EASE Grid

EASE Grid Coordinates

584, 1388

586, 1383

Lat Long, 1441, 721

Long = 620 < - > 900 =281

Lat = 60 <-> 260 =201

<https://nsidc.org/data/nsidc-0342/versions/2>

[https://nsidc.org/data/NSIDC-0032/versions/2#](https://nsidc.org/data/NSIDC-0032/versions/2)

<https://daacdata.apps.nsidc.org/pub/DATASETS/nsidc0342_nrt_ease_grid_tbs_v02/>

#!/bin/bash

GREP\_OPTIONS=''

cookiejar=$(mktemp cookies.XXXXXXXXXX)

netrc=$(mktemp netrc.XXXXXXXXXX)

chmod 0600 "$cookiejar" "$netrc"

function finish {

rm -rf "$cookiejar" "$netrc"

}

trap finish EXIT

WGETRC="$wgetrc"

prompt\_credentials() {

echo "Enter your Earthdata Login or other provider supplied credentials"

read -p "Username (kenan): " username

username=${username:-kenan}

read -s -p "Password: " password

echo "machine urs.earthdata.nasa.gov login $username password $password" >> $netrc

echo

}

exit\_with\_error() {

echo

echo "Unable to Retrieve Data"

echo

echo $1

echo

echo "https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h16v06.006.2018022025812.hdf"

echo

exit 1

}

prompt\_credentials

detect\_app\_approval() {

approved=`curl -s -b "$cookiejar" -c "$cookiejar" -L --max-redirs 2 --netrc-file "$netrc" https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h16v06.006.2018022025812.hdf -w %{http\_code} | tail -1`

if [ "$approved" -ne "302" ]; then

# User didn't approve the app. Direct users to approve the app in URS

exit\_with\_error "Please ensure that you have authorized the remote application by visiting the link below "

fi

}

setup\_auth\_curl() {

# Firstly, check if it require URS authentication

status=$(curl -s -z "$(date)" -w %{http\_code} https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h16v06.006.2018022025812.hdf | tail -1)

if [[ "$status" -ne "200" && "$status" -ne "304" ]]; then

# URS authentication is required. Now further check if the application/remote service is approved.

detect\_app\_approval

fi

}

setup\_auth\_wget() {

# The safest way to auth via curl is netrc. Note: there's no checking or feedback

# if login is unsuccessful

touch ~/.netrc

chmod 0600 ~/.netrc

credentials=$(grep 'machine urs.earthdata.nasa.gov' ~/.netrc)

if [ -z "$credentials" ]; then

cat "$netrc" >> ~/.netrc

fi

}

fetch\_urls() {

if command -v curl >/dev/null 2>&1; then

setup\_auth\_curl

while read -r line; do

curl -f -b "$cookiejar" -c "$cookiejar" -L --netrc-file "$netrc" -Og -- $line && echo || exit\_with\_error "Command failed with error. Please retrieve the data manually."

done;

elif command -v wget >/dev/null 2>&1; then

# We can't use wget to poke provider server to get info whether or not URS was integrated without download at least one of the files.

echo

echo "WARNING: Can't find curl, use wget instead."

echo "WARNING: Script may not correctly identify Earthdata Login integrations."

echo

setup\_auth\_wget

while read -r line; do

wget --load-cookies "$cookiejar" --save-cookies "$cookiejar" --keep-session-cookies -- $line && echo || exit\_with\_error "Command failed with error. Please retrieve the data manually."

done;

else

exit\_with\_error "Error: Could not find a command-line downloader. Please install curl or wget"

fi

}

fetch\_urls <<'EDSCEOF'

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h16v06.006.2018022025812.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h16v05.006.2018022025806.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h15v05.006.2018022025842.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h16v02.006.2018022025907.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h17v06.006.2018022025846.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h18v06.006.2018022025813.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h18v04.006.2018022025846.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h17v05.006.2018022025814.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h18v05.006.2018022025846.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h17v04.006.2018022025846.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h17v01.006.2018022025814.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h17v03.006.2018022025841.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h18v03.006.2018022025848.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h17v02.006.2018022025840.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h20v06.006.2018022025818.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h19v05.006.2018022025846.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h19v06.006.2018022025846.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h20v05.006.2018022025816.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h19v04.006.2018022025848.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h18v01.006.2018022025810.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h18v02.006.2018022025848.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h19v03.006.2018022025847.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h22v06.006.2018022025850.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h21v06.006.2018022025817.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h21v05.006.2018022025850.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h21v04.006.2018022025851.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h20v04.006.2018022025851.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h20v03.006.2018022025843.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h19v01.006.2018022030120.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h19v02.006.2018022025844.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h20v02.006.2018022025817.hdf

https://n5eil01u.ecs.nsidc.org/DP5/MOST/MOD10A1.006/2018.01.20/MOD10A1.A2018020.h20v01.006.2018022025816.hdf

