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 (NPV) of home elevation under the assumption that homeowners will pay for their home elevation out of pocket. In reality, people have to choose between paying out of pocket to whatever maximum height they can afford, taking out a loan, or saving up.

Each of these options has tradeoffs in total cost, annual cost, the cost of damages, and in NPV. We seek to analyze these tradeoffs by running simulations of each action in multiple possible futures with respect to flooding.

1.2 Selected Feature

The new feature I've chosen to implement is a struct called "Finance". This struct describes the financial conditions of the homeowner, and the plan to elevate the home. 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), 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 US Dollars.

In addition to this new struct, run_sim has been modified to take it into account. So if we choose 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 for however long it takes us to save money. Money already in the savings account reduces the amount of money we need to save.

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 present value of these payments.

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

2 Literature Review

Whether this Finance 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. Right now, their options include giving homeowners money to elevate their homes, forcing them to pay immediately for home elevation, setting up loan programs for home elevation, and forcing homeowners to save up for home elevation. Our new Finance feature can be used to compare these three strategies.

Tools that communicate the propery-level impacts of flooding to homeowners and community leaders are directly 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

Running this isn’t usually necessary, but sometimes if changes to the Finance package aren’t being seen in the code here, running it can fix that problem.

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

Load packages

```julia
using CSV
using DataFrames
using DataFramesMeta
using Distributions
using LaTeXStrings
using Metaheuristics
using Plots
using Random
using Unitful

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

Load Revise and our custom package, HouseElevation

```julia
using Revise
using HouseElevation
#include("Finance.jl")
#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/GyanSMA2fp9r

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

Create functions that define our possible futures

```julia
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

function draw_surge_distribution()
      = rand(Normal(5, 1))
      = rand(Exponential(1.25))
      = rand(Normal(0.1, 0.05))
    return GeneralizedExtremeValue( ,  ,  )
end

function draw_discount_rate()
    return 0.055  #fix interest rate at 5.5%
    #return rand(Normal(0.05, 0.03))
end

function draw_sow()
    slr = rand(slr_scenarios)
    surge_params = draw_surge_distribution()
    discount = draw_discount_rate()
    return SOW(slr, surge_params, discount)
end
```

Generate our possible states of the world

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

3.2 Financial_NPVS function

Write our function we'll use to evaluate financial plans

```
#assume physical house and SLR scenarios stay constant, only financial conditions change
function financial_npvs(finance)
    p = ModelParams(; house=house, years=2024:2083, finance=finance)  #make our model

    elevations_try = 0:0.5:14  #establish actions to try
    actions_try = Action.(elevations_try)


    if p.finance.loan == 1 #if taking out a loan
        #Get all the contruction costs we can afford through equity
        construction_costs = [elevation_cost(house, elevation) for elevation in elevations_try
    elseif p.finance.loan == 0 #if paying out of pocket
        #must have enough in savings to pay
        construction_costs = [elevation_cost(house, elevation) for elevation in elevations_try
    else # if we're saving up, we can try all possible outcomes
        construction_costs = [elevation_cost(house, elevation) for elevation in elevations_try]
    end


    actions_try = actions_try[1:length(construction_costs)]  #get the list of actions we can a
    elevations_try = elevations_try[1:length(construction_costs)]  #limit elevations as well s

    #Run simulations
    npvs_opt = [mean([run_sim(a, sow, p) for sow in sows_opt]) for a in actions_try]

    #get height that minimizes NPV
    min_npv, min_idx = findmax(npvs_opt)
    minimizer = elevations_try[min_idx]

    #return elevations we tried (x), npvs(y) and minimizing elevation
    return elevations_try, npvs_opt, minimizer

end
```

3.3 Finance_Basic

Define what basic financing for someone who can pay upfront looks like

```julia
finance_basic = let
    loan = 0
    loan_years = 0
    loan_rate = 0.0
    paid_off_percent = 1.0
    amnt_paid_off = paid_off_percent * house.value_usd  # Calculate amnt_paid_off
    savings = 250_000

    # Create Finance object using keyword arguments
    Finance(;
        loan = loan,
        loan_years = loan_years,
        loan_rate = loan_rate,
        paid_off_percent = paid_off_percent,
        amnt_paid_off = amnt_paid_off,
        savings = savings
    )
end
```

Now let's get the results of paying up front with 250,000 in our savings account

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

3.4 plot_many

Define our function to plot many financial scenarios

```julia
function plot_many(finance_list, initial_plot)


    for finance in finance_list

        #println(finance)
        elevations_tried, npvs, min = financial_npvs(finance)
        if finance.loan == 1  #if taking out a loan
            label = "$(round(finance.loan_rate*100))% loan over $(finance.loan_years) years"
        elseif finance.loan > 1
            label = "Saving over $(finance.loan_years) years"
        else
            label = "Paying out of pocket"
        end
        plot!(
        initial_plot,
        elevations_tried,
        npvs ./ 1000;
        xlabel="Elevation [ft]",
        ylabel="NPV [1000 USD]",
        label=label,
        marker=:circle,
        #color=line_color
        )

         #make the vertical line color the same as the horizontal line
        line_color = initial_plot.series_list[end][:linecolor]
        #line_color = plot_object.series_list[end].plotseries[:linecolor]
        vline!([min]; label="$(min)ft elevation", linestyle=:dash, color=line_color)
    end
end
```

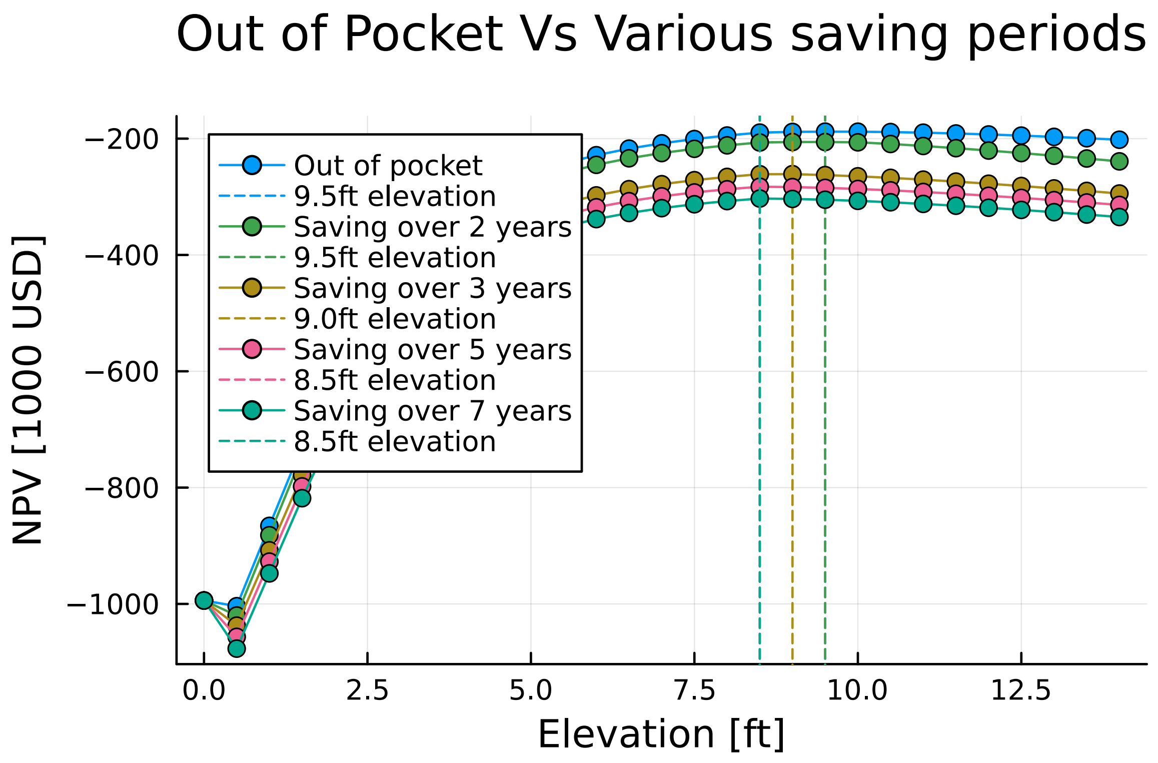
3.5 reasonable saving time

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

```
finance_saving = []

for i in [2, 3, 5, 7]
    #print(i)
    paid_off_percent = 1.0
    savings = 180_000
    fin = Finance(; loan=2, loan_years=i, loan_rate=0.0, paid_off_percent=paid_off_percent,
        amnt_paid_off=(house.value_usd*paid_off_percent), savings=savings
        )
    push!(finance_saving, fin)
end
```

Plot our saving options

```
p = plot(
    elevations_tried,
    npvs ./ 1000;
    xlabel="Elevation [ft]",
    ylabel="NPV [1000 USD]",
    title="Out of Pocket Vs Various saving periods",
    label="Out of pocket",
    marker=:circle#, color=line_color
)


line_color = p.series_list[end][:linecolor]  #make the vertical line color the same as the hor
vline!([min]; label="$(min)ft elevation", linestyle=:dash, color=line_color)

plot_many(finance_saving, p)
display(p)
```



Note that these results assume we already having 180,000 USD in our savings account, which is quite a lot.

