Final Project Report

Jonah Schaechter (JS336)

Tue., Apr. 30

1 Introduction

1.1 Problem Statement

So far, our simulations have considered the Net Present Value of home elevation under the assumption that homeowners will pay for their home elevation out of pocket. In reality, people have to chose between paying out of pocket (if this is even something they can afford), taking out a loan, or saving up.

Each of these options has tradeoffs in total cost, yearly cost, the cost of damages,and in NPV. In this paper, we will analize these tradeoffs.

1.2 Selected Feature

Describe the feature you have selected to add to the existing decision-support tool. Discuss how this feature relates to the problem statement and its potential to improve climate risk assessment.

The new feature I've chosen to implement is a struct called "Finance". This struct describes the financial conditions of the home, and the plan to elevate it. It includes a loan variable, that describes if the owners will take out a loan to finance the elevation, save up for it, or if they will pay out of pocket. It also contains a loan_rate variable for the interest on potential loans, loan_years, which describes how long homeowners will either save up for or pay out a loan for, paid_off_percent, the amount of the house that's paid off as a decimal, and paid_off_amnt (the total value of the home that's paid off in US dollars), and savings, the amount of money in the homeowners's savings account.

In addition to this new struct, run_sim has been modified to take it into account. So if we chose to save up, our action of elevating the home is delayed until enough money has been saved to do so, and the cost of construction will be increased with the discount rate of that scenario.

If we choose to take out a loan, we will elevate the home immediately, but then pay off the loan annualy according to the pre-determined interest rate, and the discount rate will again be accounted for in the value of these payments.

If we pay out of pocket, then nothing changes, and run_sim operates as it did before, but we can't elevate above what we can afford in our savings account.

If we save up, then we make annual payments until we've saved up enough money to elevate. Whatever's already in the savings account acts as a base we can save on top of.

2 Literature Review

Provide a brief overview of the theoretical background related to your chosen feature. Cite at least two relevant journal articles to support your approach (see Quarto docs for help with citations). Explain how these articles contribute to the justification of your selected feature.

Whether this feature is used to inform policy or the decisions of individual property owners, it has justification in pre-existing literature.

Chenet et. Al (2021) [2] discusses how policy makers ought to take a precautionary approach to addressing the financial impacts of climate change, and that policy "should help justify immediate preventative action". They argue for using scenario analysis and stress testing to evaluate potential policies. To this end, this feature could give policy makers a clearer picture about what they ought to do: give homeowners money to elevate their homes or force them to pay for home elevation, set up loan programs for home elevation, and forcing homeowners to save up for home elevation. This feature can be used to compare these three strategies.

Tools that communicate the propery-level impacts of flooding to homeowners and community leaders are advocated for in [1] . This project can be thought of as an extension of this idea, that the same benefits that can be extracted from communicating the costs and benefits of home elevation can also be extracted from providing more information about how those costs and benefits change under different financial plans.

3 Methodology

3.1 Implementation

```
1  import Pkg
2  Pkg.status()  # Note the paths for your local packages
3  Pkg.gc()  # Garbage collect and delete cached package files
```

Status `C:\Users\Jonah\OneDrive\Documents\Rice PhD\Climate Damage Mitigation\final-project-js3`
  [336ed68f] CSV v0.10.13
  [a93c6f00] DataFrames v1.6.1
  [1313f7d8] DataFramesMeta v0.15.2
  [31c24e10] Distributions v0.25.107
  [49e40289] HouseElevation v0.1.0 `HouseElevation`
  [7073ff75] IJulia v1.24.2
  [b964fa9f] LaTeXStrings v1.3.1
  [bcdb8e00] Metaheuristics v3.3.5
  [b6ca2b7d] ParkingGarage v0.1.0 `ParkingGarage`
  [91a5bcdd] Plots v1.40.2
  [295af30f] Revise v3.5.14
  [2913bbd2] StatsBase v0.34.3
  [1986cc42] Unitful v1.19.0
  [9a3f8284] Random
Info Packages marked with have new versions available and may be upgradable.

      Active manifest files: 10 found
      Active artifact files: 121 found

```
        Active scratchspaces: 2 found
        Deleted no artifacts, repos, packages or scratchspaces
```

```julia
1  using CSV
2  using DataFrames
3  using DataFramesMeta
4  using Distributions
5  using LaTeXStrings
6  using Metaheuristics
7  using Plots
8  using Random
9  using Unitful
10
11 Plots.default(; margin=5Plots.mm)
```

```julia
1  using Revise
2  using HouseElevation
3  #include("Finance.jl")
4  #using house
```

We put in our house in galveston

Choosing Galveston Pier 21, Texas The guage is at 29° 18.6 N, 94° 47.6 W https://maps.app.goo.gl/GyanSMA2fp9rl

Our building is 302 17th St, Galveston, TX 77550, Home area as estimated by google maps: 30ftx50ft home = 1500ft^2 Home value from zillow: 247,700 (Round up to 250,000)

The home is 4.41 feet or 1.34 meters above sea level in elevation. Looking at it on street view, the house appears to be on concrete blocks about 6 inches tall, giving it an effective height of 4.91 feet. Round this up to 5 so that it works.

Row 98 from the data is two-story, no basement in Galveston, so we'll be using that for our depth-damage curve. The home is on concrete blocks, so we can be confident that it doesn't have a basement.

```julia
1  house = let
2      haz_fl_dept = CSV.read("data/haz_fl_dept.csv", DataFrame) # read in the file
3      desc = "Two-story, no basement in Galveston"
4      row = @rsubset(haz_fl_dept, :Column1 == 98)[1, :,] # select the row I want
5      area = 1500u"ft^2"
6      height_above_gauge = height_above_gauge = 5u"ft"
7      House(row; area=area, height_above_gauge=height_above_gauge, value_usd=250_000)
8  end
```

```julia
1  slr_scenarios = let
2      df = CSV.read("data/slr_oddo.csv", DataFrame)
3      [Oddo17SLR(a, b, c, tstar, cstar) for (a, b, c, tstar, cstar) in eachrow(df)]
4  end
5
6  function draw_surge_distribution()
7      = rand(Normal(5, 1))
8      = rand(Exponential(1.25))
```

```julia
 9          = rand(Normal(0.1, 0.05))
10      return GeneralizedExtremeValue( ,  ,  )
11  end
12
13  function draw_discount_rate()
14      return 0.055  #fix interest rate at 5.5%
15      #return rand(Normal(0.05, 0.03))
16  end
17
18  function draw_sow()
19      slr = rand(slr_scenarios)
20      surge_params = draw_surge_distribution()
21      discount = draw_discount_rate()
22      return SOW(slr, surge_params, discount)
23  end
```

Generate our possible states of the world

```julia
1  Random.seed!(421521)
2  N_SOW = 10
3  N_SOW_opt = 50 # to start
4  sows = [draw_sow() for _ in 1:N_SOW]
5  sows_opt = first(sows, N_SOW_opt)
```

```
10-element Vector{SOW{Float64}}:
 SOW{Float64}(Oddo17SLR{Float64}(58.66628415, 2.709557857, 0.003600726, 2061.196955, 24.7855783
 SOW{Float64}(Oddo17SLR{Float64}(33.33247304, 2.22256303, 0.002696663, 2019.563642, 7.701098619
 SOW{Float64}(Oddo17SLR{Float64}(17.79192072, 1.709726426, -0.000614246, 2023.462876, 21.690865
 SOW{Float64}(Oddo17SLR{Float64}(25.13154386, 2.037480231, 0.002113323, 2028.879, 25.92627768)
 SOW{Float64}(Oddo17SLR{Float64}(28.67897009, 2.14841496, 0.002289295, 2046.357094, 34.59089608
 SOW{Float64}(Oddo17SLR{Float64}(40.9256403, 2.858600773, 0.008671168, 2024.544892, 18.75223102
 SOW{Float64}(Oddo17SLR{Float64}(40.57374578, 2.382721317, 0.003969053, 2030.295205, 17.0218615
 SOW{Float64}(Oddo17SLR{Float64}(23.63225858, 2.054085462, 0.002399709, 2044.044368, 13.3610314
 SOW{Float64}(Oddo17SLR{Float64}(27.54342393, 1.669043327, -0.002135507, 2036.034311, 26.046298
 SOW{Float64}(Oddo17SLR{Float64}(26.99654461, 2.09595483, 0.000472662, 2032.836775, 20.6220305
```

