Lab 3: Depth-Damage Models

DataFrames and Distributions

Lucia Romero-Alston (lmr12)

Fri., Jan. 26

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

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

1 Site information

This is an analysis of the Fisherman's Wharf at 2200 Harborside Dr, Galveston using data from the 8771450 Galveston Pier 21, TX guage. This building is "near" the water guage as it is located right off of the pier where the guage is placed. Building Elevation: 3.74 ft

```
haz_fl_dept = CSV.read("data/haz_fl_dept.csv", DataFrame)
include("depthdamage.jl")
```

```
DepthDamageData
```

2 Depth-Damage

Depth-Damage Function: USACE-Galveston Cafeteria Restaurant, Structure I am using this as the depth damage function because the building I am analyzing fits under the classification of a restaurant. More importantly, it is in Galveston, so in order to accurately represent water levels in the area, I have to use a function that is based off of this geographical location.

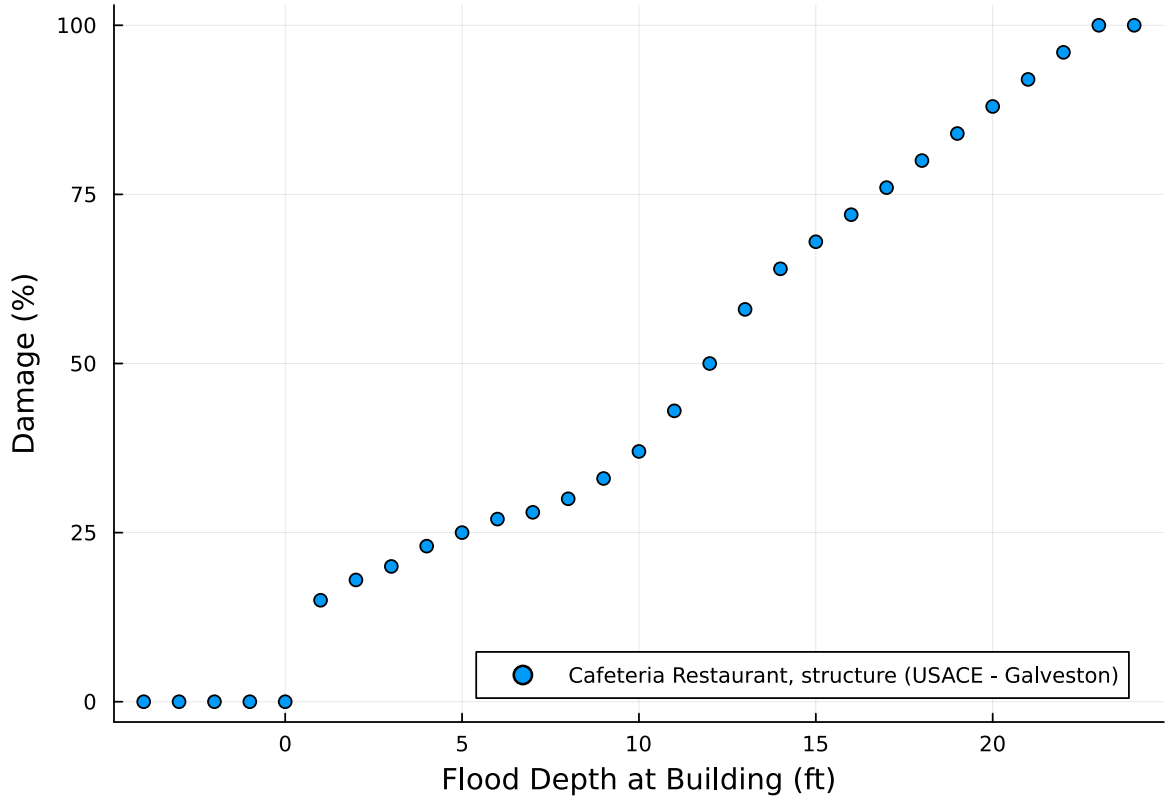
```
row = @rsubset(
    haz_fl_dept, :Description == "Cafeteria Restaurant, structure"
)[
    1,:,
]
dd = DepthDamageData(row)
```

```
DepthDamageData(Quantity{Float64,  , Unitful.FreeUnits{(ft,),  , nothing}}[-4.0 ft, -3.0 ft, -2
```

```
fieldnames(typeof(dd))
```

```
(:depths, :damages, :occupancy, :dmg_fn_id, :source, :description, :comment)
```

```
scatter(
    dd.depths,
    dd.damages;
    xlabel="Flood Depth at Building",
    ylabel="Damage (%)",
    label="$(dd.description) ($(dd.source))",
    legend=:bottomright,
    size=(700, 500),
)
```




```
itp = let
    depth_ft = ustrip.(u"ft", dd.depths)
    damage_frac = dd.damages
    Interpolations.LinearInterpolation(
        depth_ft,
        damage_frac;
        extrapolation_bc=Interpolations.Flat(),
    )
```

```
end
```

```
29-element extrapolate(interpolate((::Vector{Float64},), ::Vector{Float64}, Gridded(Linear())
   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
```

```
let
    dmg_fn(x) = itp(ustrip.(u"ft", x))
    dmg_fn.([3.1u"ft", 2.2u"m", 91.4u"inch"])
end
```

```
3-element Vector{Float64}:
 20.3
 28.435695538057743
 29.23333333333333
```

```
function get_depth_damage_function(
    depth_train::Vector{<:T}, dmg_train::Vector{<:AbstractFloat}
) where {T<:Unitful.Length}

    # interpolate
    depth_ft = ustrip.(u"ft", depth_train)
    interp_fn = Interpolations.LinearInterpolation(
        depth_ft,
        dmg_train;
```

