Homework 1 Solutions

> Due Date
>
> Friday, 9/8/23, 9:00pm

```
import Pkg
Pkg.activate(@__DIR__)
Pkg.instantiate()
```

```
using Plots
using GraphRecipes
using LaTeXStrings
```

```
[ Info: Precompiling GraphRecipes [bd48cda9-67a9-57be-86fa-5b3c104eda73]
```

Problems (Total: 40 Points)

Problem 1 (8 points)

You've been tasked with writing code to identify the minimum value in an array. You cannot use a predefined function. Your colleague suggested the function below, but it does not return the minimum value.

```
function minimum(array)
    min_value = 0
    for i in 1:length(array)
        if array[i] < min_value
            min_value = array[i]
        end
    end
    return min_value
```

```
end

array_values = [89, 90, 95, 100, 100, 78, 99, 98, 100, 95]
@show minimum(array_values);
```

```
minimum(array_values) = 0
```

Problem 1.1 (3 points)

Describe the logic error.

Solution: `min_value` is initialized at 0, which is less than any of the array elements, and so it will never be updated.

Problem 1.2 (3 points)

Write a fixed version of the function.

Solution:

```
function fixed_minimum(array)
    # this time initialize min_value to the first element
    min_value = array[1]
    # now compare to rest and update
    for i in 2:length(array)
        if array[i] < min_value
            min_value = array[i]
        end
    end
    return min_value
end
```

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

Problem 1.3 (2 points)

Use your fixed function to find the minimum value of `array_values`.

Solution:

```
@show fixed_minimum(array_values);
```

```
fixed_minimum(array_values) = 78
```

Problem 2 (8 points)

Your team is trying to compute the average grade for your class, but the following code produces an error.

```
student_grades = [89, 90, 95, 100, 100, 78, 99, 98, 100, 95]
function class_average(grades)
    average_grade = mean(student_grades)
    return average_grade
end

@show average_grade;
```

```
LoadError: UndefVarError: `average_grade` not defined
```

Problem 2.1 (3 points)

Describe the logic and/or syntax error.

Solution: There are two errors:

1. `average_grade` is a quantity from within the function, but then is referenced outside of it, which causes an error.
2. Within `class_average`, the input `grades` is not actually used, but just `student_grades`.

Problem 2.2 (3 points)

Write a fixed version of the code.

```
function fixed_class_average(grades)
    average_grade = sum(grades) / length(grades)
    return average_grade
end
```

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

Problem 2.3 (2 points)

Use your fixed code to compute the average grade for the class.

Solution:

```
@show fixed_class_average(student_grades);
```

```
fixed_class_average(student_grades) = 94.4
```

Problem 3 (8 points)

You've been handed some code to analyze. The original coder was not very considerate of other potential users: the function is called `mystery_function` and there are no comments explaining the purpose of the code. It appears to take in an array and return some numbers, and you've been assured that the code works as intended. :::

```
function mystery_function(values)
    y = []
    for v in values
        if !(v in y)
            append!(y, v)
        end
    end
    return y
end

list_of_values = [1, 2, 3, 4, 3, 4, 2, 1]
@show mystery_function(list_of_values);
```

```
mystery_function(list_of_values) = Any[1, 2, 3, 4]
```

Problem 3.1 (4 points)

Explain the purpose of `mystery_function`.

Solution: `mystery_function(v)` creates a new array which contains the unique values in the input array `v`.

Problem 3.2 (4 points)

Add comments to the code, explaining why and how it works. Refer to "Best Practices for Writing Code Comments", and remember that bad comments can be just as bad as no comments at all. You do not need to add comments to every line (in fact, this is very bad practice), but you should note the *purpose* of every "section" of code, and add comments explaining any code sequences that you don't immediately understand.

```
function mystery_function(values)
    # Starting with an empty array y, add new values
    # if they are not in y
    y = []
    # Check if the values do not exist in y and if they don't, append
    for v in values
        if !(v in y)
            append!(y, v)
        end
    end
    return y
end

list_of_values = [1, 2, 3, 4, 3, 4, 2, 1]
@show mystery_function(list_of_values);
```

```
mystery_function(list_of_values) = Any[1, 2, 3, 4]
```

Problem 4 (16 points)

Cheap Plastic Products, Inc. is operating a plant that produces 100m^3/day of wastewater that is discharged into Pristine Brook. The wastewater contains 1kg/m^3 of YUK, a toxic substance. The US Environmental Protection Agency has imposed an effluent standard on the plant prohibiting discharge of more than 20kg/day of YUK into Pristine Brook.

