shortcourse_hw_0

March 11, 2022

1 2021 gnssrefl short course

1.1 Homework 0

Due date: This homework is to be completed **before** the short course given on October 21. You need to make sure the software has been properly installed and you have successfully completed the "homework 0" assignment.

Purpose: To test if environment and code is ready for gnssrefl processing

NOTE: if you plan to use this jupyter notebook then please follow the instructions here for running the notebook in a docker container OR running locally.

TIP: When running cells in jupyter notebook,the In[*] on the top left means that the cell is currently running.

1.1.1 Step 1: Check that we can import gnssrefl and other required imports

- If you are running the docker image, then the gnssrefl code and other required imports should be installed and the following cell should import it's functions with no issues.
- If you are running this notebook locally, then make sure that in your terminal, you run `pip install -r requirements.txt` in the main directory of this repository where the requirements.txt file is stored.

Now run the following cell:

[4]:
```
# all of these imports should be installed
# and no errors will return when the cell is run
# If there are no errors then you are all set to move forward

import os
import sys
import re
import json
import pandas as pd
import numpy as np
import seaborn as sns; sns.set_theme(style="whitegrid");
import matplotlib.pyplot as plt
```

```
%matplotlib inline
```

The following cell is necessary to import python modules from the repository. These python modules are stored in the 'bin' folder. If this notebook has been placed somewhere other than the location where it started in the repository, then you can manually change the path below to where the bin has been placed. Otherwise, it should be in the two directories 'behind' from this notebook.

If the following cell does not throw any errors then we can move forward.

[5]:
```
# We are including our repository bin to the system path so that we can import␣
 ↪the following python modules
path = '../../bin'
bin_path = os.path.abspath(os.path.join(path))
if bin_path not in sys.path:
    sys.path.append(bin_path)

import run_gnssrefl
import gnssrefl_helpers
```

1.1.2 Step 2. Set the environmnet variables

This next cell will set your environment variables. If they are not already set (done previously or via docker) - then they will be set for you - assuming the directory structure has not changed from the repository. There are three required environment variables:

- EXE - where various RINEX executables will live.
- ORBITS - where the GPS/GNSS orbits will be stored. They will be listed under directories by year and sp3 or nav depending on the orbit format.
- REFL_CODE - where the reflection code inputs (SNR files and instructions) and outputs (RH) will be stored (see below). Both SNR files and results will be saved here in year subdirectories.

If you are running the docker container then the environment variables should look like * ORBITS = /home/jovyan/gnssir_jupyter/orbits * EXE = /home/jovyan/gnssir_jupyter/bin/exe * REFL_CODE = ORBITS = /home/jovyan/gnssir_jupyter

You can also define parameters orbits=, exe=, refl_code= with environment.set_environment() to manually set the locations for these environment variables.

Once you run the following cell, it will print out the locations that these environment variables are set to. If these locations are satisfactory then we can move forward.

[6]:
```
#Making sure environment variables are set - this is required to run the␣
 ↪gnssrefl code
exists = gnssrefl_helpers.check_environment()
```

```
# if the environment variables are not set already then the exists variable␣
 ↪will return as False.
if exists == False:
    gnssrefl_helpers.set_environment()
else:
     print('environment variable ORBITS path is', os.environ['ORBITS'],
          '\nenvironment variable REFL_CODE path is', os.environ['REFL_CODE'],
          '\nenvironment variable EXE path is', os.environ['EXE'])
```

```
environment variable ORBITS path is /home/jovyan/gnssrefl_jupyter/orbits
environment variable REFL_CODE path is /home/jovyan/gnssrefl_jupyter
environment variable EXE path is /home/jovyan/gnssrefl_jupyter/bin/exe
```

1.1.3 Step 3. Download and check EXE dependencies are present:

use environment.download_crx2rnx to import the crx2rnx file(Required translator for compressed (Hatanaka) RINEX files) which is dependant on your working OS - this is required to run the gnssrefl code.

If this does not properly find your running os, then it will print out an error and instruct you how to add a parameter to manually set which os you are using.

Note that this function relies on your environment variables to be properly set.

[ ]:
```
# import the crx2rnx file which is dependant on your working OS - this is␣
 ↪required to run the gnssrefl code
gnssrefl_helpers.download_crx2rnx()

print('files in exe folder:', os.listdir(os.environ['EXE']))
```

If you see 'CRX2RNX' and 'gfzrnx' in your EXE folder then you are all set.

Note* that the gfzrnx file was in the exe when you pulled the repository - it currently is set for a linux environment and can only be used with the docker version of the jupyter notebooks or if you are running linux. The gfzrnx file is not required to run the code - but is needed if you want to work with RINEX3 files. If you need to download the correct version for your os then download from here and then place it in your exe folder.

1.1.4 Step 4. Run a quick Analysis

a. simple use case that requires CRX2RNX and broadcast orbits:

[8]:
```
station = 'p042'
year = 2018
doy = 150
```

To understand what rinex2snr returns, lets look at the function's available and default parameters:

[ ]:
```
run_gnssrefl.rinex2snr?
```

Now lets run the function without changing any of the defaults.

