Lab 5: Sea-Level Rise

Lucia Romero-Alston (lmr12)

Fri., Feb. 16

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
```

1.4 House

This creates the House object which contains all of the relevant information four our building, including: depth-damage function, area, cost (USD), elevation relative to gauge, and metadata The building I am studying is Fisherman's Wharf at 2200 Harborside Drive Galveston, TX. I got the information for the building area from the cvent website for event space. The value of the property results from searches on Zillow for average building prices in the area. Finally, the depth-damage curve which I chose is a result of location, building type, and what I am looking to analyze for damages.

```
house = let
    haz_fl_dept = CSV.read("data/haz_fl_dept.csv", DataFrame) # read in the file
    desc = "Cafeteria Restaurant, structure"
    row = @rsubset(haz_fl_dept, :Description == desc)[1, :] # select the row I want
    area = 4004u"ft^2"
    height_above_gauge = 4*u"ft" # height is actually 3.74ft but this function only takes integ
    House(
        row;
        area=area,
        height_above_gauge=height_above_gauge,
        value_usd=400_000,
    )
end
```

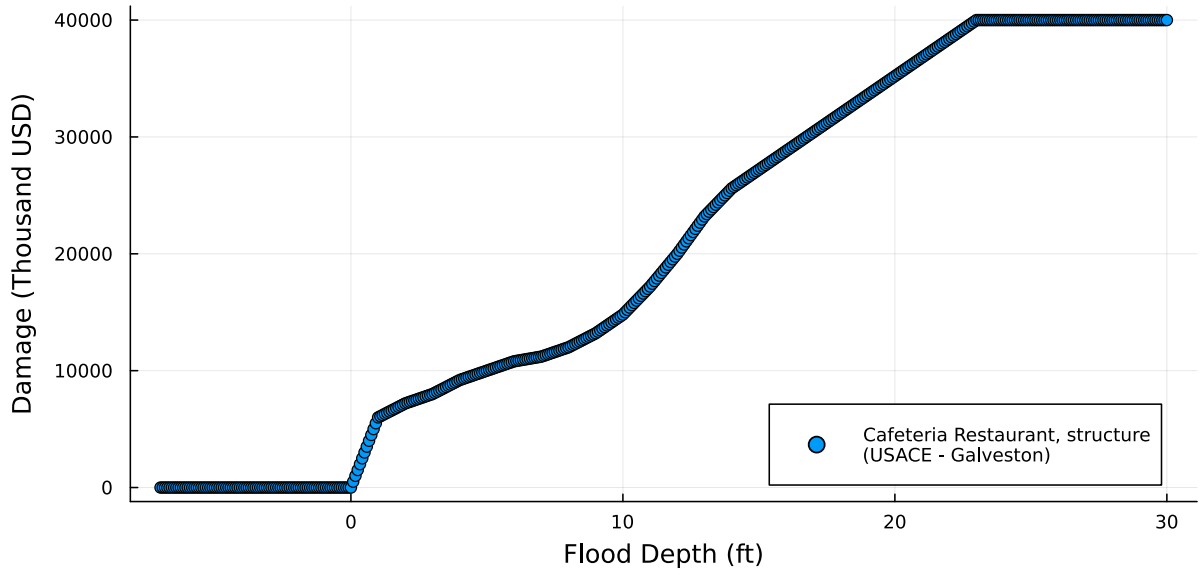
```
House{Int64}(4004, 400000, 4, DepthDamageFunction{Interpolations.Extrapolation{Float64, 1, Inte
   0.0
   0.0
   0.0
   0.0
   0.0
  15.0
  18.0
  20.0
  23.0
  25.0
  27.0
  28.0
  30.0

  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, structu
```

We can use this House object to find a relationship between flood depth and home damage.

