Lab 5: Sea-Level Rise

<Leanh Nguyen (Ln14)>

Fri., Feb. 16

Table of contents

# 1. Setup

## 1.1 The usual

As always:

1. Clone the lab repository to your computer
2. Open the lab repository in VS Code
3. Open the Julia REPL and activate, then instantiate, the lab environment
4. Make sure you can render: quarto render template.qmd in the terminal.
   * If you run into issues, try running ] build IJulia in the Julia REPL (] enters the package manager).
   * If you still have issues, try opening up blankfile.py. That should trigger VS Code to give you the option to install the Python extension, which you should do. Then you should be able to open a menu in the bottom right of your screen to select which Python installation you want VS Code to use.

## 1.2 Load packages

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

## 1.3 Local package

using Revise  
using HouseElevation

# 2. Building the model

## 2.1 House

|  |  |
| --- | --- |
|  | We will consider a single house, and will ignore uncertainty in the depth-damage function or other house parameters |

* Neglect uncertainty in depth-damage function
* Consider a single building
* We’re going to put all relevant information into a House object:
  + Depth-damage function
  + Area
  + Cost (USD)
  + Elevation relative to gauge
  + Metadata

We can create a House as follows – note that we’re using a let...end block to create the House object.

house = let  
 haz\_fl\_dept = CSV.read("data/haz\_fl\_dept.csv", DataFrame) # read in the file  
 desc = "Gift Shop, structure"   
 row = @rsubset(haz\_fl\_dept, :Description == desc)[1, :] # select the row I want  
 area = 795u"ft^2"  
 height\_above\_gauge = 6u"ft" #can only be a whole number  
 House(  
 row;  
 area=area,  
 height\_above\_gauge=height\_above\_gauge,  
 value\_usd=221\_600,  
 )  
end

Line 5

Area was obtained from Google Earth’s measurement tools

Line 11

From Google Earth, I calculated/estimated my target structure’s area (795 sq ft). Then, I found a structure near my target structure on zillow that had a similar area (750 sq ft). Thus, I decided to use this as the house structure value. https://www.zillow.com/homedetails/101-21st-St-STE-214-Galveston-TX-77550/2132158257\_zpid/

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

Line 2

Depth-damage curve obtained from data from data/haz\_fl\_dept.csv. We were given a range of depth to consider. Then, we use the House object to calculate the damage to the house for a given flood depth. We then convert the damage to dollars by multiplying the fraction (given by our depth-damage function) by the value of the house.

Use the House object to calculate the cost of raising the house to a given elevation. Use the elevation\_cost function

elevation\_cost(house, 6u"ft")

87184.28571428571

Plot This

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

## 2.2 Sea-level rise

|  |  |
| --- | --- |
|  | We will sample many different scenarios of sea-level rise |

We’re modeling sea-level rise following the approach of Oddo et al. (2017). Essentially, we use five parameters: , , , , and . The local sea-level in year is given by equation 6 of Oddo et al. (2017):

The authors note:

In this model, the parameters , , and represent the reasonably well-characterized process of thermosteric expansion as a second-order polynomial. It also accounts for more poorly understood processes, including potential abrupt sealevel rise consistent with sudden changes in ice flow dynamics.Here, represents an increase in the rate of sea-level rise that takes place at some uncertain time, , in the future.

This is, of course, a highly simplified model. However, the parameters can be calibrated to match historical sea-level rise (i.e., throwing out any parameter values that don’t match the historical record) and use a statistical inversion method to estimate the parameters. One could also calibrate the parameters to match other, more complex, physics-based models. We’ll use Monte Carlo simulations from Oddo et al. (2017), available on [GitHub](https://github.com/pcoddo/VanDantzig/blob/master/Model_Versions/Uncertainty_SLR/SLR_Module/Rejection_Sampling/beta/output/array_beta.txt). These were actually calibrated for the Netherlands, but we’ll pretend that sea-level rise in your location matches (which – as we know – it doesn’t).

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")

Plot these scenarios to get a sense of the range of sea-level rise we might expect.

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

Key insight from this plot: uncertainty in future sea level increases over time

## 2.3 Storm surge

|  |  |
| --- | --- |
|  | We will consider parametric uncertainty in the storm surge |

The next component of the model is the storm surge (i.e., the height of the flood above mean sea-level). Model the water level *at the gauge* as the sum of the local sea-level and the storm surge. Then model the water level *at the house* as the water level at the gauge minus the elevation of the house above the gauge.