Results of this graph show two interesting things:

1. NPV of saving is often 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.

3.6 Long saving time

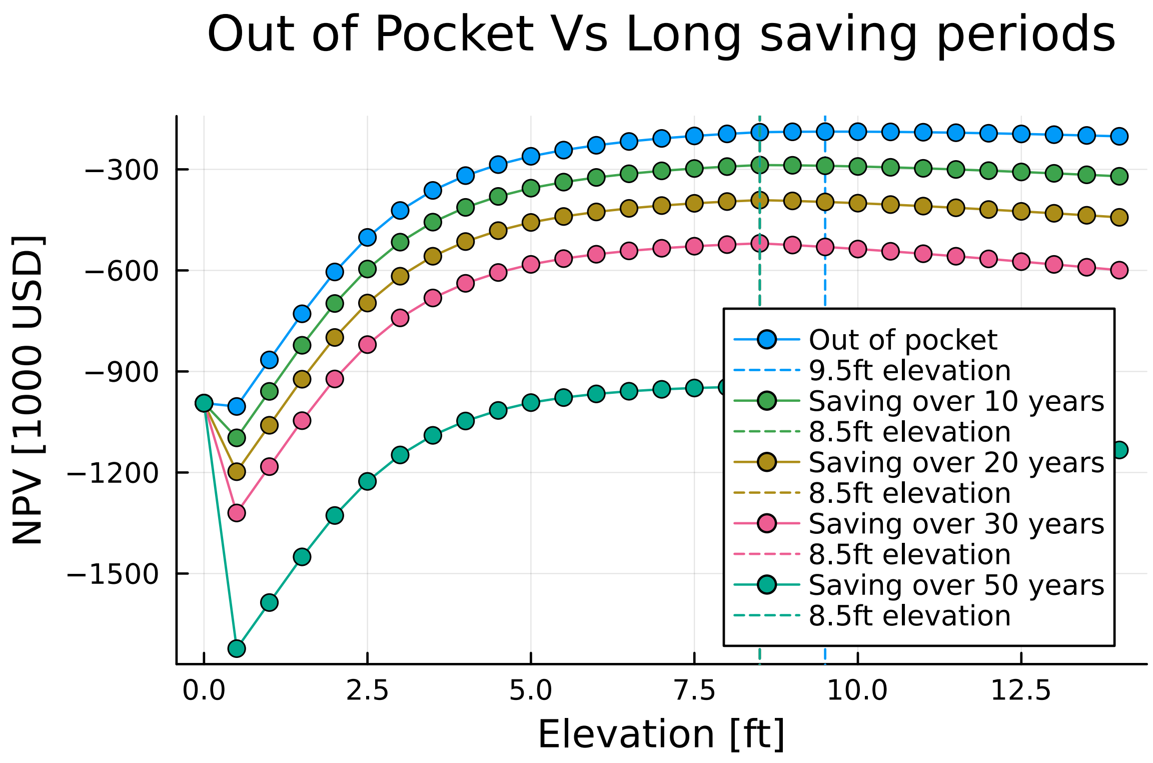
Define finances of saving for very long periods of time

```
Long_Savings = []

for i in [10, 20, 30, 50]
```

```
    #print(i)
    paid_off_percent = 1.0
    savings=180_000
    fin = Finance(; loan=2, loan_years=i, loan_rate=0.0, paid_off_percent=paid_off_percent,
        amnt_paid_off=(house.value_usd*paid_off_percent),
        savings=180_000
        )
    push!(Long_Savings, fin)
end
```

Plot long saving scenarios

```
p = plot(
    elevations_tried,
    npvs ./ 1000;
    xlabel="Elevation [ft]",
    ylabel="NPV [1000 USD]",
    title="Out of Pocket Vs Long saving periods",
    label="Out of pocket",
    marker=:circle#, color=line_color
)


line_color = p.series_list[end][:linecolor]  #make the vertical line color the same as the hor
vline!([min]; label="$(min)ft elevation", linestyle=:dash, color=line_color)

plot_many(Long_Savings, p)
display(p)
```



Again, keep in mind we're doing this with 180,000 dollars already in our savings account.

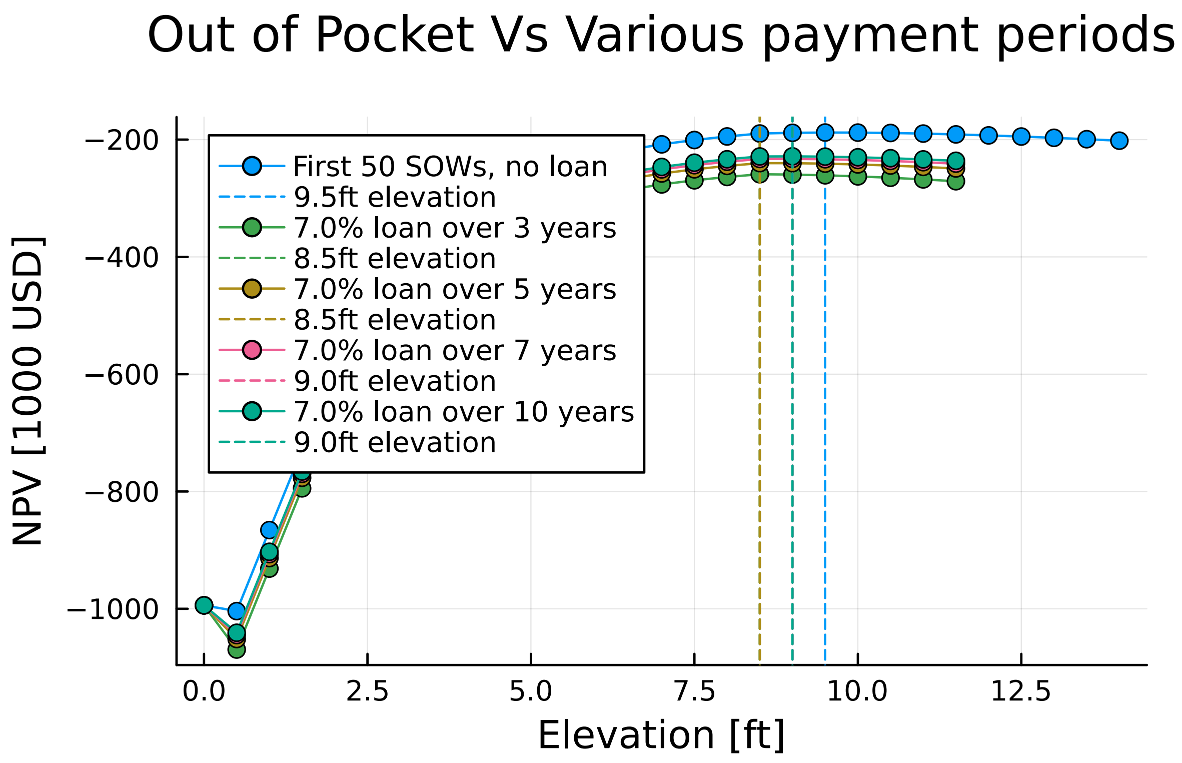
3.7 Different loan durations

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

```julia
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
```

Plot our scenarios of different amounts of time to pay off the loan

```julia
p = plot(
    elevations_tried,
    npvs ./ 1000;
    xlabel="Elevation [ft]",
    ylabel="NPV [1000 USD]",
    title="Out of Pocket Vs Various payment periods",
    label="First $(N_SOW_opt) SOWs, no loan",
    marker=:circle#, color=line_color
)


line_color = p.series_list[end][:linecolor]  #make the vertical line color the same as the hor
vline!([min]; label="$(min)ft elevation", linestyle=:dash, color=line_color)

plot_many(finance_list_years, p)
display(p)
```