EDSCEOFwget --load-cookies ~/.urs\_cookies --save-cookies ~/.urs\_cookies --keep-session-cookies --no-check-certificate --auth-no-challenge=on -r --reject "index.html\*" -np -e robots=off https://daacdata.apps.nsidc.org/pub/DATASETS/nsidc0032\_ease\_grid\_tbs/global/2018/EASE-F17-ML2018001D-V2.37V.gz

import os  
import pyresample  
import numpy as np  
import glob  
import h5py  
import datetime  
  
  
  
class H13\_Raw\_data\_channel\_generator(object):  
  
 def \_\_init\_\_(self):  
 lats\_bin\_file = 'EASE2\_M25km.lats.1388x584x1.double'  
 lons\_bin\_file = 'EASE2\_M25km.lons.1388x584x1.double'  
 self.width = 1388  
 self.height = 584  
 self.hsaf\_extensions = {'height': [60, 261], "width": [620,901]}  
 self.output\_location = None  
 self.input\_location = None  
 self.file\_mask = "%s\_SSMIS\_%s\_TL-25-75\_BR45-25\_res0.25.hdf"  
 self.initiation\_time = datetime.datetime.now()  
 this\_dir, this\_filename = os.path.split(\_\_file\_\_)  
 self.lats\_ease = np.fromfile(os.path.join(this\_dir, lats\_bin\_file), dtype=np.float64).reshape(  
 (self.height, self.width))  
 self.lons\_ease = np.fromfile(os.path.join(this\_dir, lons\_bin\_file), dtype=np.float64).reshape(  
 (self.height, self.width))  
 self.lons\_wgs, self.lats\_wgs = np.meshgrid(np.arange(-180, 180 + 0.25, .25), np.arange(90, -90 - 0.25, -.25))  
 # From H  
 self.orig\_def = pyresample.geometry.SwathDefinition(lons=self.lons\_ease, lats=self.lats\_ease)  
 self.targ\_def = pyresample.geometry.SwathDefinition(lons=self.lons\_wgs, lats=self.lats\_wgs)  
  
  
  
  
  
  
 @staticmethod  
 def return\_binary\_data(f\_name\_, dataset="data"):  
 f = h5py.File(f\_name\_, "r")  
 c = np.array(f[dataset])  
 c = np.delete(c, -1, 0)  
 c = np.delete(c, -1, 0)  
 c = np.c\_[np.zeros(c.shape[0]), c]  
 c = np.c\_[np.zeros(c.shape[0]), c]  
 c = np.c\_[c, np.zeros(c.shape[0])]  
 c = np.c\_[c, np.zeros(c.shape[0])]  
 c = np.c\_[c, np.zeros(c.shape[0])]  
 f.close()  
 return c  
  
 def line\_and\_datetime(func):  
 def wrapper(self, \*args, \*\*kwargs):  
 st = datetime.datetime.now()  
 print "="\*50  
 ret = func(self,\*args, \*\*kwargs)  
 en = datetime.datetime.now() - st  
 print "Has been finished in", en  
 print "="\*50  
 return ret  
 return wrapper  
  
 @line\_and\_datetime  
 def return\_data(self, f\_name\_, row=None, column=None):  
 if row is None:  
 row = self.height  
  
 if column is None:  
 column = self.width  
 with open(f\_name\_, 'rb') as binfile:  
 channel\_content = binfile.read()  
 c = np.frombuffer(bytearray(channel\_content), dtype=np.int16)  
 c = np.reshape(c, (row+2, column-5))  
 c = np.delete(c, -1, 0)  
 c = np.delete(c, -1, 0)  
 c = np.c\_[np.zeros(c.shape[0]), c]  
 c = np.c\_[np.zeros(c.shape[0]), c]  
 c = np.c\_[c, np.zeros(c.shape[0])]  
 c = np.c\_[c, np.zeros(c.shape[0])]  
 c = np.c\_[c, np.zeros(c.shape[0])]  
 return c  
  
 @staticmethod  
 def retun\_date\_string(s\_, year=2018):  
 return datetime.datetime.strptime(str(year) + "0101", "%Y%m%d") + datetime.timedelta(  
 days=int(os.path.basename(s\_)[15:18]) - 1)  
  
 @staticmethod  
 def retunflag(s\_):  
 temp = s\_.split(".")[-1]  
 if temp == '19H':  
 return '19GH'  
 elif temp == '19V':  
 return '19GV'  
 elif temp == '37H':  
 return '36GH'  
 elif temp == '37V':  
 return '36GV'  
 else:  
 raise Exception("Data Filename is not acceptable")  
  
