Journal: RAC Fuel Factsheet—Outbound

Wednesday 28.8.2019

1. Setup project and repo
2. Initialise packrat with options

- vcs.ignore.src: TRUE
- auto.snapshot:TRUE **todo: check this works!**
- initialisation successful but Warning messages:

```
1: In untar(src, exdir = target, compressed = "gzip") :
  argument 'compressed' is ignored for the internal method
2: In untar(src, compressed = "gzip", list = TRUE) :
  argument 'compressed' is ignored for the internal method
```

3. Started getting an odd error on saving: "no such file or directory" which just happened constantly, not even when saving and usually crashed R. Restart of the machine seemed to help, since the suggestions here didn't seemm to. This is what my makefile looks like graphically at the minute:

4. Install knitr and dependencies to be able to compile Rmds.

5. CHeck packrat.lock, which seems to have not changed, meaning automatic snapshot doesn't work?

6. Try manual snapshot, but get error

```
Error: Unable to retrieve package records for the following packages:
- 'base64enc', 'digest', 'htmltools', 'jsonlite', 'rmarkdown', 'tinytex'```
```

7. Ah, OK, seems these were needed by knitr, so as soon as i ran knit on this journal the packages were installed.

8. Run `packrat::snapshot()` which seems to work fine. all good. Although there is no automatic snapshots it seems.

9. Clean up makefile. I think I'll try building this whole project with make to keep it more managable and easier for RAC to handle as well. **todo: potential portability issue: the command `dot` used in creating the makefile graphic, which is from graphviz. so that would need to be installed. Run "dot -V" from the command prompt to check. but this can be removed from the makefile once porting, since it won't change later anyway**

10. Back to packrat. Seems that I am unnecessarily using the `infer.dependencies=TRUE` default setting, which checks for the versions of all dependent packages. But is it bad? This guy seems to think so. **todo: should i migrate to renv instead? seems packrat is out..**

11. In code/sandbox, walk through the crop.R script. This is pretty straightforward, crops predetermined rectangles from the factsheet and adds a source/date annotation.

12. Walk through main outbound script:

- RDCOMClient package is used to send emails from Outlook in Windows. Won't be able to test that. **todo: Unless I set up a virtual machine..**

Thursday 29.8.2019

1. walk through outbound script:

- `gs_url` works fine, no authentication required. registered googlesheet?
- `gs_read` downloads individual worksheets.

- Missing column names warning messages! **can be fixed in googlesheet**
- `googlesheets` lifecycle is down as **retired** - time to move to googlesheets4? but not on CRAN yet.
- ok, so the unit testing looks like it could be cleaned up using `tinytest`? **todo: read up on other options**

2. `tinytest` or `testthar`

- doesn't stop the script but simply saves the errors.
- unit testing outside package envirnoments e.g. here

3. Logical testing

- includes manually input numbers for max price, duty rates.. **what if these change?**

4. error/typo : (?/barrel) instead of (£/barrel)

5. sheet data testing: all or nothing, doesn't specify error.

6. saving to excel (some weird date calculations haha)

Outline of script: 1. download and save data from googlesheet. 2. functions 3. checking data * pump price data: + remove any non-numeric values + check all the cells have legal values + STOP if not. + logical tests * etc. all the worksheets 4. creating custom data frames 5. write to xls file

7. email Ivo:

   The deliverable is a script/repo/package that does the following: 1. downloads the data (this is already OK) 2. does unit testing with reporting e.g. using testthat or tidytest (this is to be completely rewritten) 3. does necessary calculations (this is already OK) 4. create a list of edits (this is to be completely rewritten) 5. sends them to the designers and archives them (this is already OK)

   So it's point 4 that I'm just not completely clear on. You want the edits to be passed on in a more user friendly format, not the current excel spreadsheet? This is what you mean by "email text and attached tables of data etc"? And do you want to use the numbering of the elements you sent in the attached pdf? The idea being presumably to make Nick's life easier?

