CS4171 Programming Assignment II

DueDate: 18.00, Tues. Week15

Objective: We saw previously how eirgrid, maintains data at 15-minute intervals on the nation's electricity supply. The assignment is to generate plots of two sets of data that originated with eirgrid but that will be available to you in `~cs4171/proj/proj02`. The two files `wind-gen.csv` and `system-demand.csv` track, respectively, the wind generation contribution to total electricity generation – both what was actually generated and what was forecasted, and the system-wide electricity demand. All figures are in megawatts, MW.
Your job: You should generate two plots of these two datasets and save them, as `.png` files, respectively, `wind-gen.png` and `system-demand.png`. The first plot should be of the wind data, showing both actual power generated and what was forecasted. These are the two "interesting" columns of `wind-gen.csv`. Then you should generate a brand new heatmap figure using the *actual* power consumption data in `system-demand.csv`. So, while you will need both the *actual* and *forecast* columns from `wind-gen.png` you will only need *actual* from `system-demand.csv`. (You can immediately throw away the unused dataframe columns if you wish.)

Because data are provided up until midnight of each day the `actual` data may not yet be known so a `-` is given instead. This can be verified by going to the bottom of the two files and observing the `-`s in that column and this presents our first problem. Since the dataframe reader will make the decision that since there are non-digits in that field *everything* should be strings, all numbers get stored in the dataframe as their string equivalent. To convert back to `Int`s here's what I did:

```
wind = CSV.File("wind-gen.csv") |> DataFrame
sysdem = CSV.File("system-demand.csv") |> DataFrame

mondays = map([wind, sysdem]) do df
    select!(df, Not([:region]))

    map([:forecast, :actual]) do data
        df[!,data] = map(df[!,data]) do d
            d == "-" ? missing : typeof(d) == Int ? d : parse(Int,d)
        end
    end

    # logic here to figure out dates, times, when is first Monday, etc.
end
```

The deal about Mondays is that to keep the figures tidy and to avoid clutter your `xticklabels` should only be displayed for Mondays, the first day of the week. Yet you cannot be guaranteed that the test data will start on Monday. Have a mooch around the `Dates` documentation and you will be able to find the function that tells what day of week `df.period[1]` occurs on. From that you can figure out when the first Monday will be. But be careful to properly count the number of periods reported for each day – that bug bit me first time out! The figure below in Fig. 1 would be an acceptable output for this part of the assignment. Table 1 later shows the marks breakdown for this part of the assignment.