Consider parametric uncertainty in the storm surge. From lab 3, you should have a GeneralizedExtremeValue distribution for the storm surge. We can then sample parameters from a range centered on this distribution. For example, in the example for lab 3 we had GeneralizedExtremeValue(5, 1.5, 0.1). We can use this function to create a distribution for the 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 (generic function with 1 method)

Storm Surge Distribution: I’ve chosen to model the storm surge distribution using the Generalized Extreme Value (GEV) distribution. The GEV distribution is commonly used to model extreme events (e.g., storm surges) because it can capture various shapes of distributions. The parameters μ, σ, and ξ represent the location, scale, and shape of the distribution. I’ve chosen to sample these parameters randomly to introduce variability in the storm surge distribution.

We can then call this function many times to get many different distributions for the storm surge.

[draw\_surge\_distribution() for \_ in 1:1000]

|  |  |
| --- | --- |
|  | **Important**  This is NOT statistical estimation. We are not saying anything at all about whether these parameters are consistent with observations. In fact, even when parameters are uncertain, sampling around a point estimate in this manner usually produces lots of parameter values that are highly implausible. Here, we are just exploring the implications of different parameter values. Building a better model for storm surge is a great idea for your final project! |

## 2.4 Discount rate

|  |  |
| --- | --- |
|  | We will consider parametric uncertainty in the discount rate. |

The discount rate is an important economic parameter in our NPV analysis. There are elements of discounting that are perhaps not random (e.g., how much do you value the future versus the present?) while there are other elements that are very much random (what is the opportunity cost of spending money now?) Model this by treating the discount rate as a random variable, but more sophisticated analyses are possible.

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

Note that we are now defining the discount rate as a proportion (from 0 to 1) rather than a percentage (from 0 to 100).

Discount Rate: I’ve chosen to model the discount rate using a normal distribution with a mean of 0.04 (4%) and a standard deviation of 0.02 (2%). This reflects typical values for discount rates used in financial modeling. By sampling from a normal distribution, I can introduce variability in the discount rate, allowing for different discount rates to be considered in the analysis.

These modeling choices aim to capture the uncertainty and variability inherent in storm surge events and discount rates, which are important factors in decision-making processes related to risk management and financial plans.

## 2.5 Running a simulation

In the notation we’ve seen in class, we have a system model that takes in a state of the world , an action , and outputs some metric or metrics. I’ve reproduced this in our model, adding one extra piece: a ModelParams object that contains all the parameters of the model that *don’t change from one simulation to the next.*

In our model, the ModelParams are the house characteristics (area, value, and depth-damage curve) and the years we’re considering. You should consider different time horizons!

p = ModelParams(  
 house=house,  
 years=2024:2083  
)

The next step is to create an object to hold our state of the world (SOW). We can create one like this. In the next step, we’ll want to sample a large ensemble of SOWs.

sow = SOW(  
 rand(slr\_scenarios),  
 draw\_surge\_distribution(),  
 draw\_discount\_rate()  
)

Last, we need to define our action. For now, our action is very simple: we’re going to raise the house to a fixed elevation. However, in the future we might have a more complex action (e.g., when the sea level exceeds some threshold , raise the house by some fixed amount , which has two parameters). We define our action as follows:

a = Action(10.0u"ft")

Finally, we have a function to run the simulation. This function takes in the model parameters, the state of the world, and the action, and returns the NPV of the action. Please have a look at [run\_sim.jl](HouseElevation/src/run_sim.jl) to see how this is implemented!

res = run\_sim(a, sow, p)

-95291.32921864453

# 3. Exploratory modeling

Now that you’ve figured out how this model works, it’s your turn to conduct some exploratory modeling. In [template.qmd](./template.qmd), I’ve provided only the code required to load packages.

## 3.1 Apply the model to your site

1. Build your own house object, based on the house you’ve been using (or you can switch if you’d like)
   1. Briefly explain where you got the area, value, and depth-damage curve from
   2. Plot the depth-damage curve
   3. Plot the cost of raising the house to different elevations from 0 to 14 ft
2. Read in the sea-level rise data
3. Modify my code to create a function to draw samples of storm surge and the discount rate. Explain your modeling choices!
4. Define an illustrative action, SOW, and model parameters, and run a simulation.

## 3.2 Large ensemble

Now that you’ve got the model working for your site, you should run a large ensemble of simulations (explain how you interpret “large”).