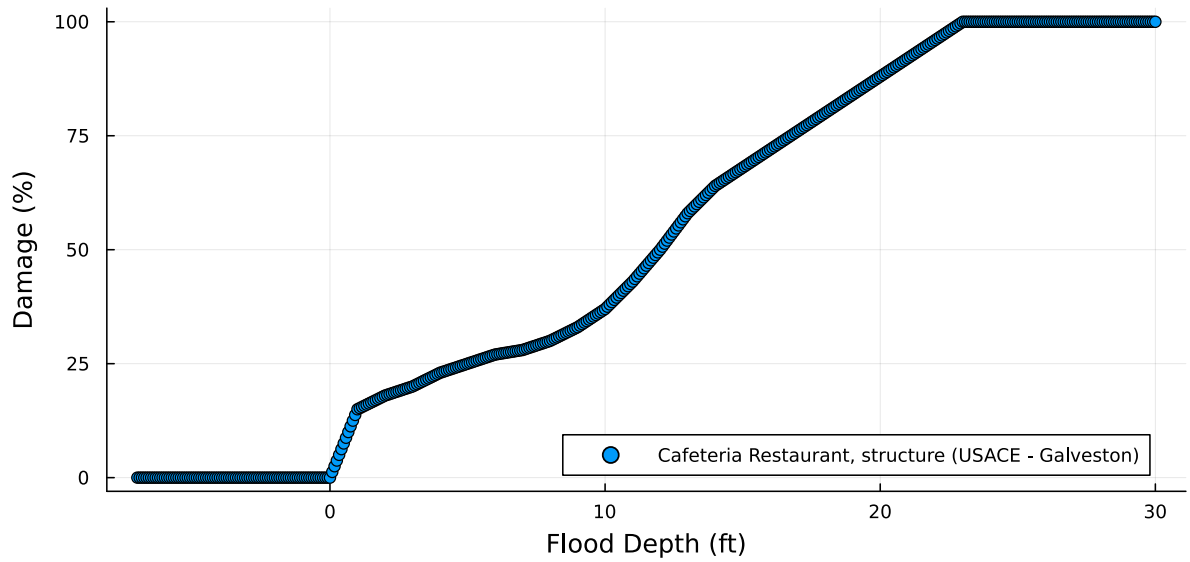
```
        extrapolation_bc=Interpolations.Flat(),
    )

    damage_fn = function (depth::T2) where {T2<:Unitful.Length}
        return interp_fn(ustrip.(u"ft", depth))
    end
    return damage_fn
end
```

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

```
damage_fn = get_depth_damage_function(dd.depths, dd.damages)
```

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

```
p = let
    depths = uconvert.(u"ft", (-7.0u"ft"):(1.0u"inch"):(30.0u"ft"))
    damages = damage_fn.(depths)
    scatter(
        depths,
        damages;
        xlabel="Flood Depth",
        ylabel="Damage (%)",
        label="$(dd.description) ($(dd.source))",
        legend=:bottomright,
        size=(800, 400),
        linewidth=2,
    )
end
p
```