Write our function we'll use to evaluate financial plans

3.2 Financial_NPVS function

```julia
1  #assume physical house and SLR scenarios stay constant, only financial conditions change
2  function financial_npvs(finance)
3      p = ModelParams(; house=house, years=2024:2083, finance=finance)  #make our model
4
5      elevations_try = 0:0.5:14  #establish actions to try
6      actions_try = Action.(elevations_try)
7
8
9      if p.finance.loan == 1 #if taking out a loan
```

```
10          #Get all the contruction costs we can afford through equity
11          construction_costs = [elevation_cost(house, elevation) for elevation in elevations_try
12      elseif p.finance.loan == 0 #if paying out of pocket
13          #must have enough in savings to pay
14          construction_costs = [elevation_cost(house, elevation) for elevation in elevations_try
15      else # if we're saving up, we can try all possible outcomes
16          construction_costs = [elevation_cost(house, elevation) for elevation in elevations_try]
17      end
18
19
20      actions_try = actions_try[1:length(construction_costs)]  #get the list of actions we can a
21      elevations_try = elevations_try[1:length(construction_costs)]  #limit elevations as well s
22
23      #Run simulations
24      npvs_opt = [mean([run_sim(a, sow, p) for sow in sows_opt]) for a in actions_try]
25
26      #get height that minimizes NPV
27      min_npv, min_idx = findmax(npvs_opt)
28      minimizer = elevations_try[min_idx]
29
30      #return elevations we tried (x), npvs(y) and minimizing elevation
31      return elevations_try, npvs_opt, minimizer
32
33  end
```

financial_npvs (generic function with 1 method)

Basic financing, just paying up fronts

println(methods(Finance)) println(fieldtypes(Finance)) ## Finance_Basic

```
1   finance_basic = let
2       loan = 0
3       loan_years = 0
4       loan_rate = 0.0
5       paid_off_percent = 1.0
6       amnt_paid_off = paid_off_percent * house.value_usd  # Calculate amnt_paid_off
7       savings = 250_000
8
9       # Create Finance object using keyword arguments
10      Finance(;
11          loan = loan,
12          loan_years = loan_years,
13          loan_rate = loan_rate,
14          paid_off_percent = paid_off_percent,
15          amnt_paid_off = amnt_paid_off,
16          savings = savings
17      )
18  end
```

```
Finance(0, 0, 0.0, 1.0, 250000.0, 250000.0)
```

Now lets calculate our basic finance options

```
1  elevations_tried, npvs, min = financial_npvs(finance_basic)
```

```
(0.0:0.5:14.0, [-994136.2797665314, -1.0039462753954036e6, -865807.0880606174, -728807.7632033
```

3.3 plot_many

Define our function to plot many financial scenarios

```julia
1  function plot_many(finance_list, initial_plot)
2
3
4     for finance in finance_list
5
6         #println(finance)
7         elevations_tried, npvs, min = financial_npvs(finance)
8         if finance.loan == 1  #if taking out a loan
9             label = "$(round(finance.loan_rate*100))% loan over $(finance.loan_years) years"
10        elseif finance.loan > 1
11            label = "Saving over $(finance.loan_years) years"
12        else
13            label = "Paying out of pocket"
14        end
15        plot!(
16        initial_plot,
17        elevations_tried,
18        npvs ./ 1000;
19        xlabel="Elevation [ft]",
20        ylabel="NPV [1000 USD]",
21        label=label,
22        marker=:circle,
23        #color=line_color
24        )
25
26         #make the vertical line color the same as the horizontal line
27        line_color = initial_plot.series_list[end][:linecolor]
28        #line_color = plot_object.series_list[end].plotseries[:linecolor]
29        vline!([min]; label="$(min)ft elevation", linestyle=:dash, color=line_color)
30     end
31  end
```

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

3.4 reasonable saving time

Generate many financial scenarios where we save up for different periods of time

```
1   finance_saving = []
2
3   for i in [2, 3, 5, 7]
4       #print(i)
5       paid_off_percent = 1.0
6       savings = 180_000
7       fin = Finance(; loan=2, loan_years=i, loan_rate=0.0, paid_off_percent=paid_off_percent,
8           amnt_paid_off=(house.value_usd*paid_off_percent), savings=savings
9           )
10      push!(finance_saving, fin)
11  end
12
13  println(finance_saving)
```

```
Any[Finance(2, 2, 0.0, 1.0, 250000.0, 180000.0), Finance(2, 3, 0.0, 1.0, 250000.0, 180000.0),
```

Plot our saving options

```
1   p = plot(
2       elevations_tried,
3       npvs ./ 1000;
4       xlabel="Elevation [ft]",
5       ylabel="NPV [1000 USD]",
6       title="Out of Pocket Vs Various saving periods",
7       label="Out of pocket",
8       marker=:circle#, color=line_color
9   )
10
11
12  line_color = p.series_list[end][:linecolor]  #make the vertical line color the same as the hor
13  vline!([min]; label="$(min)ft elevation", linestyle=:dash, color=line_color)
14
15  plot_many(finance_saving, p)
16  display(p)
```

Out of Pocket Vs Various saving periods
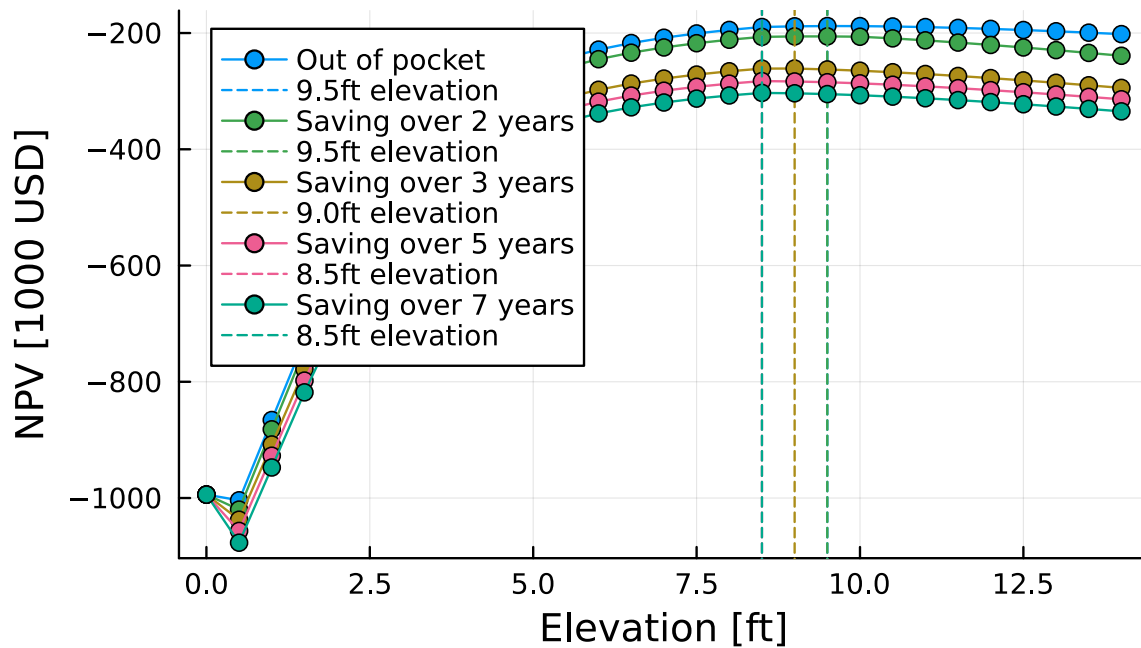
Results of this graph show two interesting things:

1. NPV of saving is always a lot lower than paying up front. This makes sense, since we're increasing construction cost with inflation, so the only change to the NPV will be more damages while we wait to elevate the house.