1. Sample many SOWs (see below)
2. Sample a range of actions. You can do this randomly, or you can look at just a couple of actions (e.g., 0, 3, 6, 9, 12 ft) – explain your choice.
3. Run the simulations for each SOW and action. You can use a for loop for this.
4. Create a DataFrame of your key inputs and results (see below)

Here’s how you can create a few SOWs and actions and run the simulations for each:

num\_sows = 10 # Number of SOWs to sample <1>  
sows = [SOW(rand(slr\_scenarios), draw\_surge\_distribution(), draw\_discount\_rate()) for \_ in 1:num\_sows] # for 200 SOWs <2>  
actions = [Action(i\*u"ft") for i in 0:3:12]  
results = [(sow, action, run\_sim(action, sow, p)) for sow in sows, action in actions]

Line 3

To sample a range of actions, I chose to select a few specific actions of interest, generating a range of actions from 0 to 12 feet with a step size of 3 feet:

Line 4

Once I have my SOWs and actions, I run simulations for each combination of SOW and action using a nested loop.

10×5 Matrix{Tuple{SOW{Float64}, Action{Int64}, Float64}}:  
 (SOW{Float64}(Oddo17SLR{Float64}(47.5576, 2.62864, 0.00443075, 2067.69, 26.0016), GeneralizedExtremeValue{Float64}(μ=4.7644, σ=1.20913, ξ=0.142451), 0.0204246), Action{Int64}(0), -2.21924e5) … (SOW{Float64}(Oddo17SLR{Float64}(47.5576, 2.62864, 0.00443075, 2067.69, 26.0016), GeneralizedExtremeValue{Float64}(μ=4.7644, σ=1.20913, ξ=0.142451), 0.0204246), Action{Int64}(12), -1.04636e5)  
 (SOW{Float64}(Oddo17SLR{Float64}(31.0652, 2.52287, 0.00585508, 2074.65, 14.0133), GeneralizedExtremeValue{Float64}(μ=3.68272, σ=0.669684, ξ=0.0959033), 0.0211398), Action{Int64}(0), -26758.1) (SOW{Float64}(Oddo17SLR{Float64}(31.0652, 2.52287, 0.00585508, 2074.65, 14.0133), GeneralizedExtremeValue{Float64}(μ=3.68272, σ=0.669684, ξ=0.0959033), 0.0211398), Action{Int64}(12), -1.03255e5)  
 (SOW{Float64}(Oddo17SLR{Float64}(-6.68213, 1.23905, -0.00275502, 2035.32, 22.2437), GeneralizedExtremeValue{Float64}(μ=4.584, σ=0.349579, ξ=0.0574241), 0.0485025), Action{Int64}(0), -46747.5) (SOW{Float64}(Oddo17SLR{Float64}(-6.68213, 1.23905, -0.00275502, 2035.32, 22.2437), GeneralizedExtremeValue{Float64}(μ=4.584, σ=0.349579, ξ=0.0574241), 0.0485025), Action{Int64}(12), -1.03226e5)  
 (SOW{Float64}(Oddo17SLR{Float64}(29.54, 2.1311, 0.00210334, 2053.23, 14.3959), GeneralizedExtremeValue{Float64}(μ=5.77801, σ=3.18924, ξ=0.0744897), 0.0502922), Action{Int64}(0), -3.78147e5) (SOW{Float64}(Oddo17SLR{Float64}(29.54, 2.1311, 0.00210334, 2053.23, 14.3959), GeneralizedExtremeValue{Float64}(μ=5.77801, σ=3.18924, ξ=0.0744897), 0.0502922), Action{Int64}(12), -125941.0)  
 (SOW{Float64}(Oddo17SLR{Float64}(33.3178, 2.04505, 0.00095543, 2066.96, 24.3176), GeneralizedExtremeValue{Float64}(μ=5.84295, σ=3.20424, ξ=0.0596975), 0.0636169), Action{Int64}(0), -2.98288e5) (SOW{Float64}(Oddo17SLR{Float64}(33.3178, 2.04505, 0.00095543, 2066.96, 24.3176), GeneralizedExtremeValue{Float64}(μ=5.84295, σ=3.20424, ξ=0.0596975), 0.0636169), Action{Int64}(12), -1.1918e5)  
 (SOW{Float64}(Oddo17SLR{Float64}(14.8342, 1.94493, 0.00229614, 2016.23, 22.3789), GeneralizedExtremeValue{Float64}(μ=4.21889, σ=0.800341, ξ=0.0453125), 0.000792102), Action{Int64}(0), -7.65503e5) … (SOW{Float64}(Oddo17SLR{Float64}(14.8342, 1.94493, 0.00229614, 2016.23, 22.3789), GeneralizedExtremeValue{Float64}(μ=4.21889, σ=0.800341, ξ=0.0453125), 0.000792102), Action{Int64}(12), -1.03247e5)  
 (SOW{Float64}(Oddo17SLR{Float64}(43.569, 2.56893, 0.00438087, 2052.77, 34.048), GeneralizedExtremeValue{Float64}(μ=3.3844, σ=2.15494, ξ=0.149088), 0.0375845), Action{Int64}(0), -2.03095e5) (SOW{Float64}(Oddo17SLR{Float64}(43.569, 2.56893, 0.00438087, 2052.77, 34.048), GeneralizedExtremeValue{Float64}(μ=3.3844, σ=2.15494, ξ=0.149088), 0.0375845), Action{Int64}(12), -1.12666e5)  
 (SOW{Float64}(Oddo17SLR{Float64}(19.1906, 1.50992, -0.00264614, 2055.24, 17.3384), GeneralizedExtremeValue{Float64}(μ=5.1971, σ=0.809722, ξ=0.151448), 0.0353417), Action{Int64}(0), -122763.0) (SOW{Float64}(Oddo17SLR{Float64}(19.1906, 1.50992, -0.00264614, 2055.24, 17.3384), GeneralizedExtremeValue{Float64}(μ=5.1971, σ=0.809722, ξ=0.151448), 0.0353417), Action{Int64}(12), -1.03437e5)  
 (SOW{Float64}(Oddo17SLR{Float64}(16.9417, 1.62079, -0.00130172, 2075.01, 31.8877), GeneralizedExtremeValue{Float64}(μ=3.68854, σ=1.6518, ξ=0.195318), 0.0565413), Action{Int64}(0), -93415.6) (SOW{Float64}(Oddo17SLR{Float64}(16.9417, 1.62079, -0.00130172, 2075.01, 31.8877), GeneralizedExtremeValue{Float64}(μ=3.68854, σ=1.6518, ξ=0.195318), 0.0565413), Action{Int64}(12), -1.07647e5)  
 (SOW{Float64}(Oddo17SLR{Float64}(21.8272, 1.91946, 0.000533337, 2044.27, 16.5012), GeneralizedExtremeValue{Float64}(μ=4.21656, σ=0.914502, ξ=0.0871205), 0.0378158), Action{Int64}(0), -73262.5) (SOW{Float64}(Oddo17SLR{Float64}(21.8272, 1.91946, 0.000533337, 2044.27, 16.5012), GeneralizedExtremeValue{Float64}(μ=4.21656, σ=0.914502, ξ=0.0871205), 0.0378158), Action{Int64}(12), -1.03259e5)