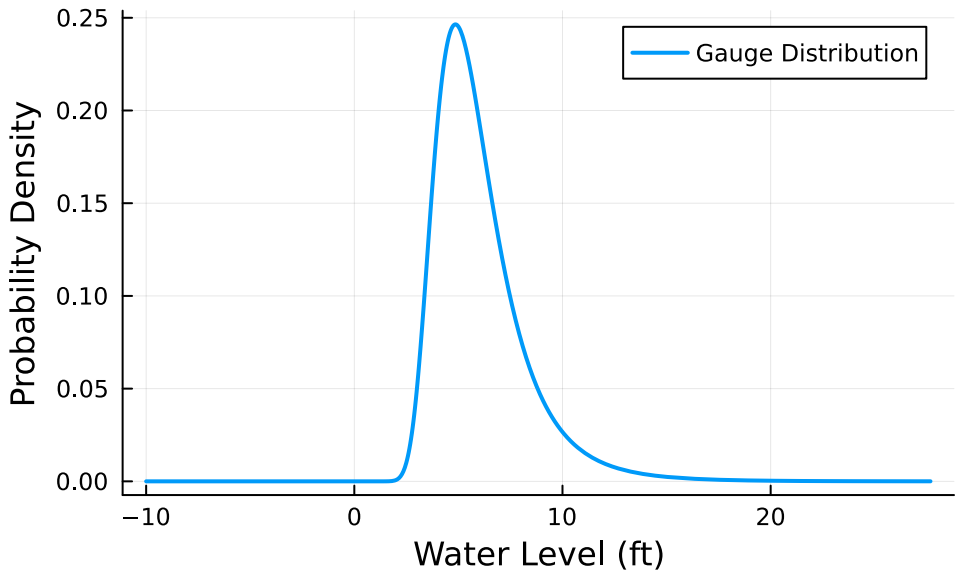
This curve is plotted in 1 inch increments from -10 to 30 feet. This graph represents an estimation of the amount of damage that a building will experience given a certain flood depth. Plotting this at 1 inch increments provides a detailed graph with the estimated damage for every inch of water rise that can be more generally used for depths that are in between foot measurments.

3 Expected annual damages

```
gauge_dist = GeneralizedExtremeValue(5, 1.5, 0.1)
```

```
GeneralizedExtremeValue{Float64}( =5.0,  =1.5,  =0.1)
```

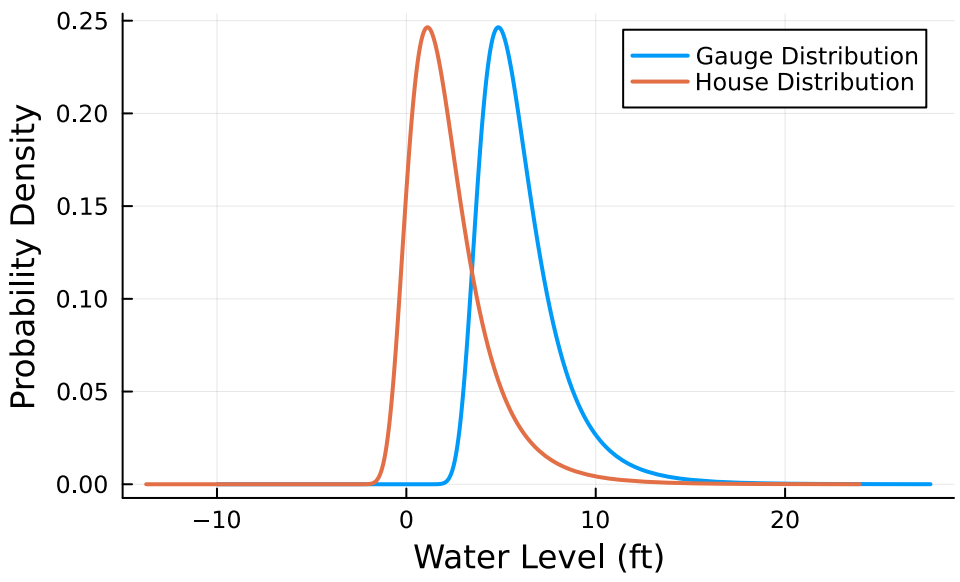
```
p1 = plot(
    gauge_dist;
    label="Gauge Distribution",
    xlabel="Water Level (ft)",
    ylabel="Probability Density",
    legend=:topright,
    linewidth=2,
)
```




```
offset = 3.74 # building is 3.74 feet above gauge
house_dist = GeneralizedExtremeValue(gauge_dist.  - offset, gauge_dist. , gauge_dist. )
```

```
GeneralizedExtremeValue{Float64}( =1.2599999999999998,  =1.5,  =0.1)
```

```
plot!(p1, house_dist; label="House Distribution", linewidth=2)
```




```
using Random
```

The following is known as a Monte Carlo Algorithm, which is estimating the expected annual damages based on 1,000,000 samples. This algorithm randomly takes samples of hazard from a distribution, which are then inpuut into the damge function in order to get a distribution of damages from randomized hazards. The mean of these damage values gives the average value of damage that a home in this location might experience during an event.

```
N=1000000
sample = rand(house_dist, N) .* 1.0u"ft"
result = damage_fn.(sample)
mean(result)
```

```
16.085853129988504
```

The result of this algorithm is 16.08%.