8. OK, what the heck, try a new repo with `renv` instead of `packrat`...

9. Still not clear on packrat - is the infer.dependencies setting really necessary? it means it fetches sources for dozens of packages in addition to the ones I use. I think.

10. OK, first i'll set ignored directories in packrat options to ignore most everything.

Friday 30.8.2019

1. OK, there was a bug in renv, solved wiht this issue but now it seems to work.

2. restart repo and reinitialise. This time before committing remove Bhavin's password...

3. Actually had to reinitialise renv, because it has to happen after cloning the repo from github, that way it has a `.gitignore` file for renv to amend.

4. Now commit

Modnay 6.1.2020

1. OK, not sure what was going on before, i'll remove the packrat folder and initialise renv.

2. This again didn't go completely smoothly and required a sequence of install.package (not successfully), remove.package, re-initialise renv before stopped giving a knitr error. Seems ok now. But it is not.

3. Could it be that i had an old version of renv? i mean i clearly did. but i also know i'd already checked it before. but i guess that was checked inside a renv project? so that means it was updated there, but not globally? I really need to get a handle on package management in R. OK, reinstalled renv, reinitialised package, hopefully it works now.
4. Now figure out what i have already done here.
5. OK, so outbound is before crop presumably, i have the order wrong in the sandbox.
6. OK, start cleaning up the 01-outbound code.

Tuesday 7.1.2020

1. clean through read and assign data code. consistency, readability.
2. Data import works, but there are warning messages about missing column names being filled in X1 etc.
3. So, many calculations are being made in the google sheet already. So there are intermediate results there, as well as non-tidy tables and cells that are used to produce outputs to then import into R. But some are imported several times over. e.g. maxmin and lastweek.
4. I don't like this stuff happening in the google sheet. I'd much rather just import the raw data and manupulate it programatically in R as opposed to using googlesheet formulas. That also reduces the possibility of googlesheet error.
5. For example the fuel price over time ten year maximums are not actually calculated in the googlesheet but are fixed. Maybe that's OK now, but what if the prices are surpassed? Who will know to manually fix that? Or what happens in 2022, when the maximum is no longer within the past ten years?
6. Some of the checking seems to be duplicating work and then comparing to itself. I'm not sure how that contributes much. Like e.g. oil max min seems to work like this: (i) the raw data is in the gs. (ii) the max and min are caclulated in the gs. (iii) the data is imported into R. (iv) the max and min. are imported into R. (v) the max and min are calculated again in R from the data imported in R. (vi) the calculated and imported max and min are compared.
7. Why would you do this? What kind of error is this anticipating? In the case of an error, which calculation wins out? I would just do the calculation in R.
8. OK, the facsheet seems to have 77 fields, some are single values, others multiple ones e.g. time series. A tidy way of outputting this is a sinlge numbered long table, with a field ID number, description and values, plus an index variable for fields with multiple values.
9. What are we checking? Sanity checks. OK. But i'm not writing tests to check my code. or the googlesheet code.
10. OK, so let's just import the raw tables that are necessary for all the other calculations. That means:

- `pump.prices` which is the last years worth of petrol and diesel prices.
- `oil.prices`, same for oil
- `basil`, which is some other raw data we need
- `taxes`, which lists duty and VAT levels
- `eu.compare` which lists the pump price ranking of all eu countries

All the other variables etc. we will calculate in the script, not import them from the googlesheet.

11. The fuel prediciton is not derived in the sheet, but somewhere else i gather. But i also don't see it on the factsheets, so i'm ignoring it for now.
12. OK, so only the raw data gets imported, now figure out the testing
13. Have a look at the unit testing and look at tinytest and testthat to decide what if anything is approopriate.