10 SOWs - 4.35 seconds 100 SOWs - 32.97 seconds 200 SOWs - 1 minute 34 seconds 1000 SOWs - > 8 minutes

Here’s how you can create a dataframe of your results. Each row corresponds to one simulation, and the columns are the inputs and outputs of the simulation.

df = DataFrame(  
 npv=vec([res[3] for res in results]), # Extract NPV results for each simulation <2> <3> <4>  
 Δh\_ft=vec([res[2].Δh\_ft for res in results]), # Extract action height for each simulation  
 slr\_a=vec([res[1].slr.a for res in results]), # Extract SLR parameter 'a' for each SOW  
 slr\_b=vec([res[1].slr.b for res in results]),  
 slr\_c=vec([res[1].slr.c for res in results]),  
 slr\_tstar=vec([res[1].slr.tstar for res in results]),  
 slr\_cstar=vec([res[1].slr.cstar for res in results]),  
 surge\_μ=vec([res[1].surge\_dist.μ for res in results]),  
 surge\_σ=vec([res[1].surge\_dist.σ for res in results]),  
 surge\_ξ=vec([res[1].surge\_dist.ξ for res in results]),  
 discount\_rate=vec([res[1].discount\_rate for res in results]),  
)  
show(df, allrows=true) # to see all table but uncomment if large SOW (will crash the program)

Line 1

Create a DataFrame of Key Inputs and Results: store the inputs and outputs in a DataFrame facilitates analysis and visualization of the results, allowing for better decision-making.