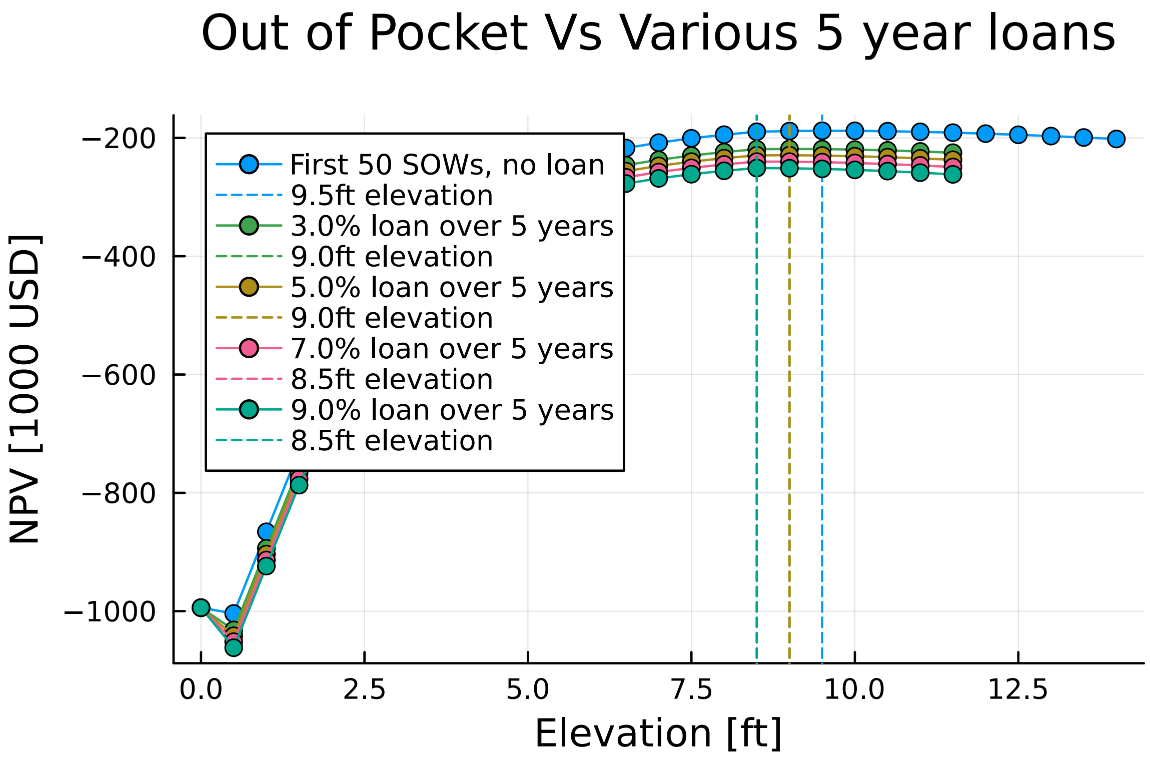
3.8 Different interest rates

Generate many financial scenarios with different interest rates

```
finance_list_rates = []

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

Plot what our finances look like with different interest rates

```
p = plot(
    elevations_tried,
    npvs ./ 1000;
    xlabel="Elevation [ft]",
    ylabel="NPV [1000 USD]",
    title="Out of Pocket Vs Various 5 year loans",
    label="First $(N_SOW_opt) SOWs, no loan",
    marker=:circle#, color=line_color
)
#println(p.series_list[end][:linecolor] )

#colors = Plots.palette()

line_color = p.series_list[end][:linecolor]  #make the vertical line color the same as the hori
vline!([min]; label="$(min)ft elevation", linestyle=:dash, color=line_color)

plot_many(finance_list_rates, p)
display(p)
```



3.9 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 elevation

```julia
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
    else #saving up
        annual_cost = construction_cost/finance.loan_years
    end

    return best_npv, annual_cost, best_elevation

end
```

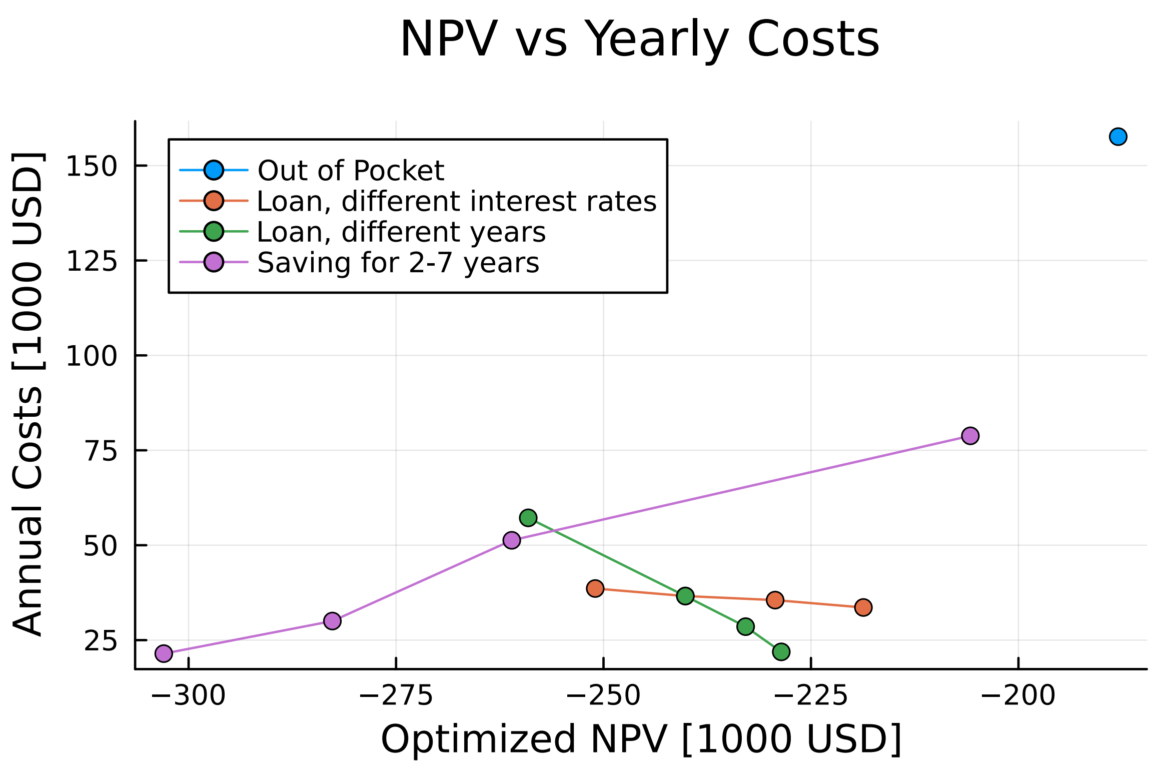
Write a function that obtain the npvs, annual costs, and optimal elevations of different scenarios.

Also write a function that'll help us plot these results.

```julia
function get_costs(finances)
    figures = annual_v_npv.(finances)
    best_npvs = [fig[1] for fig in figures]
    annual_costs = [fig[2] for fig in figures]
    best_elevations = [fig[3] for fig in figures]

    return best_npvs, annual_costs, best_elevations
end


function plot_money_values(finance, initial_plot, label)

    best_npvs, annual_costs, best_elevations = get_costs(finance)
    plot!(
        initial_plot,
        best_npvs ./ 1000,
        annual_costs ./ 1000;
        xlabel="Optimized NPV [1000 USD]",
        ylabel="Annual Costs [1000 USD]",
        label=label,
        marker=:circle,
        #color=line_color
        )

end
```

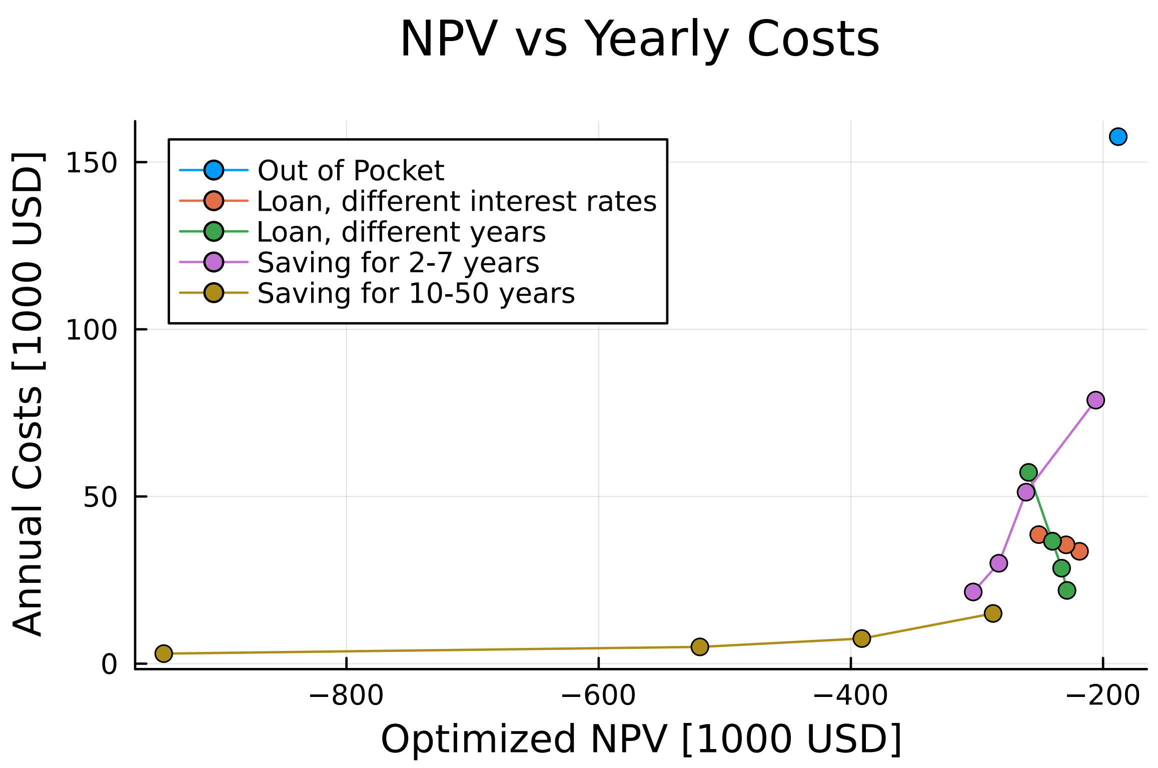
Plot our results

```
optimal_npv, annual_cost, best_elevation = annual_v_npv(finance_basic)

