MARVIC2EFI

Istem Fer, Manuel Martin

2025-06-17

MARVIC models with EFI standards

EFI standards are concerned with two main files:

1) The model output file which contains the actual predicted values from the model. This file will be in **csv** format. There is also a possibility to have the output file in netCDF format, but tabular data has certain advantages.

2) The dataset metadata file contains all the accompanying metadata information that will help interpret and use the model outputs (resolution, dimensions, variable names, units etc. but also uncertainty propagation, data assimilation, model complexity etc.). This file will be in **json** format.

Start with installing libraries

```
# install.packages(c("EML", "emld))
library(EML)
library(emld)
emld::eml_version("eml-2.2.0")

#install.packages(c("lubridate", "tibble", "dplyr", "tidyr", "reshape2"))
library(lubridate)
library(tibble)
library(dplyr)
library(tidyr)
library(reshape2)

# for writing the parquet files
# install.packages("arrow")

# also make sure you install other packages that are required to read YOUR outputs
# e.g. if your raw outputs are in netCDF format:
# install.packages("ncdf4")
# library(ncdf4)
```

Read your model outputs

Read in your model outputs in your raw format. The example below assumes your outputs are in netcdf format:

```
rm(list=ls())

raw_output_file <- ".../my_raw_model_output.nc"
out_nc <- ncdf4::nc_open(raw_output_file)
```

```r
# read example variables, here NEE, GPP, NPP
out_nee <- ncdf4::ncvar_get(out_nc, "NEE")
out_gpp <- ncdf4::ncvar_get(out_nc, "GPP") # in this example out_gpp is an "n_time x n_ensembles" matri
out_ar  <- ncdf4::ncvar_get(out_nc, "AutoResp")
out_npp <- out_gpp - out_ar
```

Reformat model outputs according to EFI standards

Convert outputs to a flat file format (CSV):

```r
# reference_datetime and pubtime can be the same for MARVIC
# just putting today's date
pub_datetime <- reference_datetime <- lubridate::today()

# actual time stamps for the predictions
n_time <- dim(out_gpp)[1]
start_date <- as.Date("2018-01-01") <== YOUR START DATE
datetimes <- seq(from=start_date, by="1 day", length=n_time)

# this example assumes ensemblized model runs
# there are other options, e.g. summary of a distribution
# number of ensemble members
n_ensembles <- dim(out_gpp)[2]
ensembles <- seq_len(n_ensembles)

obs_dim <- 1 ## 1 = latent state
## 2 = latent state + observation error

variable_names <- c("net_ecosystem_exchange",
                    "gross_primary_productivity",
                    "net_primary_productivity")
n_variables <- length(variable_names)
names(variable_names) <- paste0(rep("variable_", n_variables), 1:n_variables)

# prepare storage
output_storage <- array(NA,dim = c(n_time, n_ensembles, obs_dim, n_variables))

output_storage[,,,1] <- out_nee
output_storage[,,,2] <- out_gpp
output_storage[,,,3] <- out_npp

## data assimilation flag
data_assimilation <- rep(as.integer(0), n_time)  ## this code indicates a 'free-run' that didn't assimi

## forecast flag
forecast <- rep(as.integer(0), n_time)  ## EFI standard forcast flag. 1 = forecast, 0 = hindcast


## wrangle data from an array into a long format
df_combined <- reshape2::melt(output_storage,varnames=c("datetime","parameter","obs_flag", "variable_na
  pivot_wider(id_cols = 1:3,names_from = "variable_names", names_prefix = "variable_", values_from = "va
  pivot_longer(cols = starts_with("variable"),names_to = "variable",values_to = "prediction") %>%
  mutate(datetime = datetimes[datetime]) %>%
  mutate(variable = variable_names[variable]) %>%
```

```
  right_join(tibble(datetime=datetimes,data_assimilation = data_assimilation, forecast = forecast))
```

Here df_combined should look something like this:

```
> head(df_combined,n=10)
# A tibble: 10 × 7
   datetime   parameter obs_flag variable                    prediction data_assimilation forecast
   <date>         <int>    <int> <chr>                            <dbl>              <int>    <int>
 1 2018-01-01         1        1 net_ecosystem_exchange     0.00000000843                  0        0
 2 2018-01-01         1        1 gross_primary_productivity 0                              0        0
 3 2018-01-01         1        1 net_primary_productivity   0                              0        0
 4 2018-01-02         1        1 net_ecosystem_exchange     0.00000000885                  0        0
 5 2018-01-02         1        1 gross_primary_productivity 0                              0        0
 6 2018-01-02         1        1 net_primary_productivity   0                              0        0
 7 2018-01-03         1        1 net_ecosystem_exchange     0.00000000929                  0        0
 8 2018-01-03         1        1 gross_primary_productivity 0                              0        0
 9 2018-01-03         1        1 net_primary_productivity   0                              0        0
10 2018-01-04         1        1 net_ecosystem_exchange     0.00000000928                  0        0
```

Write EFI csv

```
##### write csv
# YOUR PATH HERE
# we can also discuss standard file naming
csv_filename <- ".../FI-Qvd_SPY-C_2018-2023_EFIstandard.csv"
write.csv(df_combined,
          file = csv_filename)

### convert to parquet format
# this is preliminary at the moment, we are discussing the S3 folder hierarchy
# YOUR PATH HERE
parquet_path <- ".../"

arrow::write_dataset(
  df_combined,
  parquet_path,
  compression = "snappy"
)
```