2. NPV results recommend lower elevations if we chose to wait. This seems strange at first, since most people might save money so they can elevate higher, and you'd think that if we do wait to elevate, it'd make more sense to do a higher elevation to compensate for the damages we experienced in the first few years.

But since we are doing this simulation over a fixed period of time, the longer we wait to elevate, the less value is gained by elevating, since we'll have spent less of our time avoiding damages. So assuming a non-infinite time duration for this experiment, the longer we wait to elevate, the lower the recommended elevation height gets.

We can demonstrate this more clearly by displaying some extremely long saving periods.

3.5 Long saving time

```
1  Long_Savings = []
2
3  for i in [10, 20, 30, 50]
4      #print(i)
5      paid_off_percent = 1.0
6      savings=180_000
7      fin = Finance(; loan=2, loan_years=i, loan_rate=0.0, paid_off_percent=paid_off_percent,
```

```
8            amnt_paid_off=(house.value_usd*paid_off_percent),
9            savings=180_000
10           )
11      push!(Long_Savings, fin)
12   end
13
14   println(Long_Savings)
```

```
Any[Finance(2, 10, 0.0, 1.0, 250000.0, 180000.0), Finance(2, 20, 0.0, 1.0, 250000.0, 180000.0)
```

```
1   p = plot(
2       elevations_tried,
3       npvs ./ 1000;
4       xlabel="Elevation [ft]",
5       ylabel="NPV [1000 USD]",
6       title="Out of Pocket Vs Various saving periods",
7       label="Out of pocket",
8       marker=:circle#, color=line_color
9   )
10
11
12   line_color = p.series_list[end][:linecolor]  #make the vertical line color the same as the hori
13   vline!([min]; label="$(min)ft elevation", linestyle=:dash, color=line_color)
14
15   plot_many(Long_Savings, p)
16   display(p)
```

Out of Pocket Vs Various saving periods
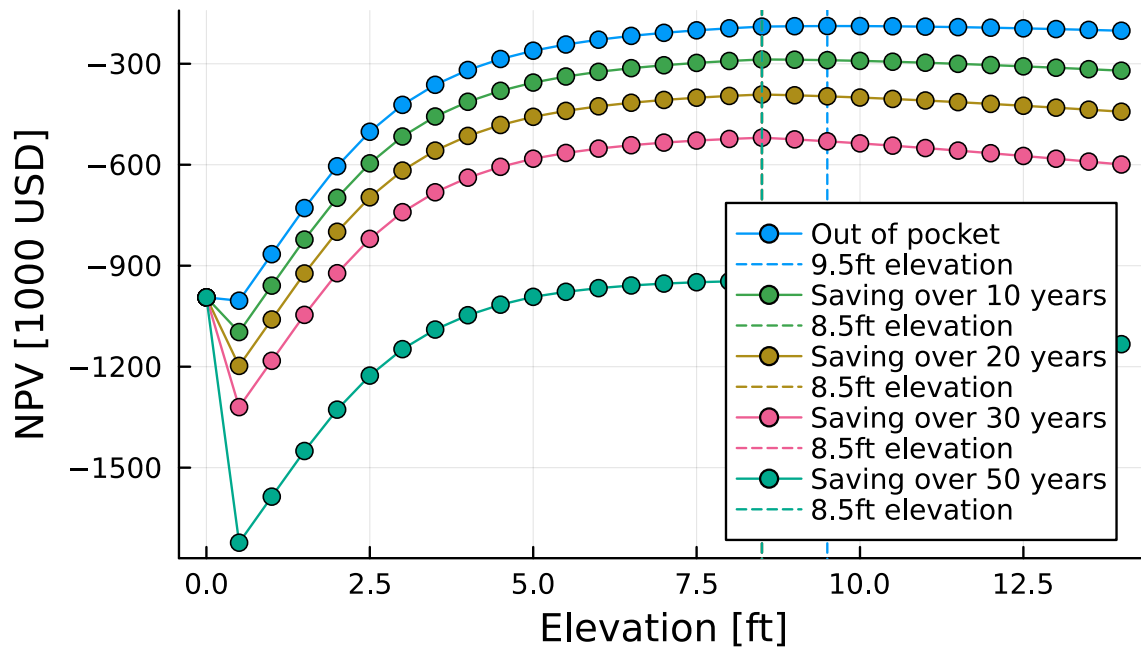
3.6 Different loan durations

Generate many financial scenarios with different years to pay off the loan at a 7% interest rate

```
finance_list_years = []

for i in [3, 5, 7, 10]
    #print(i)
    paid_off_percent = 0.7
    savings=180_000
    fin = Finance(; loan=1, loan_years=i, loan_rate=0.07, paid_off_percent=paid_off_percent,
        amnt_paid_off=(house.value_usd*paid_off_percent), savings=savings
        )
    push!(finance_list_years, fin)
end

println(finance_list_years)
```

Any[Finance(1, 3, 0.07, 0.7, 175000.0, 180000.0), Finance(1, 5, 0.07, 0.7, 175000.0, 180000.0)

Plot our scenarios of different years to pay off the loan

```
p = plot(
    elevations_tried,
    npvs ./ 1000;
    xlabel="Elevation [ft]",
```

```
5    ylabel="NPV [1000 USD]",
6    title="Out of Pocket Vs Various payment periods",
7    label="First $(N_SOW_opt) SOWs, no loan",
8    marker=:circle#, color=line_color
9  )
10
11
12 line_color = p.series_list[end][:linecolor]  #make the vertical line color the same as the hor:
13 vline!([min]; label="$(min)ft elevation", linestyle=:dash, color=line_color)
14
15 plot_many(finance_list_years, p)
16 display(p)
```

Out of Pocket Vs Various payment periods
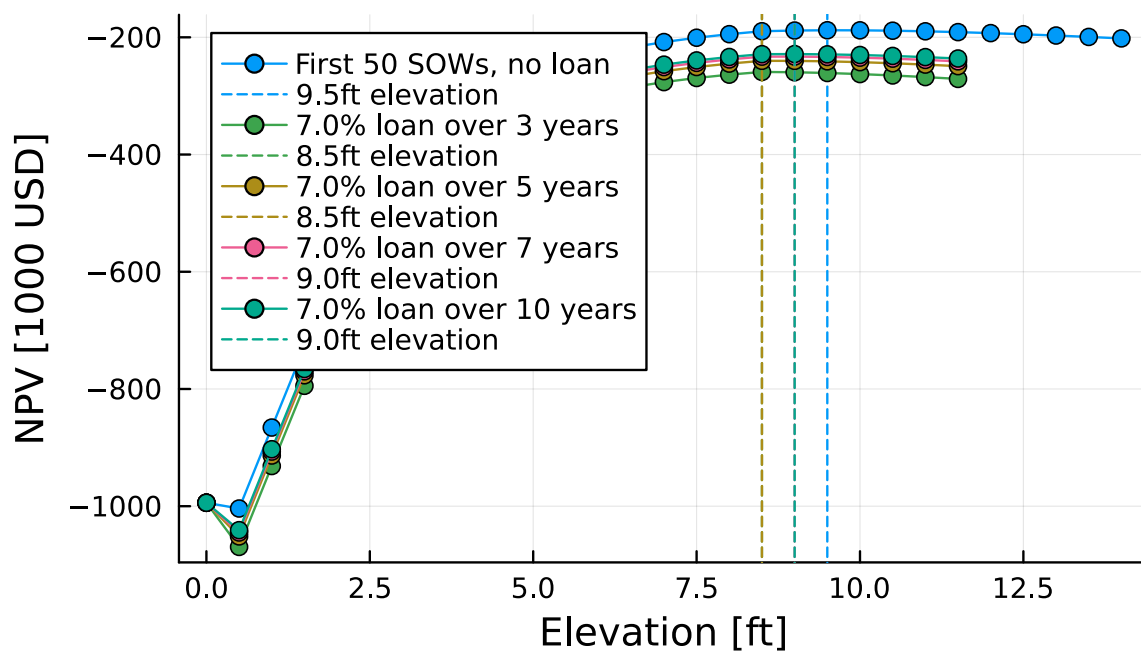
3.7 Different interest rates