14. So there are four functions in the script:

- `error check` is checking if there are any NAs in the cells.
- `sheet check` then just aggregates several error checks to see if they were all OK.
- `gg.convert` removes any remnnants from the google sheet and makes sure there are only numbers and NAs left. This one could be useful.
- `date.check` checks if the if the date is yesterday.

15. OK, so now I've got just the one function `Fun.gs.clean` that replaces any googlesheet errors (which all start with a hash #) with an `NA`. Tested it on the weekly fule worksheet, which currenlty has those errors for whatever reason. Wokrs a treat.

16. Other checks:

- are petrol and diesel more than 10p apart? - would you really want to stop, or just get a warning?
- check the petrol and diesel prices are lower than their all time highs. so what if they are, what do you want to do? are these sanity checks or should sth be done?
- duty rates are correct. if they are what you say they are. why are you importing them then? what if they actually change, why would you want the code to stop if they do?
- checking date, checking week before is really week before. but this is again r code checking googlesheet code, which is silly.
- then there is another fuel price check, repeating the lower than max check. and adding a higher than 90 check. again. what if it happens. why is this being checked.
- also check the weekly difference isn't larger than 5p. otherwise stop. these should all be warnings for sure.
-

17. OK, so it seems that looking up prices for the previous week means looking 7 days back, but if that doesn't exist, then 6 days. Is that always OK? Anyway, means I have to treat dates as dates, not characters.

18. OK, now i'm having trouble becuase the clean function is changing everythign to character vectors instead of keeping them numeric if they started out that way. OK, fixed.

19. First five fields done. still haven't looked at unit testing per se. not sure if i need to for this?

20. commit

wednesday 8.1.2020

OK: unit testing reading.

- not hugely useful, but testthat example outside package
- maybe `assertr` is what i really need for verifying assumptions about the data.
- `assertr` looks interesting, since it's focussed more on data shape and form, not on testing functions. but it doesn't look like it has a nice way of communicating. like you get these report tables, which are maybe not easy to read for someone who didin't write the tests. vignette.
- assertr just throws an error an stops execution. ah, that's not true, you can configure what it does. you cna write your own, but `error_append` seems useful also `just_warn`, which just prints the errors, but doesn't stop execution.
- to report on a series of assertions you use `chain_start` and `chain_end`, and you can write your own `error_fun`, so looking very good.
- have a look at tidytest
- Rstudio folks debate using testthat outside packages here, prefer using packages, but it's possible.

- the problem with `testthat` is that it tests the whole script. so the test file sources the code, and then runs the tests. This won't work for me. Unless I break up the script into import/clean, compute, prepare edits and email. Then each could be tested individually. Would be fun as well.
- here's another non-package tutorial on testthat here
- Maybe I could just try out both approaches. See what I like best? :D
- there might be philosophical reasons for not using testthat, since it in principle increases the difference between the developer's version and the user's (explained here).
- the `tinytest` vignette is here and seems quite similar to testthat, albeit not package focussed. Very useful vignette, some basic stuff about unit testing:
- only test exported functions, not internal ones (makes it easier to change the internals later if you have to. and avoid changing the externals).
- make the ratio of external to internal functions small = a spherical package.
- the "external surface" of a package i.e. the exported funcitons is also called the API.
- idealy test all the possible paths through your code. but can be unrealistic. second best is to at least test each line of code: *full code coverage* means each line is run at least once during testing.
- for every bug write a test. then be really freaked out if it reappears.
- beware of side effects. `on.exit()` is a useful function to make sure you don't get derailed and return everything to how it was.
- Hm so what do i do now, tinytest feels nicer, but very undocumented. Or do I want to use `assertr` - "Assertion" is computer-science jargon for a run-time check on your code.

1. OK, so splitting the files into modular subroutines and using tinytest to test each one.
2. tinytest won't let me test the working directory - it just gives the one where the file is. open issue.
3. OK, the nice thing about source and test is that source leaves the console nice and clean (if you supress all messages and warnings), so then you only see the outputs of the tests.
4. OK, now actually moved the testing into the modular scripts, so called at the end of each script. The outputs are saved so they can be viewed at the end.
5. Commit.

