Lab 5: Sea-Level Rise

Solomon Nyamekye : SN66

Wed., Feb. 28

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. 1. Building House Object

house = let  
 haz\_fl\_dept = CSV.read("data/haz\_fl\_dept.csv", DataFrame) # read in the file  
 haz\_fl\_dept, :DmgFnId == 56  
 row = @rsubset(haz\_fl\_dept, :DmgFnId == 56)[1, :] # select the row I want  
 area = 2406u"ft^2"   
 height\_above\_gauge = 2u"ft"  
 House(  
 row;  
 area=area,  
 height\_above\_gauge=height\_above\_gauge,  
 value\_usd=506\_600,  
 )  
end;

## 2.1 a. House details

* Zillow was used for the identification of a single-family residence with 4 bedrooms and 4 bathrooms situated approximately 0.56 miles from Galveston Pier 21, TX. The distance from the gauge was estimated using Google Maps, while the elevation of the house above Mean Sea Level (MSL) was determined using the USGS National Map Viewer. The estimated value of the house and area of the house were from Zillow.
* Depth-damage function from depth-damage functions from the HAZUS model developed by the [US Army Corps of Engineers](https://zenodo.org/records/10027236).

## 2.2 b. Depth-damage curve for House

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

## 2.3 c. Plot of cost for raising the house from 0 to 14 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  
 aspect\_ration=:equal,  
 xlim = (minimum(elevations), maximum(elevations)),  
 ylim = (minimum(costs ./1\_000), maximum(costs ./1\_000)),  
  
 )  
end

# 3. 2. 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;

# 4. 3. Sampling storm surge and discount rates

## 4.1 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) #using GEV distribution for uncertainty surrounding storm surge.   
end

draw\_surge\_distribution (generic function with 1 method)

## 4.2 b. Discount rates

function draw\_discount\_rate()  
#| output: false  
  
 # PDF iscounts rates between 0% to 5% base on historical data while considering the discount rate as random variable and   
 rate = rand(Normal(0.05, 0.02))   
 return max(0.0,rate) #avoiding negative discount rates  
  
 end

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

# 5. Setting up model

## 5.1 a. Setting up Model parameter object, p

p = ModelParams(  
 house=house,  
 years=2024:2100 #using 100 years period  
)

## 5.2 b. Setting up object to hold State of the world, SOW

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

## 5.3 c. Defining my action, a

a = Action(5.0u"ft")

## 5.4 d. Runing the simulation

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

-705644.274557947

# 6. Large esemble

## 6.1 Runing simulations for 10 sow and selected action of elevating the house (0.0ft, 2.0ft, 4.0ft, 6.0ft, 8.0ft)

samples = 100  
  
sows = [SOW(rand(slr\_scenarios), draw\_surge\_distribution(), draw\_discount\_rate()) for \_ in 1:samples] # for 10 SOWs  
  
#Define initial heights for action  
initial\_heights = [0,2,4,6,8] #elevation to at least cover FEMA's BFE   
  
#creating dataframe for each action  
dfs =[]  
df=[]  
npv\_results\_df =[]  
  
for height in initial\_heights  
actions = [Action(height\*u"ft") for \_ in 1:samples] #looping around each height  
results = [run\_sim(a, s, p) for (a, s) in zip(actions, sows)]  
# adding also sea-level rise to dataframe  
year = 2024:(2024-1+samples) # getting number of years sea-level rise data  
slr\_ft\_values = [sow.slr(y) for y in year] #adding Sea-level rise to DataFrame  
  
# creating datafram for results of actions  
df = DataFrame(  
 npv=results,  
 slr=slr\_ft\_values,  
 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],  
)  
push!(dfs, df)  
end  
## combining the results from each sow for each action, a into a single dataframe for analysis  
println("Results for $samples simulations")  
sim\_df = vcat(dfs...) #semicolumn to prevent output from showing

Results for 100 simulations

# 7. Analysis

Finding sow and actions that yields best/worst npv results

## 7.1 Find the sow and action that gives best npv value

using Statistics  
 #fing maximum npv value in all the overall results.  
overall\_best\_npv\_row = argmax(sim\_df.npv)  
#extracting the row that gives best npv results  
best\_npv\_result\_para = sim\_df[overall\_best\_npv\_row, :]  
println(best\_npv\_result\_para)

