Introduction to Computer Programming In Julia

Final Assignment

Deadline : 22:00 on 24 April 2022,

Write efficient, well-commented code that completes the following 4 tasks.

The total score for all tasks is 100. A minimum score of 40 is needed to pass.

Your coded answers should be saved as a single Pluto julia file with the title format:

FirstName_LastName.jl

Submit your code by email to:

matthew.flood@lih.lu and formation@lih.lu

There are several datasets needed to perform the tasks. These can be downloaded from the course GitHub repository:

https://github.com/MattWillFlood/Introduction-to-Computer-Programming-in-Julia/tree/main/Exam

To complete these tasks, you must have the following packages installed and imported:

DelimitedFiles
CSV
DataFrames
Plots
Statistics
StatsBase
StatsPlots
LinearAlgebra

Task 1: 20 points

This dataset *Olympics.txt* is a tab delimited file containing the number of medals won by countries at the summer and winter Olympic games.

The columns correspond to:

1. Country Code
2. Number of summer Olympics attended
3. Number of summer Olympic gold medals
4. Number of summer Olympic silver medals
5. Number of summer Olympic bronze medals
6. Total number of summer Olympic medals
7. Number of winter Olympic attended
8. Number of winter Olympic gold medals
9. Number of winter Olympic silver medals
10. Number of winter Olympic bronze medals
11. Total Number of winter Olympic medals

1. Import the dataset *Olympics.txt*.

2. Write a function called *MedalScore* using the simple, single line format. *MedalScore* should accept 4 required arguments: *Games*, *Gold*, *Silver* and *Bronze*. *MedalScore* should return weighted average of medals won, i.e.

$$MedalScore = \frac{3 * Gold \; + \; 2 * Silver \; + \; Bronze}{6 * Games}$$

3. Write a *for loop* that iterates through each row of the Olympics dataset.

   For each iterated row, calculate the *MedalScore* for summer and winter games, and compare them using an *if-else* statement.
   If summer > winter, print to the REPL: "*Country* prefer the heat"
   If summer < winter, print to the REPL: "*Country* prefer the snow"
   Otherwise, print to the REPL: "*Country* don't have a preference"
   where *Country* is the corresponding country code in the dataset,
   e.g. *LUX prefer the snow*

Task 2: 20 points

The file *Chris_Actors.txt* contains <u>comma</u> delimited time series of the relative google search term popularity for the actors: Chris Pine, Chris Hemsworth, Chris Evans, Chris Pratt

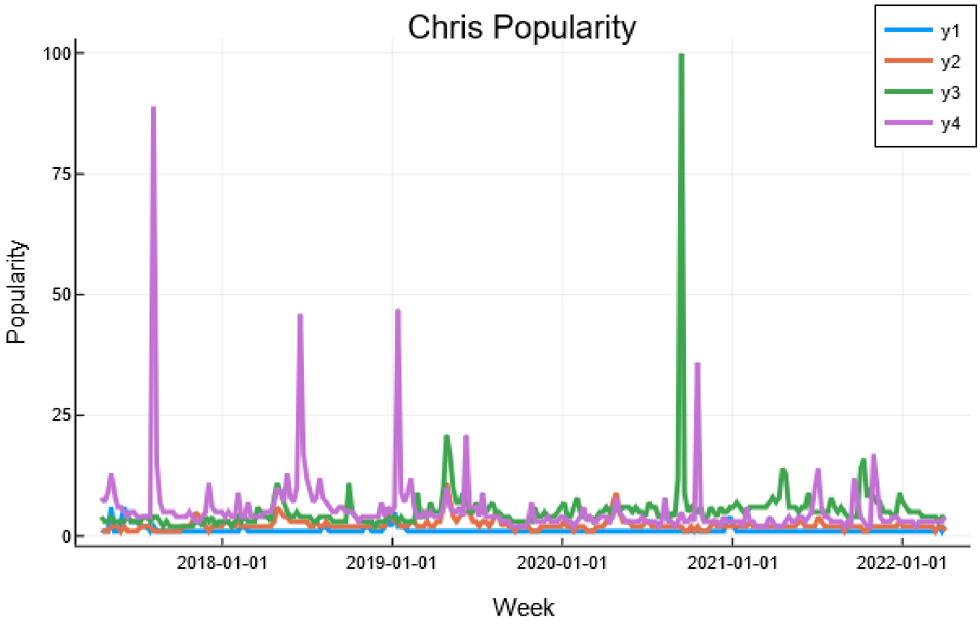
1. Import the dataset *Chris_Actors.txt*

2. Using the *Dates.jl* package (installed by default with Julia), create a new variable called *WeekX*, which is the data in the first column of the dataset, converted into a DateTime object.
   e.g. *WeekX = DateTime.(…)*

3. Create a second variable called *Popularity* which contains the remaining columns of the dataset converted into an array of integers.

