Lab 5: Sea-Level Rise

Andres Calvo (ac228)

Fri., Feb. 16

1 Load packages

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

Plots.default(; margin=5Plots.mm)
```

1.1 Local package

```julia
using Revise
using HouseElevation
```

2 Building object

The information from previous labs is integrated in a `Building` object with the following parameters:

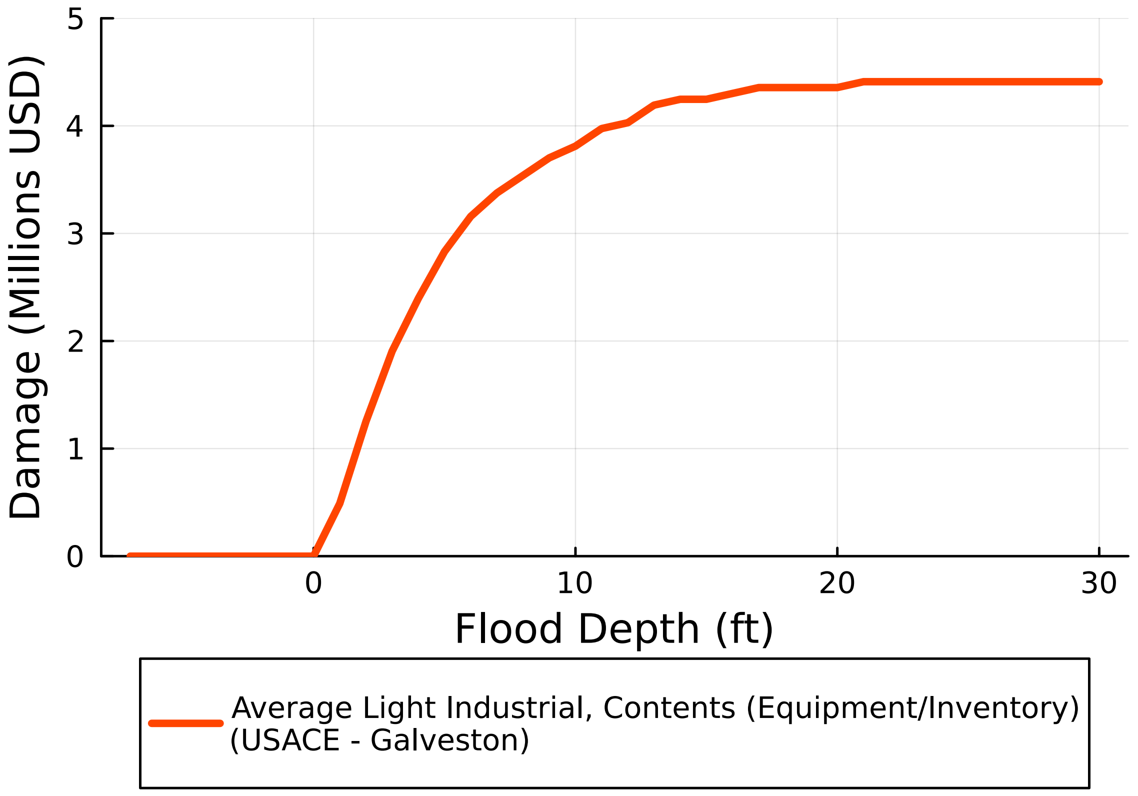
Area 33 000 ft^2
Offset from gauge 6 ft
Valuation 5'445 000 USD (*Structure + Contents*)
Offset (measure from gauge) 6 ft

```julia
offset = 6
building = let
    haz_fl_dept = CSV.read("data/haz_fl_dept.csv",DataFrame)
    desc = "Average Light Industrial, Contents (Equipment/Inventory)"
    row = @rsubset(haz_fl_dept, :Description == desc)[1,:]
    area = 33000u"ft^2"
    height_above_gauge = (offset)u"ft"
    House(
        row;
        area = area,
        height_above_gauge = height_above_gauge,
        value_usd = 5_445_000,
        )
end
```

2.1 Building Depth-Damage function

The following Depth-Damage function (DDF) is use from the USACI-Galveston data for light industrial buildings

```
let
    depths = uconvert.(u"ft", (-7.0u"ft"):(1.0u"inch"):(30.0u"ft"))
    damages = building.ddf.(depths) ./ 100
    damages_1e6_usd = damages .* building.value_usd ./ 1e6
    plot(
        depths,
        damages_1e6_usd;
        xlabel = "Flood Depth",
        ylabel = "Damage (Millions USD)",
        ylims = [0,trunc(maximum(damages_1e6_usd)) + 1],
        label = "$(building.description)\n($(building.source))",
        legend = :outerbottom,
        size = (500, 400),
        yformatter=:plain, # prevents scientific notation
        color = "orangered",
        linewidth = 3,
    )
end
```

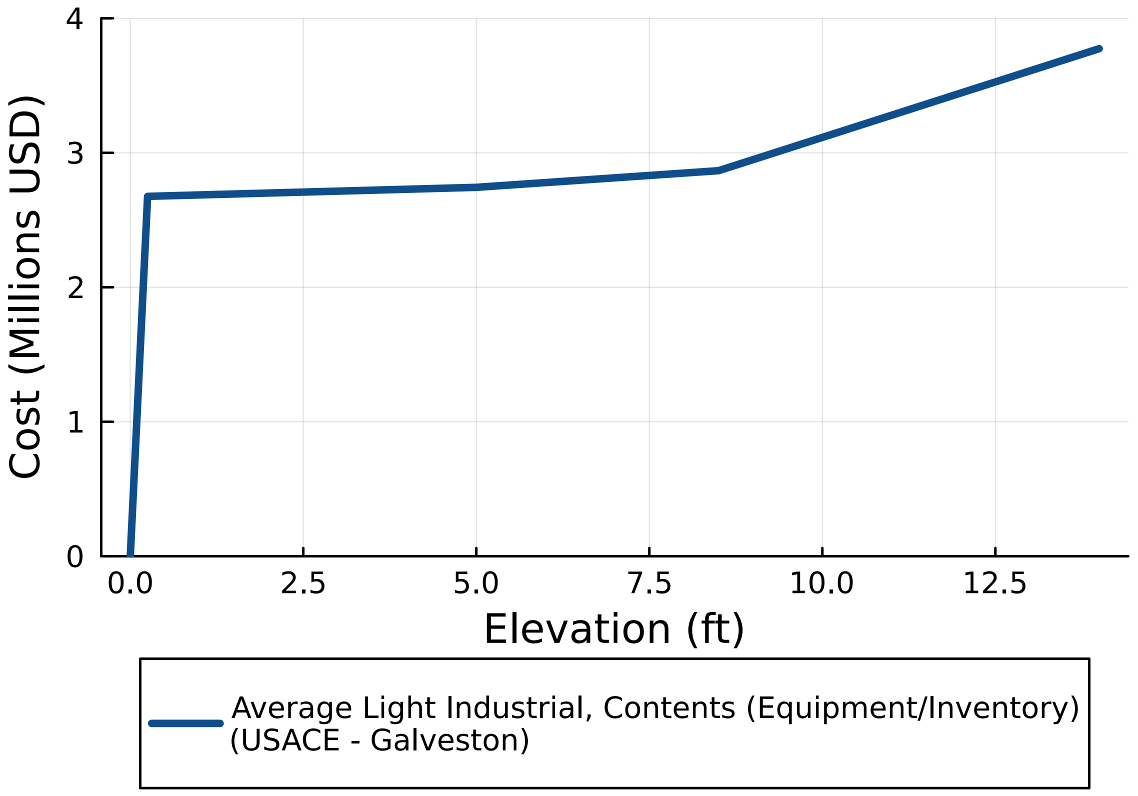


2.2 Building Elevation cost function

The elevation costs considered are the following for every elevation:

```julia
let
    elevations = 0u"ft":0.25u"ft":14u"ft"
    costs = [elevation_cost(building, e ) for e  in elevations]
    plot(
        elevations,
        costs ./ 1e6;
        xlabel="Elevation",
        ylabel="Cost (Millions USD)",
        ylims = [0,trunc(maximum(costs ./ 1e6)) + 1],
        label="$(building.description)\n($(building.source))",
        legend=:outerbottom,
        size=(500, 400),
        yformatter=:plain, # prevents scientific notation
        color = "dodgerblue4",
        linewidth = 3,
        )
```

```
end
```



2.3 Sea-level rise model

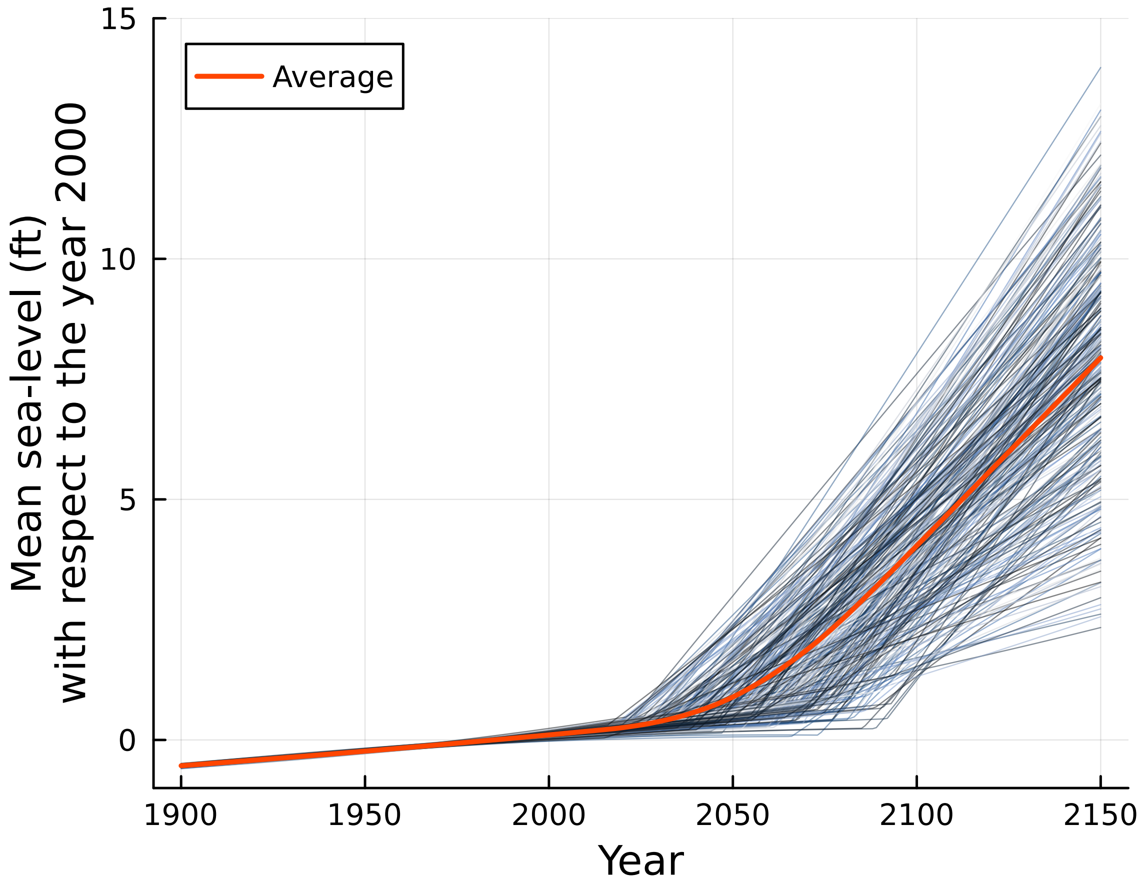
The Sea-level rise (SLR) is adapted from Oddo et al. (2017)

