Final Project Report

Kyle Olcott (kto1) and Patricia Hashimoto (pch3)

Tue., Apr. 30

Table of contents

# 1. Introduction

## 1.1 Problem Statement

Clearly define the problem statement that your chosen feature aims to address. Explain the significance of this problem in the context of climate risk management.

This project will improve the model’s ability to estimate the price to rebuild a home after flooding by accounting for more complex aspects of discount rate and price changes based on scarcity after a flood.

To obtain a more accurate understanding of cost over time, the discount rate will be broken down to have separate discount rates for the net present value calculation, the housing market, and the cost of labor and materials. Including these values will capture the way that different aspects of cost change over time in different ways; for example, perhaps inflation is higher for labor or materials. These factors will affect estimates of the cost to rebuild a house entirely from scratch. The discount rates, which may have different distributions or be constant, will be decided based on the findings of the literature review.

Additionally, the model will be modified to capture how scarcity caused by a flood affects construction cost. Including storm surge rating as a way to estimate the severity of the flood’s impact of scarcity of materials and labor will enable the model to estimate the cost of rebuilding a house from scratch given scarcity.

The components added to the model are illustrated in Figure 1.

## 1.2 Selected Feature

Describe the feature you have selected to add to the existing decision-support tool. Discuss how this feature relates to the problem statement and its potential to improve climate risk assessment.

The two features we’ve deicded to add are additional discount rates in order to model price increase for construction and labor costs as well as a factor that multiples the cost to rebuild the house after a flood in order to model scarcity after a flood event. This will improve the climate risk assessment as it more accurately shows that magnitude of the costs associated with rebuilding the house. Under our previous model without these changes, we are likely to come to an optimal solution that is lower than what the elevation should actually be in the real world. The increased cost to rebuild in the future makes solutions using a higher elevation in year 1 more robust.

# 2. Literature Review

Provide a brief overview of the theoretical background related to your chosen feature. Cite at least two relevant journal articles to support your approach (see [Quarto docs](https://quarto.org/docs/authoring/footnotes-and-citations.html) for help with citations). Explain how these articles contribute to the justification of your selected feature.

# 3. Methodology

## 3.1 Implementation

You should make your modifications in either the HouseElevation or ParkingGarage module. Detail the steps taken to implement the selected feature and integrate it into the decision-support tool. Include code snippets and explanations where necessary to clarify the implementation process.

The scarcity/demand surge feature will be added by assuming a linear relationship between the damage of the house and the demand surge magnitude. The highest value for a demand surge was found to be around 20%. Thus, if a house was damages by 50%, the demand surge will be 50% of 20%, which is 10%. This 10% will then be multiplied onto the damage that the house suffered in order to model how much more it would cost to rebuild under scarcity. For the example before, this would be 50% \* (1 + 10%) = 55% damage.

## 3.2 Validation

As we have seen in labs, mistakes are inevitable and can lead to misleading results. To minimize the risk of errors making their way into final results, it is essential to validate the implemented feature. Describe the validation techniques used to ensure the accuracy and reliability of your implemented feature. Discuss any challenges faced during the validation process and how they were addressed.

# 4. Results

Present the results obtained from the enhanced decision-support tool. Use tables, figures, and visualizations to clearly communicate the outcomes. Provide sufficient detail to demonstrate how the implemented feature addresses the problem statement. Use the #| output: false and/or #| echo: false tags to hide code output and code cells in the final report except where showing the output (e.g.g, a plot) or the code (e.g., how you are sampling SOWs) adds value to the discussion. You may have multiple subsections of results, which you can create using ##.

# 5. Conclusions

## 5.1 Discussion

Analyze the implications of your results for climate risk management. Consider the context of the class themes and discuss how your findings contribute to the understanding of climate risk assessment. Identify any limitations of your approach and suggest potential improvements for future work.

## 5.2 Conclusions

Summarize the key findings of your project and reiterate the significance of your implemented feature in addressing the problem statement. Discuss the broader implications of your work for climate risk management and the potential for further research in this area.

# 6. References

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

We also load our local package as in lab 5.

using Revise  
using HouseElevation

# 7. States of the world

We begin by defining the variables that don’t change from one SOW to the next. We load these into the ModelParams.

house = let  
 haz\_fl\_dept = CSV.read("data/haz\_fl\_dept.csv", DataFrame) # read in the file  
 desc = "one story, Contents, fresh water, short duration"  
 row = @rsubset(haz\_fl\_dept, :Description == desc)[1, :] # select the row I want  
 area = 500u"ft^2"  
 height\_above\_gauge = 12u"ft"  
 House(row; area=area, height\_above\_gauge=height\_above\_gauge, value\_usd=250\_000)  
end  
  
p = ModelParams(; house=house, years=2024:2083)

Next we define how we will sample the states of the world.