Cheap Plastic Products has analyzed two methods for reducing its discharges of YUK. Method 1 is land disposal, which costs $X_1^2/20$ dollars per day, where X_1 is the amount of wastewater disposed of on the land (m^3/day). With this method, 20% of the YUK applied to the land will eventually drain into the stream (*i.e.*, 80% of the YUK is removed by the soil).

Method 2 is a chemical treatment procedure which costs \$1.50 per m^3 of wastewater treated. The chemical treatment has an efficiency of $e = 1 - 0.005X_2$, where X_2 is the quantity of wastewater (m^3/day) treated. For example, if $X_2 = 50\text{m}^3$/day, then $e = 1 - 0.005(50) = 0.75$, so that 75% of the YUK is removed.

Cheap Plastic Products is wondering how to allocate their wastewater between these three disposal and treatment methods (land disposal, and chemical treatment, and land disposal) to meet the effluent standard while keeping costs manageable.

Problem 4.1 (3 points)

The flow of wastewater through this treatment system is shown in Figure 1. Modify the edge labels (by editing the `edge_labels` dictionary in the code producing Figure 1) to show how the wastewater allocations result in the final YUK discharge into Pristine Brook. For the `edge_label` dictionary, the tuple (i, j) corresponds to the arrow going from node i to node j. The syntax for any entry is `(i, j) => "label text"`, and the label text can include mathematical notation if the string is prefaced with an `L`, as in `L"x_1"` will produce x_1.

Solution:

```
using GraphRecipes, Plots

A = [0 1 1 1;
    0 0 0 1;
    0 0 0 1;
    0 0 0 0]

names = ["Plant", "Land Treatment", "Chem Treatment", "Pristine Brook"]
# modify this dictionary to add labels
edge_labels = Dict((1, 2) => L"X_1", (1,3) => L"X_2", (1, 4) => L"100-X_1-X_2", (2, 4) => L"
shapes=[:hexagon, :rect, :rect, :hexagon]
xpos = [0, -1.5, -0.25, 1]
ypos = [1, 0, 0, -1]

graphplot(A, names=names,edgelabel=edge_labels, markersize=0.15,
    markershapes=shapes, markercolor=:white, x=xpos, y=ypos)
```

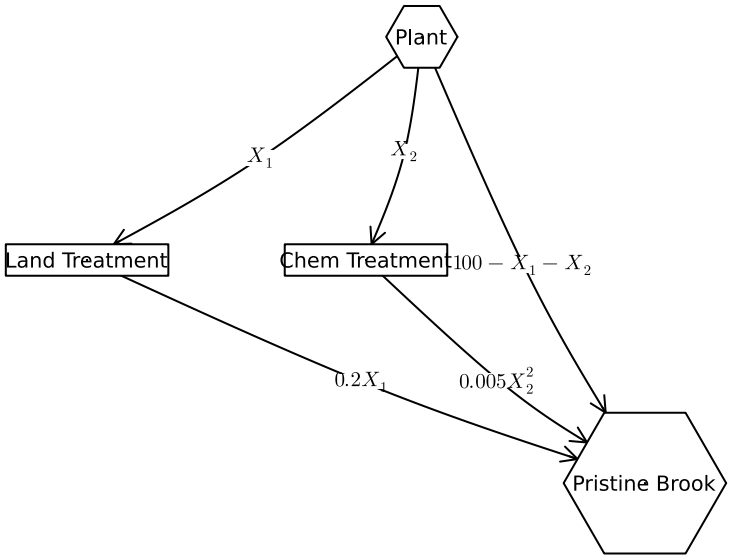

Figure 1: System diagram of the wastewater treatment options in Problem 4.

Problem 4.2 (4 points)

Formulate a mathematical model for the treatment cost and the amount of YUK that will be discharged into Pristine Brook based on the wastewater allocations. This is best done with some equations and supporting text explaining the derivation. Make sure you include, as additional equations in the model, any needed constraints on relevant values. You can find some basics on writing mathematical equations using the LaTeX typesetting syntax here, and a cheatsheet with LaTeX commands can be found on the course website's Resources page.

Solution:

These equations will be derived in terms of X_1 and X_2, where $X_1 + X_2 \leq 100$.