Generate many financial scenarios with different interest rates

```
1  finance_list_rates = []
2
3  for i in range(start=3, stop=9, length=4)
4      #print(i)
5      paid_off_percent = 0.7
6      savings = 180_000
7      fin = Finance(; loan=1, loan_years=5, loan_rate=i/100, paid_off_percent=paid_off_percent,
8          amnt_paid_off=(house.value_usd*paid_off_percent), savings=savings
```

```
9              )
10       push!(finance_list_rates, fin)
11   end
12
13   println(finance_list_rates)
```

```
Any[Finance(1, 5, 0.03, 0.7, 175000.0, 180000.0), Finance(1, 5, 0.05, 0.7, 175000.0, 180000.0)
```

```
1    #colors = [colorant"red", colorant"green", colorant"blue", colorant"orange", colorant"purple"]
2
3    #line_color = colors[1]
4
5    p = plot(
6        elevations_tried,
7        npvs ./ 1000;
8        xlabel="Elevation [ft]",
9        ylabel="NPV [1000 USD]",
10       title="Out of Pocket Vs Various 5 year loans, 70% paid off",
11       label="First $(N_SOW_opt) SOWs, no loan",
12       marker=:circle#, color=line_color
13   )
14   #println(p.series_list[end][:linecolor] )
15
16   #colors = Plots.palette()
17
18   line_color = p.series_list[end][:linecolor]  #make the vertical line color the same as the hor
19   vline!([min]; label="$(min)ft elevation", linestyle=:dash, color=line_color)
20
21   plot_many(finance_list_rates, p)
22   display(p)
```

Out of Pocket Vs Various 5 year loans, 70% paid
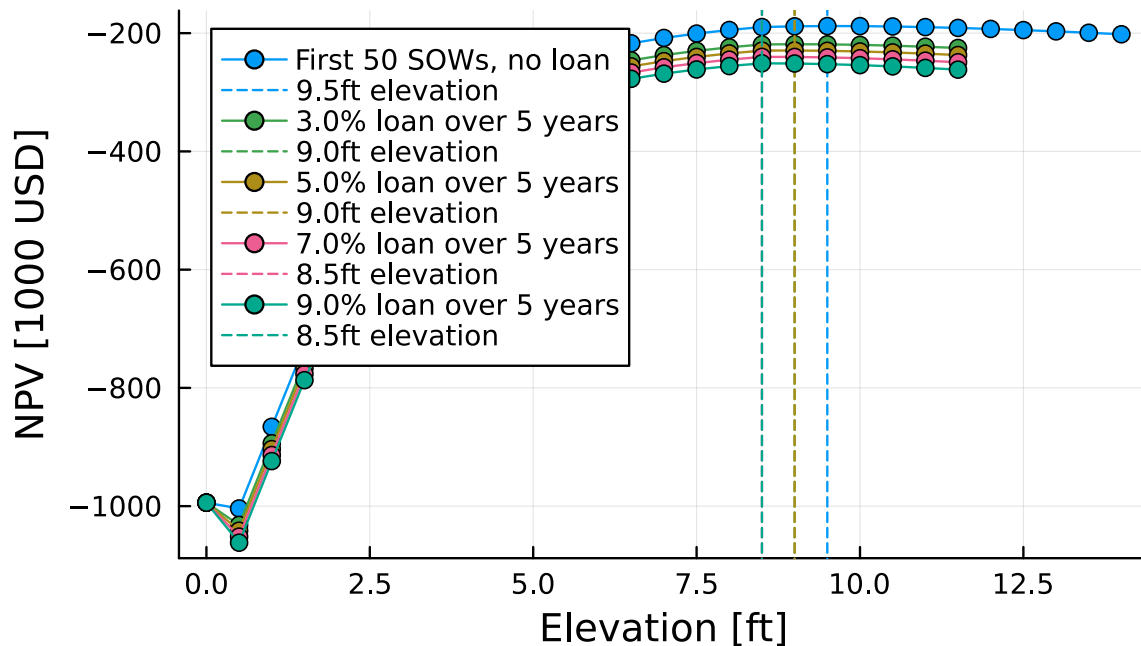
3.8 Compare NPVS and annual costs

Now we want to compare NPVS and annual costs for all of these scenarios

Make a function to get NPV, annual cost, and optimal elevatin

```
function annual_loan_cost(p, r, n)
    #P is principle amount, r is rate, n is number of years
    a = p * ( r * ((1+r)^n)  / ( (1+r)^n - 1 ) )
    return a #a is annual payments
end


function annual_v_npv(finance)

    elevations_tried, npvs, best_elevation = financial_npvs(finance)
    #An engineer is not someone who makes things maximally efficient, but someone who makes thi
    best_npv, best_idx = findmax(npvs)

    construction_cost = elevation_cost(house, best_elevation)
    #println(elevation_cost(house, best_elevation))

    if finance.loan  == 0 #out of pocket
        annual_cost = construction_cost
    elseif finance.loan == 1  #taking out a loan
        annual_cost = annual_loan_cost(construction_cost, finance.loan_rate, finance.loan_years
```

```
21        else #saving up
22            annual_cost = construction_cost/finance.loan_years
23        end
24
25        return best_npv, annual_cost, best_elevation
26
27    end
```

annual_v_npv (generic function with 1 method)

```
1   function get_costs(finances)
2       figures = annual_v_npv.(finances)
3       best_npvs = [fig[1] for fig in figures]
4       annual_costs = [fig[2] for fig in figures]
5       best_elevations = [fig[3] for fig in figures]
6
7       return best_npvs, annual_costs, best_elevations
8   end
9
10
11  function plot_money_values(finance, initial_plot, label)
12
13      best_npvs, annual_costs, best_elevations = get_costs(finance)
14      plot!(
15          initial_plot,
16          best_npvs ./ 1000,
17          annual_costs ./ 1000;
18          xlabel="Optimized NPV [1000 USD]",
19          ylabel="Annual Costs [1000 USD]",
20          label=label,
21          marker=:circle,
22          #color=line_color
23          )
24
25  end
```

plot_money_values (generic function with 1 method)