Annual_v_npv_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
)
plot_money_values(finance_list_rates, Annual_v_npv_plot, "Loan, different interest rates")
plot_money_values(finance_list_years, Annual_v_npv_plot, "Loan, different years")
plot_money_values(finance_saving, Annual_v_npv_plot, "Saving for 2-7 years")
display(Annual_v_npv_plot)
```



The chart changes radically when we display long savings times

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



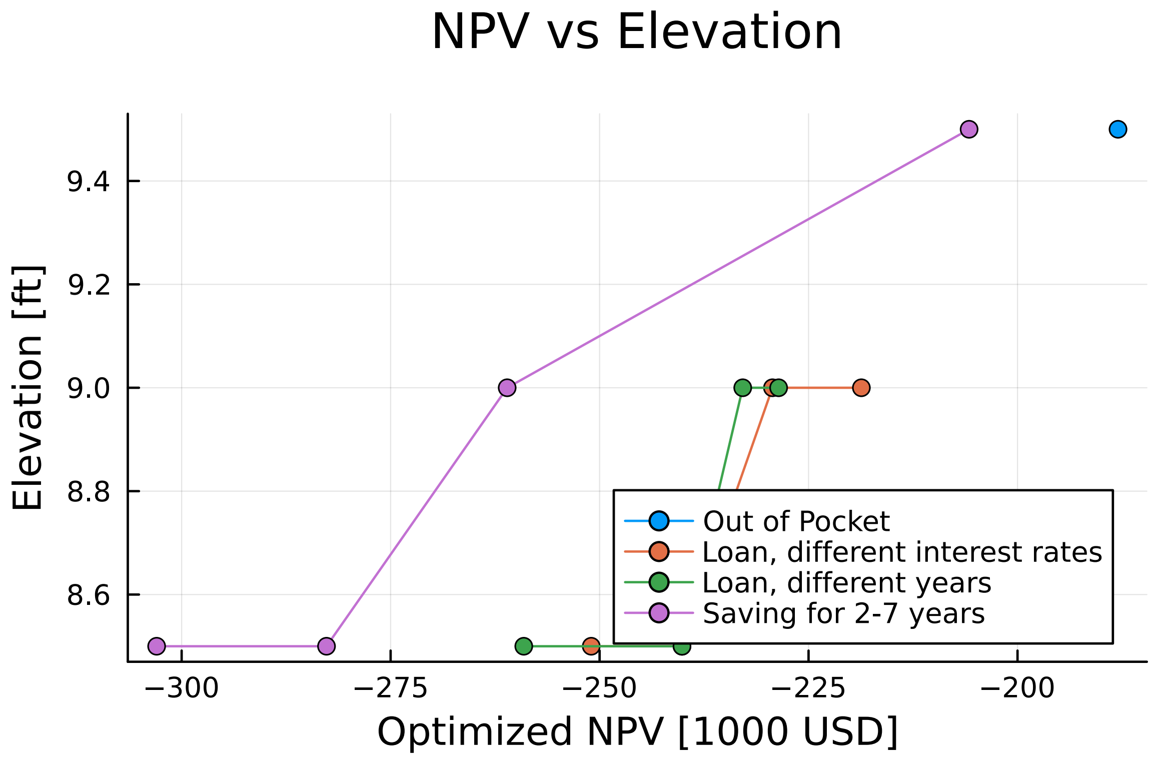
3.10 compare NPVS and construction heights for all these different scenarios

Write a more generalized plotting function since we'll be comparing a few different things now

```julia
function general_plot(x, y, finance, initial_plot, label)
    # 1 = best_npvs, 2 = annual_costs, 3 = best_elevation
    values = collect(get_costs(finance)) #get our values to plot.
    #Using collect makes sure these values are in an array so we can use ./1000 later
    #println(values)
    #println(values[2])
    values[1] = values[1] ./ 1000  #adjust npv and annual costs
    values[2] = values[2] ./1000

    plot!(
        initial_plot,
        values[x],  #decide what to plot
        values[y];
        label=label,
        marker=:circle,
        #color=line_color
        )
end
```

Plot how optimal height of different plans compares to NPV

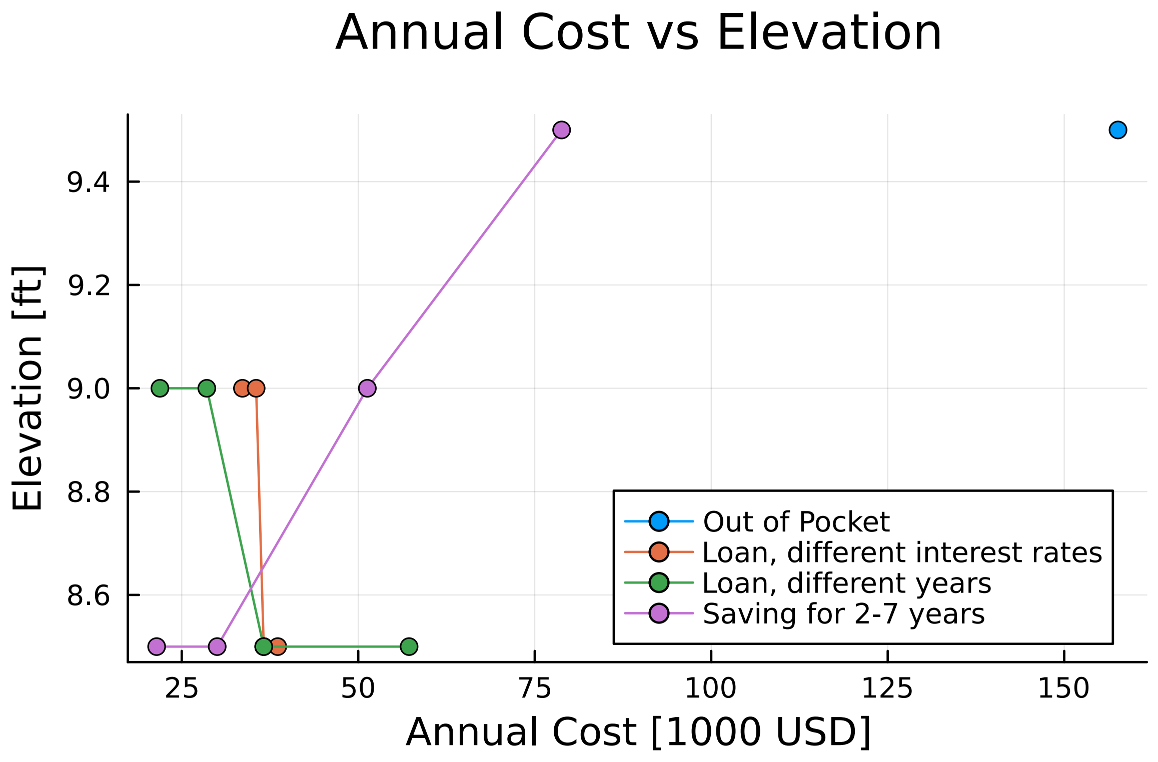
```
height_v_npv_plot = plot(
    [optimal_npv] ./ 1000,
    [best_elevation] ;
    xlabel="Optimized NPV [1000 USD]",
    ylabel="Elevation [ft]",
    title="NPV vs Elevation",
    label="Out of Pocket",
    marker=:circle,#, color=line_color
    legend=:bottomright
)

general_plot(1, 3, finance_list_rates, height_v_npv_plot, "Loan, different interest rates")
general_plot(1, 3, finance_list_years, height_v_npv_plot, "Loan, different years")
general_plot(1, 3, finance_saving, height_v_npv_plot, "Saving for 2-7 years")
display(height_v_npv_plot)
```



3.11 compare annual costs and construction heights

Plot how optimal height of different plans compares to annual costs