```
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
```

The following are 300 random samples (realizations) using the model. The average trend of these simulations is also included.

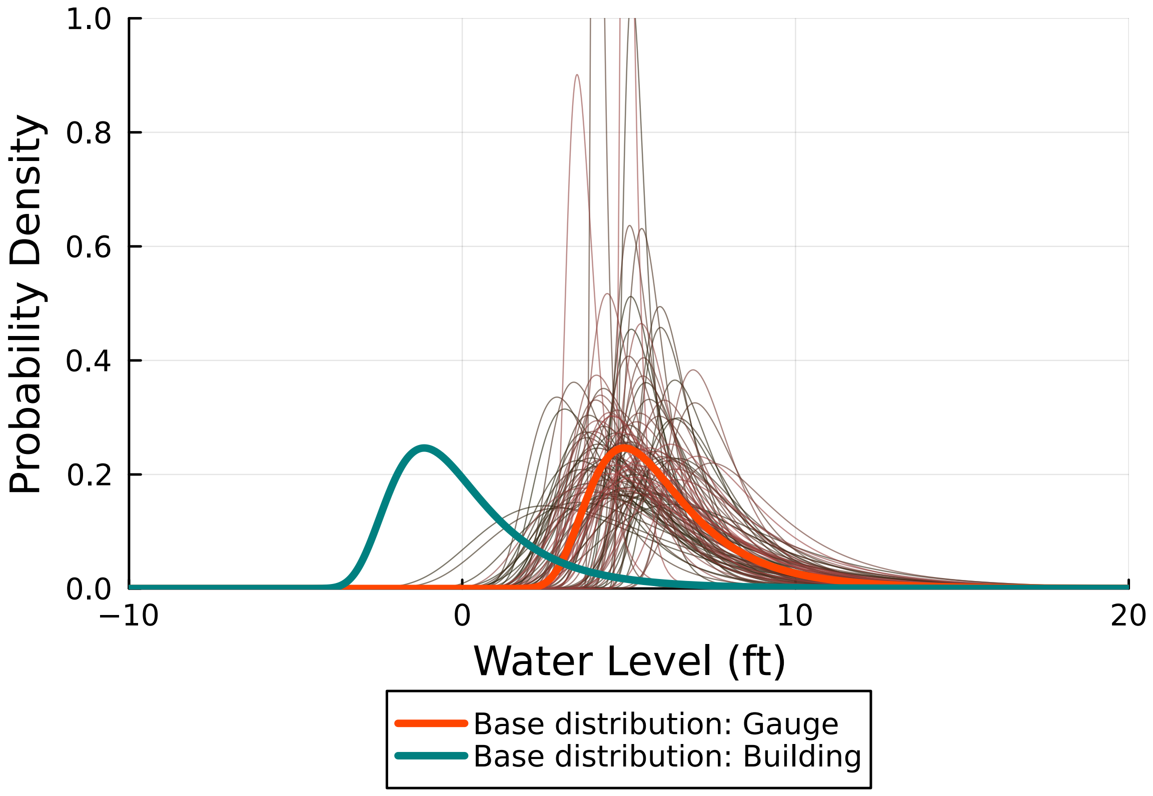
```
let
    years = 1900:2150
    p = plot(;
             xlabel = "Year",
            ylabel = "Mean sea-level (ft)\nwith respect to the year 2000",
            label = "Oddo et al. (2017)",
            legend = :topleft,
            size=(500, 400),)
    s_average = years.*0
    for s in rand(slr_scenarios,300)
        plot!(p,
             years,
             s.(years);
             palette = :oslo,
             alpha = 0.5,
             linewidth = 0.5,
             label = nothing,
             )
        s_average +=  s.(years)
    end
    s_average /= 300
    plot!(years,
          s_average;
          ylims = [-1,15],
          color = "orangered",
          label = "Average",
          linewidth = 2,
          )
    p
end
```



2.4 Storm surge hazard model

A General Extreme Value (GEV) distribution for the gauge is selected to represent the flood hazard, that is, the flood depth intensity probability. The following are the distribution parameters (, and) that can be randomly sampled from normal distributions around average values. The plot shows 100 random realizations and the based distribution for the gauge and for the building (subtracting the offset height).

```
function draw_surge_distribution()
      = rand(Normal(5, 1))
      = rand(truncated(Normal(1.5,0.5),0,Inf))
      = rand(Normal(0.1, 0.05))
    GeneralizedExtremeValue( ,  ,  )
end
let
    p = plot(;
        xlabel = "Water Level (ft)",
        ylabel = "Probability Density",
        xlims = [-10,20],
        ylims = [0,1],
        size=(500, 400),
        )
    plot!(p,
          [draw_surge_distribution() for _ in 1:100];
          palette = :lajolla,
          linewidth = 0.5,
          alpha = 0.6,
          label = nothing,
          )
    plot!(p,
          GeneralizedExtremeValue(5, 1.5, 0.1);
          color = "orangered",
          linewidth = 3,
          label = "Base distribution: Gauge",
          legend = :outerbottom,
          )
    plot!(p,
          GeneralizedExtremeValue(5-offset, 1.5, 0.1);
          color = "Teal",
          linewidth = 3,
          label = "Base distribution: Building",
          legend = :outerbottom,
          )
end
```

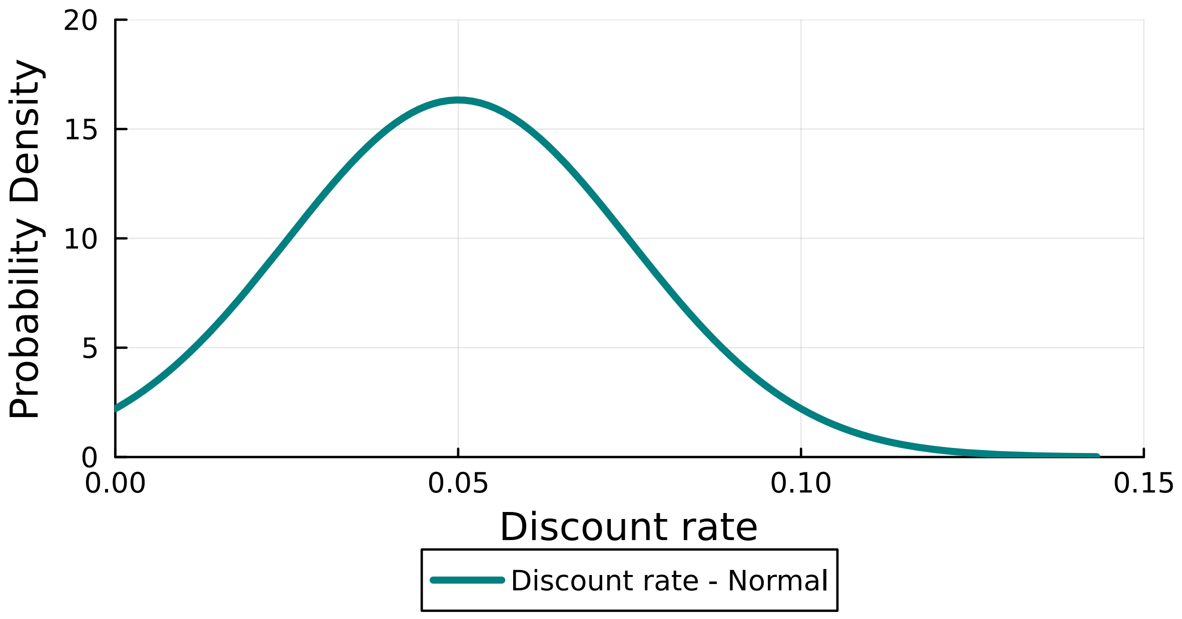


2.5 Discount rate

The discount rate is generated following normal distributions with a mean value of 5%. This higher value could be appropriate for industrial stakeholders (even larger) considering that their cost of opportunity can be higher than those for house holders.