```
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
```



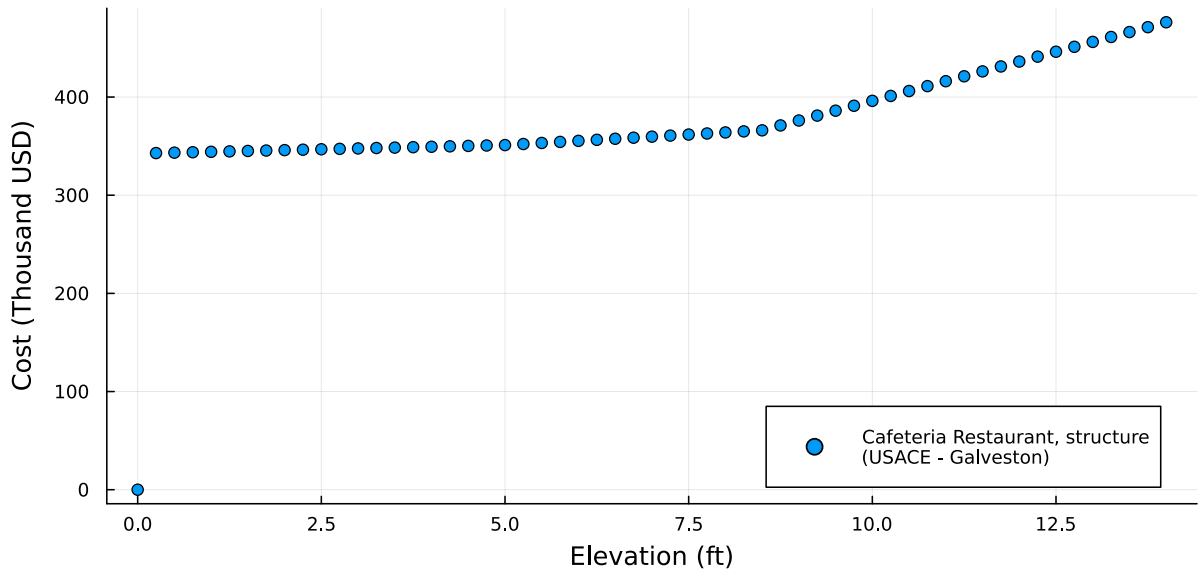
Here we will plot the cost of ekevating our building from 0 to 14 ft using the elevation_cost function.

```
elevation_cost(house, 4u"ft")
```

```
349361.288
```

```
let
    # elevations from 0 to 14 feet at 0.25ft differences
    elevations = 0u"ft":0.25u"ft":14u"ft"
    # cost of elevating the house by each of these heights
    costs = [elevation_cost(house, e ) for e  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
```



1.5 Sea-level Rise

Here we will model sea-level rise using the approach by Otto et al. (2017) that is calibrated to historical sea-level rise in the Netherlands.

```
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")
```

```
There are 34895 parameter sets
```

1.6 Storm Surge

Here we are modeling a distribution for storm surge by sampling parameteres from the range centered in the distribution from Lab 3. This helps to account for the uncertainty in storm surge. We can call this function to get different distributions for storm surge.

```
function draw_surge_distribution()
      = rand(Normal(5, 1))
      = rand(Exponential(1.5))
      = rand(Normal(0.1, 0.05))
    GeneralizedExtremeValue( ,  ,  )
end
```

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

We want a function that draws samples from the storm surge distribution.

```
function surge_dist_sample()
    surge_sample = rand(draw_surge_distribution())
end
```

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

1.7 Discount Rate

The discount rate is important in NPV analysis, but there are both random and not random factors that go into discounting, which are acounted for in the following function. This function already draws a sample from a normal distribution for discount rates.

```
function draw_discount_rate()
    return rand(Normal(0.04, 0.02))
end
```

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

1.8 Running the Simulation

We are adding the object ModelParams, which contains all of the parameters of the model that dont change from one simulation to the next. Note: it may be interesting to run this simulation considering different years.

```
p = ModelParams(
    house=house,
    years=2024:2100
)
```

```
ModelParams(House{Int64}(4004, 400000, 4, DepthDamageFunction{Interpolations.Extrapolation{Flo
   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, structu
```

Here we are creating an object that will hold our state of the world (SOW).

```
sow = SOW(
    rand(slr_scenarios),
    draw_surge_distribution(),
    draw_discount_rate()
)
```

```
SOW{Float64}(Oddo17SLR{Float64}(50.55089646, 2.774668111, 0.005648044, 2041.717541, 27.7886181
```

Here we are defininf out action, a. This action is to raise the building to a fixed elevation.

```
a = Action(3.0u"ft")
```

```
Action{Float64}(3.0)
```

This function runs the simulation, taking in the model parameters, SOW, and the action, and returning the net present value of doing the action.

```
res = run_sim(a, sow, p)
```

```
-1.7923031487824088e6
```

1.9 Large Ensemble

This will be a large ensemble of simulations which samples many SOWs for a range of actions (house elevations).