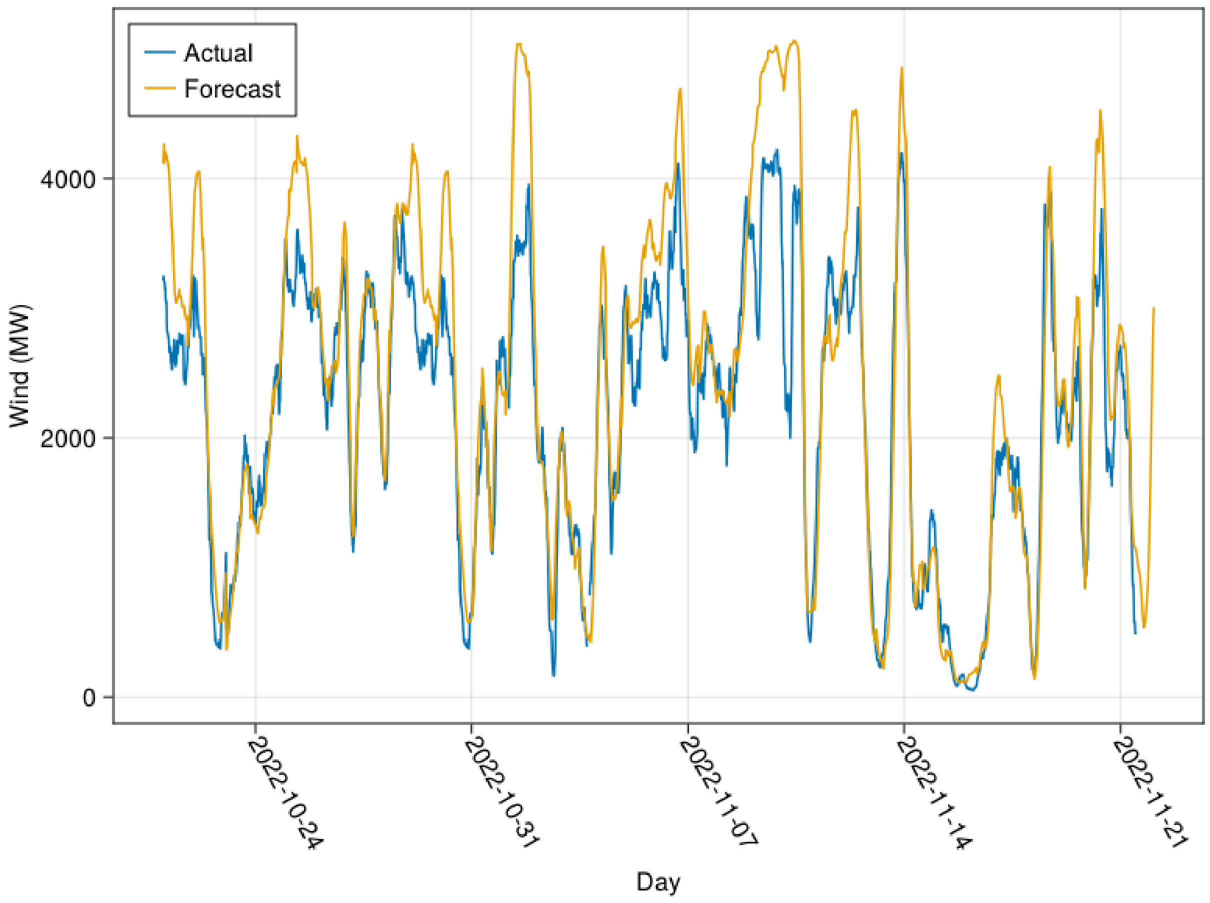


Figure 1: A typical line graph plot for wind generation.

The second part of the assignment is to display, as a heatmap, the island-wide electricity demand for the period covered by the contents of the `system-demand.csv` file. Since each day is broken into 4 × 24 periods this would translate to almost 100 rows across. This is unwieldy and probably at a finer resolution than is needed as a visual aid. So you should condense the data so that only hourly demand is shown. You can round your figures to the nearest integer when displaying in the grid, as you did in the lab.

Please don't assume that the start / end dates of the two files coincide. Also, *please be aware* that I will be adding to the two .csv files so please don't make any assumptions about start / end dates therein.

Marking: The project will be worth 15% of your final grade. It will be marked out of 30 and the marks breakdown will be as follows:

Category	Sub-Category	Mark	Details
Submission		5	
	`electric-summ.jl` present		5
Wind		10	
	Plot generated in `png`		4
	Only one day per week shown		4
	First Monday shown correctly		2
Demand		15	
	Plot generated in `png`		3
	Only one day per week shown		3
	First Monday shown correctly		1
	Text values in heatmap grid cells		3
	Hourly averages shown		5
In iomlán		30	30

Table 1: Marks breakdown for PA II.

Execution: You should put all of your code in a single file called `electric-summ.jl` and place it in `~/cs4171/proj/proj02` as this is where it will be collected from. Your program should run in its entirety by you issuing the command

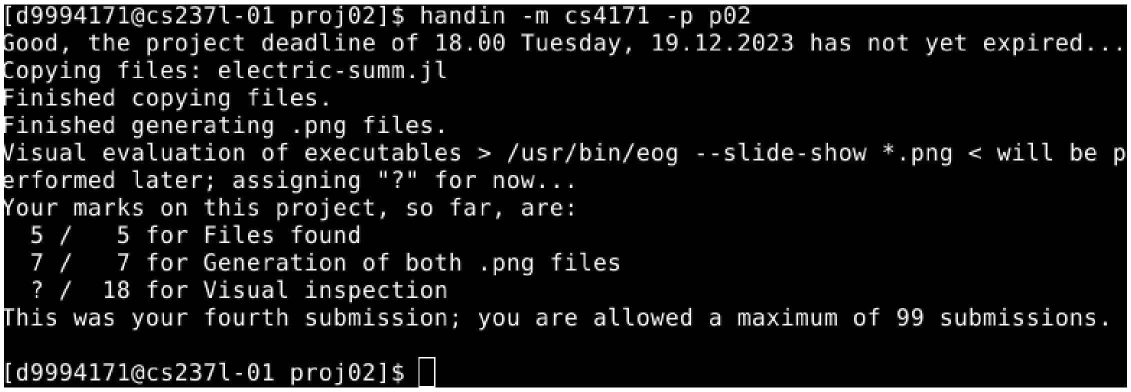
```
julia electric-summ.jl
```

Any output issued to the screen will be ignored in the marking process.
Due Date: The project will be handed in no later than 18.00, Tues. Week15.

Submitting: The project will be handed in (submitted)using the `handin` mechanism that we run here in the CS dept. The idea is that you "push" your code to us, we run it through the julia interpreter automatically and we test if the output files exist. You get a report at that point telling you if it generated the `.png` files successfully. Then, after the deadline has passed, we visually inspect the graphics for coherency, etc.

You will get a report telling you if a) your code file was found and, b) whether that code generated the two graphics. The visual inspection will be done later.

Here's a screenshot of a successful handin session:



```
[d9994171@cs237l-01 proj02]$ handin -m cs4171 -p p02
Good, the project deadline of 18.00 Tuesday, 19.12.2023 has not yet expired...
Copying files: electric-summ.jl
Finished copying files.
Finished generating .png files.
Visual evaluation of executables > /usr/bin/eog --slide-show *.png < will be p
erformed later; assigning "?" for now...
Your marks on this project, so far, are:
  5 /   5 for Files found
  7 /   7 for Generation of both .png files
  ? /  18 for Visual inspection
This was your fourth submission; you are allowed a maximum of 99 submissions.

[d9994171@cs237l-01 proj02]$ 
```

Good luck.