```julia
function draw_discount_rate()
    return rand(truncated(Normal(0.05, 0.025),0,Inf))
end
let
    p = plot(truncated(Normal(0.05, 0.025),0,Inf);
             xlabel = "Discount rate",
             ylabel = "Probability Density",
             ylims = [0,20],
             xlims = [0,0.15],
             color = "teal",
             linewidth = 3,
             label = "Discount rate - Normal",
             legend = :outerbottom,
             )
```

```
end
```



2.6 Simulations

The following is an example simulation considering a 50-year time window and random realizations of the hazard distribution, discount rate and SLR model. The action is elevating the building by 3 ft.

```
let
    p = ModelParams(
        house = building,
        years = 2024:2034
    )

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

    a = Action(3.0u"ft")

    res = run_sim(a, sow, p)/1e6

    print("The NPV cost for the action a = 3 ft and a realization of the SOW is \n $(round(res
```

```julia
end
```

```
The NPV cost for the action a = 3 ft and a realization of the SOW is
 -2.77 USD Millions
```

2.7 Exploratory modeling

The actions correspond to incremental elevation heights: Actions a : [0: 1: 14] ft

There is no elevation cost above 14 ft and would be more than one-story elevation. For each action, 100 SOWs are going to be considered to form the "*large ensemble*". Inside the functions, there are 10000 Monte Carlo Sampling realizations.

```julia
df = let
    action_scheme = 0:1:14
    time_frame = [25,50,100]
    realizations = 100
    simulations = 0
    for t in 1:size(time_frame)[1]
        for e in 1:size(action_scheme)[1]
            p = ModelParams(
                            house = building,
                            years = 2024:(2024 + time_frame[t])
                            )

            sows = [SOW(rand(slr_scenarios),
                        draw_surge_distribution(),
                        draw_discount_rate())
                    for _ in 1:realizations]

            actions = [Action((action_scheme[e])u"ft") for _ in 1:realizations]

            results = [run_sim(a, s, p) for (a, s) in zip(actions, sows)]

            if t == 1 && e == 1
                simulations = DataFrame(
                        npv = results,
                        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_  = [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],
                        years_frame = time_frame[t])
            else
```

```
                for r_i in 1:realizations
                    push!(simulations,[results[r_i],
                            [a.Δh_ft for a in actions][r_i],
                            [s.slr.a for s in sows][r_i],
                            [s.slr.b for s in sows][r_i],
                            [s.slr.c for s in sows][r_i],
                            [s.slr.tstar for s in sows][r_i],
                            [s.slr.cstar for s in sows][r_i],
                            [s.surge_dist. for s in sows][r_i],
                            [s.surge_dist. for s in sows][r_i],
                            [s.surge_dist. for s in sows][r_i],
                            [s.discount_rate for s in sows][r_i],
                            time_frame[t]])
                end
            end
        end
    end
    simulations
end
```

The following are some general statistics of the resulting from the simulations:

2.8 25 year life-span

```
describe(df[df.years_frame .== 25,[1,3,6,7,8,9,11]])
```

	variable	mean	min	median	max	nmissing	eltype
	Symbol	Float64	Float64	Float64	Float64	Int64	DataType
1	npv	-4.69235e6	-2.9013e7	-3.62476e6	-15436.1	0	Float64
2	slr__a	32.4006	-11.4581	32.7133	75.8202	0	Float64
3	slr__tstar	2050.53	2015.02	2050.87	2103.05	0	Float64
4	slr__cstar	20.2251	-14.5416	20.499	34.9954	0	Float64
5	surge__	5.00626	1.82769	5.00637	9.25888	0	Float64
6	surge__	1.51909	0.0555552	1.51772	3.32977	0	Float64
7	discount__rate	0.0524076	0.00025258	0.0518289	0.139236	0	Float64

2.9 50 year life-span

```
describe(df[df.years_frame .== 50,[1,3,6,7,8,9,11]])
```

	variable	mean	min	median	max	nmissing	eltype
	Symbol	Float64	Float64	Float64	Float64	Int64	DataType
1	npv	-5.71229e6	-7.41289e7	-3.77133e6	-615265.0	0	Float64
2	slr_a	32.8267	-17.0328	33.278	66.4874	0	Float64
3	slr_tstar	2050.95	2015.02	2051.1	2101.82	0	Float64
4	slr_cstar	20.5063	-1.61623	20.9171	34.9699	0	Float64
5	surge_	4.99408	1.61044	4.98551	8.49184	0	Float64
6	surge_	1.49318	0.0739012	1.49517	3.22325	0	Float64
7	discount_rate	0.0505592	5.85605e-5	0.0492426	0.126207	0	Float64

2.10 100 year life-span

```
describe(df[df.years_frame .== 100,[1,3,6,7,8,9,11]])
```

	variable	mean	min	median	max	nmissing	eltype
	Symbol	Float64	Float64	Float64	Float64	Int64	DataType
1	npv	-7.59944e6	-1.72541e8	-3.79911e6	-7.92874e5	0	Float64
2	slr_a	32.7464	-1.68561	32.9757	67.375	0	Float64
3	slr_tstar	2050.69	2015.03	2050.2	2102.95	0	Float64
4	slr_cstar	20.8201	-7.37344	21.1071	34.9978	0	Float64
5	surge_	4.94174	1.70614	4.929	8.24849	0	Float64
6	surge_	1.49305	0.215862	1.47805	3.12566	0	Float64
7	discount_rate	0.0506664	0.000515111	0.0493965	0.141629	0	Float64

2.11 Multi-year NPV averages

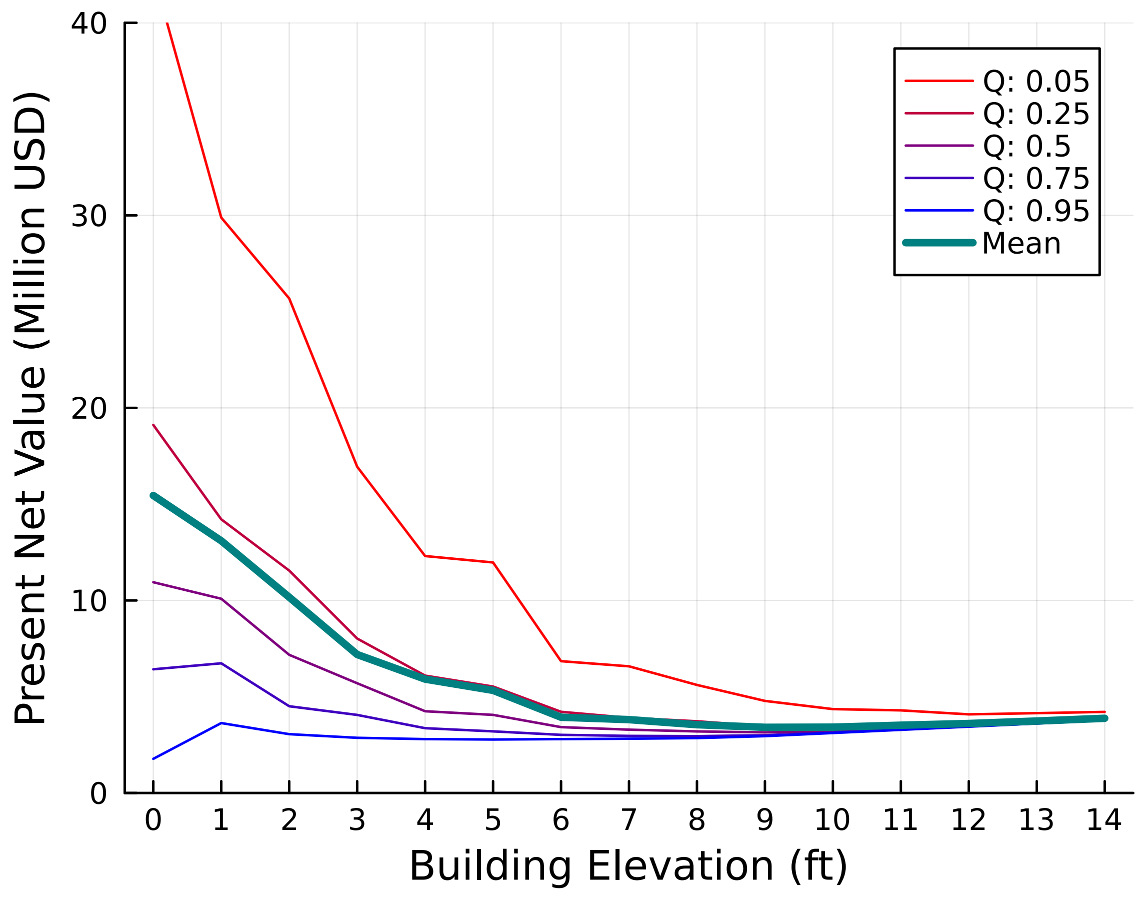
The following are the statistics of NPV results for Actions a : [0,2,4,6,8,10,12,14] ft