[ ]:
```
run_gnssrefl.rinex2snr(station, year, doy)
```

```
Will seek RINEX file  p042  year: 2018  doy: 150  translate with  hybrid  at the
all  archive
sopac did not work, so try cddis

--2022-03-11 06:34:35--
ftps://gdc.cddis.eosdis.nasa.gov/gps/data/daily/2018/150/18n/brdc1500.18n.Z
           => ‘brdc1500.18n.Z’
Resolving gdc.cddis.eosdis.nasa.gov (gdc.cddis.eosdis.nasa.gov)…
198.118.242.43
Connecting to gdc.cddis.eosdis.nasa.gov
(gdc.cddis.eosdis.nasa.gov)|198.118.242.43|:21… connected.
==> AUTH TLS … WARNING: cannot verify gdc.cddis.eosdis.nasa.gov's certificate,
issued by ‘emailAddress=support@fortinet.com,CN=FGT1KD3916800558,OU=Certificate
Authority,O=Fortinet,L=Sunnyvale,ST=California,C=US’:
  Self-signed certificate encountered.
 done.
Logging in as anonymous … Logged in!
==> PBSZ 0 … done.  ==> PROT P … done.
==> SYST … done.    ==> PWD … done.
==> TYPE I … done.  ==> CWD (1) /gps/data/daily/2018/150/18n … done.
==> SIZE brdc1500.18n.Z … 89389
==> PASV … done.    ==> RETR brdc1500.18n.Z …
No such file ‘brdc1500.18n.Z’.

--2022-03-11 06:34:55--
ftps://gdc.cddis.eosdis.nasa.gov/gps/data/daily/2018/150/18n/brdc1500.18n.gz
           => ‘brdc1500.18n.gz’
Resolving gdc.cddis.eosdis.nasa.gov (gdc.cddis.eosdis.nasa.gov)…
198.118.242.43
Connecting to gdc.cddis.eosdis.nasa.gov
(gdc.cddis.eosdis.nasa.gov)|198.118.242.43|:21… connected.

going for the gzip version of the file

==> AUTH TLS … WARNING: cannot verify gdc.cddis.eosdis.nasa.gov's certificate,
issued by ‘emailAddress=support@fortinet.com,CN=FGT1KD3916800558,OU=Certificate
Authority,O=Fortinet,L=Sunnyvale,ST=California,C=US’:
  Self-signed certificate encountered.
 done.
Logging in as anonymous … Logged in!
==> PBSZ 0 … done.  ==> PROT P … done.
==> SYST … done.    ==> PWD … done.
==> TYPE I … done.  ==> CWD (1) /gps/data/daily/2018/150/18n … done.
==> SIZE brdc1500.18n.gz … done.

==> PASV … done.    ==> RETR brdc1500.18n.gz …
```

you've successfully run the rinex2snr program that: * downloaded and uncompressed hatanaka rinex for a single station (p042) for a single day (doy 150 in 2018) * downloaded GPS broadcast orbits * calculated azimuth and elevation for each satellite at each epoch given these orbits * wrote this az/el, signal, time and CN0 information to a formatted snr output file for future analysis. Reminder, the .66 file name suffix refers to the elevation masking options.

b. simple use case that requires CRX2RNX and SP3 orbits: Here we will run rinex2snr for the same day, but lets change the 'orb' parameter to gnss.

[ ]:
```
run_gnssrefl.rinex2snr(station, year, doy=doy, orb='gnss')
```

Note* If you get: *SNR file exists...*
This is because the logic of gnssrefl checks for an snr file prior for processing - and we already processed this day earlier. Remember this fact if you ever want to **re**-process with different orbits! You can use the overwrite parameter to overwrite files if you want to reprocess. Now lets try that again.

[ ]:
```
run_gnssrefl.rinex2snr(station, year, doy=150, orb='gnss', overwrite=True)
```

If you get: SUCCESS: SNR file was created: ... you've successfully:

- downloaded and uncompressed hatanaka rinex for a single station (p042) for a single day (doy 150 in 2018)
- downloaded SP3 format GNSS orbits from the GFZ archive
- calculated azimuth and elevation for each satellite at each epoch
- wrote this az/el, signal, time and CN0 information to a formatted snr output file for future analysis.

c. (OPTIONAL - requires the gfzrnx executable mentioned previously) RINEX 3 simple use case that requires gfzrnx If you are interested in using RINEX version 3 data, please run this test:

note: this will fail if you do not have the correct system-dependant gfzrnx translation file. See the instructions above to get this file.

[ ]:
```
run_gnssrefl.rinex2snr(station='onsa00swe', year=2020, doy=1, archive='cddis',␣
↪orb='gnss')
```

If you get: *SUCCESS: SNR file was created: ...*
you've successfully: * downloaded and uncompressed rinex 3 for a single station (onsa) for a single day (doy 1 in 2020) from the cddis archive * converted rinex 3 to rinex 2 using gfzrnx executable * downloaded SP3 format GNSS orbits from the GFZ archive * calculated azimuth and elevation for each satellite at each epoch * wrote this az/el, signal, time and CN0 information to a formatted snr output file for future analysis.

[ ]:
```

```

[ ]:
```

```