get_costs(finance_list_rates)

```
1   optimal_npv, annual_cost, best_elevation = annual_v_npv(finance_basic)
2
3   Annual_v_npv_plot = plot(
4       [optimal_npv] ./ 1000,
5       [annual_cost] ./ 1000 ;
6       xlabel="Optimized NPV [1000 USD]",
7       ylabel="Annual Costs [1000 USD]",
8       title="NPV vs Yearly Costs",
9       label="Out of Pocket",
```
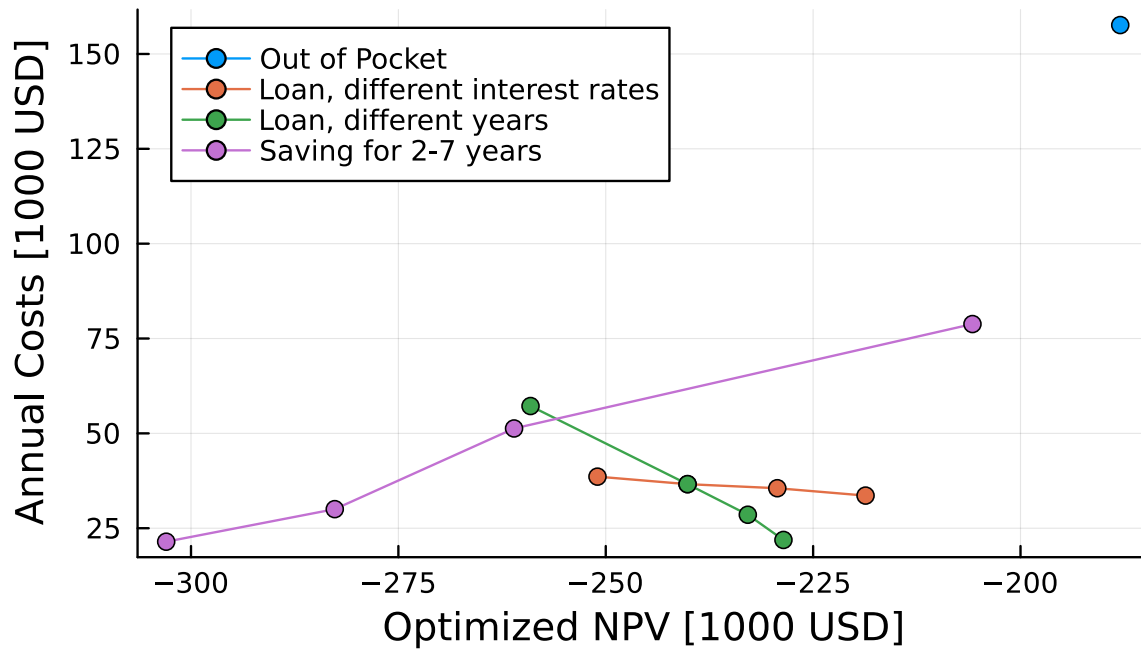
```
10      marker=:circle#, color=line_color
11  )
12  plot_money_values(finance_list_rates, Annual_v_npv_plot, "Loan, different interest rates")
13  plot_money_values(finance_list_years, Annual_v_npv_plot, "Loan, different years")
14  plot_money_values(finance_saving, Annual_v_npv_plot, "Saving for 2-7 years")
15  display(Annual_v_npv_plot)
```
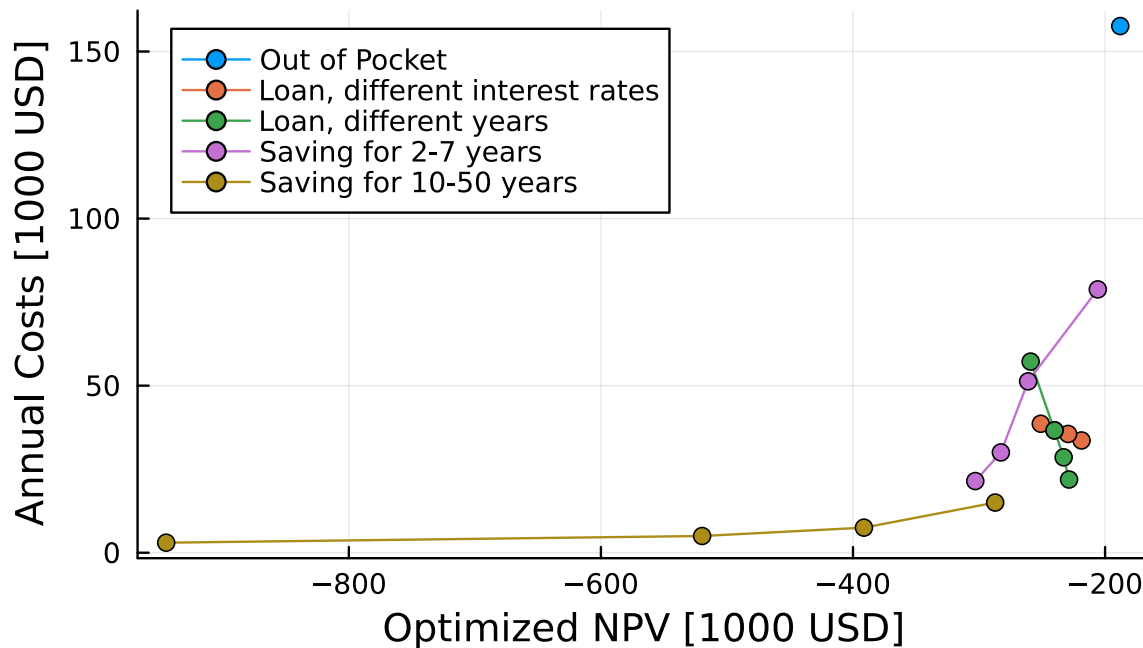
NPV vs Yearly Costs

The chart changes radically when we display long savings times

```
1  plot_money_values(Long_Savings, Annual_v_npv_plot, "Saving for 10-50 years")
2  display(Annual_v_npv_plot)
```

NPV vs Yearly Costs

3.9 compare NPVS and construction heights for all these different scenarios

```
1   function general_plot(x, y, finance, initial_plot, label)
2       # 1 = best_npvs, 2 = annual_costs, 3 = best_elevation
3       values = collect(get_costs(finance)) #get our values to plot.
4       #Using collect makes sure these values are in an array so we can use ./1000 later
5       #println(values)
6       #println(values[2])
7       values[1] = values[1] ./ 1000  #adjust npv and annual costs
8       values[2] = values[2] ./1000
9
10      plot!(
11          initial_plot,
12          values[x],  #decide what to plot
13          values[y];
14          label=label,
15          marker=:circle,
16          #color=line_color
17          )
18  end
```

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

```
1   height_v_npv_plot = plot(
2       [optimal_npv] ./ 1000,
```
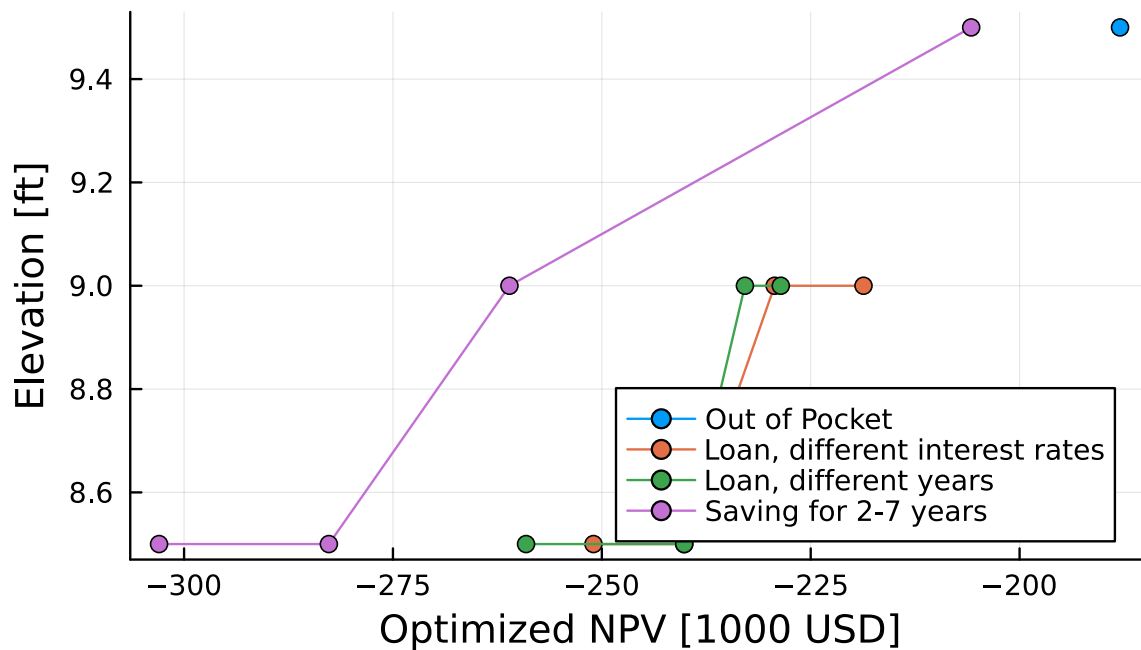
```
3       [best_elevation] ;
4       xlabel="Optimized NPV [1000 USD]",
5       ylabel="Elevation [ft]",
6       title="NPV vs Elevation",
7       label="Out of Pocket",
8       marker=:circle,#, color=line_color
9       legend=:bottomright
10  )
11
12  general_plot(1, 3, finance_list_rates, height_v_npv_plot, "Loan, different interest rates")
13  general_plot(1, 3, finance_list_years, height_v_npv_plot, "Loan, different years")
14  general_plot(1, 3, finance_saving, height_v_npv_plot, "Saving for 2-7 years")
15  display(height_v_npv_plot)
```