 @line\_and\_datetime  
 def run(self):  
 if self.output\_location is None or self.input\_location is None:  
 raise Exception("Input or Output locations must be set...")  
 input\_path = self.input\_location  
 files = [os.path.join(input\_path, row) for row in glob.glob1(input\_path, "EASE\*")]  
 for file\_ in files:  
 try:  
 inner = datetime.datetime.now()  
 print file\_, "is being processed"  
 c = self.return\_data(file\_)  
 c\_gauss = pyresample.kd\_tree.resample\_gauss(self.orig\_def, c, self.targ\_def, radius\_of\_influence=25000,  
 neighbours=10,  
 sigmas=12500, fill\_value=None)  
 a = c\_gauss.filled(0)  
 # d = a[60:261, 620:901]  
 d = a[self.hsaf\_extensions['height'][0]:self.hsaf\_extensions['height'][1],  
 self.hsaf\_extensions['width'][0]:self.hsaf\_extensions['width'][1]]  
 flag = self.retunflag(file\_)  
 f\_name\_ = self.file\_mask % (self.retun\_date\_string(file\_).strftime("%Y-%m-%d"), flag)  
 with h5py.File(os.path.join(self.output\_location, f\_name\_), 'w') as channel\_file:  
 channel\_file.create\_dataset("data", data=d / 10.0, compression="gzip")  
 channel\_file.close()  
 print datetime.datetime.now() - inner  
 except BaseException as be\_message:  
 print "ERROR ==> ", file\_  
 print be\_message.message  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 channel\_regenerate = H13\_Raw\_data\_channel\_generator()  
 path\_ = "/media/sf\_I\_DRIVE/EASE\_GRID/DATA"  
 channel\_regenerate.input\_location = os.path.join(path\_, "input")  
 channel\_regenerate.output\_location = os.path.join(path\_, "output")  
 channel\_regenerate.run()  
  
#  
# lats = np.fromfile(, dtype = np.float64).reshape((584, 1388))  
# lons = np.fromfile('', dtype=np.float64).reshape((584, 1388))  
#  
# st = datetime.datetime.now()  
# lats = np.fromfile('EASE2\_M25km.lats.1388x584x1.double', dtype=np.float64).reshape((584, 1388))  
# lons = np.fromfile('EASE2\_M25km.lons.1388x584x1.double', dtype=np.float64).reshape((584, 1388))  
#  
# # region alternative methods  
#  
# #  
# # path\_ = r""  
# # fr = Dataset(os.path.join(path\_, 'air.mon.mean.nc'))  
# # air = fr.variables['air'][0, :, :]  
# # lat = fr.variables['lat'][:]  
# # lon = fr.variables['lon'][:]  
# # fr.close()  
# #  
# # from ease\_grid import EASE2\_grid  
# #  
# # egrid = EASE2\_grid(25000)  
# # # these two attributes contain the longitude and latitude coordinate dimension  
# # print(egrid.londim)  
# # egrid.latdim  
#  
# # endregion  
#  
# lon2d, lat2d = lons, lats  
# lon\_curv, lat\_curv = np.meshgrid(np.arange(-180, 180 + 0.25, .25), np.arange(90, -90 - 0.25, -.25))  
# orig\_def = pyresample.geometry.SwathDefinition(lons=lon2d, lats=lat2d)  
# targ\_def = pyresample.geometry.SwathDefinition(lons=lon\_curv, lats=lat\_curv)  
#  
#  
# def return\_binary\_data(f\_name\_, dataset="data"):  
# f = h5py.File(f\_name\_, "r")  
# c = np.array(f[dataset])  
# c = np.delete(c, -1, 0)  
# c = np.delete(c, -1, 0)  
# c = np.c\_[np.zeros(c.shape[0]), c]  
# c = np.c\_[np.zeros(c.shape[0]), c]  
# c = np.c\_[c, np.zeros(c.shape[0])]  
# c = np.c\_[c, np.zeros(c.shape[0])]  
# c = np.c\_[c, np.zeros(c.shape[0])]  
# f.close()  
# return c  
#  
#  
# # c\_nearest = pyresample.kd\_tree.resample\_nearest(orig\_def, c, targ\_def, radius\_of\_influence=25000, fill\_value=None)  
#  
# # wf = lambda r: 1 / r \*\* 2  
# # c\_idw = pyresample.kd\_tree.resample\_custom(orig\_def, c, targ\_def, radius\_of\_influence=30000, neighbours=30,  
# # weight\_funcs=wf, fill\_value=None)  
# # plt.imshow(c\_idw)  
#  
#  
# #  
# # transformation\_matrix = {  
# # # "H13": [(-25.12500, 0.25, 0.0, 75.125, 0.0, -0.25), (201, 281), [75.000, -25.000], [25.000, 45.0000]],  
# # "EASE": [(-180.12500, 0.25, 0.0, 90.125, 0.0, -0.25), (586, 1383), [90.000, -180.000], [-90.000, 180.0000]],  
# # "GLOBAL": [(-180.12500, 0.25, 0.0, 90.125, 0.0, -0.25), (721, 1441), [90.000, -180.000], [-90.000, 180.0000]],  
# # }  
#  
# def return\_data(f\_name\_, row=586, column=1383):  
# with open(f\_name\_, 'rb') as binfile:  
# channel\_content = binfile.read()  
# c = np.frombuffer(bytearray(channel\_content), dtype=np.int16)  
# c = np.reshape(c, (row, column))  
# c = np.delete(c, -1, 0)  
# c = np.delete(c, -1, 0)  
# c = np.c\_[np.zeros(c.shape[0]), c]  
# c = np.c\_[np.zeros(c.shape[0]), c]  
# c = np.c\_[c, np.zeros(c.shape[0])]  
# c = np.c\_[c, np.zeros(c.shape[0])]  
# c = np.c\_[c, np.zeros(c.shape[0])]  
# return c  
#  
#  
# def retun\_date\_string(s\_):  
# return datetime.datetime.strptime("20180101", "%Y%m%d") + datetime.timedelta(  
# days=int(os.path.basename(s\_)[15:18]) - 1)  
#  
#  
# def retunflag(s\_):  
# temp = s\_.split(".")[-1]  
# if temp == '19H':  
# return '19GH'  
# elif temp == '19V':  
# return '19GV'  
# elif temp == '37H':  
# return '36GH'  
# elif temp == '37V':  
# return '36GV'  
# else:  
# raise Exception("Data Filename is not acceptable")  
  