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  
  
function draw\_surge\_distribution()  
 μ = rand(Normal(5, 1))  
 σ = rand(Exponential(1.25))  
 ξ = rand(Normal(0.1, 0.05))  
 return GeneralizedExtremeValue(μ, σ, ξ)  
end  
  
function draw\_discount\_rate()  
 return rand(Normal(0.05, 0.03))  
end  
  
function draw\_sow()  
 slr = rand(slr\_scenarios)  
 surge\_params = draw\_surge\_distribution()  
 discount = draw\_discount\_rate()  
 return SOW(slr, surge\_params, discount)  
end

Finally we can sample the SOWs

Random.seed!(421521)  
N\_SOW = 10\_000  
N\_SOW\_opt = 10 # to start  
sows = [draw\_sow() for \_ in 1:N\_SOW]  
sows\_opt = first(sows, N\_SOW\_opt)

# 8. Optimization

## 8.1 Bounds

We have a single decision variable, the height of the house above the ground. This can be any real number between 0 and 14 feet.

bounds = boxconstraints(; lb=[0.0], ub=[14.0])

BoxConstrainedSpace{Float64}([0.0], [14.0], [14.0], 1, true)

## 8.2 Objective function

We next need an objective function. Recall that we want to *maximize* NPV, but the optimization package we are using is set up to *minimize*.

function objective\_function(Δh::Vector{Float64})  
 a = Action(Δh[1])  
 npvs = [run\_sim(a, sow, p) for sow in sows\_opt]  
 return -mean(npvs)  
end

objective\_function (generic function with 1 method)

## 8.3 Running

We can throw this straight into the optimize function:

result = optimize(objective\_function, bounds)

Optimization Result  
===================  
 Iteration: 55  
 Minimum: 80168.3  
 Minimizer: [10.9924]  
 Function calls: 385  
 Total time: 2.3260 s  
 Stop reason: Due to Convergence Termination criterion.

We can view the minimum of the objective function with

minimum(result)

80168.32817561974

and the value of the decision variable that achieves that minimum with:

minimizer(result)

1-element Vector{Float64}:  
 10.992360598674209

This seems like it’s working plausibly. Let’s try now with more SOWs.

N\_SOW\_opt = 100  
sows\_opt = first(sows, N\_SOW\_opt)

