# Download CDS Climate, ECMWF Global Enviornmental Data via Python API

Fan Wang

2020-07-07

## Contents



# 1 ECMWF ERA5 Data

Go to the **RMD**, **R**, **PDF**, or **HTML** version of this file. Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

This files uses R with the reticulate package to download ECMWF ERA5 data. See this file for instructions and tutorials for downloading the data.

## 1.1 Program to Download, Unzip, Convert to combined CSV, derived-utci-historical data

The data downloaded from CDS climate could become very large in size. We want to process parts of the data one part at a time, summarize and aggregate over each part, and generate a file output file with aggregate statistics over the entire time period of interest.

This code below accompalishes the following tasks:

1. download data from derived-utci-historical as ZIP
2. unzip
3. convert *nc* files to *csv* files
4. individual csv files are half year groups

Parameter Control for the code below:

1. *spt_root*: root folder where everything will be at
2. *spth_conda_env*: the conda virtual environment python path, eccodes and cdsapi packages are installed in the conda virtual environment. In the example below, the first env is: wk_ecmwf
3. *st_nc_prefix*: the downloaded individual nc files have dates and prefix before and after the date string in the nc file names. This is the string before that.
4. *st_nc_suffix*: see (3), this is the suffix
5. *ar_years*: array of years to download and aggregate over
6. *ar_months_g1*: months to download in first half year
7. *ar_months_g2*: months to download in second half year

