Download CDS Climate, ECMWF Global Enviornmental Data via Python API

Fan Wang

2020-06-23

Contents

1 ECMWF ERA5 Data

Go to the **RMD**, **PDF**, or **HTML** version of this file. Go back to Python Code Examples Repository (bookdown site) or the pyfan Package (API).

1.1 Basic Conda Setup

1. Download Anaconda for Python 3. For more involved conda instructions see here
2. Open up *anaconda prompt* with admin rights (right click choose as admin).

```
# Inside anaconda prompt
where python
where anaconda
# C:/ProgramData/Anaconda3/Scripts/anaconda.exe
# C:/ProgramData/Anaconda3/python.exe
```

3. Add to Path
4. Install cdsapi and eccodes

```
conda config --add channels conda-forge
conda install -c anaconda pandas
conda install -c conda-forge eccodes -y
```

```
conda install -c conda-forge cfgrib -y
conda install -c anaconda xarray
```

1.2 Account Registration

1. Register for an account
2. Agree to Licence
3. Go to your CDS user page copy the url and key: Get url and key
   - this has UID, 4XXXX, and API KEY, 4XXXfXXX-XXXf-4XXX-9XXX-7XXXebXXfdXX
   - together they should look like: 4XXXX:4XXXfXXX-XXXf-4XXX-9XXX-7XXXebXXfdXX
4. Open up an editor (notepad++ for example), create an empty file, paste the url and your UID:APIKEY into the file as below. Save file as: *C:/Users/fan/.cdsapirc*. Under user root, as *.cdsapirc* file. Note *.cdsapirc* is the file name, you are saving that under the directory *C:/Users/fan/*.

```
url: https://cds.climate.copernicus.eu/api/v2
key: 4XXXX:4XXXfXXX-XXXf-4XXX-9XXX-7XXXebXXfdXX
```

1.3 Run API Request via Jupyter Notebook

1. open up Jupyter Notebook (this opens up a browser page)
   - cd "C:/Users/fan/Downloads"
   - jupyter notebook
2. create a new *python3* file somewhere you like
3. name the file *cdstest* (saved as ipynb file)
4. paste the code below inside the *ipynb* file you opened (modify *spt_root*):

```
import cdsapi
import urllib.request
# download folder
spt_root = "C:/Users/fan/downloads/_data/"
spn_dl_test_grib = spt_root + "test_china_temp.grib"
# request
c = cdsapi.Client()
res = c.retrieve("reanalysis-era5-pressure-levels",
  {
    'product_type': 'reanalysis',
    'variable': 'temperature',
    'pressure_level': '1000',
    'year': '2008',
    'month': '01',
    'day': '01',
    'time': '12:00',
    'format': 'netcdf',
    'area'          : [53.31, 73, 4.15, 135],
    'grid'          : [1.0, 1.0],
    "format": "grib"
  },
  spn_dl_test_grib
)
# show results
```

```
print('print results')
print(res)
print(type(res))
```

5. click run

1.4 Run API request via Ipython

1. In Anaconda Prompt: *ipython*
2. Open a file in notepad++ or elsewhere, copy the code above over and edit the spt_root to reflect your directories
3. Select the entire code in the notepad++ page, and copy all lines
4. Now inside ipython, type percentage and paste: %paste
5. This should run the file above and save the grib file in the folder you specified with the name you specified.

1.5 Convert CRIB data to CSV

1. inside conda prompt cd into the folder where you downloaded the grib file
2. *grib_ls* shows what is in the grib file
3. *grib_get_data* translates grib to csv

```
cd "C:/Users/fan/downloads/_data/"
grib_ls test_china_temp.grib
grib_get_data test_china_temp.grib > test_china_temp_raw.csv
```

1.6 More Advanced Download Setup and Instructions

1.6.1 Conda Enviornment and Installation

In conda, set up a conda environment for downloading ECMWF data using the ECMWF API. (Conda Set-up)

```
# Set up
conda deactivate
conda list env
conda env remove -n wk_ecmwf
conda create -n wk_ecmwf -y
conda activate wk_ecmwf

# Add conda-forge to channel in env
conda config --env --add channels conda-forge
conda config --get channels
conda config --get channels --env

# Install
conda install cdsapi -y
conda install -c anaconda pandas
conda install -c conda-forge eccodes -y
conda install -c conda-forge cfgrib -y
conda install -c anaconda xarray
```

This creates the conda env that we are using here for python.

1.6.2 Config File .cdsapirc

Open up the *cdsapirc*, create new if does note exist. Below, open up the file and save the text. See Python Reading and Writing to File Examples.

First, get the text for the config file:

```
stf_cds_cdsapirc = """\
url: https://cds.climate.copernicus.eu/api/v2
key: 4XXXX:4XXXfXXX-XXXf-4XXX-9XXX-7XXXebXXfdXX\
"""
print(stf_cds_cdsapirc)
```

Second save text to file:

```
# Relative file name
spt_file_cds = "C:/Users/fan/"
snm_file_cds = ".cdsapirc"
spn_file_cds = spt_file_cds + snm_file_cds
# Open new file
fl_cdsapirc_contents = open(spn_file_cds, 'w')
# Write to File
fl_cdsapirc_contents.write(stf_cds_cdsapirc)
# Close
fl_cdsapirc_contents.close()
```

```
# Open the config file to check
code "C:/Users/fan/.cdsapirc"
```

1.7 Generate API Requests

Go to the sites below, choose download data, pick what is needed, and then select *Show API request* at the bottom of page:

ERA5 pressure levels from 1979 to present

- ERA5 hourly pressure
- ERA5 monthly pressure

ERA5 single levels from 1979 to present

- ERA5 hourly pressure
- ERA5 monthly pressure

1.7.1 API Request China Temp Test

API function is here.