100-element Vector{SOW{Float64}}:  
 SOW{Float64}(Oddo17SLR{Float64}(58.66628415, 2.709557857, 0.003600726, 2061.196955, 24.78557835), GeneralizedExtremeValue{Float64}(μ=5.1592657251467795, σ=1.786085589594579, ξ=0.04909541822954309), 0.043912364344874985)  
 SOW{Float64}(Oddo17SLR{Float64}(18.98238754, 2.076689405, 0.002459112, 2017.216294, 15.58401579), GeneralizedExtremeValue{Float64}(μ=4.801196796558416, σ=0.6923008755715017, ξ=0.23052884287490827), 0.07869168998582142)  
 SOW{Float64}(Oddo17SLR{Float64}(13.54334493, 1.576769557, -0.001892038, 2019.630551, 13.56306606), GeneralizedExtremeValue{Float64}(μ=5.840579770203203, σ=3.5558601124004285, ξ=0.08723607897000375), 0.06488106252216932)  
 SOW{Float64}(Oddo17SLR{Float64}(38.14589083, 2.623948719, 0.005085519, 2016.508542, 8.134629056), GeneralizedExtremeValue{Float64}(μ=3.8315083460098016, σ=3.554939606231098, ξ=-0.0028238758770310346), 0.04051324148798102)  
 SOW{Float64}(Oddo17SLR{Float64}(40.9256403, 2.858600773, 0.008671168, 2024.544892, 18.75223102), GeneralizedExtremeValue{Float64}(μ=5.503318936597412, σ=0.6057691505230971, ξ=0.14893794145853154), 0.07318252811671196)  
 SOW{Float64}(Oddo17SLR{Float64}(36.12774556, 2.258399655, 0.002399946, 2066.78925, 19.7923147), GeneralizedExtremeValue{Float64}(μ=4.0973209937081165, σ=0.5519168910230567, ξ=0.07677316316267815), 0.022963396163057218)  
 SOW{Float64}(Oddo17SLR{Float64}(54.44686357, 2.440676913, 0.003209864, 2039.323894, 22.89705148), GeneralizedExtremeValue{Float64}(μ=4.976640818675022, σ=2.929128322094571, ξ=0.0748038905239377), 0.07234831771584435)  
 SOW{Float64}(Oddo17SLR{Float64}(42.54886307, 2.374354031, 0.003786949, 2018.717192, 9.587090849), GeneralizedExtremeValue{Float64}(μ=6.304505229243999, σ=0.6796814459767018, ξ=0.13313994855425473), 0.04084383078870752)  
 SOW{Float64}(Oddo17SLR{Float64}(5.951503134, 1.501882544, -0.002460307, 2022.058777, 10.78953073), GeneralizedExtremeValue{Float64}(μ=5.612056817161585, σ=2.81935964343091, ξ=0.026574484379920854), 0.05338896379200811)  
 SOW{Float64}(Oddo17SLR{Float64}(17.66182918, 1.54415968, -0.002055399, 2058.500619, 31.64125888), GeneralizedExtremeValue{Float64}(μ=5.974257340053487, σ=1.3745730137960008, ξ=-0.004411514151707152), 0.02982395279108502)  
 SOW{Float64}(Oddo17SLR{Float64}(39.36100099, 2.426238607, 0.00403237, 2027.055898, 10.85150744), GeneralizedExtremeValue{Float64}(μ=5.706722362384169, σ=0.4475118422886198, ξ=0.08349999654312897), 0.025689250596865044)  
 SOW{Float64}(Oddo17SLR{Float64}(51.98651863, 2.760247424, 0.00577138, 2031.033814, 27.0621288), GeneralizedExtremeValue{Float64}(μ=4.307007231689129, σ=1.9945414455727353, ξ=0.20881284794970267), 0.07615161997445703)  
 SOW{Float64}(Oddo17SLR{Float64}(31.20844945, 2.11554997, 0.001930065, 2027.473128, 21.54471462), GeneralizedExtremeValue{Float64}(μ=6.433525656652894, σ=0.28491381216716927, ξ=0.042331800536263224), 0.05989456228664741)  
 ⋮  
 SOW{Float64}(Oddo17SLR{Float64}(47.00832611, 2.647493943, 0.004804853, 2057.476657, 22.22390011), GeneralizedExtremeValue{Float64}(μ=5.852048346873611, σ=0.46933006204308647, ξ=0.08830977838825098), 0.020802045320114662)  
 SOW{Float64}(Oddo17SLR{Float64}(36.73834466, 2.205458479, 0.002122161, 2034.741468, 20.80094387), GeneralizedExtremeValue{Float64}(μ=4.936344528332246, σ=0.29964843733494845, ξ=0.061382696771434375), 0.05374843832753328)  
 SOW{Float64}(Oddo17SLR{Float64}(44.05127787, 2.338748756, 0.002183976, 2057.606653, 15.06497881), GeneralizedExtremeValue{Float64}(μ=3.7820151093398553, σ=0.1403473868398733, ξ=0.06665005213494601), 0.012138111249493516)  
 SOW{Float64}(Oddo17SLR{Float64}(28.75735666, 2.444711243, 0.004489676, 2073.837867, 23.35242685), GeneralizedExtremeValue{Float64}(μ=5.591290057672752, σ=0.022244130648271264, ξ=0.055152538774899895), 0.03510151584563126)  
 SOW{Float64}(Oddo17SLR{Float64}(35.84037864, 2.395483919, 0.003342545, 2020.868925, 20.06540289), GeneralizedExtremeValue{Float64}(μ=5.484879083835489, σ=3.0099405201654132, ξ=0.08636514474849032), 0.04019400052648379)  
 SOW{Float64}(Oddo17SLR{Float64}(46.47782198, 2.652879425, 0.004551166, 2083.94697, 31.71562617), GeneralizedExtremeValue{Float64}(μ=4.098889452454249, σ=0.11111117305426037, ξ=0.022236381784040596), 0.06672330608824631)  
 SOW{Float64}(Oddo17SLR{Float64}(47.06379093, 2.686402018, 0.005309616, 2030.347586, 20.32702337), GeneralizedExtremeValue{Float64}(μ=5.6632834428719585, σ=0.30204647767371096, ξ=0.14034773727908867), 0.02816427502575243)  
 SOW{Float64}(Oddo17SLR{Float64}(20.60575154, 1.814858117, 0.001384537, 2096.479489, 33.51825669), GeneralizedExtremeValue{Float64}(μ=5.644533129673197, σ=0.7939976999990541, ξ=0.09857385622856223), 0.04469711813736867)  
 SOW{Float64}(Oddo17SLR{Float64}(31.47268467, 2.161480684, 0.001951459, 2041.194249, 17.51020409), GeneralizedExtremeValue{Float64}(μ=4.243520195003441, σ=3.900339636627553, ξ=0.15984690402869664), 0.0977582976083996)  
 SOW{Float64}(Oddo17SLR{Float64}(33.88063067, 2.470390408, 0.003485804, 2063.105751, 24.99438311), GeneralizedExtremeValue{Float64}(μ=4.740965199481158, σ=0.49909292679914224, ξ=0.0061093216506191705), 0.09350454479095838)  
 SOW{Float64}(Oddo17SLR{Float64}(14.57458321, 1.356668977, -0.004109447, 2041.776196, 22.79443674), GeneralizedExtremeValue{Float64}(μ=5.046108868294086, σ=0.22009332585830527, ξ=0.13780160468736796), 0.04567843528244755)  
 SOW{Float64}(Oddo17SLR{Float64}(52.97055604, 2.796456852, 0.006349309, 2096.336577, 22.77508265), GeneralizedExtremeValue{Float64}(μ=5.31457800663226, σ=4.7149979394504316, ξ=0.055416193099212314), 0.05945120410893797)

