Pacific Northwest
NATIONAL LABORATORY

Creating XMLs and using user-modification functions

Pralit Patel
Paul Wolfram
Ian Pimenta

U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

Modifying the xml input files – Where to start?

- The best approach for modifying GCAM depends on what you are changing
- What your intention is
- Whether your changes need to be modified over the long term – between GCAM versions

Modifying the xml input files – Where to start?

- Possible things to change:
  - The XML inputs directly
  - Parameter that is either specified in `input/gcamdata/inst/extdata` or the `constants.R` file in the gcamdata package.
  - Parameter or file that is derived in the gcamdata package
  - Structure of a region or sector
- Possible intentions:
  - Quick understanding
  - Project or paper

Wait.. what did I do again?

- No matter what, think carefully about “tracking” your changes
- Version control!
- Separate your assumptions from standard assumption
- Creating copies and backups
  - my_input_final_final_reallyFINAL.xml

Pacific Northwest
NATIONAL LABORATORY

Modify the XML Directly

- Can make sense for simple inputs such as policy files
- Gets out of hand pretty quickly

- Again, the question of quick test / something to be maintained

I need something different than what gcamdata produces

- But perhaps you still want to feed in some data that was produced as a matter of course in the gcamdata processing
  - driver(write_outputs = TRUE)
  - load_from_cache()
- Keep the processing in R and wse the XML utilities from gcamdata
- The old school Excel and Model Interface approach still works

Changing an assumption file

- Go in and change it
- Re-run driver_drake() (or driver)

- **May not be predictable which XMLs get affected!**

- If this is something you want to maintain over the long haul and / propose to bring back to the Core GCAM then this is what you want.

User Modification Functions

- Allows users to write their own function to modify a gcamdata input/output, without modifying input CSVs or gcamdata chunks directly
- Modified object gets “plugged into” datasystem and passed to all dependent chunks
- New XML(s) get created with user-specified suffixes, to distinguish them from core gcamdata XMLs
  - Note: Remember to update configuration.xml to include the custom XMLs before running GCAM
- Motivation: Keep track of gcamdata changes from user vs. core GCAM assumptions, ensuring user-implemented changes can be automated and are reproducible

Writing a User-Mod Function

```
usermod_fert <- function(command, ...) {
  if(command == driver.DECLARE_MODIFY) {
    return(c(FILE = "L2322.SubsectorShrwtFllt_Fert"))

  } else if(command == driver.DECLARE_INPUTS) {
    return()

  } else if(command == driver.MAKE) {
    all_data <- list(...)[[1]]
    L2322.SubsectorShrwtFllt_Fert <- get_data(all_data, "L2322.SubsectorShrwtFllt_Fert")

    # Read in additional inputs from outside gcamdata, if necessary
    Fert_Shwt_Additions <- read.csv("mod_inputs/Fert_Shwt_Additions.csv", header = TRUE)

    # Make changes
    L2322.SubsectorShrwtFllt_Fert %>%
      bind_rows(Fert_Shwt_Additions) -> L2322.SubsectorShrwtFllt_Fert

    # Return modified gcamdata object
    return_modified("L2322.SubsectorShrwtFllt_Fert" = L2322.SubsectorShrwtFllt_Fert)

  } else {
    stop("Unknown command")
  }
}

# Run driver_drake with new chunk in the call
# Include a suffix to append to any affected objects
driver_drake(user_modifications = c("usermod_fert"), xml_suffix = "_1")
```

Declare input/output gcamdata object you want to change (dstrace function may be useful for finding the object initially)

Declare any other gcamdata inputs that you need but won't modify

Read in other custom inputs. (Don't include custom files in driver.DECLARE_INPUTS since we don't want to mix custom files with core gcamdata files)

Make any changes and return object. Note returned object name must match the original object name we asked for in driver.DECLARE_MODIFY

Run driver_drake and include the new chunk in our function call, along with a suffix to append to any affected XMLs (currently mandatory to include suffix)

Writing a User-Mod Function: Creating Multiple XMLs

- We can also generate multiple modified XMLs with this feature.
  - Note: We must include an argument to the user mod function that we can update in a loop. For this example, instead of reading in the file directly, we modify usermod_fert() to read in a CSV with the name of the value of “file_name”

```
Fert_Shwt_Additions <- read.csv(file_name, header = TRUE)
```

```
# Loop to create multiple user-modified XMLs
for (i in 1:length(list.files("mod_inputs"))){
  # Ensures that drakes knows to run usermod_fert
  drake::clean(list="usermod_fert")

  file_name <- list.files("mod_inputs", full.names = TRUE)[i]

  driver_drake(user_modifications = c("usermod_fert"),
         xml_suffix = paste0("_", i))
}
```

Run through all files in mod_inputs folder

Clear the usermod_fert object from drake's cache as drake doesn’t recognize changes to the argument file_name. If you do not include this call, drake may assume that all downstream objects/xmls do not need to be updated.

Get new file name

Run driver_drake once for each file, ensuring each run is associated with a different suffix

- This creates separate outputs for each custom file in the user_mod folder

Modifying the structure of a region or sector

- It is difficult to give general advice in this case, because exactly how you make this change depends on what you are trying to do.
- Some general advice:
  - Start small. Make the smallest change possible at each step so you can identify when things go wrong.
  - If possible use the “add-on” approach so your changes are in its own XML and delete/replace some part of the model (GCAM-USA approach)
  - You can create your test xml file in whatever manner is easiest (e.g., typing by hand or copying and editing an existing one). However, if you are planning to use the end result in a paper or in the GCAM master, then you will need to create it via the data system at some point. This is likely to be easier sooner rather than later.
  - Be very careful when changing calibration values.
  - Use GitHub Discussions to ask for more help: https://github.com/jgcri/gcam-core/discussions