50×11 DataFrame  
 Row │ npv Δh\_ft slr\_a slr\_b slr\_c slr\_tstar slr ⋯  
 │ Float64 Int64 Float64 Float64 Float64 Float64 Flo ⋯  
─────┼──────────────────────────────────────────────────────────────────────────  
 1 │ -2.21924e5 0 47.5576 2.62864 0.00443075 2067.69 2 ⋯  
 2 │ -26758.1 0 31.0652 2.52287 0.00585508 2074.65 1  
 3 │ -46747.5 0 -6.68213 1.23905 -0.00275502 2035.32 2  
 4 │ -3.78147e5 0 29.54 2.1311 0.00210334 2053.23 1  
 5 │ -2.98288e5 0 33.3178 2.04505 0.00095543 2066.96 2 ⋯  
 6 │ -7.65503e5 0 14.8342 1.94493 0.00229614 2016.23 2  
 7 │ -2.03095e5 0 43.569 2.56893 0.00438087 2052.77 3  
 8 │ -122763.0 0 19.1906 1.50992 -0.00264614 2055.24 1  
 9 │ -93415.6 0 16.9417 1.62079 -0.00130172 2075.01 3 ⋯  
 10 │ -73262.5 0 21.8272 1.91946 0.000533337 2044.27 1  
 11 │ -1.29355e5 3 47.5576 2.62864 0.00443075 2067.69 2  
 12 │ -87137.8 3 31.0652 2.52287 0.00585508 2074.65 1  
 13 │ -85970.9 3 -6.68213 1.23905 -0.00275502 2035.32 2 ⋯  
 14 │ -2.73825e5 3 29.54 2.1311 0.00210334 2053.23 1  
 15 │ -2.31091e5 3 33.3178 2.04505 0.00095543 2066.96 2  
 16 │ -2.33742e5 3 14.8342 1.94493 0.00229614 2016.23 2  
 17 │ -1.69397e5 3 43.569 2.56893 0.00438087 2052.77 3 ⋯  
 18 │ -97717.7 3 19.1906 1.50992 -0.00264614 2055.24 1  
 19 │ -1.21211e5 3 16.9417 1.62079 -0.00130172 2075.01 3  
 20 │ -92238.0 3 21.8272 1.91946 0.000533337 2044.27 1  
 21 │ -98105.2 6 47.5576 2.62864 0.00443075 2067.69 2 ⋯  
 22 │ -87350.2 6 31.0652 2.52287 0.00585508 2074.65 1  
 23 │ -87185.5 6 -6.68213 1.23905 -0.00275502 2035.32 2  
 24 │ -1.7816e5 6 29.54 2.1311 0.00210334 2053.23 1  
 25 │ -1.55789e5 6 33.3178 2.04505 0.00095543 2066.96 2 ⋯  
 26 │ -94057.2 6 14.8342 1.94493 0.00229614 2016.23 2  
 27 │ -1.25028e5 6 43.569 2.56893 0.00438087 2052.77 3  
 28 │ -89301.1 6 19.1906 1.50992 -0.00264614 2055.24 1  
 29 │ -1.02911e5 6 16.9417 1.62079 -0.00130172 2075.01 3 ⋯  
 30 │ -88067.6 6 21.8272 1.91946 0.000533337 2044.27 1  
 31 │ -94913.1 9 47.5576 2.62864 0.00443075 2067.69 2  
 32 │ -91340.2 9 31.0652 2.52287 0.00585508 2074.65 1  
 33 │ -91301.2 9 -6.68213 1.23905 -0.00275502 2035.32 2 ⋯  
 34 │ -1.36078e5 9 29.54 2.1311 0.00210334 2053.23 1  
 35 │ -1.24527e5 9 33.3178 2.04505 0.00095543 2066.96 2  
 36 │ -91706.3 9 14.8342 1.94493 0.00229614 2016.23 2  
 37 │ -1.09936e5 9 43.569 2.56893 0.00438087 2052.77 3 ⋯  
 38 │ -91868.7 9 19.1906 1.50992 -0.00264614 2055.24 1  
 39 │ -99038.6 9 16.9417 1.62079 -0.00130172 2075.01 3  
 40 │ -91453.0 9 21.8272 1.91946 0.000533337 2044.27 1  
 41 │ -1.04636e5 12 47.5576 2.62864 0.00443075 2067.69 2 ⋯  
 42 │ -1.03255e5 12 31.0652 2.52287 0.00585508 2074.65 1  
 43 │ -1.03226e5 12 -6.68213 1.23905 -0.00275502 2035.32 2  
 44 │ -125941.0 12 29.54 2.1311 0.00210334 2053.23 1  
 45 │ -1.1918e5 12 33.3178 2.04505 0.00095543 2066.96 2 ⋯  
 46 │ -1.03247e5 12 14.8342 1.94493 0.00229614 2016.23 2  
 47 │ -1.12666e5 12 43.569 2.56893 0.00438087 2052.77 3  
 48 │ -1.03437e5 12 19.1906 1.50992 -0.00264614 2055.24 1  
 49 │ -1.07647e5 12 16.9417 1.62079 -0.00130172 2075.01 3 ⋯  
 50 │ -1.03259e5 12 21.8272 1.91946 0.000533337 2044.27 1  
 5 columns omitted

