ERA-Interim: How to calculate daily total precipitation

ERA-Interim production to stop on 31st August 2019

As ERA5 is now available (What are the changes from ERA-Interim to ERA5?), we are preparing to **stop the production of ERA-Interim on 31st August 2019**. This means that the complete span of ERA-Interim data will be from 1st January 1979 to 31st August 2019.

Keeping in mind that ERA-Interim is published with an offset of about three months from the dataset's reference date, ERA-Interim August 2019 data will be made available towards the end of 2019.

For the time being and until further notice, ERA-Interim shall continue to be accessible through the ECMWF Web API. ERA5 is available from the Climate Data Store (CDS).

This knowledge base article shows you how to calculate daily total precipitation using ERA-Interim data. If you just want monthly means, then you can simply download it from http://apps.ecmwf.int/datasets/data/interim-mdfa/levtype=sfc/.

Before you continue, make sure you read through knowledge base articles listed below:

- How to download ERA-Interim data from the ECMWF data archive
- ERA-Interim: 'time' and 'steps', and instantaneous, accumulated and min/max parameters

You are also supposed to know how to work with Python under Linux, in particular, how to install packages using pip. You are recommended to use the latest release of packages listed here:

- ECMWF WebAPI (tested with 1.5.0) - required for step 1
- netCDF4 (tested with 1.4.0) - required for step 2
- numpy (tested with 1.14.5) - required for step 2

Step-by-step guide

1. Use script below to download daily total precipitation ERA-Interim data for 1st January 2018. This script will download total precipitation from two forecasts at 00 UTC and 12 UTC, both at step 12. This will give you 24 hours coverage.
    a. Time 00 UTC and step 12 will give you 00 - 12 UTC total precipitation data
    b. Time 12 UTC and step 12 will give you 12 - 24 UTC total precipitation data

```
#!/usr/bin/env python
"""
Save as get-tp.py, then run "python get-tp.py".

Input file : None
Output file: tp_20180101.nc
"""
from ecmwfapi import ECMWFDataServer

server = ECMWFDataServer()
server.retrieve({
    "class"  : "ei",
    "dataset": "interim",
    "date"   : "2018-01-01",
    "expver" : "1",
    "grid"   : "0.75/0.75",
    "levtype": "sfc",
    "param"  : "228.128",
    "step"   : "12",
    "stream" : "oper",
    "time"   : "00:00:00/12:00:00",
    "type"   : "fc",
    "format" : "netcdf",
    "target" : "tp_20180101.nc",
})
```

2. Run a second script to calculate daily total precipitation. All it does is to add up the two values for 00 - 12 UTC and 12 - 24 UTC for a given day.

```
#!/usr/bin/env python
"""
Save as file calculate-daily-tp.py and run "python calculate-daily-tp.py".

Input file : tp_20180101.nc
Output file: daily-tp_20180101.nc
```

```
"""
import time, sys
from datetime import datetime, timedelta

from netCDF4 import Dataset, date2num, num2date
import numpy as np

day = 20180101
f_in = 'tp_%d.nc' % day
f_out = 'daily-tp_%d.nc' % day

d = datetime.strptime(str(day), '%Y%m%d')
time_needed = [d + timedelta(hours = 12), d + timedelta(days = 1)]

with Dataset(f_in) as ds_src:
    var_time = ds_src.variables['time']
    time_avail = num2date(var_time[:], var_time.units,
            calendar = var_time.calendar)

    indices = []
    for tm in time_needed:
        a = np.where(time_avail == tm)[0]
        if len(a) == 0:
            sys.stderr.write('Error: precipitation data is missing/incomplete - %s!\n'
                    % tm.strftime('%Y%m%d %H:%M:%S'))
            sys.exit(200)
        else:
            print('Found %s' % tm.strftime('%Y%m%d %H:%M:%S'))
            indices.append(a[0])

    var_tp = ds_src.variables['tp']
    tp_values_set = False
    for idx in indices:
        if not tp_values_set:
            data = var_tp[idx, :, :]
            tp_values_set = True
        else:
            data += var_tp[idx, :, :]

    with Dataset(f_out, mode = 'w', format = 'NETCDF3_64BIT_OFFSET') as ds_dest:
        # Dimensions
        for name in ['latitude', 'longitude']:
            dim_src = ds_src.dimensions[name]
            ds_dest.createDimension(name, dim_src.size)
            var_src = ds_src.variables[name]
            var_dest = ds_dest.createVariable(name, var_src.datatype, (name,))
            var_dest[:] = var_src[:]
            var_dest.setncattr('units', var_src.units)
            var_dest.setncattr('long_name', var_src.long_name)

        ds_dest.createDimension('time', None)

        # Variables
        var = ds_dest.createVariable('time', np.int32, ('time',))
        time_units = 'hours since 1900-01-01 00:00:00'
        time_cal = 'gregorian'
        var[:] = date2num([d], units = time_units, calendar = time_cal)
        var.setncattr('units', time_units)
        var.setncattr('long_name', 'time')
        var.setncattr('calendar', time_cal)
        var = ds_dest.createVariable(var_tp.name, np.double, var_tp.dimensions)
        var[0, :, :] = data
        var.setncattr('units', var_tp.units)
        var.setncattr('long_name', var_tp.long_name)

        # Attributes
        ds_dest.setncattr('Conventions', 'CF-1.6')
        ds_dest.setncattr('history', '%s %s'
                % (datetime.now().strftime('%Y-%m-%d %H:%M:%S'),
                ' '.join(time.tzname)))
        print('Done! Daily total precipitation saved in %s' % f_out)
```

For simplicity, data in the output NetCDF file of the second script is unpacked. You may want to pack the data to save some disk spaces. Refer to https://www.unidata.ucar.edu/software/netcdf/docs/BestPractices.html#bp_Packed-Data-Values for detailed information.

Related articles

- ERA-Interim documentation
- What is ERA-Interim
- Differences between ERA-Interim and ERA-Interim/Land
- How to specify dates for ERA-Interim daily and monthly data using Python
- What is the data volume of all ERA-Interim data