# Lab 4: House Elevation NPV Analysis

Kyle Olcott kto1

Thu., Feb. 8

```
using CSV
using DataFrames
using DataFramesMeta
using Distributions
using Interpolations
using Plots
using StatsPlots
using Unitful

Plots.default(; margin=6Plots.mm)

include("depthdamage.jl")
```

## 1 Defines Damage_fn in Function

```
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
dd = DepthDamageData(row) # extract the depth-damage data
damage_fn = get_depth_damage_function(dd.depths, dd.damages) # get the depth-damage function
```

```
#13 (generic function with 1 method)
```

## 2 Defines Elevation_Cost in Function

```
elevation_cost = get_elevation_cost_function() # gives us a fitted interpolator
```

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

## 3 Defines Flood_Dist in Function

```
    gauge_dist = GeneralizedExtremeValue(5, 1, 0.1) # hypothetical gauge distribution
    offset = 7.5 # hypothetical height from house to gauge
    flood_dist = GeneralizedExtremeValue(gauge_dist.  - offset, gauge_dist. , gauge_dist. )
```

```
GeneralizedExtremeValue{Float64}( =-2.5,  =1.0,  =0.1)
```

# 4 Defines house area, value, and desired elevation for function

```julia
house_area = 1000u"ft^2" # Change these values later
house_value = 250000
elevation = 1
flood_dist = GeneralizedExtremeValue(gauge_dist.  - (offset + elevation), gauge_dist. , gauge_di
Δh = elevation * 1u"ft"
```

```
1 ft
```

# 5 Function for year 0 analysis

```julia
function single_year_cost_benefit(flood_dist, damage_fn, elevation_cost, house_area, house_valu

    # calculate the expected damages
    samples = rand(flood_dist, 100_000) .* 1u"ft"
    damages = damage_fn.(samples)
    expected_damages_pct = mean(damages)
    c_dmg = expected_damages_pct * house_value / 100

    # calculate the cost of elevating
    c_constr = elevation_cost.(Δh, house_area)

    # return the total cost and benefit
    return -c_constr - c_dmg
end
```

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

# 6 Testing function for year 0

```julia
yearonecost = single_year_cost_benefit(flood_dist, damage_fn, elevation_cost, house_area, house
print(yearonecost)
```

```
-106683.28200120892
```

# 7 Defines time period and discount rate

```julia
T = 10
discount_rate = 0.05
```

```
0.05
```

# 8 Function for NPV analysis

```
function npv_cost_benefit(flood_dist, damage_fn, elevation_cost, house_area, house_value, Δh, T
    # calculate the costs and benefits for each year, and then discount
    time = 0
    npv = 0
    while time < T
    time = time + 1
    cost = single_year_cost_benefit(flood_dist, damage_fn, elevation_cost, house_area, house_va
    yearcost = cost * (1-discount_rate)^time
    npv = npv + yearcost
    Δh=0u"ft"
    end

    return npv
end
```

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

# 9 Test for NPV function

```
npvcost = npv_cost_benefit(flood_dist, damage_fn, elevation_cost, house_area, house_value, Δh,
print(npvcost)
```

```
-134990.72826536026
```

# 10 One SOW, several actions

For 0ft of elevation

```
elevation = 0
flood_dist = GeneralizedExtremeValue(gauge_dist.  - (offset + elevation), gauge_dist. , gauge_di
Δh = elevation * 1u"ft"
npvcost = npv_cost_benefit(flood_dist, damage_fn, elevation_cost, house_area, house_value, Δh,
print(npvcost)
```

```
-78471.30983427571
```

For 1 ft of elevation

```
elevation = 1
flood_dist = GeneralizedExtremeValue(gauge_dist.  - (offset + elevation), gauge_dist. , gauge_di
Δh = elevation * 1u"ft"
npvcost = npv_cost_benefit(flood_dist, damage_fn, elevation_cost, house_area, house_value, Δh,
print(npvcost)
```

```
-135189.04094402914
```