```
#Create actions and then take a random actions to which we will apply a SOW
heights = [0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0, 7.5, 8.0
sows = [SOW(rand(slr_scenarios), draw_surge_distribution(), draw_discount_rate()) for _ in 1:2
actions = [Action(rand(heights)) for _ in 1:200] # these are all the same
results = [run_sim(a, s, p) for (a, s) in zip(actions, sows)]
```

```
200-element Vector{Float64}:
      -1.4446118475694186e6
```

```
-467491.33570209827
-579434.4336674333
-396120.0
-361800.0
     -1.183380021493698e6
-555957.0697487155
-768098.0725936799
-436268.78927315277
-793935.3495204633
-434498.3596007331
-355845.8237651329
     -2.2772990988736893e6

-361800.0
-394401.74878020963
-446170.0
-391669.4979701457
-406130.0
-446394.61165104836
     -1.0694633976894226e6
     -1.0768354274883731e6
-612054.9877149891
-397956.39415514685
-774194.5354794017
-656577.226029453
```

Here we create a dataframe for our results.

```
years = 2024:2100
df = DataFrame(
    npv=results,
    Δh_ft=[a.Δh_ft for a in actions],
    slr_s=[mean(s.slr.(years)) for s in sows],
    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_val=[mean(s.surge_dist) for s in sows],
    surge_ =[s.surge_dist.  for s in sows],
    surge_ =[s.surge_dist.  for s in sows],
    surge_ =[s.surge_dist.  for s in sows],
    discount_rate=[s.discount_rate for s in sows],
)
```

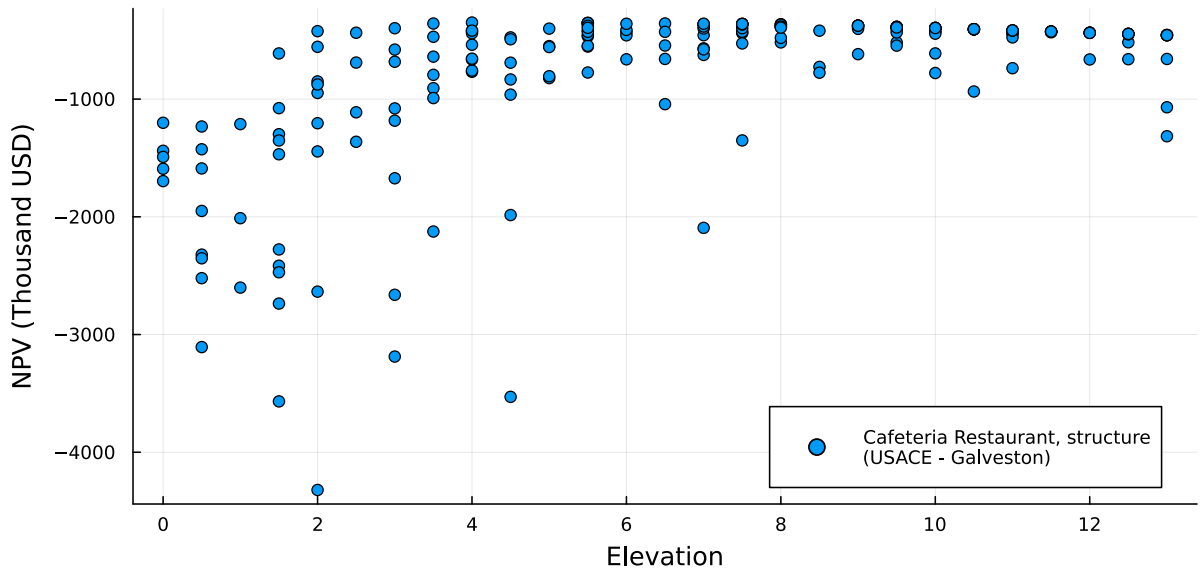
	npv	Δh_ft	slr_s	slr_a	slr_b	slr_c	slr_tstar	slr_cstar	
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	
1	-1.44461e6	2.0	1.36227	6.19152	1.4988	-0.00116971	2018.86	7.44476	...
2	-4.67491e5	5.5	3.30941	30.2822	2.20558	0.00265387	2035.77	30.5083	...
3	-5.79434e5	3.0	0.582561	16.2039	1.44674	-0.0027846	2075.6	20.8036	...
4	-396120.0	10.0	1.41694	21.5675	2.13133	0.00319444	2048.01	14.7724	...
5	-361800.0	7.5	1.29796	42.2341	2.71401	0.00596354	2072.15	30.5108	...
6	-1.18338e6	3.0	1.58043	29.1026	2.16651	0.00259064	2059.65	28.332	...
7	-5.55957e5	2.0	0.851327	34.1427	2.34793	0.00406517	2082.54	29.6809	...
8	-7.68098e5	4.0	0.694862	30.1199	1.81533	-0.000933501	2079.49	25.5172	...
9	-4.36269e5	2.5	0.651807	22.3505	1.86294	0.000221747	2083.16	30.6848	...