Note: *area* below corresponds to *North, West, South, East.*

```
################################################
# ------------ Parameters
```

```
##################################################

# Where to store everything
spt_root <- "C:/Users/fan/Downloads/_data/"
spth_conda_env <- "C:/ProgramData/Anaconda3/envs/wk_ecmwf/python.exe"

# nc name prefix
st_nc_prefix <- "ECMWF_utci_"
st_nc_suffix <- "_v1.0_con.nc"

# Years list
# ar_years <- 2001:2019
ar_years <- c(2005, 2015)
# ar_months_g1 <- c('01','02','03','04','05','06')
ar_months_g1 <- c('01', '03')
# ar_months_g2 <- c('07','08','09','10','11','12')
ar_months_g2 <- c('07', '09')

# Area
# # China
# fl_area_north <- 53.31
# fl_area_west <- 73
# fl_area_south <- 4.15
# fl_area_east <- 135
fl_area_north <- 53
fl_area_west <- 73
fl_area_south <- 52
fl_area_east <- 74

# folder to download any nc zips to
nczippath <- spt_root
# we are changing the python api file with different requests stirngs and storing it here
pyapipath <- spt_root
# output directory for AGGREGATE CSV with all DATES from this search
csvpath <- spt_root

##################################################
# ------------ Packages
##################################################

library("ncdf4")
library("chron")
library("lattice")
library("RColorBrewer")
library("stringr")
library("tibble")
library("dplyr")
Sys.setenv(RETICULATE_PYTHON = spth_conda_env)
library("reticulate")

##################################################
# ------------ Define Loops
##################################################
```

```
for (it_yr in ar_years) {
  for (it_mth_group in c(1,2)) {
    if(it_mth_group == 1) {
      ar_months = ar_months_g1
    }
    if(it_mth_group == 2) {
      ar_months = ar_months_g2
    }

    ################################################
    # ------------ Define Python API Call
    ################################################

    # name of zip file
    nczipname <- "derived_utci_2010_2.zip"
    unzipfolder <- "derived_utci_2010_2"

    st_file <- paste0("import cdsapi
import urllib.request
# download folder
spt_root = '", nczippath, "'
spn_dl_test_grib = spt_root + '", nczipname, "'
# request
c = cdsapi.Client()
res = c.retrieve(
    'derived-utci-historical',
    {
        'format': 'zip',
        'variable': 'Universal thermal climate index',
        'product_type': 'Consolidated dataset',
        'year': '",it_yr, "',
        'month': [
            ", paste("'", ar_months, "'", sep = "", collapse = ", "), "
        ],
        'day': [
            '01','03'
        ],
        'area'  : [", fl_area_north ,", ", fl_area_west ,", ", fl_area_south ,", ", fl_area_east ,"],
        'grid'  : [0.25, 0.25],
    },
    spn_dl_test_grib)
# show results
print('print results')
print(res)
print(type(res))")

    # st_file = "print(1+1)"

    # Store Python Api File
    fl_test_tex <- paste0(pyapipath, "api.py")
    fileConn <- file(fl_test_tex)
    writeLines(st_file, fileConn)
    close(fileConn)
```

```
################################################
# ------------ Run Python File
################################################
# Set Path
setwd(pyapipath)
# Run py file, api.py name just defined
use_python(spth_conda_env)
source_python('api.py')

################################################
# ------------ uNZIP
################################################
spn_zip <- paste0(nczippath, nczipname)
spn_unzip_folder <- paste0(nczippath, unzipfolder)
unzip(spn_zip, exdir=spn_unzip_folder)

################################################
# ------------ Find All files
################################################
# Get all files with nc suffix in folder
ncpath <- paste0(nczippath, unzipfolder)
ls_sfls <- list.files(path=ncpath, recursive=TRUE, pattern=".nc", full.names=T)

################################################
# ------------ Combine individual NC files to JOINT Dataframe
################################################
# List to gather dataframes
ls_df <- vector(mode = "list", length = length(ls_sfls))
# Loop over files and convert nc to csv
it_df_ctr <- 0
for (spt_file in ls_sfls) {
  it_df_ctr <- it_df_ctr + 1

  # Get file name without Path
  snm_file_date <- sub(paste0('\\',st_nc_suffix,'$'), '', basename(spt_file))
  snm_file_date <- sub(st_nc_prefix, '', basename(snm_file_date))

  # Dates Start and End: list.files is auto sorted in ascending order
  if (it_df_ctr == 1) {
    snm_start_date <- snm_file_date
  }
  else {
    # this will give the final date
    snm_end_date <- snm_file_date
  }

  # Given this structure: ECMWF_utci_20100702_v1.0_con, sub out prefix and suffix
  print(spt_file)
  ncin <- nc_open(spt_file)

  nchist <- ncatt_get(ncin, 0, "history")

  # not using this missing value flag at the moment
```

```
  missingval <- str_match(nchist$value, "setmisstoc,\\s*(.*?)\\s* ")[,2]
  missingval <- as.numeric(missingval)

  lon <- ncvar_get(ncin, "lon")
  lat <- ncvar_get(ncin, "lat")
  tim <- ncvar_get(ncin, "time")
  tunits <- ncatt_get(ncin, "time", "units")

  nlon <- dim(lon)
  nlat <- dim(lat)
  ntim <- dim(tim)

  # convert time -- split the time units string into fields
  # tustr <- strsplit(tunits$value, " ")
  # tdstr <- strsplit(unlist(tustr)[3], "-")
  # tmonth <- as.integer(unlist(tdstr)[2])
  # tday <- as.integer(unlist(tdstr)[3])
  # tyear <- as.integer(unlist(tdstr)[1])
  # mytim <- chron(tim, origin = c(tmonth, tday, tyear))

  tmp_array <- ncvar_get(ncin, "utci")
  tmp_array <- tmp_array - 273.15

  lonlat <- as.matrix(expand.grid(lon = lon, lat = lat, hours = tim))
  temperature <- as.vector(tmp_array)
  tmp_df <- data.frame(cbind(lonlat, temperature))

  # extract a rectangle
  eps <- 1e-8
  minlat <- 22.25 - eps
  maxlat <- 23.50 + eps
  minlon <- 113.00 - eps
  maxlon <- 114.50 + eps
  # subset data
  subset_df <- tmp_df[tmp_df$lat >= minlat & tmp_df$lat <= maxlat &
                        tmp_df$lon >= minlon & tmp_df$lon <= maxlon, ]

  # add Date
  subset_df_date <- as_tibble(subset_df) %>% mutate(date = snm_file_date)

  # Add to list
  ls_df[[it_df_ctr]] <- subset_df_date

  # Close NC
  nc_close(ncin)
}

# List of DF to one DF
df_all_nc <- do.call(rbind, ls_df)

# Save File
fname <- paste0(paste0(st_nc_prefix,
                       snm_start_date, "_to_", snm_end_date,
```

```
                            ".csv"))
    csvfile <- paste0(csvpath, fname)
    write.table(na.omit(df_all_nc), csvfile, row.names = FALSE, sep = ",")

    # Delete folders
    unlink(spn_zip, recursive=TRUE, force=TRUE)
    unlink(spn_unzip_folder, recursive=TRUE, force=TRUE)

  # end loop months groups
  }
# end loop year
}
```