The amount of YUK which will be discharged is

$$D(X_1, X_2) = 100 - X_1 - X_2 + 0.2X_1 + 0.005X_2^2 = 100 - 0.8X_1 + (0.005X_2 - 1)X_2.$$

The cost is

$$C(X_1, X_2) = X_1^2/20 + 1.5X_2.$$

Problem 4.3 (4 points)

Implement this systems model as a Julia function which computes the resulting YUK concentration and cost for a particular treatment plan. You can return multiple values from a function with a tuple, as in:

```
function multiple_return_values(x, y)
    return (x+y, x*y)
end

a, b = multiple_return_values(2, 5)
@show a;
@show b;
```

```
a = 7
b = 10
```

Make sure you comment your code appropriately to make it clear what is going on and why.

Solution:

```
# we will assume that X , X  are vectors so we can vectorize
# the function; hence the use of broadcasting. This makes unpacking
# the different outputs easier as each will be returned as a vector.
# Note that even though this is vectorized, passing scalar inputs
# will still work fine.
function yuk_discharge(X , X )
    # Make sure X  + X  <= 100! Throw an error if not.
    if any(X  .+ X  .> 100)
        error("X  + X  must be less than 100")
    end
    # These calculations use broadcasting
    yuk = 100 .- 0.8X  .+ (0.005X  .- 1) .* X
    cost = X .^2/20 .+ 1.5X
    return (yuk, cost)
end
```

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

Problem 4.4 (5 points)

Use your function to experiment with some different combinations of wastewater discharge and treatment. Can you find one that satisfies the YUK effluent standard? What was the cost? You don't have to find an "optimal" solution to this problem, but what do you think would be needed to find a better solution?

Solution:

We left this intentionally open for you to conceptualize how to generate combinations and to look into different ways of implementing these in Julia. For a more systematic approach, we can sample combinations from a Dirichlet distribution, which samples combinations which add up to 1. This will require installing and loading the `Distributions.jl` package (we will spend more time working with `Distributions.jl` later).

```
# Install and load Distributions.jl
Pkg.add("Distributions")
using Distributions
```

```
# The 3D Dirichlet distribution with parameters equal to 1
# is basically uniform over these combinations.
# See: https://juliastats.org/Distributions.jl/stable/multivariate/#Distributions.Dirichlet
yuk_distribution = Dirichlet(3, 1)
# Need to scale samples from 0 to 100, not 0 to 1
yuk_samples = 100 * rand(yuk_distribution, 1000)
D, C = yuk_discharge(yuk_samples[1,:], yuk_samples[2, :])

# Plot the discharge vs. cost and add a line for the regulatory limit
# We also turn off the legend as we don't need it for this plot
scatter(D, C, markersize=2, legend=:false)
vline!([20], color=:red)
# Label axes
xaxis!("YUK Discharge (kg/day)")
# For the y-axis label, we need to "escape" the $ by adding a slash
# otherwise it interprets that as starting math mode
yaxis!("Treatment Cost (\$/day)")
```

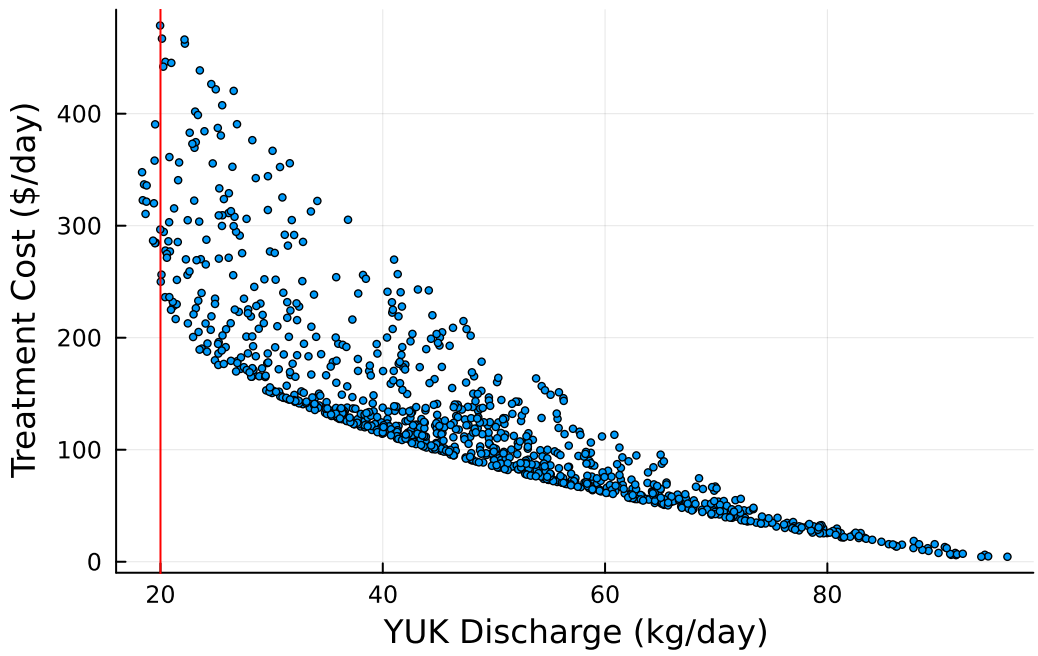

Figure 2: Sampled solutions for the wastewater allocation problem in Problem 4, showing cost vs. YUK concentration. The red line marks the regulatory discharge limit of 20kg/day.