```
height_v_npv_plot = plot(
    [annual_cost] ./ 1000,
    [best_elevation] ;
    xlabel="Annual Cost [1000 USD]",
    ylabel="Elevation [ft]",
    title="Annual Cost vs Elevation",
    label="Out of Pocket",
    marker=:circle,#, color=line_color
    legend=:bottomright
)

general_plot(2, 3, finance_list_rates, height_v_npv_plot, "Loan, different interest rates")
general_plot(2, 3, finance_list_years, height_v_npv_plot, "Loan, different years")
general_plot(2, 3, finance_saving, height_v_npv_plot, "Saving for 2-7 years")
display(height_v_npv_plot)
```



3.12 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.

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 financial outcomes for these different plans given a set equity and savings:

```
function gen_finances(equity, savings, house)

    loan_finances = [] #7.0% bank loan
    gov_loan = []  #7.5% government loan for those in need
    saving_finances = []

    for i in [2, 3, 5, 7]  #saving scenarios
        fin = Finance(; loan=2, loan_years=i, loan_rate=0.0, paid_off_percent=equity,
            amnt_paid_off=(house.value_usd*equity), savings=savings
            )
        push!(saving_finances, fin)
    end

    for i in [3, 5, 7, 10]  #bank loan scenarios
        fin = Finance(; loan=1, loan_years=i, loan_rate=0.07, paid_off_percent=equity,
        amnt_paid_off=(house.value_usd*equity), savings=savings
        )
        push!(loan_finances, fin)
    end

    for i in [3, 5, 7, 10]  #government loan scenarios
        fin = Finance(; loan=1, loan_years=i, loan_rate=0.075, paid_off_percent=1.0,  #gov loa
        amnt_paid_off=(house.value_usd*1.0), savings=savings
        )
        push!(gov_loan, fin)
    end

    # Our of pocket elevation
    OOP = Finance(;
        loan = 0,
        loan_years = 0,
        loan_rate = 0.0,
        paid_off_percent = equity,
        amnt_paid_off = (house.value_usd*equity),
        savings = savings
    )

    return OOP, saving_finances, loan_finances, gov_loan

end
```

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

```
function plot_LHSE_scenario(LHSE_options, title)

    OOP, saving_finances, loan_finances, gov_loan = LHSE_options

    optimal_npv, annual_cost, best_elevation = annual_v_npv(OOP)

    Annual_v_npv_plot = plot(
        [optimal_npv] ./ 1000,
        [annual_cost] ./ 1000 ;
        xlabel="Optimized NPV [1000 USD]",
        ylabel="Annual Costs [1000 USD]",
        title=title,
        label="Out of Pocket",
        marker=:circle#, color=line_color
    )
    plot_money_values(gov_loan, Annual_v_npv_plot, "Government Loan")
    plot_money_values(loan_finances, Annual_v_npv_plot, "Bank Loan")
    plot_money_values(saving_finances, Annual_v_npv_plot, "Saving for 2-7 years")
    display(Annual_v_npv_plot)

end
```

3.13 Validation

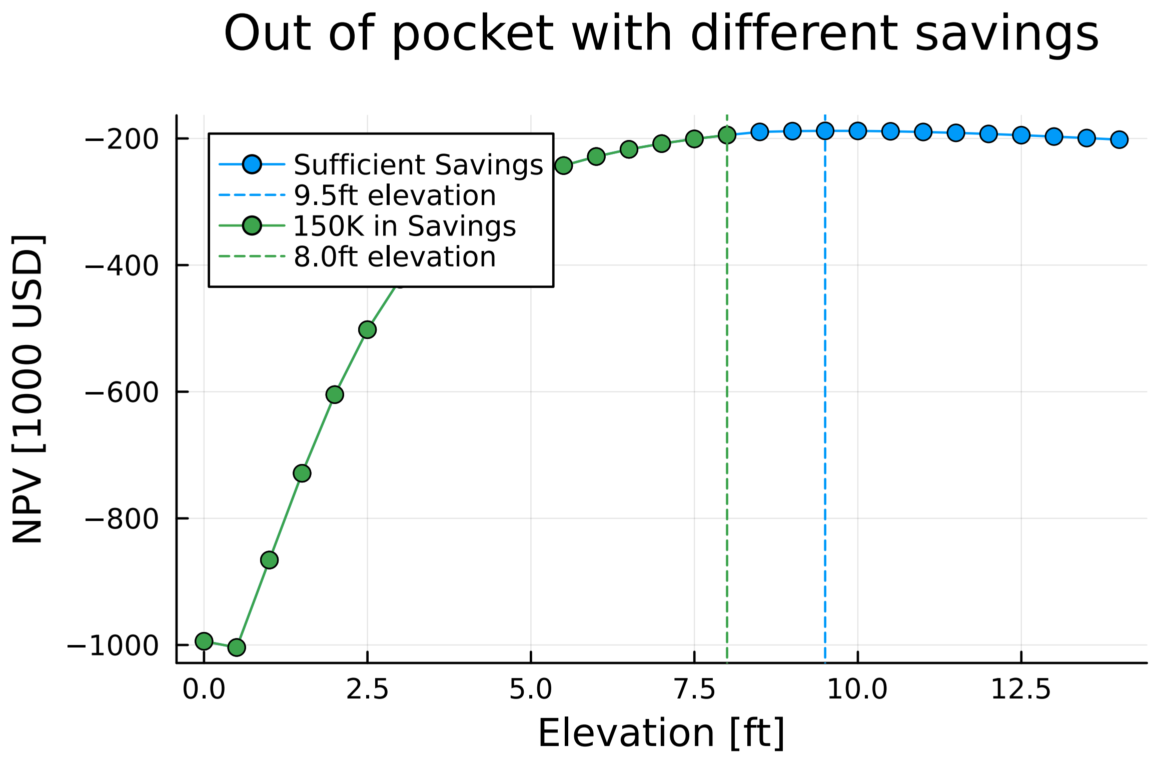
We should validate that the out-of-pocket elevation limiting is working as expected, and that homeowners can elevate up to heights they can afford with their savings, but not beyond that.

Make a household that can pay for some but not all elevation heights out of pocket:

```
finance_OOP = let
    loan = 0
    loan_years = 0
    loan_rate = 0.0
    paid_off_percent = 1.0
    amnt_paid_off = paid_off_percent * house.value_usd  # Calculate amnt_paid_off
    savings = 150_000

    # Create upgraded Finance object using keyword arguments
    Finance(;
        loan = loan,
        loan_years = loan_years,
        loan_rate = loan_rate,
        paid_off_percent = paid_off_percent,
        amnt_paid_off = amnt_paid_off,
        savings = savings
    )
end

elevations_oop, npvs_oop, min_oop = financial_npvs(finance_OOP)
```

Plot a comparison between it, and a household that can elevate to the maximum height out of pocket:

```
p = plot(
    elevations_tried,
    npvs ./ 1000;
    xlabel="Elevation [ft]",
    ylabel="NPV [1000 USD]",
    title="Out of pocket with different savings",
    label="Sufficient Savings",
    marker=:circle#, color=line_color
)
line_color = p.series_list[end][:linecolor]  #make the vertical line color the same as the hor
vline!([min]; label="$(min)ft elevation", linestyle=:dash, color=line_color)

p = plot!(
    elevations_oop,
    npvs_oop ./ 1000;
    xlabel="Elevation [ft]",
    ylabel="NPV [1000 USD]",
    #title="Out of pocket with different savings",
    label="150K in Savings",
    marker=:circle#, color=line_color
)
line_color = p.series_list[end][:linecolor]  #make the vertical line color the same as the hor
vline!([min_oop]; label="$(min_oop)ft elevation", linestyle=:dash, color=line_color)