```
let
    df_stats = DataFrame(a0 = df.npv[(df.Δh_ft .== 0)]./-1e6,
                         a2 = df.npv[(df.Δh_ft .== 2)]./-1e6,
                         a4 = df.npv[(df.Δh_ft .== 4)]./-1e6,
                         a6 = df.npv[(df.Δh_ft .== 6)]./-1e6,
                         a8 = df.npv[(df.Δh_ft .== 8)]./-1e6,
                         a10 = df.npv[(df.Δh_ft .== 10)]./-1e6,
                         a12 = df.npv[(df.Δh_ft .== 12)]./-1e6,)
    describe(df_stats)
end
```

	variable	mean	min	median	max	nmissing	eltype
	Symbol	Float64	Float64	Float64	Float64	Int64	DataType
1	a0	15.4514	0.0154361	10.9463	168.654	0	Float64
2	a2	10.1668	2.70336	7.17584	172.541	0	Float64
3	a4	5.91409	2.72912	4.24558	117.935	0	Float64
4	a6	3.93333	2.7786	3.42053	19.6506	0	Float64
5	a8	3.54774	2.84932	3.19785	8.32253	0	Float64
6	a10	3.41918	3.11449	3.22788	5.92	0	Float64
7	a12	3.60853	3.44449	3.51953	5.09027	0	Float64

The results can also be visualized for every action and their corresponding statistics. For example, the following is the NPV of the cost for all 25-, 50- and 100-year lifespans. It can be concluded that for the modeled SOW's the large majority shows that elevating the building led to lower NPVs.

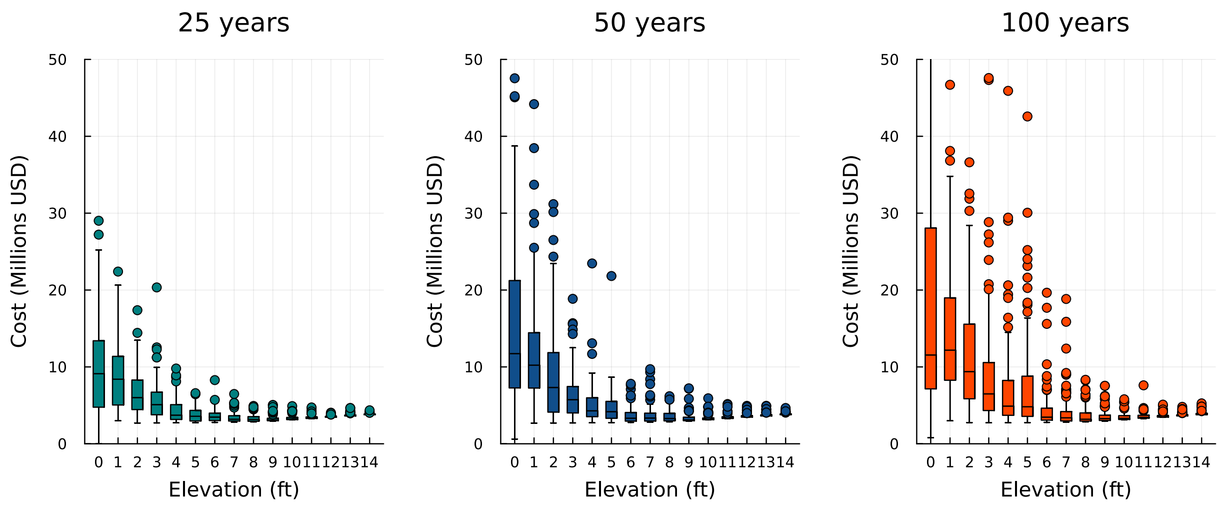
```julia
action_scheme = 0:1:14
let
    p = plot(;
             xlabel = "Building Elevation (ft)",
             ylabel = "Present Net Value (Million USD)",
             ylims = [0,trunc(quantile(-df.npv ./ 1e6,0.99)) + 5],
             xticks = 0:1:14,
             legend = :topright,
             size=(500, 400),)

    quan = [0.05,0.25,0.50,0.75,0.95]
    statistic =  ones(size(action_scheme)[1],1)
    for q in 1:size(quan)[1]
        for e in 1:size(action_scheme)[1]
            statistic[e,1] = quantile(df.npv[(df.Δh_ft .== action_scheme[e])],quan[q])
        end