Thursday 9.1.2020

1. add `here`to preliminaries and its test.
2. OK, testing for the download script. What happens if a table does not exist. all the subsequent tests don't actually fail, they don't work, since the object doesn't exist. SO you have to add the test `exists("object")` to each test. annoying. tip.
3. OK, now start on 03-calculations. What might one test? Just what was in the original.
4. Hm, i'm *not checking for NAs yet*

Modnay 13.1.2020

1. ok, move on to breakdown of average uk prices, which means cleaning up the tax/duty table
2. OK, i'm confused, the "Fuel Data" sheet, I was going to use it to get the VAT and duty levels, but the most recent data in there is 1 year old. What is this table doing there? Should I just hardcode the VAT and duty rates? What if they change?

3. Don't really know what to test on the taxes/VAT table. Except column types.
4. OK, let's download all the tables, finish the 02-download.R file.
5. The eu compare table is downloaded separately for petrol and diesel.
6. The problem is all the tables in the googlesheet are so untidy..
7. OK, all data are imported, cleaned and this is tested. Except for NAs. *where should i test for NAs?*
8. What happens on a Monday? Added a test
9. A bit worried about these tests. If they don't fail, but trhow errors, how useful is that going to be?
10. I mean how useful are these tests in the first place..
11. How do i do "price one month ago"? The google sheet does sth weird, where it goes one month back, but if that is a sunday it takes the friday before it, but if it's a saturday, it takes the thursday before it?! Surely that's an error and should take the Friday in both cases.
12. I'll keep it simple. Go one month back, if data doesn't exist, go back an extra day or two until data exists. So it is alwyas one month or just a bit more.
13. Another werid thing the google sheet does is for 6 months ago, it takes 6 months + 1 day. No idea why, it's inconsistent with how one month ago works. I'm ignoring it.
14. OK, finished first page. But mosth of the calculated values do not get tested, don't really know what to test them for. Await instructions.
15. Commit.

Tuesday 14.1.2019

1. Start working on export script.
2. For starters let's have all the data as a list, each number a list element. Each element a dataframe with the following vars:

- id number
- sequence number within ID
- description
- value

3. But what happens with the tables with two or more variables, e.g. the price data for the chart. That has three values. There are two tables like that, and then there's the EU country comparison, which also has two values.
4. If I put it all into one csv file, it will have to be in a tidy table, but that won't be practical for the designers, because the three tables will need to be "spread". So best if the three tables are exported separately. WHich means i also don't ned the sequence number for all the other ones.
5. Question: e.g. in fields 2, 4 or 7, should i add the "p" to the prices or just leave the numbers?
6. OK, prepared export for all first page data. Commit.
7. Continue 03-calculations.
8. 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. But it seems rather unlikely?
9. The oil price data has tricky column names, i'll change those on import to prevent issues further on.

10. Rest of graph: does that mean the actual charts need to be provided or only the data? I'm assuming only the data tables for now.
11. is the first date on the pump.price table always the same as the frist one in the oil.price table? Can i assume that? should i check it?
12. how do i do the arrows?
13. is the oil price table always exactly 365 days long? i mean can i just take the last row in the table to get last year's price?. Same qyestion for pump prices. last row?
14. Oh, there's a whole other table hidden on the eu sheet, the one with pre-tax numbers, which i need. Added to downloads. still need to add tests. ok added some tests.

Wednesday 15.1.2020

1. finish export script. can make each subseciton a separate table, which can all be merged together in the end, but if need be I can split them up as well.
2. OK, all 04.export done.
3. commit
4. Add some tests for the exports. Mainly number of rows? What else might there be. As long as all of them exist, that's all I need to know.
5. Start overall flow, stopping progres from one step to anohter
6. not full proof yet though.
7. commit

Appendix - make file