Since I’m using more SOWs here, I’ll also increase the time limit for the optimization to three minutes.

options = Options(; time\_limit=180.0, f\_tol\_rel=10.0)

Options  
=======  
 rng: TaskLocalRNG()  
 seed: 423339202  
 x\_tol: 1.0e-8  
 f\_tol: 1.0e-12  
 g\_tol: 0.0  
 h\_tol: 0.0  
 debug: false  
 verbose: false  
 f\_tol\_rel: 10.0  
 time\_limit: 180.0  
 iterations: 0  
 f\_calls\_limit: 0.0  
 store\_convergence: false  
 parallel\_evaluation: false

To use options, we have to choose an algorithm. See list of algorithms [here](https://jmejia8.github.io/Metaheuristics.jl/stable/algorithms/). The ECA algorithm is suggested as a default, so we’ll use that.

algorithm = ECA(; options=options)

Algorithm Parameters  
====================  
 ECA(η\_max=2.0, K=7, N=0, N\_init=0, p\_exploit=0.95, p\_bin=0.02, ε=0.0, adaptive=false, resize\_population=false)  
  
Optimization Result  
===================  
 Empty status.

Before we run the optimization, let’s set a random seed. This will make our results more reproducible. We can then vary the seed to see how sensitive our results are to the random seed.

Random.seed!(421521)  
result = optimize(objective\_function, bounds, algorithm)

Optimization Result  
===================  
 Iteration: 72  
 Minimum: 80637.5  
 Minimizer: [10.1205]  
 Function calls: 504  
 Total time: 23.7970 s  
 Stop reason: Due to Convergence Termination criterion.

We can view our result with

display(minimum(result))  
display(minimizer(result))

80637.52834867896

1-element Vector{Float64}:  
 10.120502588289005

# 9. Validation

In this case, we don’t really *need* optimization – we can use brute force. We can compare by plotting the objective function for a range of elevations (from 0 to 14 ft) using all SOWs.

elevations\_try = 0:0.5:14  
actions\_try = Action.(elevations\_try)  
  
N\_more = 500  
npvs\_opt = [mean([run\_sim(a, sow, p) for sow in sows\_opt]) for a in actions\_try]  
npvs\_moore = [  
 mean([run\_sim(a, sow, p) for sow in first(sows, N\_more)]) for a in actions\_try  
]

and plot

Key insights:

1. Our optimization appears to be working well, and maximizes the blue curve as it should
2. There is a substantial difference between the blue and red lines, indicating that using different SOWs (from the same distribution!) can make a big difference
3. Going from zero (don’t elevate) to a small elevation is always bad, as you gain little flood protection but have to pay the fixed costs of elevation
4. The optimal elevation is highly sensitive to assumptions about the SOWs