        plot!(p,
            action_scheme,
            -statistic / 1e6;
            label = "Q: $(quan[q])",
            palette = palette([:red, :blue], 5),)
    end
    for e in 1:size(action_scheme)[1]
            statistic[e,1] = mean(df.npv[(df.Δh_ft .== action_scheme[e])])
    end
    plot!(p,
                action_scheme,
                -statistic / 1e6;
                label = "Mean",
                linewidth = 3,
                color = "teal",)
    p
end
```



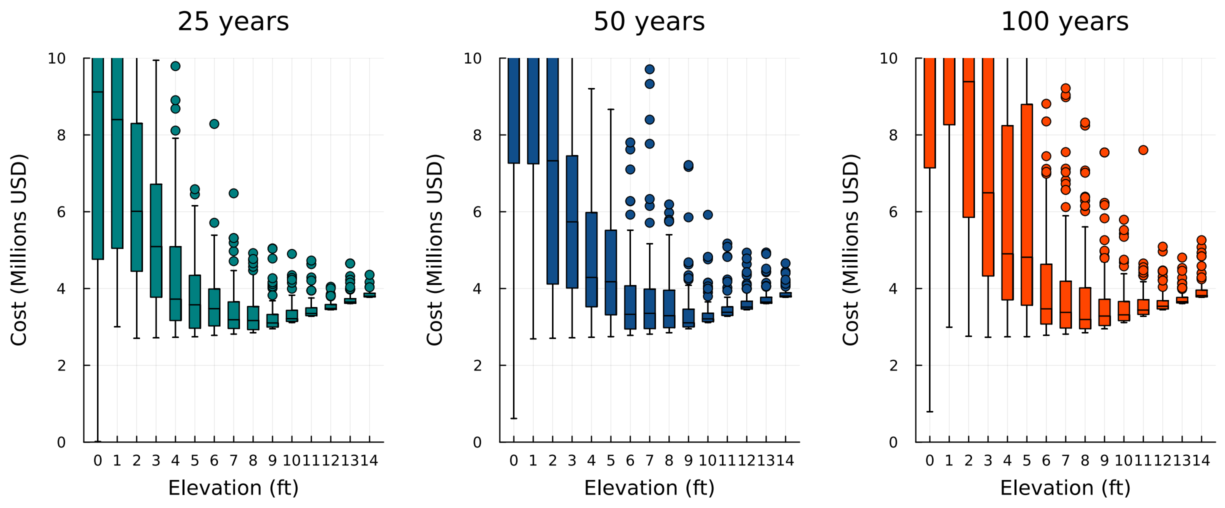
These values can be appreciated using some boxplots for every lifespan. Some conclusions are the following: 1. The uncertainty grows with the lifespan mainly due to the SRL model that contains large uncertainties for longer timeframes. 2. From these graphs, it is also evident that the longer lifespan is considered, the NPV grows. 3. Following the 50% quantile, the is evidence of a convex behavior having a minimum at ~8 ft of elevation

```
let
    y25 = boxplot(;
                xlabel="Elevation (ft)",
                ylabel="Cost (Millions USD)",
                legend = false,
                ylims = [0,50],
                title = "25 years",
                size=(1000, 400),
                )
    for e in 1:size(action_scheme)[1]
    boxplot!(y25,["$(action_scheme[e])"],
            -df.npv[(df.Δh_ft .== action_scheme[e]).*(df.years_frame .== 25)]./1e6,
            color = "teal",
            )
```