4 Discussion

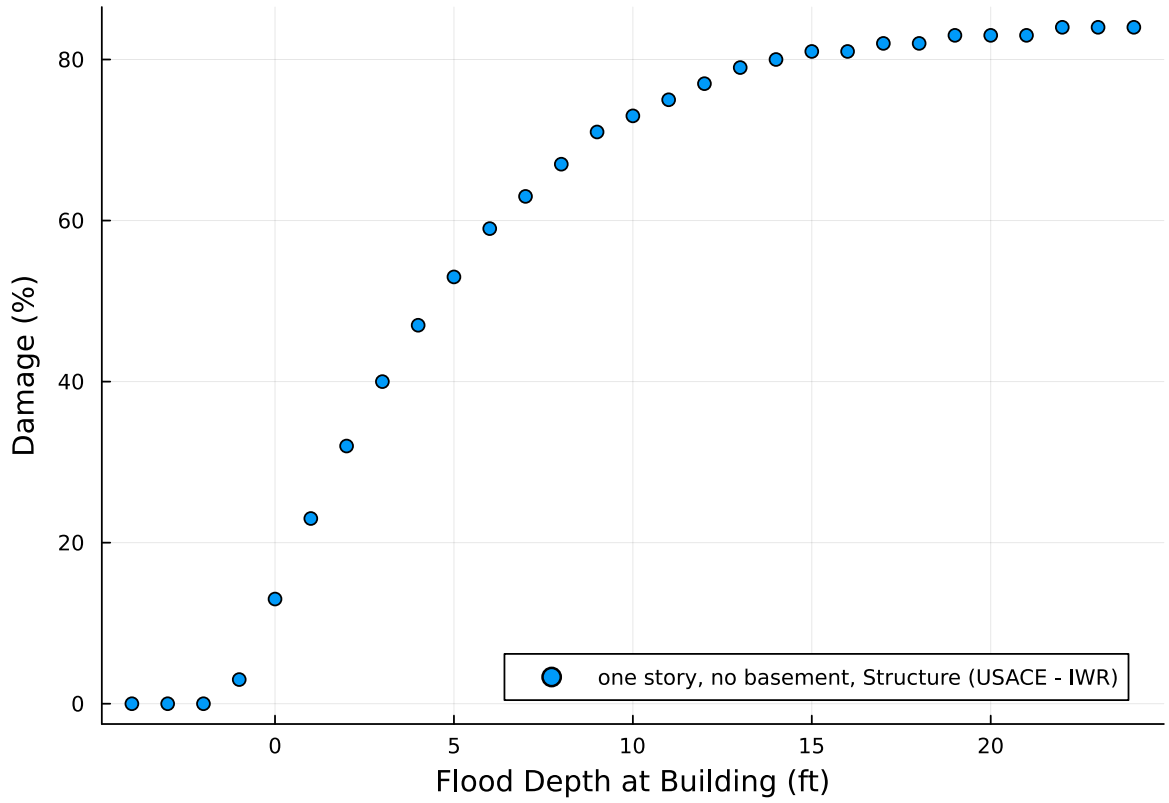
Here I am perfomring another trial on the same location using a different Depth-Damage function. For this analysis I am using the USACE-Galveston "one story, no basement, Structure" function. This trial can demonstrate how changes in depth damage function can change results.

```
row2 = @rsubset(
    haz_fl_dept, :Description == "one story, no basement, Structure"
)[
    1,:,
]
```

```
dd2 = DepthDamageData(row2)
fieldnames(typeof(dd2))
```

```
(:depths, :damages, :occupancy, :dmg_fn_id, :source, :description, :comment)
```

```
scatter(
    dd2.depths,
    dd2.damages;
    xlabel="Flood Depth at Building",
    ylabel="Damage (%)",
    label="$(dd2.description) ($(dd2.source))",
    legend=:bottomright,
    size=(700, 500),
)
```




```
itp = let
    depth_ft2 = ustrip.(u"ft", dd2.depths)
    damage_frac2 = dd2.damages
    Interpolations.LinearInterpolation(
        depth_ft2,
        damage_frac2;
        extrapolation_bc=Interpolations.Flat(),
    )
```

```
end
```

```
29-element extrapolate(interpolate((::Vector{Float64},), ::Vector{Float64}, Gridded(Linear()))
  0.0
  0.0
  0.0
  3.0
 13.0
 23.0
 32.0
 40.0
 47.0
 53.0
 59.0
 63.0
 67.0

 79.0
 80.0
 81.0
 81.0
 82.0
 82.0
 83.0
 83.0
 83.0
 84.0
 84.0
 84.0
```

```
let
    dmg_fn2(x) = itp(ustrip.(u"ft", x))
    dmg_fn2.([3.1u"ft", 2.2u"m", 91.4u"inch"])
end
```

```
3-element Vector{Float64}:
 40.7
 63.871391076115486
 65.46666666666667
```

```
function get_depth_damage_function(
    depth_train2::Vector{<:T}, dmg_train2::Vector{<:AbstractFloat}
) where {T<:Unitful.Length}

    # interpolate
    depth_ft2 = ustrip.(u"ft", depth_train2)
    interp_fn2 = Interpolations.LinearInterpolation(
        depth_ft2,
        dmg_train2;
```