10	-7.93935e5	3.5	1.70244	34.7945	2.40169	0.00390445	2042.01	14.3282	...
11	-4.34498e5	11.0	1.69939	47.4219	2.56404	0.00351103	2059.34	26.9446	...
12	-3.55846e5	5.5	1.35591	33.0265	2.33661	0.00268709	2042.62	10.2845	...
13	-2.2773e6	1.5	1.87607	17.2825	1.28192	-0.00484004	2046.46	26.1554	...
14	-1.07918e6	3.0	1.65622	18.829	2.2335	0.00469539	2052.2	21.5936	...
15	-396120.0	10.0	2.33102	44.5027	2.63376	0.0055453	2054.06	34.1761	...
16	-3.66366e5	8.0	1.46759	58.2264	3.00132	0.00767891	2034.5	5.99994	...
17	-3.53645e5	5.5	1.65395	38.0762	2.40114	0.00393204	2036.85	11.4086	...
18	-2.12498e6	3.5	2.13513	21.5101	1.57715	-0.00347967	2041.6	24.2634	...
19	-4.47356e5	11.0	1.86316	23.4518	1.77039	-0.00064349	2045.47	22.2486	...
20	-1.20119e6	0.0	0.582661	27.5169	2.05	0.00144275	2080.68	6.55668	...
21	-3.52961e6	4.5	0.973315	12.7379	1.67705	-0.00034333	2065.73	23.1114	...
22	-3.96122e5	10.0	1.61262	34.2535	2.00534	0.000700356	2059.53	30.2665	...
23	-7.78424e5	10.0	0.986923	17.8691	1.63714	-0.00160911	2063.75	21.4848	...
24	-4.36169e5	12.0	1.58644	69.2595	3.37742	0.0100966	2069.03	25.0594	...
25	-357993.0	3.5	0.724199	25.1785	1.81173	-0.000606214	2060.64	8.3209	...
26	-3.88416e5	9.5	0.514489	27.1022	1.26455	-0.00517329	2066.01	9.55037	...
27	-4.22188e5	4.0	1.28169	11.8843	1.75419	0.000212822	2050.43	16.5298	...
28	-456180.0	13.0	2.18785	10.0674	1.95896	0.00344447	2031.65	16.9047	...
29	-4.39174e5	7.5	2.42277	31.8959	1.87864	-0.000393667	2035.86	21.8154	...
30	-3.53228e5	5.5	1.95418	50.6033	2.93355	0.00750302	2048.21	18.6214	...
...	...	...	...	...	...	...	...	...	

It can be helpful to keep some parameters constant and vary only one at a time to wee which has the greatest evvect on the NPV. Here we will vary ## Analysis

In order to underestand our results we can represent the data through plots. The following is the plot if the relationship between house elevation and NPV.

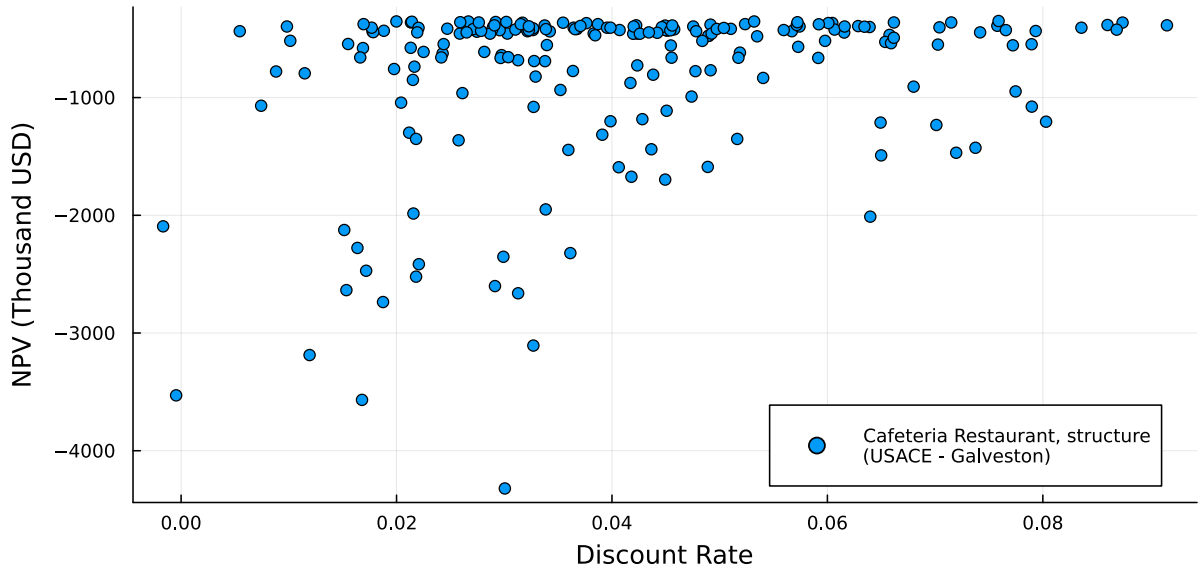
```
scatter(
    df.Δh_ft,
    df.npv ./ 1_000;
    xlabel="Elevation",
    ylabel="NPV (Thousand USD)",
    label="$(house.description)\n($(house.source))",
    legend=:bottomright,
    size=(800, 400),
    yformatter=:plain, # prevents scientific notation
```

