CEVE 101: Project 03

Carmina Andrade ca77

2024-10-23

We begin by loading the required packages.

using Revise  
using RainwaterHarvesting

Line 1

Revise allows us to update code without restarting Julia.

Line 2

We load the RainwaterHarvesting package that contains our simulation tools.

### Setting the file path

filepath = "data/2.txt"  
rainfall\_data = RainfallData(filepath)  
rainfall\_data.annual\_data

Line 1

This is the path to the rainfall data file for the station of interest.

Line 2

This creates a RainfallData object containing the rainfall data for the specified station.

Dict{Int64, AnnualRainfallData} with 50 entries:  
 1985 => AnnualRainfallData([Date("1985-01-01"), Date("1985-01-02"), Date("198…  
 2004 => AnnualRainfallData([Date("2004-01-01"), Date("2004-01-02"), Date("200…  
 2002 => AnnualRainfallData([Date("2002-01-01"), Date("2002-01-02"), Date("200…  
 2001 => AnnualRainfallData([Date("2001-01-01"), Date("2001-01-02"), Date("200…  
 2013 => AnnualRainfallData([Date("2013-01-01"), Date("2013-01-02"), Date("201…  
 2015 => AnnualRainfallData([Date("2015-01-01"), Date("2015-01-02"), Date("201…  
 1991 => AnnualRainfallData([Date("1991-01-01"), Date("1991-01-02"), Date("199…  
 1976 => AnnualRainfallData([Date("1976-01-01"), Date("1976-01-02"), Date("197…  
 1988 => AnnualRainfallData([Date("1988-01-01"), Date("1988-01-02"), Date("198…  
 2008 => AnnualRainfallData([Date("2008-01-01"), Date("2008-01-02"), Date("200…  
 1974 => AnnualRainfallData([Date("1974-01-01"), Date("1974-01-02"), Date("197…  
 1994 => AnnualRainfallData([Date("1994-01-01"), Date("1994-01-02"), Date("199…  
 2018 => AnnualRainfallData([Date("2018-01-01"), Date("2018-01-02"), Date("201…  
 2017 => AnnualRainfallData([Date("2017-01-01"), Date("2017-01-02"), Date("201…  
 2011 => AnnualRainfallData([Date("2011-01-01"), Date("2011-01-02"), Date("201…  
 2005 => AnnualRainfallData([Date("2005-01-01"), Date("2005-01-02"), Date("200…  
 1979 => AnnualRainfallData([Date("1979-01-01"), Date("1979-01-02"), Date("197…  
 1997 => AnnualRainfallData([Date("1997-01-01"), Date("1997-01-02"), Date("199…  
 1986 => AnnualRainfallData([Date("1986-01-01"), Date("1986-01-02"), Date("198…  
 1993 => AnnualRainfallData([Date("1993-01-01"), Date("1993-01-02"), Date("199…  
 2019 => AnnualRainfallData([Date("2019-01-01"), Date("2019-01-02"), Date("201…  
 1998 => AnnualRainfallData([Date("1998-01-01"), Date("1998-01-02"), Date("199…  
 1981 => AnnualRainfallData([Date("1981-01-01"), Date("1981-01-02"), Date("198…  
 1982 => AnnualRainfallData([Date("1982-01-01"), Date("1982-01-02"), Date("198…  
 1987 => AnnualRainfallData([Date("1987-01-01"), Date("1987-01-02"), Date("198…  
 ⋮ => ⋮

### Plotting the Data

plot(rainfall\_data)

Line 1

This plot function is defined in [viz.jl](./RainwaterHarvesting/src/viz.jl) and is a thin wrapper around the Plots.jl package.

### Discussion

The years kept after data cleaning are 1985, 2004, 2002, 2001, 2013, 2015, 1991, 1976, 1988, and 2008. I believe that the available data is sufficient for meaningful risk assessments because the years span over a nearly 40-year period.

### Discussion

The years that Ceara, Brazil had major droughts are 1998, 2002, 2012-2018. My data includes only some of these years (2002, 2013, 2015.) Missing data during drought years can yield an inaccurate assessment, often an underestimation, of the impact of droughts on communities. This can lead to potential water shortages because water allocation strategies are based only on non-drought years.

## Understanding the Theoretical Model

Let’s explore the model that simulates the rainwater harvesting system.

### Discussion

The major limiting assumption in the model is that all households will consume the same amount, beginning in the 150th day of the year. So I would create a household coefficient that would be the coefficient for consumption, which would be equivalent to the number of household members. This would however, also assume that one person consumes the same amount. Unfortunately, there is no way to quantify that. Thus, the modified equation would be

## Model Implementation and Testing

### Defining the Model Parameters

param = ModelParameters(  
 runoff\_coefficient=0.85,  
 roof\_area\_m2=40.0,  
 first\_flush\_mm=4.0,  
 tank\_capacity\_L=16000.0  
)

Line 1

This creates a ModelParameters object with the specified parameters.

Line 2

The runoff coefficient () is a measure of how much of the rain that falls on the roof ends up in the tank.

Line 3

The roof area () is the area of the roof that the rain falls on.

Line 4

The first flush volume () is the volume of rain that is discarded because it is dirty.

Line 5

The tank capacity () is the maximum volume of the tank.

### Discussion

I decided to change roof area and first flush. Because many communities in Brazil will be lower-income, the roof areas will be smaller as houses will be smaller. I also increased first flush because it is more likely that the water will be dirtier, which means more water will need to be discarded. This will decrease the flow in of the mass balance equation, leading to the tank emptying much faster.

### Running the Simulation for One Year

Let’s run the model for the year 1981.

rainfall\_1981 = rainfall\_data.annual\_data[1981]  
results\_1981 = run\_timesteps(rainfall\_1981, param)  
p1 = plot(results\_1981)

Line 1

We select the annual rainfall data for the year 1981.

Line 2

We run the simulation for the year 1981.

Line 3

We plot the results, again using a plot function defined in [viz.jl](./RainwaterHarvesting/src/viz.jl).

To compare rainfall and tank volume:

p2 = plot(rainfall\_1981)  
plot(p2, p1, layout=(2, 1), size=(1000, 700), link=:x)

### Reliability Analysis

We can run simulations for all years, one at a time, and then check how many times the tank runs dry.

all\_years = sort(collect(keys(rainfall\_data.annual\_data)))  
all\_results = [run\_timesteps(rainfall\_data.annual\_data[year], param) for year in all\_years]  
any\_failures = [!isempty(result.failure\_dates) for result in all\_results]  
println("Number of years with failures: ", sum(any\_failures), " out of ", length(all\_years))

Line 1

We get all the years in order.

Line 2

We run the simulation for each year.

Line 3

We check if any failures occurred.

Line 4

We print the number of years with failures and the total number of years.

Number of years with failures: 10 out of 50

### Discussion

The system failed for 10 years. This means the system has a reliability percentage of 80%. This shows that increasing the first flush and decreasing the roof area has a negative impact on the functionality of the system. Because it is unlikely that most people in Brazil will have enough money to increase their roof areas, it is necessary to implement standards of cleanliness in neighborhoods in order to keep the first flush under control.

## Identify a Gap in the Methods

### Discussion

One assumption in the current model is that many drought years are not included. This underestimates the amount of years that will be failures because the flow in will actually be less than expected. In order to address this gap, data must be regularly collected every year and standardized. In doing so, we will achieve reliable data that we can then share to the public and implement water allocation measures for.