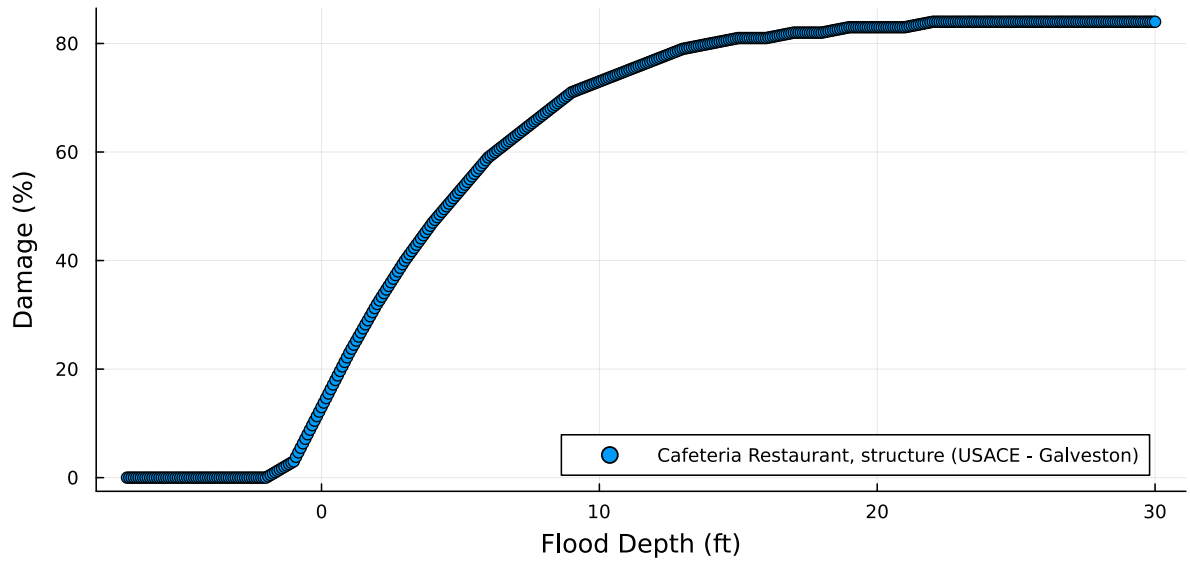
```
        extrapolation_bc=Interpolations.Flat(),
    )

    damage_fn2 = function (depth::T2) where {T2<:Unitful.Length}
        return interp_fn2(ustrip.(u"ft", depth))
    end
    return damage_fn2
end
```

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

```
damage_fn2 = get_depth_damage_function(dd2.depths, dd2.damages)
```

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

```
p = let
    depths2 = uconvert.(u"ft", (-7.0u"ft"):(1.0u"inch"):(30.0u"ft"))
    damages2 = damage_fn2.(depths2)
    scatter(
        depths2,
        damages2;
        xlabel="Flood Depth",
        ylabel="Damage (%)",
        label="$(dd.description) ($(dd.source))",
        legend=:bottomright,
        size=(800, 400),
        linewidth=2,
    )
end
p
```



```
N=1000000
sample2 = rand(house_dist, N) .* 1.0u"ft"
result2 = damage_fn2.(sample2)
mean(result2)
```

```
32.18863986049266
```

The result of this algorithm is 32.14%

This result is double that from the prefious depth-damage function. This extreme shift in results demonstrates high sensibiolity to the kind of depth-damage function used. This is important to take into consideration when choosing a function because the results can impact the way in which communities prepare for flood events, whether it is building codes or insurance.