For 2 ft of elevation

```
elevation = 2
flood_dist = GeneralizedExtremeValue(gauge_dist.  - (offset + elevation), gauge_dist. , gauge_di
Δh = elevation * 1u"ft"
npvcost = npv_cost_benefit(flood_dist, damage_fn, elevation_cost, house_area, house_value, Δh,
print(npvcost)
```

```
-116709.5585213904
```

For 3 ft of elevation

```
elevation = 3
flood_dist = GeneralizedExtremeValue(gauge_dist.  - (offset + elevation), gauge_dist. , gauge_di
Δh = elevation * 1u"ft"
npvcost = npv_cost_benefit(flood_dist, damage_fn, elevation_cost, house_area, house_value, Δh,
print(npvcost)
```

```
-107930.6242180151
```

For 4 ft of elevation

```
elevation = 4
flood_dist = GeneralizedExtremeValue(gauge_dist.  - (offset + elevation), gauge_dist. , gauge_di
Δh = elevation * 1u"ft"
npvcost = npv_cost_benefit(flood_dist, damage_fn, elevation_cost, house_area, house_value, Δh,
print(npvcost)
```

```
-103464.04518432065
```

For 5 ft of elevation

```
elevation = 5
flood_dist = GeneralizedExtremeValue(gauge_dist.  - (offset + elevation), gauge_dist. , gauge_di
Δh = elevation * 1u"ft"
npvcost = npv_cost_benefit(flood_dist, damage_fn, elevation_cost, house_area, house_value, Δh,
print(npvcost)
```

```
-101347.63765986939
```

For 6 ft of elevation

```
elevation = 6
flood_dist = GeneralizedExtremeValue(gauge_dist.  - (offset + elevation), gauge_dist. , gauge_di
Δh = elevation * 1u"ft"
npvcost = npv_cost_benefit(flood_dist, damage_fn, elevation_cost, house_area, house_value, Δh,
print(npvcost)
```

```
-101071.07710118765
```

For 7 ft of elevation

```
elevation = 7
flood_dist = GeneralizedExtremeValue(gauge_dist.  - (offset + elevation), gauge_dist. , gauge_di
Δh = elevation * 1u"ft"
```

```
npvcost = npv_cost_benefit(flood_dist, damage_fn, elevation_cost, house_area, house_value, Δh,
print(npvcost)
```

```
-101359.15844428517
```

## 11 Sensitivity Test

```
NPVdist = Normal(4, 2)
total = 0
n = 0

while n < 1000000
n = n + 1
sample = rand.(NPVdist, 1)
total = total .+ sample
end

expectedNPV = total ./ n
print(expectedNPV)
```

```
[4.0028514859826645]
```

## 12 Discussion

1: Suprisingly, not elevating the house was calculated to be the most cost effective. Once elevated, the NPV sudden jumps up and slowly decreases the more you elevate until eventually, it begins to rise up. This makes sense as at a point in elevating the house, you will have made it safe from the majority of flood events, meaning any further elevation just increases the cost without providing any benefits. If the house costed more or the flood distribution was more hazardous, the NPV for elevating the house could have been better than not elevating it. Additionally, if the analysis was conducted over a greater time line for the no elevation scenario, the cost to rebuilt the house every year will eventually outpace the money saved by not elevating the house in year 1.

2: The sensitivity test returned approx. 4. This makes sense as the expected value for a variable is just another way of saying its mean. Since the distribution used had a mean of 4, we should expect 4 to be returned.

3: A limitation of the analysis is that the expected flood damage is used every year despite this might not being realistic. For example if I elevate my house and it floods the next year, I will have a greater benefit than if it had flooded only after 10 years from now because of the discount rate.

One things that is missing from the analysis is building cost fluctuations. For this analysis we only used the current price of the house, however, that price is likely to rise as labor and material costs go up. They may even go up much more due to the limited reasources that are present directly after a flood. Implementing this would likely decrease the benefit from elevating and not elevating, however, it would likely favor elevating more as you would then avoid the much higher rebuilding costs from your non-elevated house being damaged.

One way that we could address this limitation is by introducing a factor that we multiply the house's value by to simulate increased material costs after a flood. Additionally, a discount rate for the housing market could be multiplied with the house value each year to simulate increasing labor costs and the house naturally gaining value.