#plot_many([finance_OOP], p)
display(p)
```

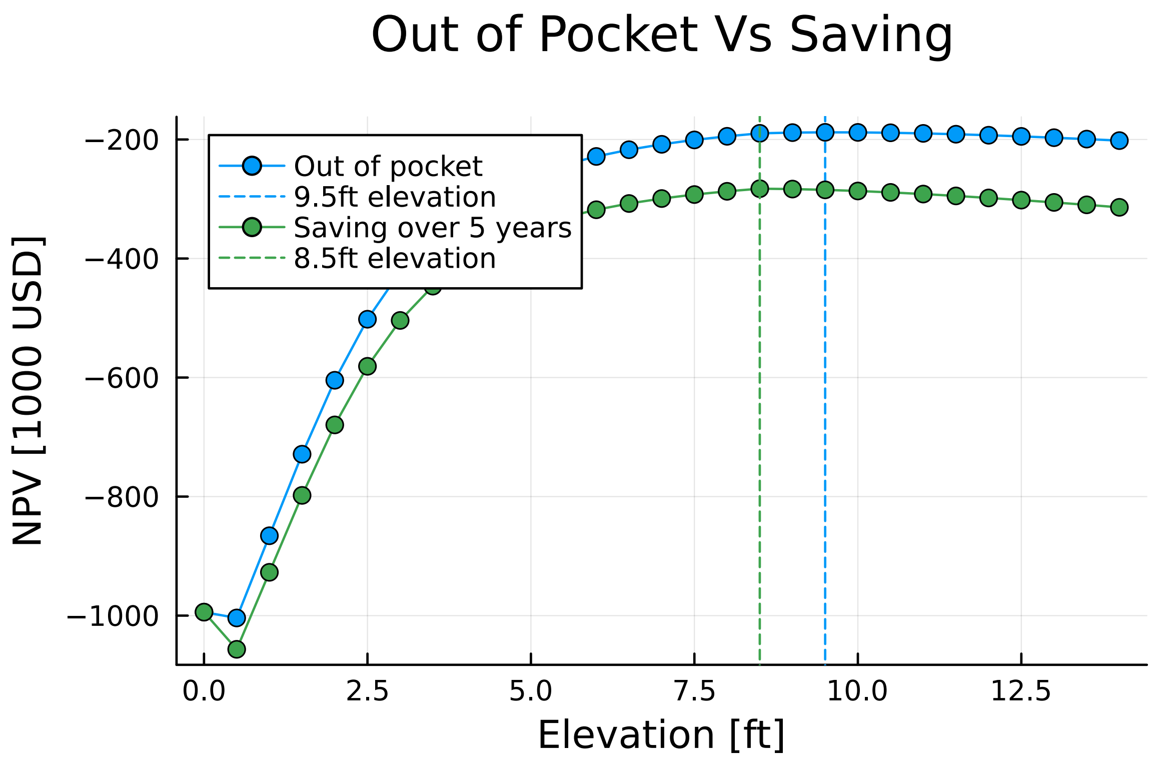


We can see that our home that is being elevated out of a shallower pocket will have deeper flooding! This aligns with what we'd expect.

We can also make sure our loan and savings features are working as expected by analyzing the impact they have on outcomes.

Let's look at how things change when we save for several years:

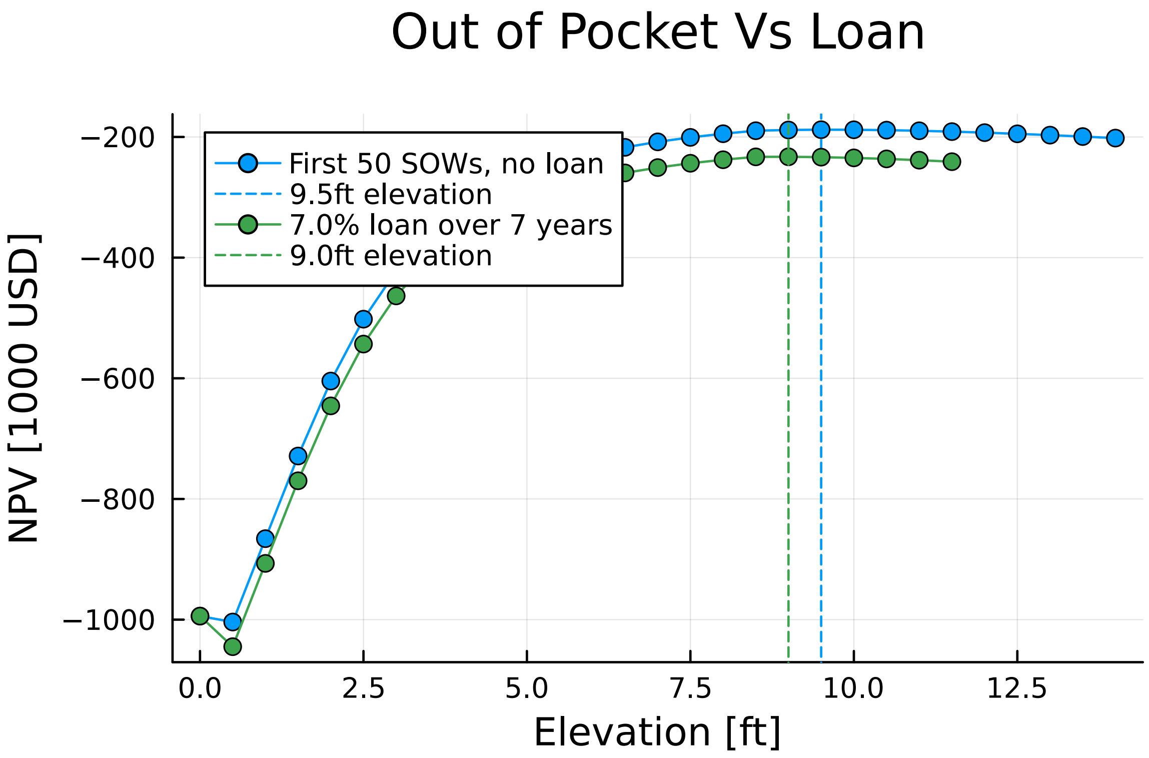
```julia
p = plot(
    elevations_tried,
    npvs ./ 1000;
    xlabel="Elevation [ft]",
    ylabel="NPV [1000 USD]",
    title="Out of Pocket Vs Saving",
    label="Out of pocket",
    marker=:circle#, color=line_color
)


line_color = p.series_list[end][:linecolor]  #make the vertical line color the same as the hori
vline!([min]; label="$(min)ft elevation", linestyle=:dash, color=line_color)

plot_many([finance_saving[3]], p)
display(p)
```



When we save, elevation costs increase over time with inflation, so the cost to elevate is higher, lowering both NPV and the ideal elevation height. Additionally, the benefits of elevating are diminished, because any flooding we would've been protected from by elevating might've already happened while we were saving up. This lowers both the NPV and the ideal elevation height even more.

Let's see how things change when we take out a loan:

```
p = plot(
    elevations_tried,
    npvs ./ 1000;
    xlabel="Elevation [ft]",
    ylabel="NPV [1000 USD]",
    title="Out of Pocket Vs Loan",
    label="First $(N_SOW_opt) SOWs, no loan",
    marker=:circle#, color=line_color
)


line_color = p.series_list[end][:linecolor]  #make the vertical line color the same as the hor
vline!([min]; label="$(min)ft elevation", linestyle=:dash, color=line_color)

plot_many([finance_list_years[3]], p)
display(p)
```



When we take out a loan, our maximum NPV is lower, the actions we can take are limited by our finances, and the ideal elevation height is lower. The lower NPV and optimal elevation are again due to the "Up-front" cost of elevation now being higher, this time because instead of just paying the cost of elevation, we now have to pay that cost, plus interest on the loan.

4 Results

We'll present our results by looking at four scenarios: One where the homeowner has high equity and high savings, one where they have low equity and high savings, high equity and low savings, and low equity and low savings.

```julia
println("The cost to elevate our home 9.5 feet is ", elevation_cost(house, 9.5))
```

