Fuel Factsheet Notes and Queries

---

1 Overview

[Questions/querries are in the second part of this document.]

1.1 Outline

The `/code/` folder contains:

- `outbound-master.R` - the script to be run: this is the only file you need to open and run.
- `/scripts/` - there are five sub-scripts in here, which are the workhorses of the master script.They
  - 1. load the required packages,
  - 2. download the data and clean it up,
  - 3. perform the calculations,
  - 4. create the excel file and
  - 5. send the email.
- `/tests/` - there are four test scripts in here, which are called from their respective scripts to test everything went OK in each one (except the email script doesn't have a test).

The idea is that you place the `/code/` folder next to the `/data/` folder in `K://RESEARCH/Data/Fuel Factsheet/`.

1.2 Flow

In order to run the script you simply source the `outbound-master.R` file. But read the instructions at the top.

There are two decision points in the flow:

1. Whether or not the script is being run from a work computer or a remote computer.
2. Whether or not to proceed if sanity checks fail.

ad 1. Work/home PC: the first decision is dealt with manually in the `outbound-master.R` script by setting the logical flag `work.computer` to `TRUE/FALSE`, depending on your circumstance. Additionally, if you are not at work, you need to make sure the data folder, and email and password data are also entered correctly. **Right now they are not, because I don't have working credentials.**

ad 2. Insanity checks: The second decision happens during the script execution. All other tests–if they fail–stop the script. But if the sanity checks fail e.g. the petrol price went up by more than 5p in a week, you get the option to either stop the script or continue it anyway. Because insane things happen sometimes.

1.3 Testing

During testing of this script keep in mind the following:

1. Careful about the folder structure.
2. You are not using on the real googlesheet.
3. I wasn't able to test the emailing script.

ad 1. working directory: the `/code/` folder with the scripts should be at the same level as the `/data/` folder (otherwise a new `data` folder will be created). The script will then automatically set the working directory once it is sourced (to the folder that contains both of these). Even if the /data/ folder does not exist, it will create it. This I believe, only works if you use RStudio. If that is a problem, we can find another solution where you manually do `setwd()` or sth.

ad 2. The test googlesheet: at the moment the data is being pulled from a test googlesheet, not the actual one. The reason for this is so that the test googlesheet, which is found on this link, can be manipulated to cause tests to fail and test the script.

The test googlesheet contains the four sheets required to perform the script, as they appeared in the googlesheet from 22.1.2020. This means one of the tests that will fail is the one to test if the data is from yesterday. Because this is just a sanity test, you can proceed anyway, and this also shows you how sanity tests work.

If you don't want to play around with the test googlesheet you can switch to the real one by going into `/scripts/02-download.R` and comment out the test gs assignment in rows 15-17 while uncommenting the actual gs assignment in rows 9-12.

ad 3. The emailing: since I don't have working credentials I wasn't able to test the remote emailing script, and since I am also not on a "work PC" I wasn't able to test the Outlook one either. In principle they should work though, since they are the same as they were before, although I might have made some aesthetic changes, hopefully it will work anyway..

Either way, the script can be tested up until that point even if the email fails. If the email works, it is currently only set to send to Ivo, not everyone else.

2 Queries and Notes

Overall what I've done is moved all possible calculations from the googlesheet to the script. This is cleaner and more robust. It also means I have removed a bunch of redundancy where a calculation was performed in the google sheet and then again in the script and the results were compared.. This sort of testing did not really serve any purpose, it is not clear what sort of error it is anticipating and what one would do if it occurred even. So none of this is happening any more.

I am therefore only importing data from the following four sheets from the googlesheet. They are listed here along with the numbered outputs they relate to.