4. Calculate the standard deviation of the popularity of each Chris.

5. Plot *Weekx* against *Popularity.* Include a title, x-axis label and y-axis label. Specify a line thickness of 3pt. It should look similar to this:



Task 3: 30 points

The file *Population.csv* contains semi-colon delimited array of European countries and their projected populations between 2025 and 2100.

1. Using *readdlm()*, import the dataset *Population.csv.*
   The data should be imported as a variable called *Population* and the headers should be imported as a variable called *Years.*

2. Create a new variable called *PopNumbers* from all but the first column of *Population* (the numeric data). Convert the values of *PopNumbers* to integers by:

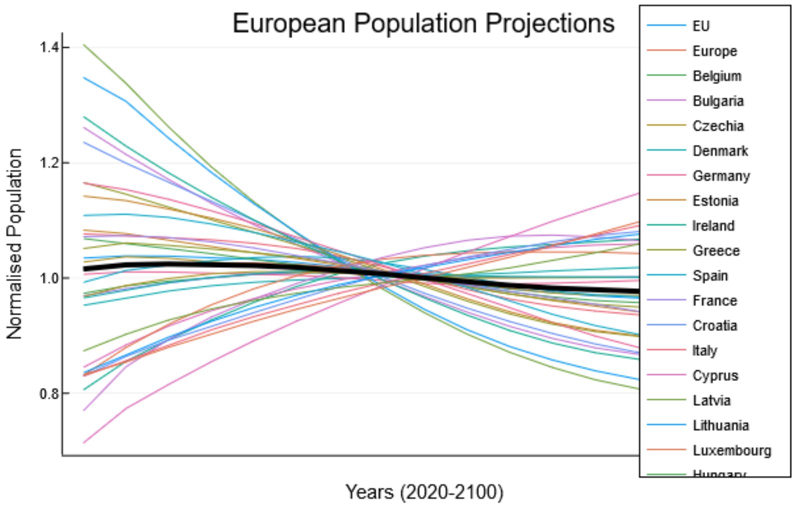
   a. splitting each string element by empty spaces into substrings [split()]
   b. rejoining the substrings [join()]
   c. parsing the rejoined string elements into integers [parse()]

   Bonus points if you do all these steps in a single line of code.

3. Create a new variable, *PopNorm*, that is *PopNumbers* normalised by dividing the population of each country (Pop$_x$) by its mean over all years (N).

$$\widehat{Pop_X} = \frac{Pop_X}{\sum Pop_X / N}$$

4. Recreate (as much as you can) the plot below with the *PopNorm* data, where the black line represents the mean of all countries.



   Hints: *adjoint(), plot!(), label=reshape(__ , 1, 34), linewidth=3, colour=:black*

5. Find the year where the variance of all normalised populations reaches a minimum.

Task 4: 30 points

The file *Mercury_Prize_Winners.csv* contains the track listings for each Mercury Prize winning albums from 2014-2021.

1. Using CSV, import *Mercury_Prize_Winners.csv* as a dataframe object called *Mercury*.

2. Group *Mercury* into sub-dataframes by album name [groupby()]

3. Create a composite type (struct) called *Hit* with 4 fields:

• *NumTracks*	an integer, representing the number of songs in the album
• *Artist*	a string, representing the name of the artist
• *MinLength*	a Time object, representing the shortest song length
• *MaxLength*	a Time object, representing the longest song length

4. Write a function called *HitMaker* that takes in a sub-dataframe from step 2 (required argument), and a second keyword string argument.
   Hitmaker should check to see if the keyword string argument appears in any of the song titles of the album. If it does, then a message should be printed to the REPL to say that word ___ appears in song ___ by artist ___.

   For example: “Bullet appears in ‘Just Another Bullet’ by Young Fathers”

   Bonus points if you use the single line *if-else* statement here.

   Include a try-catch statement in your function to catch if a user passes a non-string type as the keyword argument.