```julia
    end
    y50 = boxplot(;
                xlabel="Elevation (ft)",
                ylabel="Cost (Millions USD)",
                legend = false,
                ylims = [0,50],
                title = "50 years",
                )
    for e in 1:size(action_scheme)[1]
    boxplot!(["$(action_scheme[e])"],
            -df.npv[(df.Δh_ft .== action_scheme[e]).*(df.years_frame .== 50)]./1e6,
            color = "dodgerblue4",
            )
    end
    y100 = boxplot(;
                xlabel="Elevation (ft)",
                ylabel="Cost (Millions USD)",
                legend = false,
                ylims = [0,50],
                title = "100 years",
                )
    for e in 1:size(action_scheme)[1]
    boxplot!(["$(action_scheme[e])"],
            -df.npv[(df.Δh_ft .== action_scheme[e]).*(df.years_frame .== 100)]./1e6,
            color = "orangered",
            )
    end
    p = plot(y25,y50,y100, layout = (1,3))
end
```



Detail

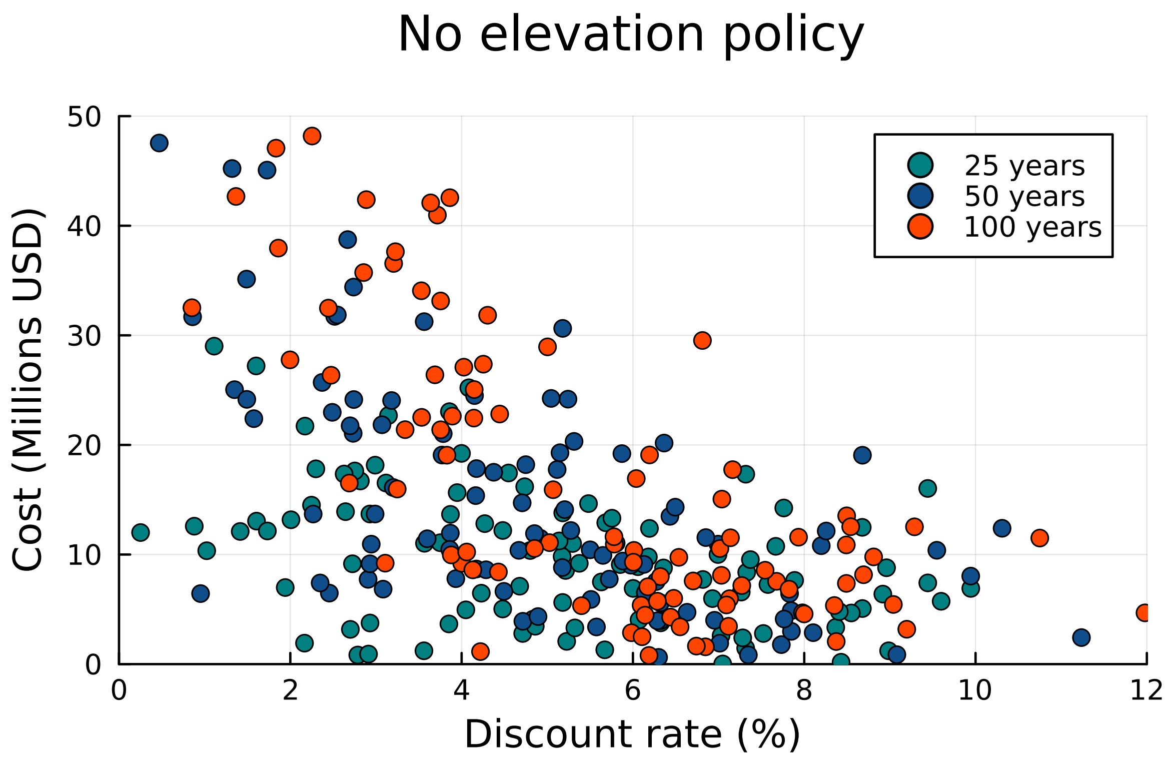


Considering that the highest variability correspond to the action of 0 ft elevation, some analysis can be done to understand the source of these uncertainty and the parameters for which variation the model is more sensible.

2.12 Discount rate

The discount rate for every analyzed lifespan has a negative trend where the NPV decreases as the DR grows.

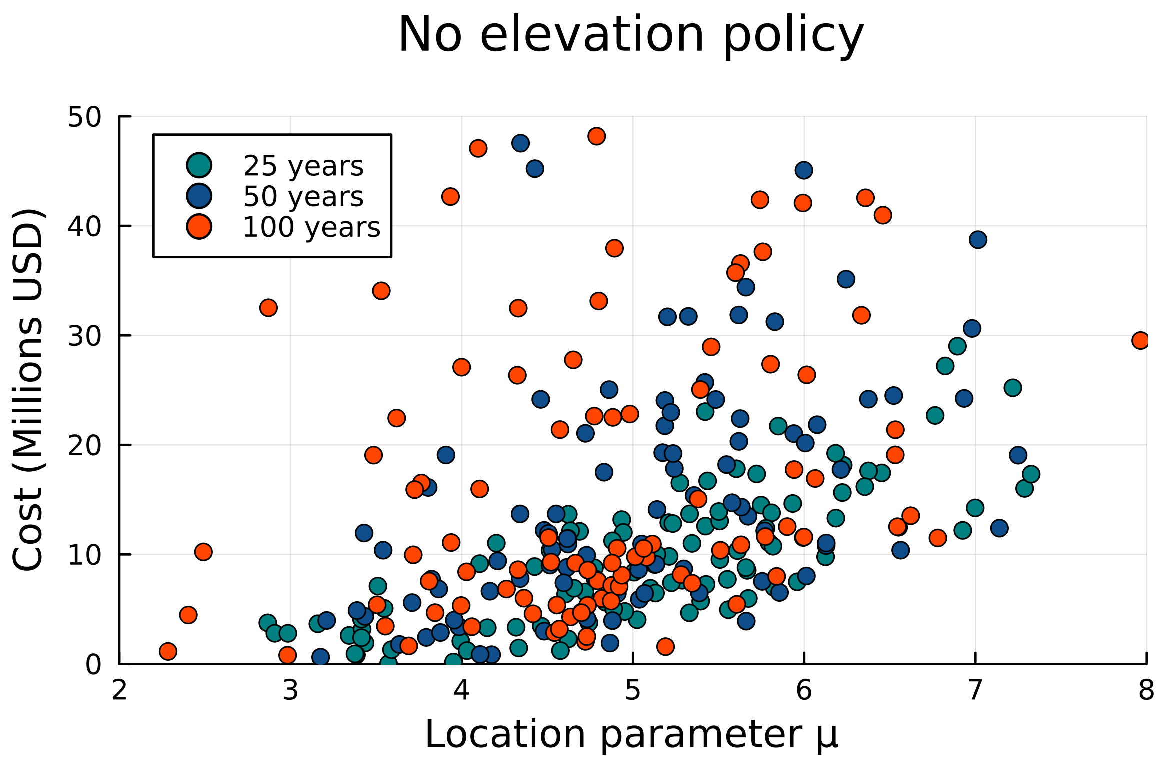
```julia
let
    s = scatter(;
                xlabel="Discount rate (%)",
                ylabel="Cost (Millions USD)",
                ylims = [0,50],
                xlims = [0,12],
                title = "No elevation policy",)
    years = [25,50,100]
    colors = ["teal","dodgerblue4","orangered"]
    for y in 1:3
    scatter!(s,100 .* df.discount_rate[(df.Δh_ft .== 0) .* (df.years_frame .==years[y])],
            -df.npv[(df.Δh_ft .== 0) .* (df.years_frame .==years[y])]/ 1e6;
            label = "$(years[y]) years",
            color = colors[y],
            )
    end
    s
end
```



2.13 Flood frequency

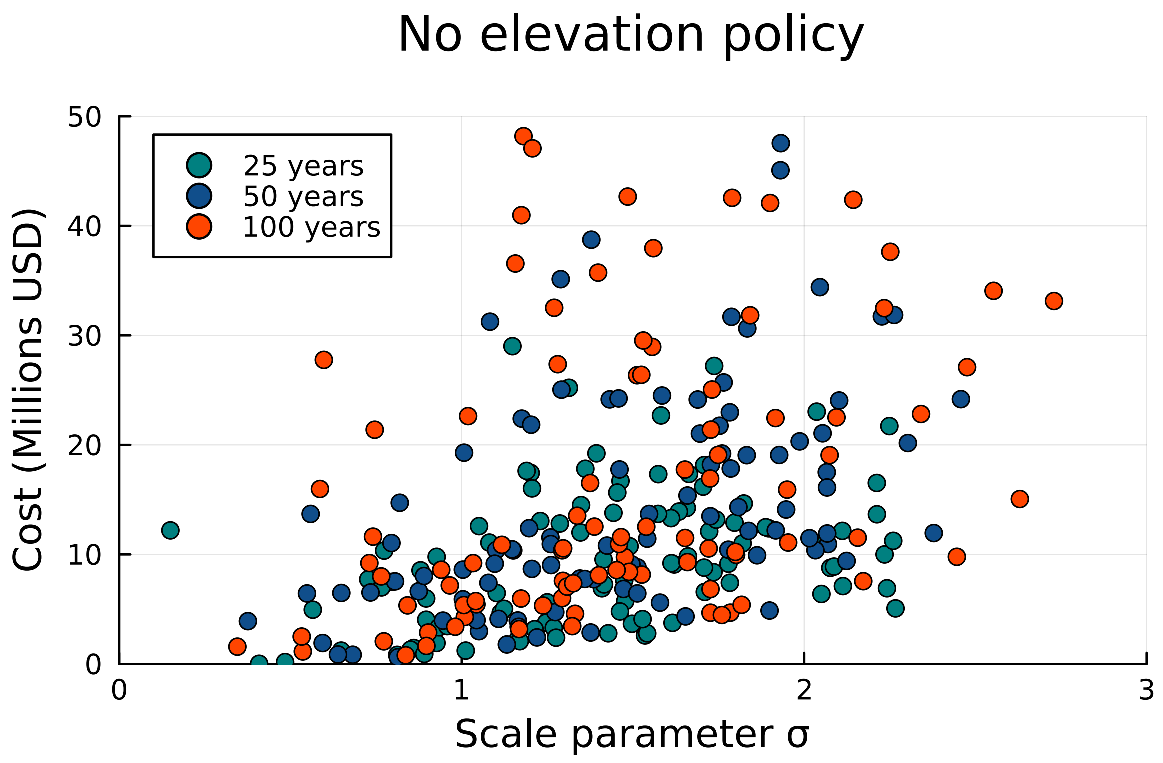
The higher the location parameter of the GEV distribution is the bigger is the NPV. This trend is more evident for shorter lifespans.