## 3.3 Analysis

Now, analyze your results. You can use scatterplots and other visualizations, or any other statistical analyses that you think may be helpful. Remember that the goal is to understand how different parameter values affect the success or failure of different actions.

Some questions to consider:

* When do you get the best results? - When do you get the worst results?
  + Negative NPVs indicate that the costs outweigh the benefits when discounted back to the present value.
  + In all the cases, for my particular structure and its given characteristics, all options result in a negative NPV, suggesting that elevating the house may not be a good economic decision for the owner as well as elevating the house may be a worse economic decision for the owner as well. However, the 0 ft (no elevation decision) and the 12 ft (elevate 12 feet) actions have resulted in the lowest negative NPV (~ -1). This is much better than the 6 ft and 9 ft actions, which contains multiple instances of large negative NPVs (4 or 5 times)
* What are the most important parameters?
  + Δh\_ft: This represent a change in height or elevation of a house. As the action variable is how the house is raised and impacts the NPV, Δh\_ft is one of the most important parameter for this project, impacting upfront costs and benefits from elevation.
  + slr\_a, slr\_b, slr\_c: These columns represent parameters or coefficients related to sea level rise (SLR), which was given by the dataset.
  + slr\_tstar, slr\_cstar: These could denote specific threshold or critical values associated with the sea level rise model, which can indicate significant points of change or impact in the projections.
  + surge\_μ, surge\_σ, surge\_ξ: These columns relate to the storm surge modeling, with μ (mean), σ (standard deviation), and ξ (shape parameter or the distribution tail) describing the statistical characteristics of storm surge events, suggesting that the analysis includes considerations of extreme weather events.
  + Different discount rates can significantly affect the NPV outcome, reflecting the sensitivity of the investment’s value to the cost of capital or opportunity cost.
* If you had unlimited computing power, would you run more simulations? How many?
  + If I had unlimited computing power, I would run more simulations. Specifically, I would run thousands more. As you can see when I run my SOWs, actions, and results, it takes my computer half a minute to load 100 SOWs and it exponentially increases as I use more loads. If possible, it would be good to run more simulations (e.g., 1000s) to identify and reduce uncertainties with the NPV calculations and results. However, I am limited by computing power. On the other hand, in certain situations, running more simulations may be unncecessary or inappropriate for something that is inherently/deeply uncertain (i.e., discount rates and depth-damage curves) so utilizing different models might be better.
* What are the implications of your results for decision-making?
  + This kind of analysis is crucial for long-term planning in sectors like coastal infrastructure development, where investments must consider the changing environment and associated financial risks. The use of NPV and discount rates ties the environmental projections to their economic implications, helping stakeholders make informed decisions based on both current and future expectations. However, it also shows how deep uncertainties (like depth-damage curves and discount rates) can have large impacts on the data and must be analyzed for accurate NPV estimations of decisions. However, including these uncertainties is complicated and requires the use of more simulations or the use of different models.
  + All in all, if the results generated here were given to a policymaker, they would not fully help the policymaker make informed decisions due to the complex nature of the code as well as the uncertainties deeply rooted in the code, which may make underestimation and overestimation of damages and benefits from a given action or policy.

Oddo, Perry C., Ben S. Lee, Gregory G. Garner, Vivek Srikrishnan, Patrick M. Reed, Chris E. Forest, and Klaus Keller. 2017. “Deep Uncertainties in Sea-Level Rise and Storm Surge Projections: Implications for Coastal Flood Risk Management.” *Risk Analysis* 0 (0). <https://doi.org/ghkp82>.