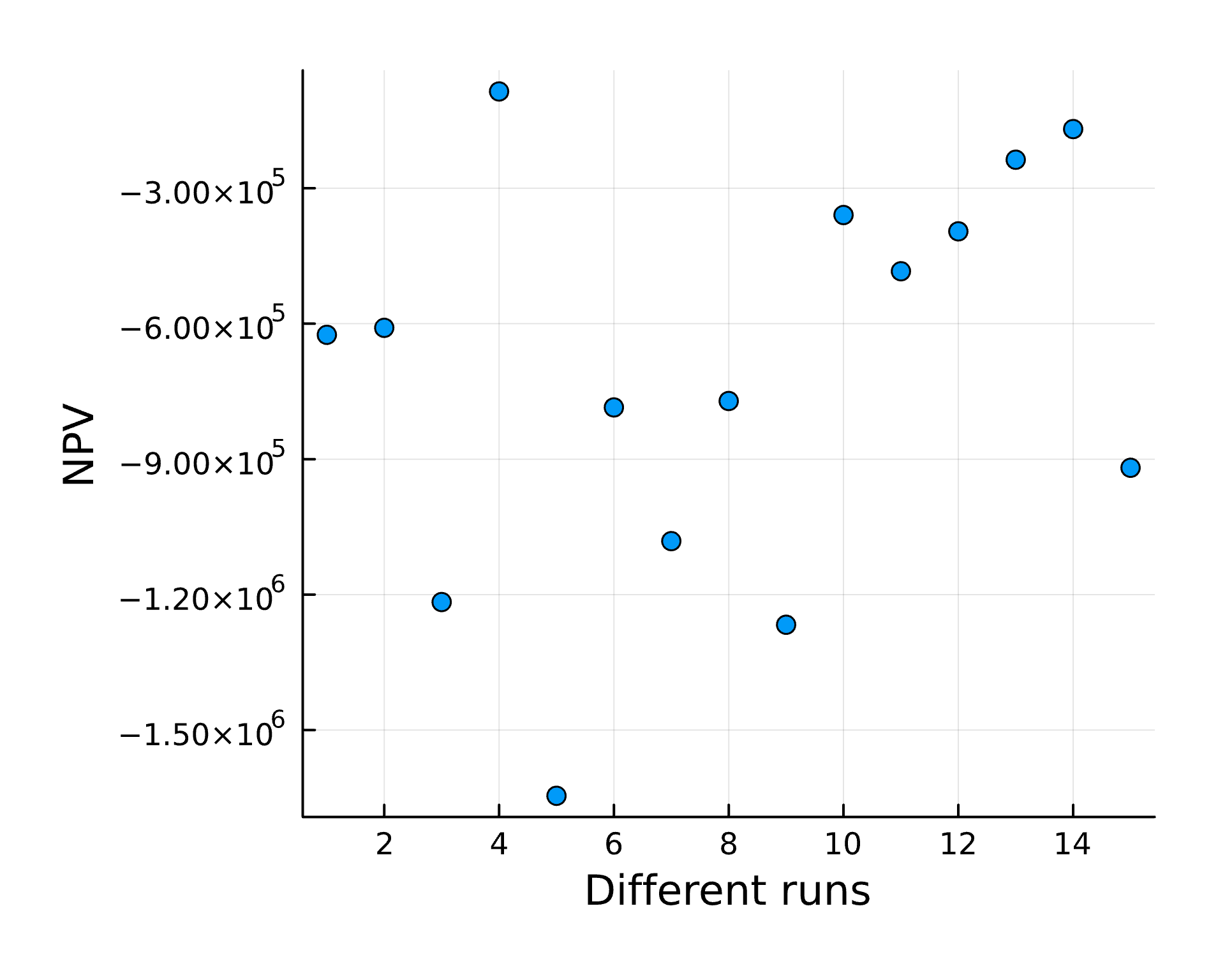
When actions = 3 ft：

sows = [SOW(rand(slr\_scenarios), draw\_surge\_distribution(), draw\_discount\_rate()) for \_ in 1:15]   
actions = [Action(3.0u"ft") for \_ in 1:15]   
results = [run\_sim(a, s, p) for (a, s) in zip(actions, sows)]