Prepare standard Metadata

These functions use R EML/emld packages.

```
## define variable names, units, etc
## in practice, this might be kept in a spreadsheet
attributes <- tibble::tribble(
  ~attributeName,               ~attributeDefinition,                          ~unit,           ~fo
  "datetime",                   "[dimension]{datetime}",                       "year",          "YY
  "parameter",                  "[dimension]{index of ensemble member}",       "dimensionless", NA,
  "obs_flag",                   "[dimension]{observation error}",              "dimensionless", NA,
  "net_ecosystem_exchange",     "[variable]{Net Ecosystem Exchange}",          "kg C m-2 s-1",  NA,
  "gross_primary_productivity", "[variable]{Gross Primary Productivity}",      "kg C m-2 s-1",  NA,
  "net_primary_productivity",   "[variable]{Net Primary Productivity}",        "kg C m-2 s-1",  NA,
  "data_assimilation",          "[flag]{whether time step assimilated data}",  "dimensionless", NA,
```

```
  "forecast",                    "[flag]{whether time step forecast or hindcast}", "dimensionless", NA,
)
attributes

## note: EML uses a different unit standard than UDUNITS. MARVIC / EFI needs to provide a custom unitLi
## ignoring warnings for now
attrList <- set_attributes(attributes,
                           col_classes = c("Date", "numeric", "numeric","numeric",
                                           "numeric", "numeric", "numeric", "numeric"))
```

More metadata:

```
## sets metadata about the file itself (name, file type, size, MD5, etc)
physical <- set_physical(csv_filename)


## set metadata for the file as a whole
dataTable <- eml$dataTable(
  entityName = "MARVIC T3.2? outputs",  ## this is a standard name to allow us to distinguish this enti
  entityDescription = "Agro-ecosystem carbon budget predictions",
  physical = physical,
  attributeList = attrList)

# who to contact about this output
creator_info <- list(individualName = list(givenName = "YourName",
                                           surName = "YourSurname"),
                     electronicMailAddress = "YourEmail",
                     id = "https://orcid.org/YourOrcid")

# can also set taxonomic, temporal, and geographic coverage of the outputs
taxa <- tibble::tribble(
  ~Genus,      ~Species,
  "Your", "Species") # UPDATE ACCORDINGLY

coverage <-
  set_coverage(begin = first(datetimes),
               end = last(datetimes),
               sci_names = taxa,
               geographicDescription = "Qvidja, FI ", # UPDATE ACCORDINGLY
               west = 22.3932251, east = 22.39017,    # UPDATE ACCORDINGLY
               north = 60.29531,  south = 60.29331).  # UPDATE ACCORDINGLY

# Set key words.  We will need to develop a MARVIC controlled vocabulary
keywordSet <- list(
  list(
    keywordThesaurus = "MARVIC controlled vocabulary",
    keyword = list("hindcast",
                   "fluxes",
                   "timeseries")
  ))
```

Combine the above bits to document the output dataset as a whole:

```
dataset = eml$dataset(
  title = "MARVIC T3.2 model outputs",  # can discuss how to name this
```

```
  creator = creator_info,
  pubDate = reference_datetime,
  intellectualRights = "https://creativecommons.org/licenses/by/4.0/", # choose
  abstract =  "An illustration of how we might use EML metadata to describe a MARVIC output",
  dataTable = dataTable,
  keywordSet = keywordSet,
  coverage = coverage
)
```

Additional metadata:

```
## Global attributes
target_id <- "YOUR_TARGET_ID" # this is a unique identifier (e.g., URL or DOI) that links to data or me
model_name <- "YOUR_MODEL_NAME"
model_version <- "YOUR_MODEL_VERSION"
iteration_id <- "20180101T000000" #`iteration_id` represents a unique ID for each run. Examples might b

# Additional metadata
additionalMetadata <- eml$additionalMetadata(
  metadata = list(
    forecast = list(
      ## Basic elements
      timestep = "1 day", ## should be udunits parsable
      horizon = paste0(n_time, " days"),
      reference_datetime = reference_datetime,
      iteration_id = iteration_id,
      target_id = target_id,
      metadata_standard_version = "1.0",
      model_description = list(
        model_id = model_version,
        name = model_name,
        type = "process-based", # OR?
        repository = "https://github.com/huitang-earth/SVMC"
      ),
      ## MODEL STRUCTURE & UNCERTAINTY CLASSES
      initial_conditions = list(
        # Possible values: absent/present, data_driven, propagates, assimilates
        #### ============>
        present = FALSE  # UPDATE ACCORDINGLY
      ),
      drivers = list(
        present = FALSE # UPDATE ACCORDINGLY
      ),
      parameters = list(
        present = TRUE, # UPDATE ACCORDINGLY
        data_driven = TRUE, # UPDATE ACCORDINGLY
        #### ============>
        complexity = 4,  ## number of parameters being varied, UPDATE ACCORDINGLY
        propagation = list(
          type = "ensemble",
          size = 256 # UPDATE ACCORDINGLY
        )
      ),
      random_effects = list(
```

```
        present = FALSE # UPDATE ACCORDINGLY
      ),
      obs_error = list(
        present = FALSE # UPDATE ACCORDINGLY
      ),
      process_error = list(
        present = FALSE # UPDATE ACCORDINGLY
      )
    ) # forecast
  ) # metadata
) # eml$additionalMetadata
```

Write:

```
my_eml <- eml$eml(dataset = dataset,
                  additionalMetadata = additionalMetadata,
                  packageId = iteration_id ,
                  system = "datetime"  ## system used to generate packageId
)

## check base EML
## currently getting warnings about units
emld::eml_validate(my_eml)

# YOUR PATHS
eml_filename <- ".../my_model.xml"
write_eml(my_eml, eml_filename)
json_filename <- ".../my_model.json"
emld::as_json(as_emld(eml_filename), file = json_filename)
```