```
The cost to elevate our home 9.5 feet is 157620.0
```

Note that the cost to elevate our house 9.5 feet, what was previously considered optimal when paying out of pocket, is more than 157,000 US dollars, putting it out of reach for out of pocket payment even 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 120,000 are doing fairly well, but still can't afford the "optimal" elevation. [3]

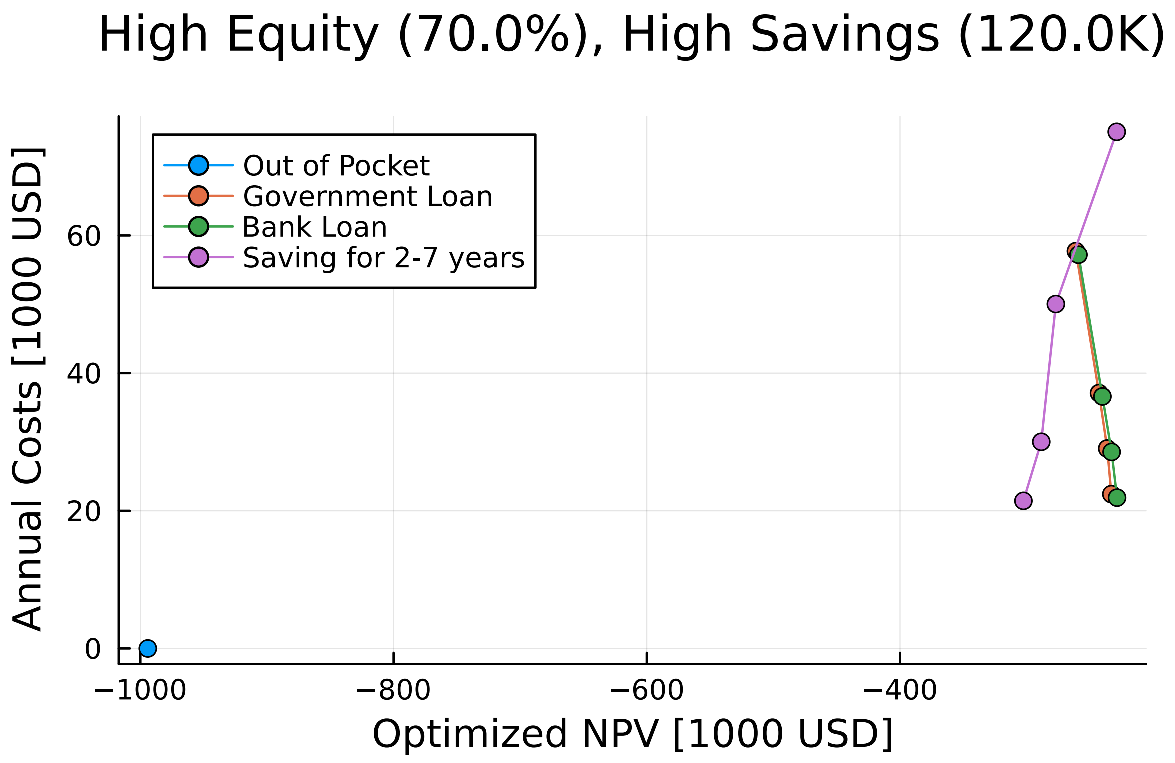
Also, it's worth pointing out that our "low savings" individual has 10k in their bank account, but this is still more than 60% of americans [3]

Our high-equity individuals will have 70% equity in their home, and low-equity will have 30%

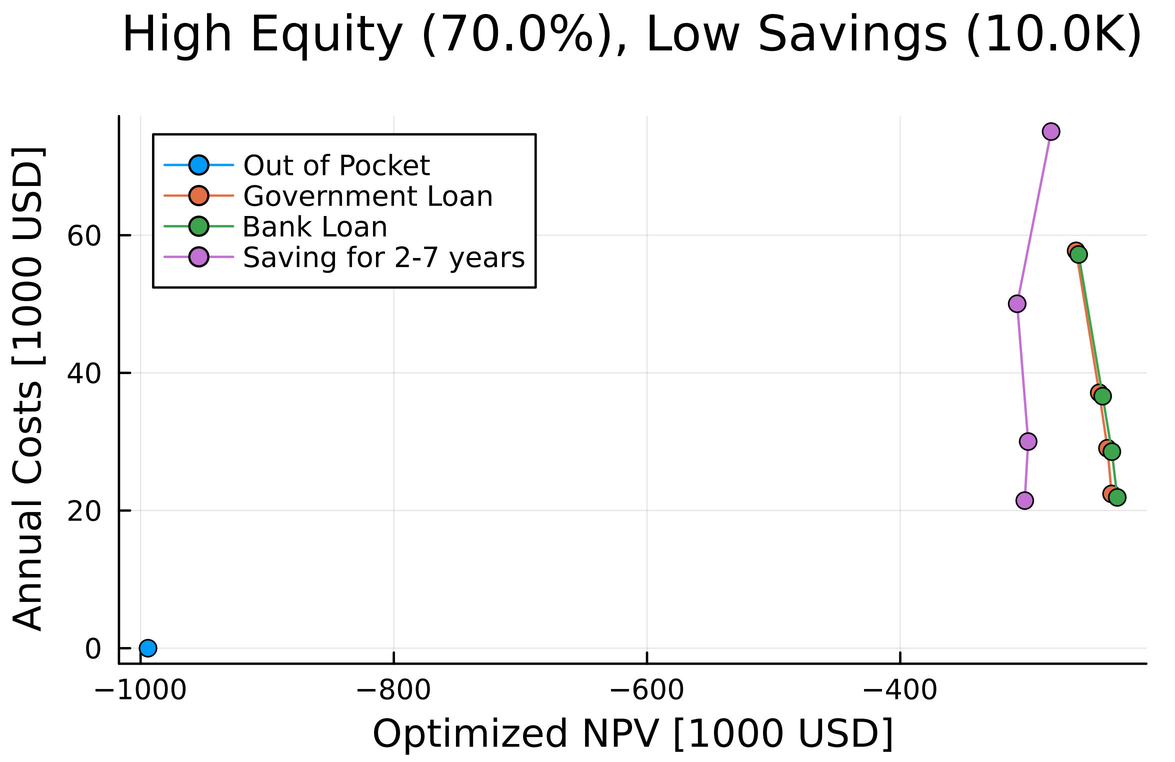
```julia
#high equity, high savings
high_equity = 0.7
high_savings = 120_000

low_equity = 0.3
low_savings = 10_000
#10k is actally still more than 60% of americans according to the motley fool, which is... sad