15-element Vector{Float64}:  
 -624626.5845157949  
 -609166.0847281386  
 -1.2164802255585766e6  
 -85641.59172842067  
 -1.6455912239428712e6  
 -785417.2043362445  
 -1.0815880300690937e6  
 -771173.9953882194  
 -1.266763233939359e6  
 -359282.7012035918  
 -483866.3048332154  
 -395445.2025679443  
 -236801.7976909419  
 -168942.80516919348  
 -919123.0657487903

df = DataFrame(  
 npv=results,  
 Δh\_ft=[a.Δh\_ft for a in actions],  
 slr\_a=[s.slr.a for s in sows],  
 slr\_b=[s.slr.b for s in sows],  
 slr\_c=[s.slr.c for s in sows],  
 slr\_tstar=[s.slr.tstar for s in sows],  
 slr\_cstar=[s.slr.cstar for s in sows],  
 surge\_μ=[s.surge\_dist.μ for s in sows],  
 surge\_σ=[s.surge\_dist.σ for s in sows],  
 surge\_ξ=[s.surge\_dist.ξ for s in sows],  
 discount\_rate=[s.discount\_rate for s in sows],  
)  
df

npv\_data = [df[i, :npv] for i in 1:nrow(df)]  
scatter(npv\_data, xlabel="Different runs", ylabel="NPV", legend=false)