DataFrameRow  
 Row │ npv slr Δh\_ft slr\_a slr\_b slr\_c slr\_tstar slr\_cstar surge\_μ surge\_σ surge\_ξ discount\_rate   
 │ Float64 Float64 Int64 Float64 Float64 Float64 Float64 Float64 Float64 Float64 Float64 Float64   
─────┼───────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────  
 317 │ -2.22106e5 0.424087 6 32.4789 1.98925 3.96e-5 2077.21 22.4511 4.71161 0.187763 0.0910918 0.0662683

The table above contains combinations of parameters that gives best npv value

## 7.2 Finding the sow and action that gives worst npv result

#finding minimum npv results   
overall\_worst\_npv\_row = argmin(sim\_df.npv)  
#extracting the row that gives best npv results  
worst\_npv\_result\_para = sim\_df[overall\_worst\_npv\_row , :]  
println(worst\_npv\_result\_para)

DataFrameRow  
 Row │ npv slr Δh\_ft slr\_a slr\_b slr\_c slr\_tstar slr\_cstar surge\_μ surge\_σ surge\_ξ discount\_rate   
 │ Float64 Float64 Int64 Float64 Float64 Float64 Float64 Float64 Float64 Float64 Float64 Float64   
─────┼────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────  
 54 │ -1.56655e7 2.09296 0 11.9235 1.48949 -0.00244635 2034.59 22.931 5.38867 1.08929 0.140484 0.00826185

The table above contains combinations of parameters that gives worst npv value

## 7.3 What is the most important parameters

#print(Int(global\_df.Δh\_ft))  
scatter(sim\_df.discount\_rate,   
 sim\_df.npv;   
 zcolor=Int.(sim\_df.Δh\_ft),  
 xlabel="Discount Rates",  
 ylabel="NPV",  
 markersize=Int.(sim\_df.Δh\_ft),  
 size=(800, 400),  
 legend=:bottomright,  
 yformatter=:plain,  
  
 )

scatter(sim\_df.slr,   
 sim\_df.npv;   
 zcolor=Int.(sim\_df.Δh\_ft),  
 xlabel="sea-level rise",  
 ylabel="NPV",  
 markersize=Int.(sim\_df.Δh\_ft),  
 size=(800, 400),  
 legend=:bottomright,  
 yformatter=:plain,  
  
 )

scatter(sim\_df.surge\_μ,   
 sim\_df.npv;   
 zcolor=Int.(sim\_df.Δh\_ft),  
 xlabel="surge\_μ",  
 ylabel="NPV",  
 markersize=Int.(sim\_df.Δh\_ft),  
 size=(800, 400),  
 legend=:bottomright,  
 yformatter=:plain,  
  
 )

scatter(sim\_df.surge\_σ,   
 sim\_df.npv;   
 zcolor=Int.(sim\_df.Δh\_ft),  
 xlabel="surge\_σ",  
 ylabel="NPV",  
 markersize=Int.(sim\_df.Δh\_ft),  
 size=(800, 400),  
 legend=:bottomright,  
 yformatter=:plain,  
)

scatter(sim\_df.surge\_σ,   
 sim\_df.npv;   
 zcolor=Int.(sim\_df.Δh\_ft),  
 markersize=Int.(sim\_df.Δh\_ft),  
 size=(800, 400),  
 xlabel="surge\_σ",  
 ylabel="NPV",  
 yformatter=:plain,  
)

* The results DataFrame and the plots for discount rate, surge\_σ, surge\_μ, and sea-level shows the elevation height has more impact on the npv, followed by the discount rate, the sea-level rise , and then storm surge.
* Also running more simulations increases the resolution and makes it easy to see patterns in the data. I tried 10 and 100 simulations and found out that the 100 gave better picture of the output then the 10 simulations.
* The implies the elevation height impact the npv more than other parameters. However decision makers should also consider the other factors like, discount rate, sea-level rise, and storm surge.