Select based on China's area, some testing data and download grib file. The data is from 2008, Jan 1st, at 12 noon?

Open up Jupyter notebook: *jupyter notebook*

```
# import module in conda env wk_ecmwf
import cdsapi
import urllib.request

# download folder
spt_root = "C:/Users/fan/py4econ/vig/getdata/envir/"
spn_dl_test_grib = spt_root + "_data/test/test_china_temp.grib"
```

```
# request
c = cdsapi.Client()
res = c.retrieve("reanalysis-era5-pressure-levels",
  {
    'product_type': 'reanalysis',
    'variable': 'temperature',
    'pressure_level': '1000',
    'year': '2008',
    'month': '01',
    'day': '01',
    'time': '12:00',
    'format': 'netcdf',
    'area'          : [53.31, 73, 4.15, 135],
    'grid'          : [1.0, 1.0],
    "format": "grib"
  },
  spn_dl_test_grib
)
# show results
print('print results')
print(res)
print(type(res))

# download
# response = urllib.request.urlopen('http://www.example.com/')
# html = response.read()
```

2020-06-17 23:51:35,107 INFO Welcome to the CDS 2020-06-17 23:51:35,107 INFO Sending request to https://cds.climate.copernicus.eu/api/v2/resources/reanalysis-era5-pressure-levels 2020-06-17 23:51:36,441 INFO Request is queued 2020-06-17 23:51:39,183 INFO Request is running 2020-06-17 23:51:45,059 INFO Request is completed 2020-06-17 23:51:45,060 INFO Downloading > http://136.156.133.25/cache-compute-0008/cache/data2/adaptor.mars.internal-1592455900.8655114-11162-11-68e1ea23-8985-4926-95e6-9f181cc7792> 7.grib to C:/Users/fan/pyfan/vig/getdata/envir/_data/test/test_china_temp.grib (6.3K) 2020-06-17 23:51:45,441 INFO Download rate 16.6K/s print results Result(content_length=6480,content_type=application/x-grib,location=http://136.156.133.25/cache-compute-0008/cache/data2/adaptor.mars.inte> rnal-1592455900.8655114-11162-11-68e1ea23-8985-4926-95e6-9f181cc77927.grib) <class 'cd-sapi.api.Result'>

Convert grib to raw csv, open up command line:

```
cd "C:/Users/fan/pyfan/vig/getdata/envir/_data/test/"
grib_ls test_china_temp.grib
grib_get_data test_china_temp.grib > test_china_temp_raw.csv
```

Format the CSV file (is not comma separated)

```
spt_root = "C:/Users/fan/pyfan/vig/getdata/envir/_data/test/"
spn_csv_raw = spt_root + "test_china_temp_raw.csv"
spn_csv_edi = spt_root + "test_china_temp.csv"

with open(spn_csv_raw, 'r') as f_in, open(spn_csv_edi, 'w') as f_out:
    f_out.write(next(f_in))
    [f_out.write(','.join(line.split()) + '\n') for line in f_in]
```

Show CSV results:

```
# Path and Read
spt_root = "C:/Users/fan/pyfan/vig/getdata/envir/"
spn_dl_test_csv = paste0(spt_root, "_data/test/test_china_temp.csv")
china_weather_data <- read.csv(spn_dl_test_csv)

# Top 50 rows
kable(head(china_weather_data, 50),
      caption="Chinese Long and Lat, Temperature Pressure, 2008 Jan 1st at 12 noon?") %>%
  kable_styling_fc()
```

Chinese Long and Lat, Temperature Pressure, 2008 Jan 1st at 12 noon?

time	latitude	longitude	u10	v10	d2m	t2m	msl	sp
2008-01-01 02:00:00	23.25	113	-2.6031342	-4.829605	265.4314	284.8363	102918.8	102616.0
2008-01-01 02:00:00	23.25	114	-2.7173920	-3.808121	262.7693	284.2719	102862.1	101628.0
2008-01-01 02:00:00	22.25	113	-2.6246185	-6.311050	266.9294	284.2602	102796.6	102059.0
2008-01-01 02:00:00	22.25	114	-2.6285248	-6.152847	267.4978	285.3168	102710.1	102201.0
2008-01-01 12:00:00	23.25	113	-1.1495056	-2.728592	265.8091	286.1729	102588.9	102290.7
2008-01-01 12:00:00	23.25	114	-1.4454040	-2.477615	265.3033	285.0987	102591.6	101374.7
2008-01-01 12:00:00	22.25	113	-0.6924744	-4.270584	268.0396	286.5753	102482.9	101757.7
2008-01-01 12:00:00	22.25	114	-1.9668884	-4.906326	266.7486	288.0030	102440.1	101940.7

"ERA5 is a comprehensive reanalysis, from 1979 (soon to be backdated to 1950) to near real time, which assimilates as many observations as possible in the upper air and near surface. The ERA5 atmospheric model is coupled with a land surface model and a wave model."

1. Register for an account
2. Agree to Licence

1.8 Learning

1.8.1 Terminologies

Links:

- status of the CDS queue.

Terminologies:

- single level parameters

1.8.2 Single Level Parameters

ERA5 Variables?

1. Table 1: surface and single level parameters: invariants
2. Table 9: pressure level parameters: instantaneous

- Temperature

ER5 Data Download Instructions.

1.9 UTCI, NC Format Data, Download, Unzip, Convert to combined CSV

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: API request by itself
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

```
################################################
# ------------ 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')


# 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'  : [53.31, 73, 4.15, 135],
        '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
}
```