We can see (to the left of the line in Figure 2) that we have several wastewater allocations which do comply with the regulation. We can look more closely as to which solutions these are and how much they cost.

```
# Note some of the options:
# - zcolor colors the points based on another variable (in this case cost)
# - label=:false turns off the legend for the points (we don't have multiple series)
# - seriescolor sets the colorscheme for the colorbar (in this case, reds)
#   see https://docs.juliaplots.org/latest/generated/colorschemes/
# - in the colorbar title, we need to "escape" the $ by adding a slash
#   otherwise it interprets that as starting math mode
scatter(yuk_samples[1, D .<= 20], yuk_samples[2, D .<= 20], zcolor=C[ D .<= 20],
colorbar_title="Cost (\$/day)", seriescolor=:reds, label=:false)
# Add axis labels
# We use the LaTeXStrings.jl package to render variables, subscripts
# and superscripts. The key is the use of the L in front of the "",
# which will render anything inside $ $ as though they are LaTeX.
xlabel!(L"Land Disposal (m$^3$/day)")
ylabel!(L"Chemical Disposal (m$^3$/day)")
```

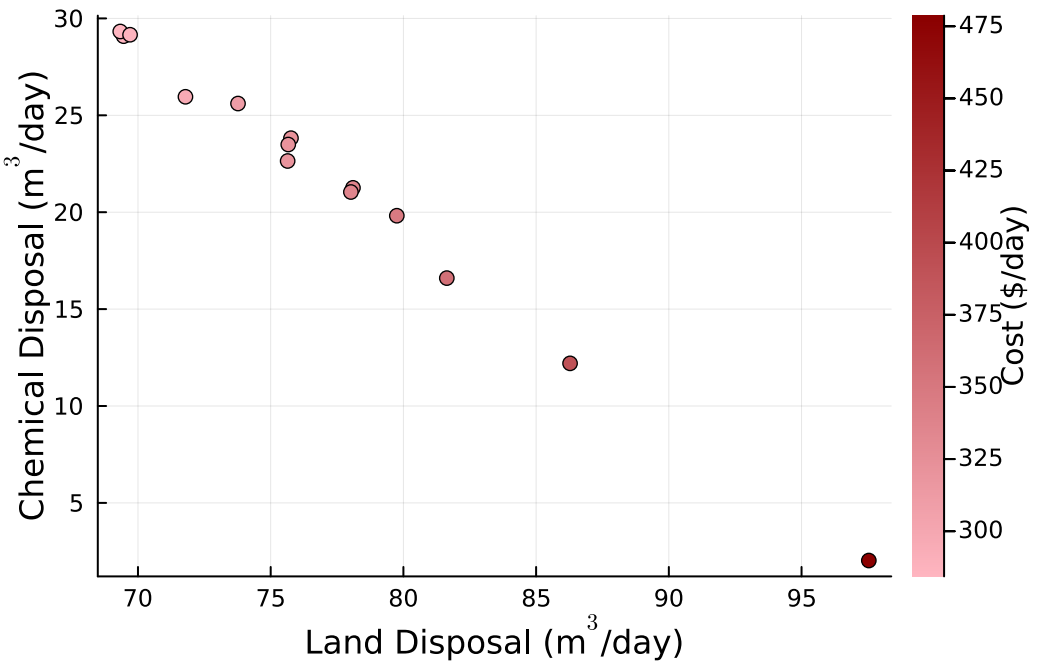

Figure 3: Solutions for our wastewater problem which comply with the YUK regulation. The x-axis (X_1) is the land disposal allocation, the y-axis (X_2) is the chemical disposal allocation, and the colors correspond to the cost of that strategy.

From Figure 3, we can see that compliant solutions which rely more on the chemical disposal method (larger X_2) are generally more cost-effective, though the minimum cost is still around \$275/day.

Depending on how you analyzed the problem, there are different solutions to the question of what additional information is needed. In our case, we might benefit from having some idea of the treatment budget, as some (or all!) of the solutions might be cost-prohibitive.

References

List any external references consulted, including classmates.