3.10   compare annual costs and construction heights

```
1   height_v_npv_plot = plot(
2       [annual_cost] ./ 1000,
3       [best_elevation] ;
4       xlabel="Annual Cost [1000 USD]",
5       ylabel="Elevation [ft]",
6       title="Annual Cost vs Elevation",
7       label="Out of Pocket",
8       marker=:circle,#, color=line_color
```
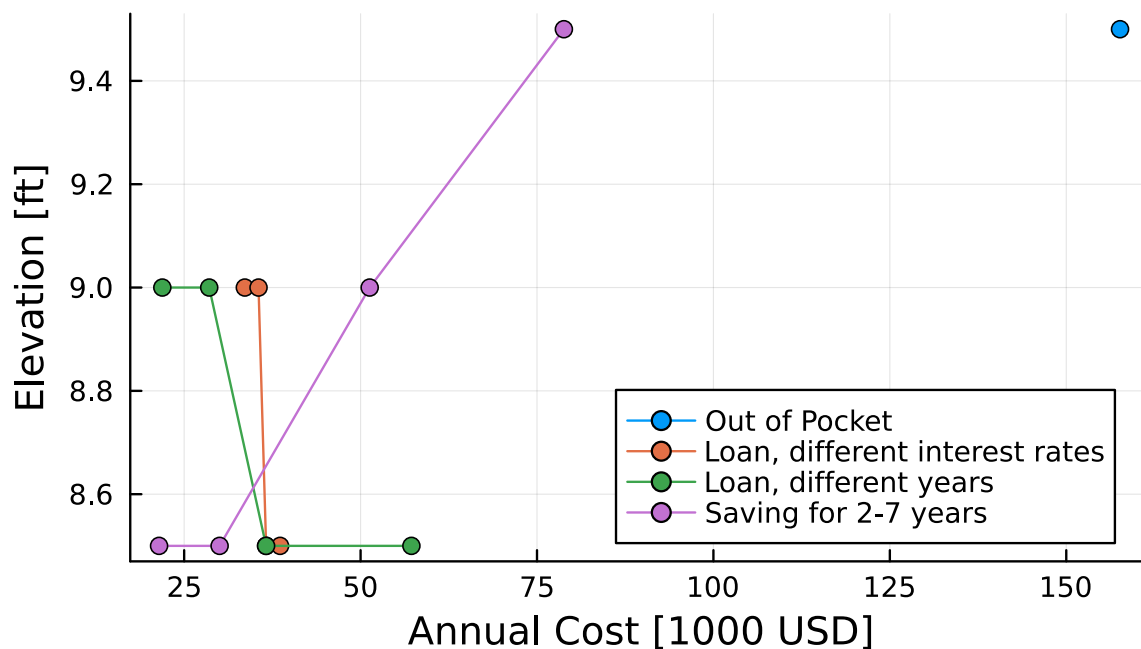
```
 9       legend=:bottomright
10   )
11
12   general_plot(2, 3, finance_list_rates, height_v_npv_plot, "Loan, different interest rates")
13   general_plot(2, 3, finance_list_years, height_v_npv_plot, "Loan, different years")
14   general_plot(2, 3, finance_saving, height_v_npv_plot, "Saving for 2-7 years")
15   display(height_v_npv_plot)
```

Annual Cost vs Elevation

3.11 Low and high savings and equity (LHSE) homes

Now that we're all set up, lets look at scenarios where we have high and low savings and high and low equity (LHSE).

We'll assume that market loans are run at 7% interest and not consider other interest rates, except for govenrment loans for people with low equity and low savings at 7.5%.

Make a function that generates the finance outcomes for these different plans given a set equity and savings:

```
1   function gen_finances(equity, savings, house)
2
3       loan_finances = []  #7.0% bank loan
4       gov_loan = []   #7.5% government loan for those in need
5       saving_finances = []
6
7       for i in [2, 3, 5, 7]   #saving scenarios
```

```
8          fin = Finance(; loan=2, loan_years=i, loan_rate=0.0, paid_off_percent=equity,
9              amnt_paid_off=(house.value_usd*equity), savings=savings
10             )
11         push!(saving_finances, fin)
12     end
13
14     for i in [3, 5, 7, 10]  #bank loan scenarios
15         fin = Finance(; loan=1, loan_years=i, loan_rate=0.07, paid_off_percent=equity,
16         amnt_paid_off=(house.value_usd*equity), savings=savings
17         )
18         push!(loan_finances, fin)
19     end
20
21     for i in [3, 5, 7, 10]  #government loan scenarios
22         fin = Finance(; loan=1, loan_years=i, loan_rate=0.075, paid_off_percent=1.0,  #gov loar
23         amnt_paid_off=(house.value_usd*1.0), savings=savings
24         )
25         push!(gov_loan, fin)
26     end
27
28     # Our of pocket elevation
29     OOP = Finance(;
30         loan = 0,
31         loan_years = 0,
32         loan_rate = 0.0,
33         paid_off_percent = equity,
34         amnt_paid_off = (house.value_usd*equity),
35         savings = savings
36     )
37
38     return OOP, saving_finances, loan_finances, gov_loan
39
40 end
```

gen_finances (generic function with 1 method)

Now make a function that plots the outcomes for all of these scenarios:

```
1 function plot_LHSE_scenario(LHSE_options, title)
2
3     OOP, saving_finances, loan_finances, gov_loan = LHSE_options
4
5     optimal_npv, annual_cost, best_elevation = annual_v_npv(OOP)
6
7     Annual_v_npv_plot = plot(
8         [optimal_npv] ./ 1000,
9         [annual_cost] ./ 1000 ;
10        xlabel="Optimized NPV [1000 USD]",
```

```
11          ylabel="Annual Costs [1000 USD]",
12          title=title,
13          label="Out of Pocket",
14          marker=:circle#, color=line_color
15      )
16      plot_money_values(gov_loan, Annual_v_npv_plot, "Government Loan")
17      plot_money_values(loan_finances, Annual_v_npv_plot, "Bank Loan")
18      plot_money_values(saving_finances, Annual_v_npv_plot, "Saving for 2-7 years")
19      display(Annual_v_npv_plot)
20
21  end
```

plot_LHSE_scenario (generic function with 1 method)

```
1  elevation_cost(house, 9.5)
```

157620.0

Note that the cost to elevate our house 9.5 feet, what was previously considered optimal, costs more than 157,000 US dollars, putting it even out of reach for out of pocket payment by our high-savings individuals.

Only the wealthiest 10% of americsn have more than 111,000 USD in their bank accounts, so our high-savings individuals with more than 120k are doing fairly well, but still can't afford the "optimal" elevation.