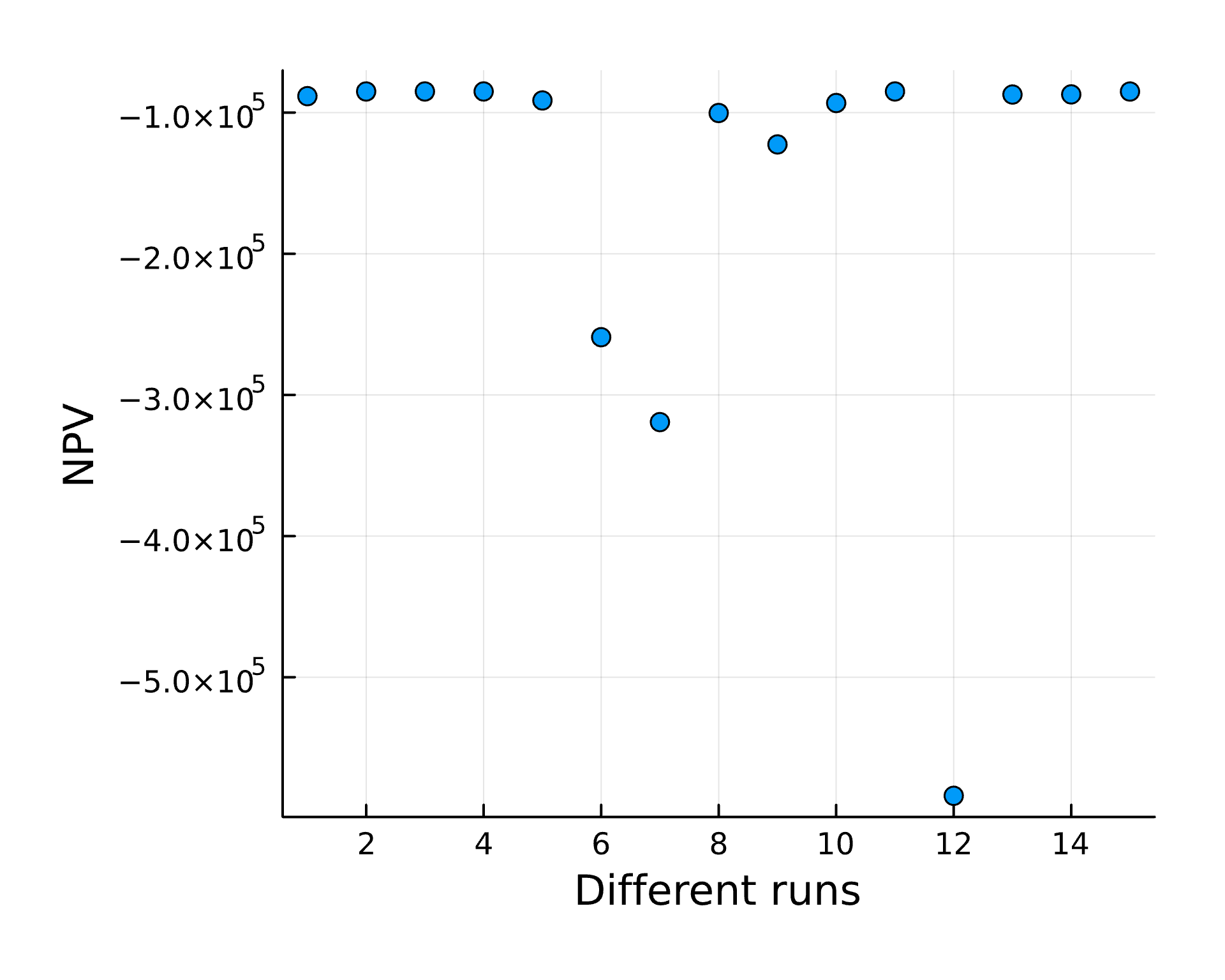
When actions = 8 ft：

sows = [SOW(rand(slr\_scenarios), draw\_surge\_distribution(), draw\_discount\_rate()) for \_ in 1:15]  
actions = [Action(8.0u"ft") for \_ in 1:15] # these are all the same  
results = [run\_sim(a, s, p) for (a, s) in zip(actions, sows)]