```
)
```



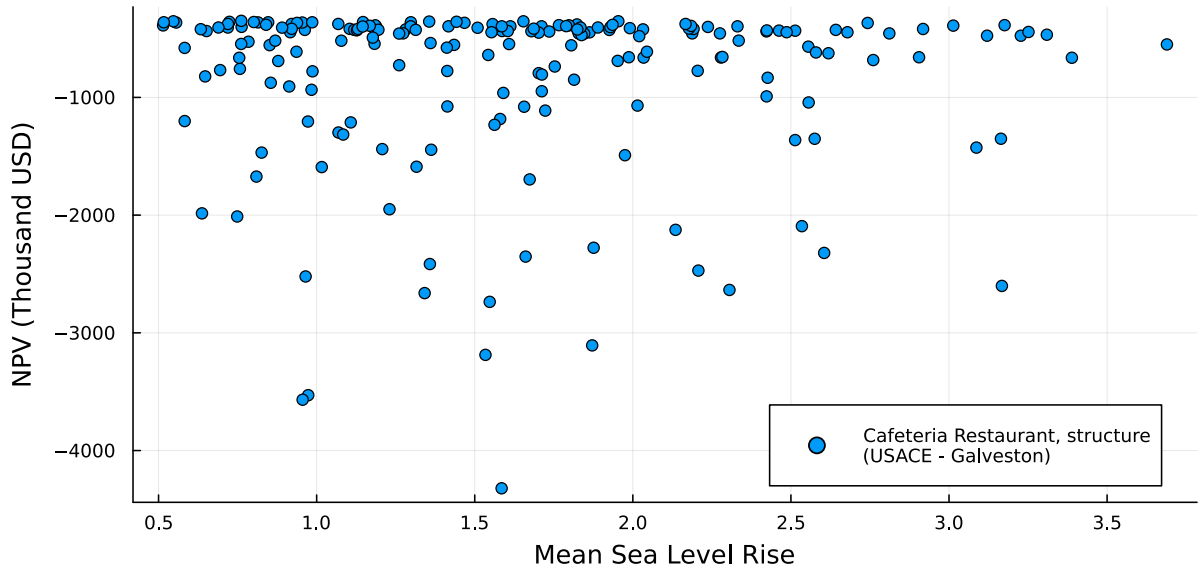
We can also find the relationship between the NPV and other parameters to determine which is the most important and which has the greatest influence on the NPV. Effect of the discount rate on NPV:

```
scatter(
    df.discount_rate,
    df.npv ./ 1_000;
    xlabel="Discount Rate",
    ylabel="NPV (Thousand USD)",
    label="$(house.description)\n($(house.source))",
    legend=:bottomright,
    size=(800, 400),
    yformatter=:plain, # prevents scientific notation
)
```



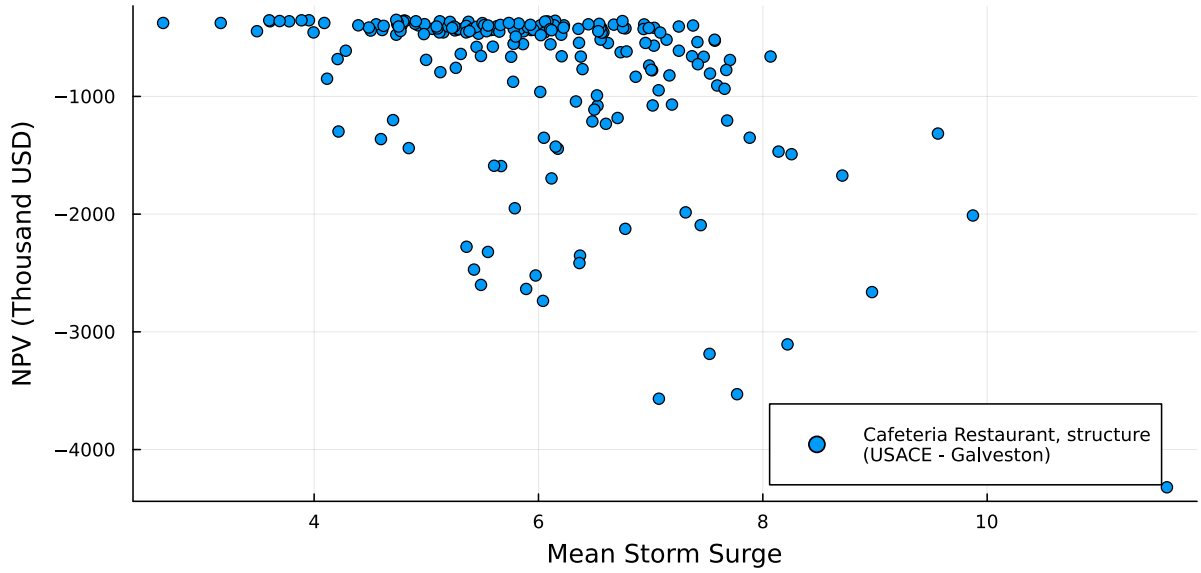
Effect of sea level rise on NPV:

```julia
scatter(
    df.slr_s,
    df.npv ./ 1_000;
    xlabel="Mean Sea Level Rise",
    ylabel="NPV (Thousand USD)",
    label="$(house.description)\n($(house.source))",
    legend=:bottomright,
    size=(800, 400),
    yformatter=:plain, # prevents scientific notation
)
```



Effect of storm surge on NPV:

```
scatter(
    df.surge_val,
    df.npv ./ 1_000;
    xlabel="Mean Storm Surge",
    ylabel="NPV (Thousand USD)",
    label="$(house.description)\n($(house.source))",
    legend=:bottomright,
    size=(800, 400),
    yformatter=:plain, # prevents scientific notation
)
```



When do we get the best results? The best results seem to be at around the 11-ft area of the graph. In this reigon, the scatter points demonstrate a smaller negative NPV, meaning that in the long run, we are still spending money, but less money than for many of the other cases.

When do you get the worst resutls? The worst results appear between 0- and 2-ft. This is the region in which we find the highest lows and the highest highs. This is most likely a result of the cost of elevating the house an elevation which is not enough to make a singificant difference when it comes to flood depth and damage.

What are the most important parameters? Using the graphs comparing the different parameters to the net present value, we can analyze which parameters have a greater effect on NPV. While it is clear that all aprameters can have drastic effects on the NPV, the one which most affects it seems to be the discount rate. This is because for the smallest variations in discount rate, there seems to be the largest differences in NPV. This is important information that can be gained from this sort of analysis because it will inform homewoners, developers and planners that they must be attentive to economic shifts and possibilities, because it is likely to have a large effect on the NPV of their project.

If you had unlimited computing power, would you run more simulations? How many? If it were

possible to have unlimited computing power, I would run trillions of simulations. This is because it would be extrememly to understand and have an exact outcome for every single situation. This would provide homeowners, developers, and policy makers with exact knowledge. This is not realistic of course, so the information we have is a great source of information into what are the more important parameters and on trends and correlations in what we are looking for,

What are the implications of your results for decision-making? The implications of these results is that it is clear that elevating ones home results in monetary saving, but that below a certain threshold, it there is a chance that it is actually more expensive than doing nothing at all. This can be a good guide for people in regions of flood risk to understand what elevating ones home can do, but it should not be taken as law. This is because there are considerations that are not factored in, and uncertainty in many places including future economies, climate change, and flood depth among many other things.