- **Max/min fuel working**, "E1:G260": 2, 3, 6–13, 22–34, 50–67
- **Oil Price**, "A2:D260": 4, 5, 14–21, 36–49,
- **UK vs EU Fuel**, "A46:I73": 68–69
- **UK vs EU Fuel**, "A3:G30": 70–77

Current checks include:

- Data integrity tests: whether the tables exist, and whether there are any missing values or wrong data types in any of the four tables mentioned above. This stops the script.
- whether the tables are appropriately sized (col and row numbers) and contain e.g. a year's worth of data. This also stops the script.
- Sanity checks: if prices are in reasonable ranges. This does not stop the script, but warns the user and asks if they want to proceed.
- Export checks: to see if export tables are appropriately sized (these checks are actually completely redundant, they are only here for sake of symmetry: so each script has a test script haha).

2.1 Questions:

1. I am not clear on how exactly does the googlesheet come into being? This would be good to know in order to be able to anticipate the sorts of errors I should be testing for.
2. What explicit tests/checks would you like included?
3. We are currently checking the prices are lower than the all time highs. I've moved these values as parameters at the start of the main script, so you can change them if need be. There's max petrol, max diesel and max oil. Are there any others that might be useful here?

4. The VAT and duty rates are now entered as parameters so you can change them if need be. Is it OK to assume they will be the same for both petrol and diesel?

5. If it's a Monday when the script is run, should that stop the script or just trigger a warning?

6. If the data is not from yesterday when the script is run, should that stop the script or just trigger a warning?

7. There seem to be some inconsistencies in the googlesheet about how to determine which day is 1 week, 1 month or 6 months ago. e.g.:

- pump prices one month ago: currently goes one month back, if that is a Sunday it takes the Friday before it, but if it's a Saturday, it takes the Thursday before it?!
- full tank price: the 6 months ago calculation in the google sheets seems to be 6months + 1 day. Always, regardless of the day. So 6months ago from 28.1.20 is 29.7.20. Is this OK? Because that's not what's happening with one month.
- full tank price: if the day falls on a Saturday or Sunday it takes the Friday. Which makes sense but is not the same as how the pump price is done.
- pump prices for the previous week are done looking 7 days back, but if that doesn't exist, then 6 days. will this always work? Also, it's not consistent with the month/6 month approach, where an earlier date is looked up if there is no data for the exact date, not a later one like here.

9. Do you want 1 week/1 month/6 month lookups to all be performed the same way? Do you want pump price and tank price lookups to be performed the same way?

10. Do you want the data for the first available date that is *at least* X days earlier? Or *at the most* X days earlier?

11. The oil price over last 12 months chart: There's an error in the visualisation: the minimum is for $/barrel, but is drawn on the £/barrel line. Maybe it's not an error, but it's an awkwardness. One option is to print out both the £/barrel and the $/barrel prices, and draw a vertical dashed line connecting both points. It is theoretically possible that lowest price in $ is not the same day as the lowest price in £ though. So just a warning, this isn't really related to the script per se.

12. Am I ok to assume the first date on the petrol/diesel price table always the same as the first one in the oil price table?

13. is the oil price table always exactly 12 months long? i mean can i just take the last row in the table to get last year's price?. Same question for pump prices (**Max/min fuel working**, "E1:G260"). How are these worksheets created, can I rely on them being 12 months long?

14. The output is now a single Excel file with five worksheets:

- *master* - which contains, numbered and labelled, all the values except the chart and EU tables.
- *pump chart* - contains the petrol and diesel price data
- *oil chart* - contains the oil price data
- *eu.rank.p* - contains the eu ranking by petrol price
- *eu.rank.d* - contains the eu ranking by diesel price

Is this OK or would you prefer sth different? I can split the master sheet into sub-tables if it's easier for the designers?

15. In the master worksheet, should there be units next to the values?

16. Flow and folder structure. Right now the script places the excel file into "../data/2020" relative to the `outbound.master.R` script. This means that wherever you put the `code` folder, the `data` folder will be created (if it's not already there) next to it. Regardless of whether you are at home or at the office. Is that cool or would you prefer sth different?