#  
# save\_path = "/media/sf\_I\_DRIVE/EASE\_GRID/DATA/daacdata.apps.nsidc.org/pub/DATASETS/nsidc0032\_ease\_grid\_tbs/global"  
# save\_path = "/media/sf\_I\_DRIVE/EASE\_GRID/DATA/daacdata.apps.nsidc.org/pub/DATASETS/nsidc0032\_ease\_grid\_tbs/global/daacdata.apps.nsidc.org/pub/DATASETS/nsidc0032\_ease\_grid\_tbs/global"  
# input\_path = os.path.join(save\_path, "2018")  
# files = [os.path.join(input\_path, row) for row in glob.glob1(input\_path, "EASE\*")]  
# # files = ["data.h5"]  
#  
# for file\_ in files:  
# try:  
# inner = datetime.datetime.now()  
# print file\_, "is being processed"  
# c = return\_data(file\_)  
# c\_gauss = pyresample.kd\_tree.resample\_gauss(orig\_def, c, targ\_def, radius\_of\_influence=25000, neighbours=10,  
# sigmas=12500, fill\_value=None)  
# a = c\_gauss.filled(0)  
# d = a[60:261, 620:901]  
# flag = retunflag(file\_)  
#  
# f\_name\_place\_holder = "%s\_SSMIS\_%s\_TL-25-75\_BR45-25\_res0.25.hdf"  
# f\_name\_ = f\_name\_place\_holder % (retun\_date\_string(file\_).strftime("%Y-%m-%d"), flag)  
# channel\_file = h5py.File(os.path.join(save\_path, f\_name\_), 'w')  
# channel\_file.create\_dataset("data", data=d / 10.0, compression="gzip")  
# channel\_file.close()  
# print datetime.datetime.now() - inner  
# except BaseException as be:  
# print "ERORRRRRRR ==> ", file\_  
#  
# print datetime.datetime.now() - st  
# target = osr.SpatialReference()  
# target.ImportFromEPSG(4326)  
# export\_data = gdal.GetDriverByName("GTiff").Create("z\_%d.tif" %counter, transformation\_matrix['GLOBAL'][1][1],  
# transformation\_matrix['GLOBAL'][1][0], 1, gdal.GDT\_Float32)  
# export\_data.GetRasterBand(1).SetNoDataValue(999)  
# export\_data.GetRasterBand(1).WriteArray((c\_gauss.filled(999)))  
# export\_data.SetGeoTransform(transformation\_matrix['GLOBAL'][0])  
# export\_data.SetProjection(target.ExportToWkt())  
# export\_data.FlushCache()  
# gdal.ReprojectImage(src, export\_data, src\_proj, target.ExportToWkt(), gdalconst.GRA\_NearestNeighbour)  
# input\_data.FlushCache()

<http://earthpy.org/interpolation_between_grids_with_pyresample.html>

<https://spectraldifferences.wordpress.com/2015/11/24/convert-ease-2-grid-cell-to-latitude-and-longitude/>

<https://nsidc.org/pmesdr/files/2016/02/2016_Brodzik_et_al_OceanSciences.final_.pdf>

https://nsidc.org/support/how/Points-Pixels-Grids-and-Cells-A-Mapping-and-Gridding-Primer