https://www.fool.com/the-ascent/banks/articles/heres-how-much-the-average-rich-person-has-in-the-bank/

```
1  #high equity, high savings
2  high_equity = 0.8
3  high_savings = 120_000
4
5  low_equity = 0.2
6  low_savings = 10_000
7
8  ultra_high_savings = 150_000
9  #10k is actally still more than 60% of americans according to the motley fool, which is... sad
```
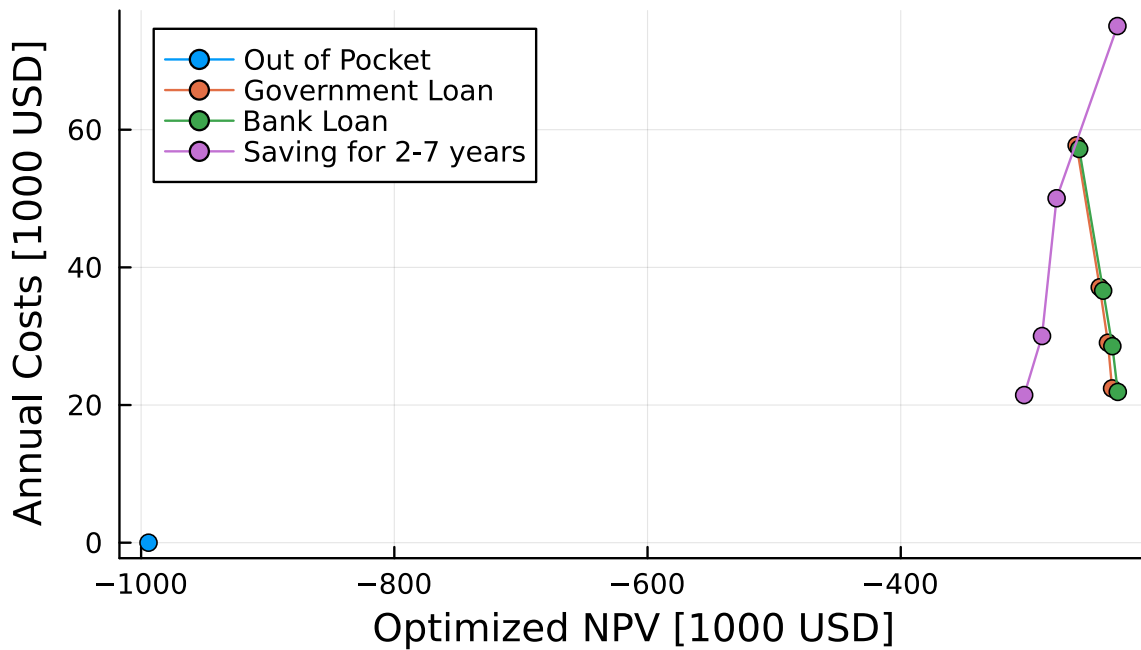
150000

```
1  HEHS = gen_finances(high_equity, high_savings, house)
2  plot_LHSE_scenario(HEHS, "High Equity ($(high_equity*100)%), High Savings ($(high_savings/1000)
```
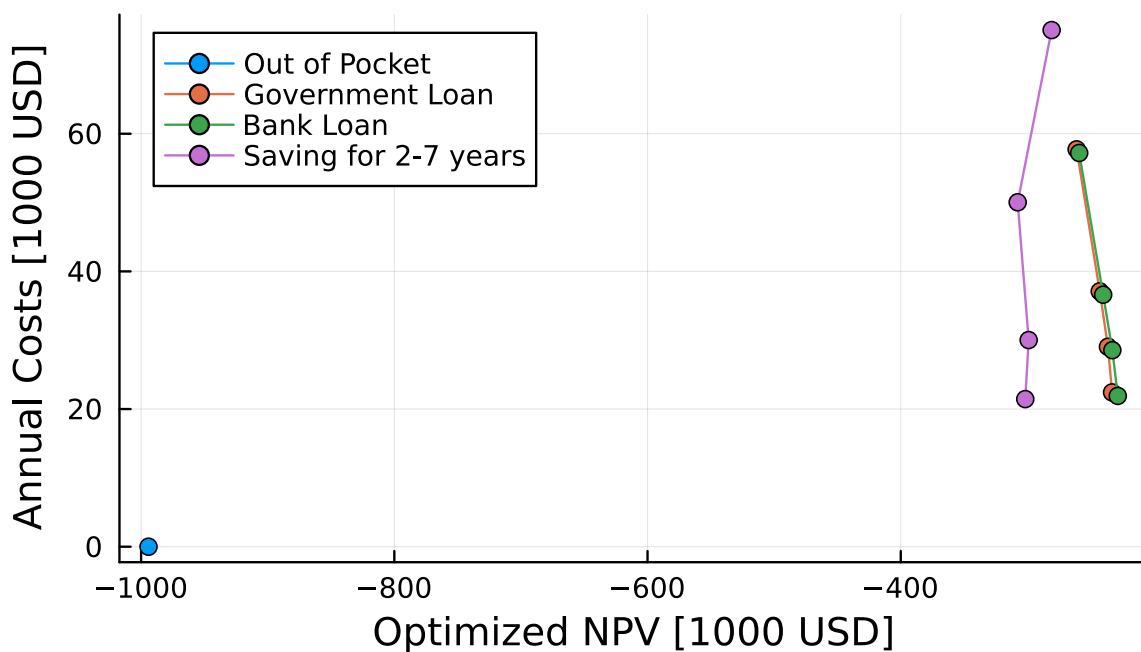
High Equity (80.0%), High Savings (120.0K)


```
1  HELS = gen_finances(high_equity, low_savings, house)
2  plot_LHSE_scenario(HELS, "High Equity ($(high_equity*100)%), Low Savings ($(low_savings/1000)K)
```

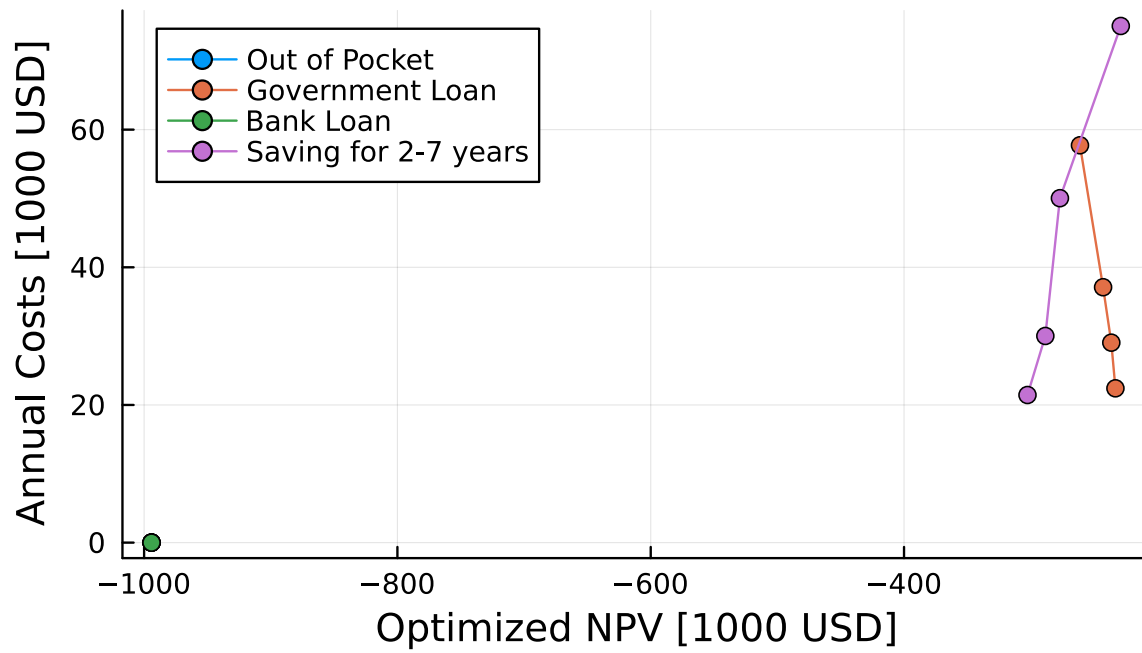# High Equity (80.0%), Low Savings (10.0K)

```
1  LEHS = gen_finances(low_equity, high_savings, house)
2  plot_LHSE_scenario(LEHS, "Low Equity ($(low_equity*100)%), High Savings ($(high_savings/1000)K)
```
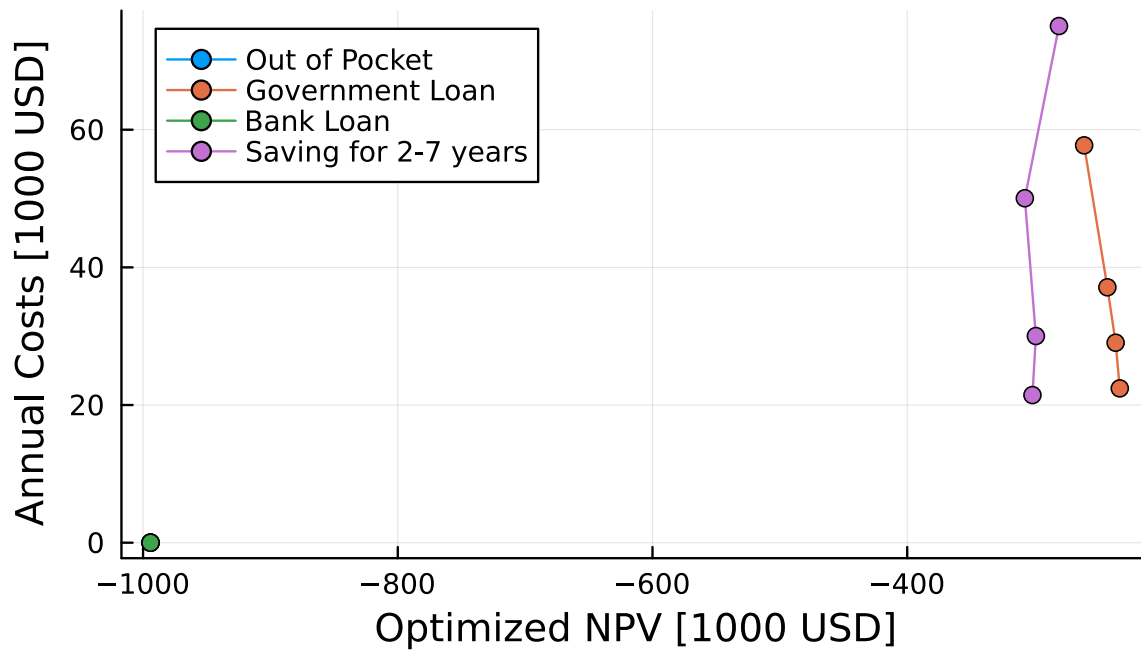