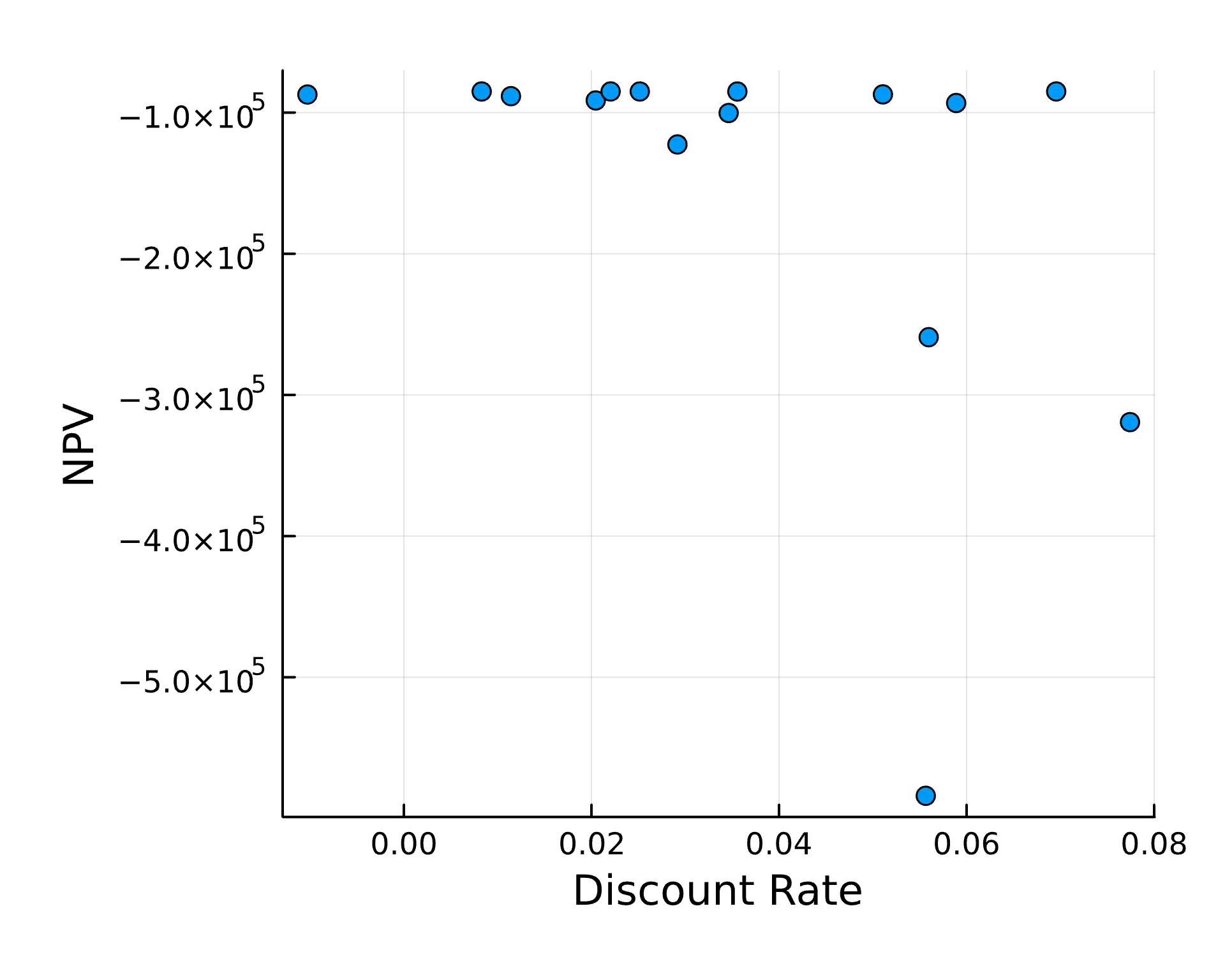
15-element Vector{Float64}:  
 -88355.81144666774  
 -85056.27282121949  
 -85030.71428571429  
 -85030.71428571429  
 -91404.35340552163  
 -259110.50248518234  
 -319267.9031800892  
 -100287.59488076139  
 -122552.05769879225  
 -93216.35933211778  
 -85030.71428571429  
 -584001.7302773366  
 -87196.8722059013  
 -87106.1826100337  
 -85076.88005584334

df = DataFrame(  
 npv=results,  
 Δh\_ft=[a.Δh\_ft for a in actions],  
 slr\_a=[s.slr.a for s in sows],  
 slr\_b=[s.slr.b for s in sows],  
 slr\_c=[s.slr.c for s in sows],  
 slr\_tstar=[s.slr.tstar for s in sows],  
 slr\_cstar=[s.slr.cstar for s in sows],  
 surge\_μ=[s.surge\_dist.μ for s in sows],  
 surge\_σ=[s.surge\_dist.σ for s in sows],  
 surge\_ξ=[s.surge\_dist.ξ for s in sows],  
 discount\_rate=[s.discount\_rate for s in sows],  
)  
df

npv\_data = [df[i, :npv] for i in 1:nrow(df)]  
scatter(npv\_data, xlabel="Different runs", ylabel="NPV", legend=false)



discounts = [df[i, :discount\_rate] for i in 1:nrow(df)]  
scatter(discounts, npv\_data, xlabel="Discount Rate", ylabel="NPV", legend=false)



## results and analysis

The 8ft lift generally achieves a lower NPV than 3ft, which can be intuitively reflected in the figure. At the same time, the discount rate also has a great impact on NPV. We might calculate the correlation between these variables. Although The house elevation height seems to be the most important variable for NPV. The model, does not take into account the future appreciation or depreciation of the house, nor does it take into account changes in the surrounding communities. Also, the future change of climate didn’t take into consideration.  
Based on the existing model, the number of simulations currently used is sufficient. If you consider adding more variables or performing more complex calculations in the future, you can consider increasing the number of simulations.