```
let
    s = scatter(;
                xlabel="Location parameter  ",
                ylabel="Cost (Millions USD)",
                ylims = [0,50],
                xlims = [2,8],
                title = "No elevation policy",)
    years = [25,50,100]
    colors = ["teal","dodgerblue4","orangered"]
    for y in 1:3
    scatter!(s, df.surge_ [(df.Δh_ft .== 0) .* (df.years_frame .==years[y])],
            -df.npv[(df.Δh_ft .== 0) .* (df.years_frame .==years[y])]/ 1e6;
            label = "$(years[y]) years",
            color = colors[y],
            )
    end
    s
end
```



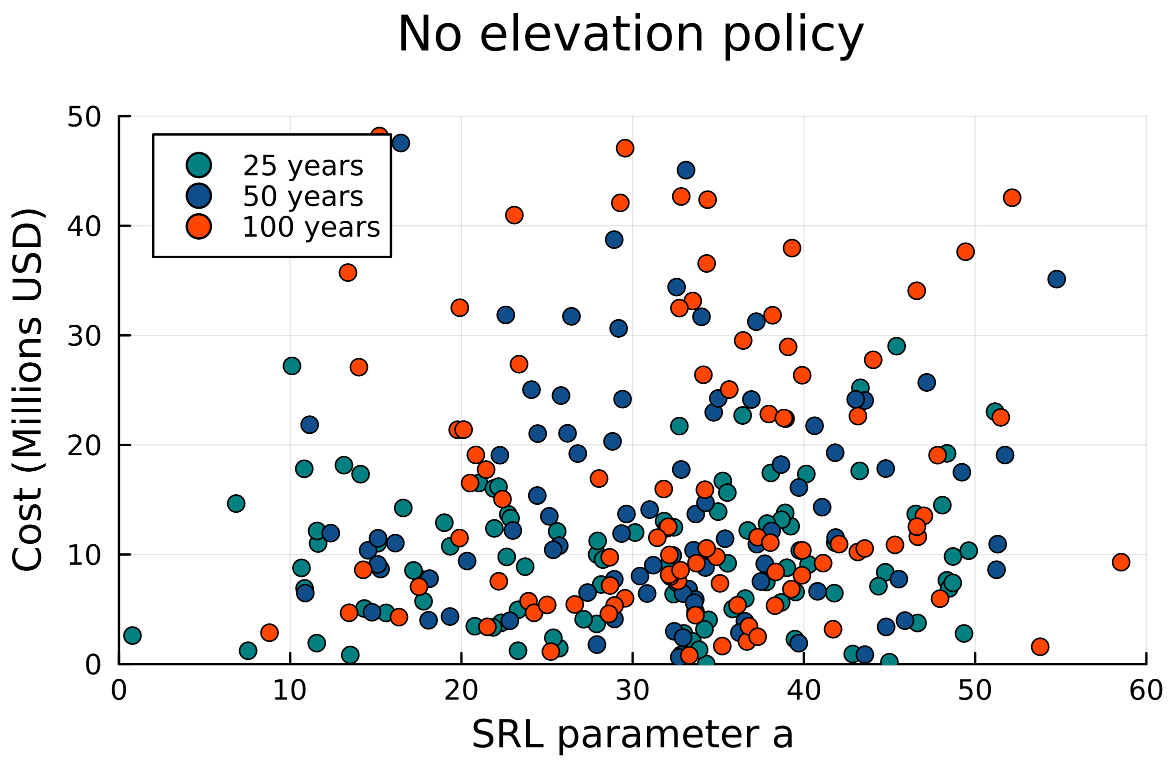
Simillarly, the shape factor , also has a important positive trend.

```
let
    s = scatter(;
                xlabel="Scale parameter  ",
                ylabel="Cost (Millions USD)",
                ylims = [0,50],
                xlims = [0,3],
                title = "No elevation policy",)
    years = [25,50,100]
    colors = ["teal","dodgerblue4","orangered"]
    for y in 1:3
    scatter!(s, df.surge_ [(df.Δh_ft .== 0).* (df.        years_frame .==years[y])],
            -df.npv[(df.Δh_ft .== 0) .* (df.years_frame .==years[y])]/ 1e6;
            label = "$(years[y]) years",
            color = colors[y],
            )
    end
    s
end
```



There is no identifiable trend regarding the SLR model parameters considering perhaps its large variation.

```
let
    s = scatter(;
                xlabel="SRL parameter a",
                ylabel="Cost (Millions USD)",
                ylims = [0,50],
                xlims = [0,60],
                title = "No elevation policy",
                )
    years = [25,50,100]
    colors = ["teal","dodgerblue4","orangered"]
    for y in 1:3
    scatter!(s, df.slr_a[(df.Δh_ft .== 0).* (df.        years_frame .==years[y])],
            -df.npv[(df.Δh_ft .== 0) .* (df.years_frame .==years[y])]/ 1e6;
            label = "$(years[y]) years",
            color = colors[y],
            )
    end
    s
end
```



3 Analysis

When do you get the best results?

As previously identified (past lab), the best overall results is around 8 ft of elevation (no important difference in 7-9 ft interval).

When do you get the worst results? The worst results are concentrated in the 0 and 1 ft elevation actions for all the lifespans considered.

What are the most important parameters? The flood frequency GEV parameters are very important in the modeling as well as the discount rate. As also shown, the time window considered is also a huge source of both uncertainty and NPV difference.

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

I would not run that many simulations considering that the identified trend is already giving good information. The number 1000 or 10000 sounds enough. Similarly, for the present case study the alternative of not-elevating is clearly the one with the worst outcome for any timeframe.

What are the implications of your results for decision-making?

The exploratory modeling shows that for the majority of possible SOW, the building is expected to experience major losses if no action is taken. When considering elevating the building, the prevented losses are more important than the elevation costs. From a financial, cost-benefit analysis there are enough arguments to support the elevation project. Regarding the ammount of elevation, there is enough evidence to elevate the bulding ~ 7 ft to reach cost optimallity.

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.