Low Equity (20.0%), High Savings (120.0K)

```
1  LELS = gen_finances(low_equity, low_savings, house)
2  plot_LHSE_scenario(LELS, "Low Equity ($(low_equity*100)%), Low Savings ($(low_savings/1000)K)")
```
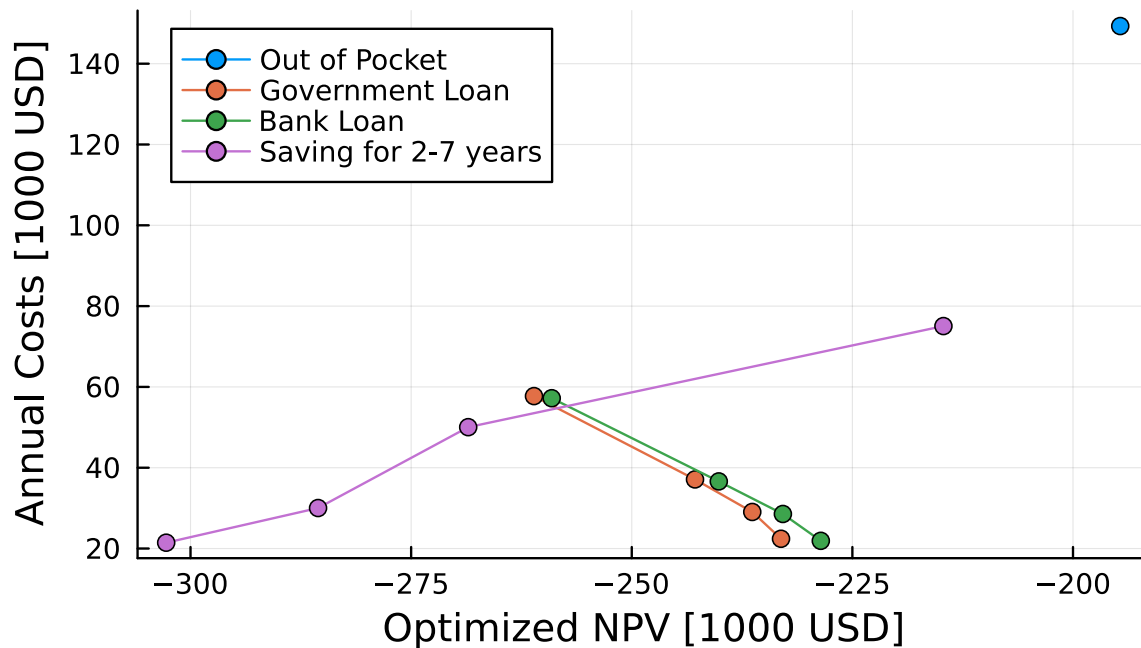
Low Equity (20.0%), Low Savings (10.0K)

It's notable that you need about 150k in your savings to have all elevation options open to you

```julia
HEUHS = gen_finances(high_equity, ultra_high_savings, house)
plot_LHSE_scenario(HEUHS, "High Equity ($(high_equity*100)%), Ultra High Savings ($(ultra_high_
```

High Equity (80.0%), Ultra High Savings (150.0k

3.12 testing

```julia
#{julia} test_plot = plot(    [optimal_npv] ./ 1000,    [annual_cost] ./ 1000
;    xlabel="Optimized NPV [1000 USD]",    ylabel="Annual Costs [1000 USD]",
title="NPV vs Yearly Costs",    label="Out of Pocket",    marker=:circle#,
color=line_color ) general_plot(1, 2, finance_list_rates, test_plot, "Loan,
different interest rates") general_plot(1, 2, finance_list_years, test_plot,
"Loan, different years") general_plot(1, 2, finance_saving, test_plot, "Saving
for 2-7 years") #display(Annual_v_npv_plot) #
```

Now we want to compare NPVS and construction heights for all these different scenarios

3.13 Next: add indicators to annual_v_npv plot that shows what thing is what

You should make your modifications in either the `HouseElevation` or `ParkingGarage` module. Detail the steps taken to implement the selected feature and integrate it into the decision-support tool. Include code snippets and explanations where necessary to clarify the implementation process.

3.14 Validation

In this case, we don't really *need* optimization – we can use brute force. We can compare by plotting the objective function for a range of elevations (from 0 to 14 ft) using all SOWs.

As we have seen in labs, mistakes are inevitable and can lead to misleading results. To minimize the risk of errors making their way into final results, it is essential to validate the implemented feature. Describe the validation techniques used to ensure the accuracy and reliability of your implemented feature. Discuss any challenges faced during the validation process and how they were addressed.

4 Results

Present the results obtained from the enhanced decision-support tool. Use tables, figures, and visualizations to clearly communicate the outcomes. Provide sufficient detail to demonstrate how the implemented feature addresses the problem statement. Use the `#| output: false` and/or `#| echo: false` tags to hide code output and code cells in the final report except where showing the output (e.g.g, a plot) or the code (e.g., how you are sampling SOWs) adds value to the discussion. You may have multiple subsections of results, which you can create using `##`.

5 Conclusions

5.1 Discussion

Analyze the implications of your results for climate risk management. Consider the context of the class themes and discuss how your findings contribute to the understanding of climate risk assessment. Identify any limitations of your approach and suggest potential improvements for future work.

5.2 Conclusions

Summarize the key findings of your project and reiterate the significance of your implemented feature in addressing the problem statement. Discuss the broader implications of your work for climate risk management and the potential for further research in this area.

References

[1] Saman Armal et al. "Assessing Property Level Economic Impacts of Climate in the US, New Insights and Evidence from a Comprehensive Flood Risk Assessment Tool". In: *Climate* 8.10 (2020). ISSN: 2225-1154. DOI: 10.3390/cli8100116. URL: https://www.mdpi.com/2225-1154/8/10/116.

[2] Hugues Chenet, Josh Ryan-Collins, and Frank van Lerven. "Finance, climate-change and radical uncertainty: Towards a precautionary approach to financial policy". In: *Ecological Economics* 183 (2021), p. 106957. ISSN: 0921-8009. DOI: https://doi.org/10.1016/j.ecolecon.2021.106957. URL: https://www.sciencedirect.com/science/article/pii/S092180092100015X.