ultra_high_savings = 250_000
```

```
250000
```

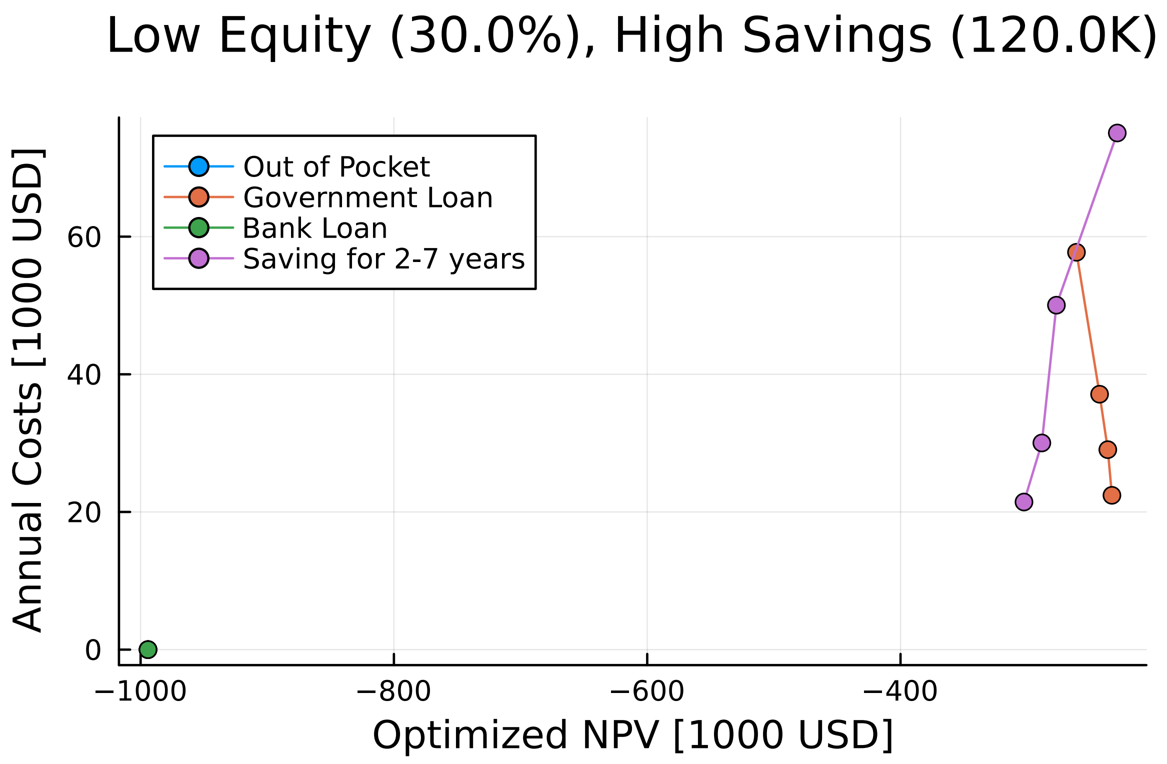
```julia
HEHS = gen_finances(high_equity, high_savings, house)
plot_LHSE_scenario(HEHS, "High Equity ($(high_equity*100)%), High Savings ($(high_savings/1000
```



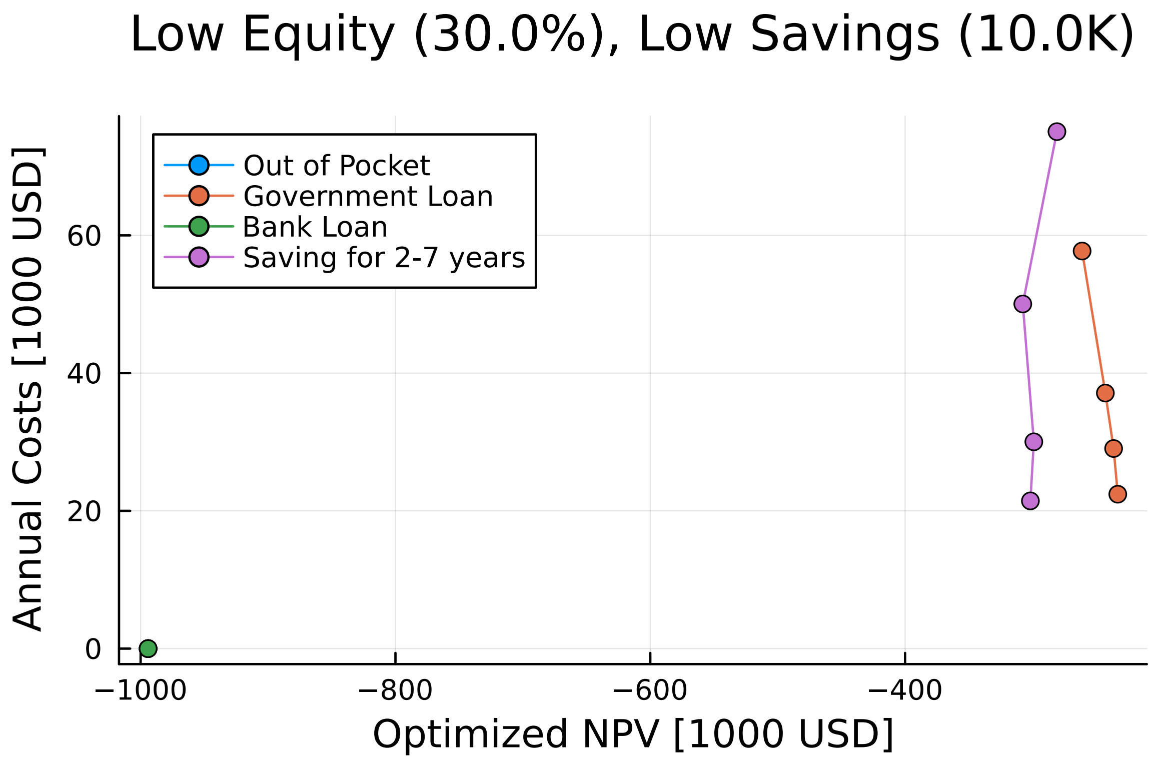
```julia
HELS = gen_finances(high_equity, low_savings, house)
plot_LHSE_scenario(HELS, "High Equity ($(high_equity*100)%), Low Savings ($(low_savings/1000)K
```



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

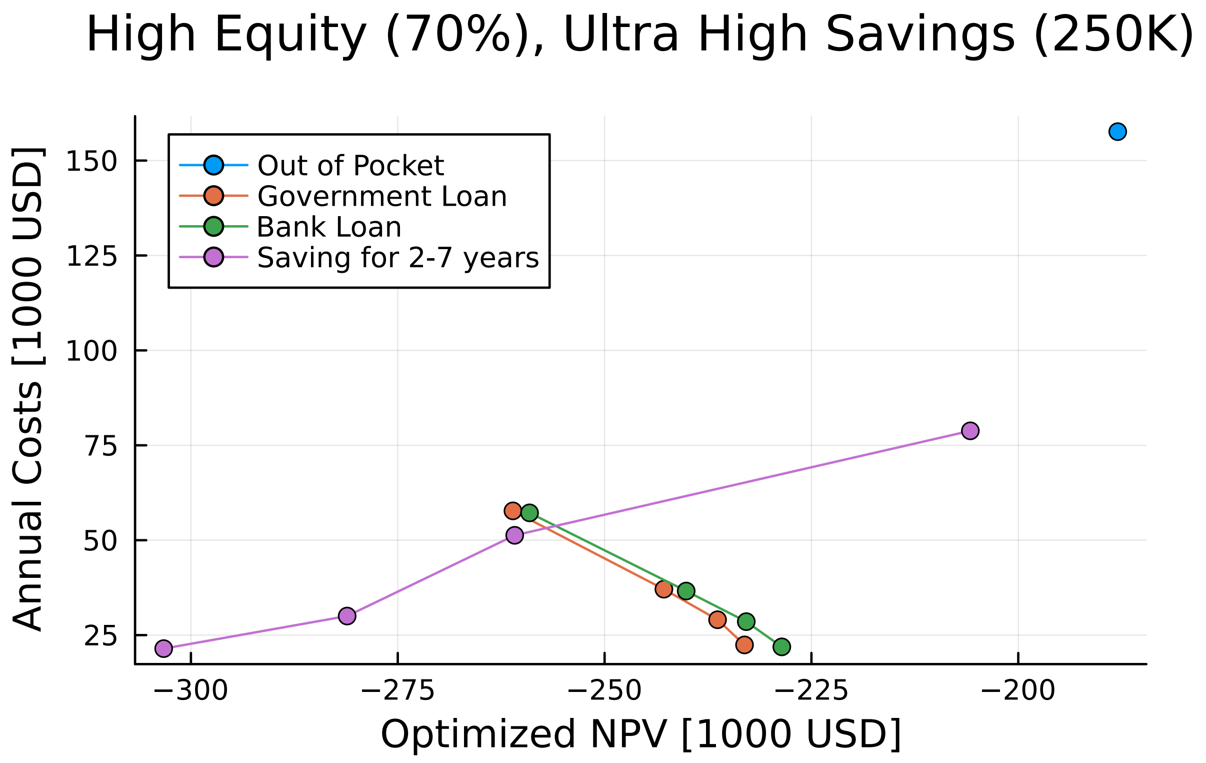


```
LELS = gen_finances(low_equity, low_savings, house)
plot_LHSE_scenario(LELS, "Low Equity ($(low_equity*100)%), Low Savings ($(low_savings/1000)K)"
```



It's notable that you need about 150k in your savings to have all elevation options open to you. We can see this if we make a chart for someone with "ultra high" savings.

```
HEUHS = gen_finances(high_equity, ultra_high_savings, house)
plot_LHSE_scenario(HEUHS, "High Equity ($(Int64(high_equity*100))%), Ultra High Savings ($(Int6
```



Inspite of the high potential for NPV when paying out of pocket, in all of the other scenarios we looked at, the NPV of paying out of pocket was lower than that of saving up or taking out a loan. This is due to the high cost of construction making paying for an optimal elevation without some type of finanical plan out of reach even for wealthy individuals. The Ultra-high savings individual is the only scenario where the NPV of paying out of pocket is higher than loans or saving because it's the only one where this strategy was able to be used to its full potential.

5 Conclusions

5.1 Discussion

The results indicate that although paying up front for elevation *can* return the highest Net Present Value, this is only true if the person paying to elevate their home has enough money to elevate it to an appropriate height.

This might incline someone to take out a loan instead of paying up front. And since taking out a loan increases the cost of elevation through incurring interest on the initial cost, the elevation that yields an ideal NPV is lower than what we previoulsy saw when we were looking only at out-of-pocket elevation simulations.

While saving money may seem like a potential solution to this, since one does not incur interest if they do so, our simulations reveal it to be a very bad option. Saving to elevate devalues the elevation itself becasue less years are spent out of harm's way, causing the results to favor lower elevations while also having lower NPVs.

Our findings imply that if a government wanted to encourage people to elevate their homes to make their community more resilient to floods, the best way to do that in terms of return on total value is to pay for some or all of the cost of elevating people's homes. If this were not affordable however, then the next best option would be to offer sub-market rate loans that anyone can take out, so that even people with low equity can elevate their properties.

Future studies could investigate how savings interact with loans if people have the option to put their savigns into their loans, thus reducing the amount they need to borrow and pay interest on.

A main limitation of this work is that it concludes that the solution to flood damage is government intervention either through direct payments or loans, but doesn't conduct an analysis on the costs and benefits of these options from a governement's perspective. After a flood, great financial costs are incurred due to physical damages on the home, but there is also a cost in commerce and damage to the local economy when people are rebuilding their communities instead of engaging in status quo economic activities, like working and purchasing goods. Additionally, money that otherwise would've circulated in the local economy is spent on repairing homes. While the government would certainly unlock benefits from avoiding the opportunity cost in taxes lost due to a flood damaged community, how large those benefits are in comparison to the costs of elevation could be analyzed in future work.

5.2 Conclusions

Our key findings are that paying to elevate your home up front is the best option, if you can afford it. Otherwise, the second best option is to take out a loan, and the worst option is to save up to elevate your home. These are real decisions that real people will need to make as our climate changes and storms become more intense, and analyzing how our models interact with household finances can help homeowners make better choices.

We also saw implications for government action, and that a government-backed loan for homewoners with low equity can have extremely positive impacts in helping people elevate their homes.

Further research could investigate how pre-existing savings interact with loans when the two are combined, and what the financial circumstances look like from the government's point of view; how the cost of paying to elevate a community or to give out a loan compares to the savings of not having that community's tax base harmed by flood damage.

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.

[3] Kailey Hagen. *Here's How Much the Average Rich Person Has in the Bank*. Nov. 1, 2023. URL: https://www.fool.com/the-ascent/banks/articles/heres-how-much-the-average-rich-person-has-in-the-bank/ (visited on 04/25/2024).