Install

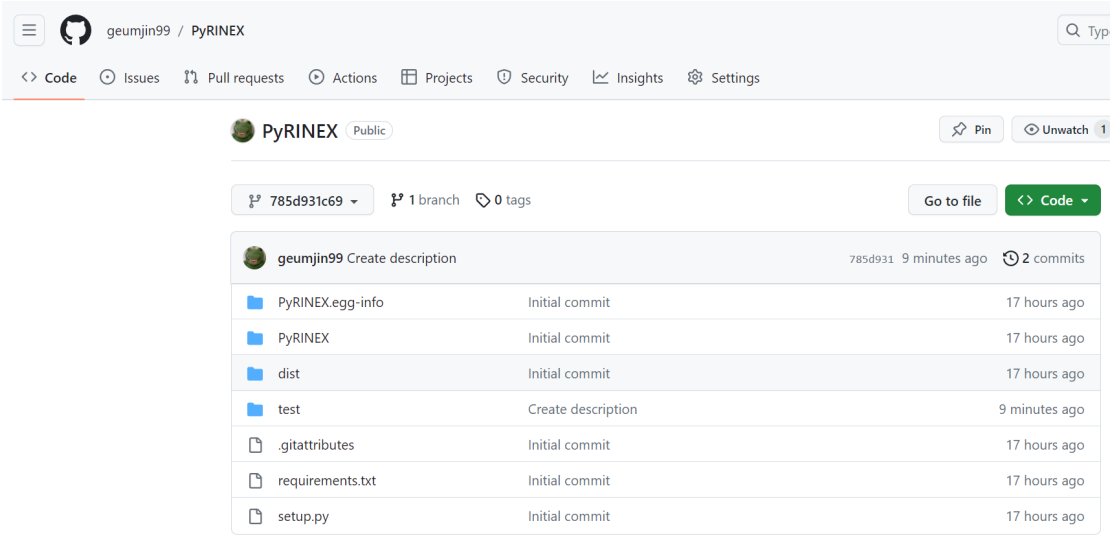


Figure 1 GitHub repository

The GitHub repository link is https://github.com/geumjin99/PyRINEX, you can download it or just clone it. The Test folder contains examples used in the paper.

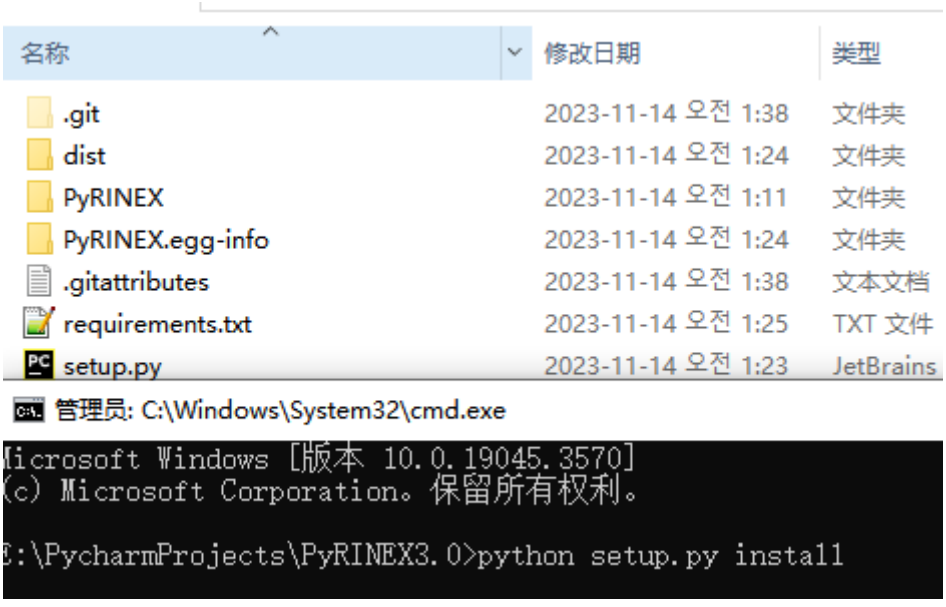


Figure 2 How to install with setup.py

If you are downloading, clock CMD in the downloaded path and enter the command phrase as shown in Figure 2.

The PyRINEX software project (hereinafter referred to as the "Project") is released under the Apache Software License 2.0 (hereinafter referred to as the "License"). Anyone using this Project must comply with the terms of this License.

Reader

Table 1 Functions in Reader module

Functions	Parameter
oheader	observation file path
observations	observation file path
navigaions	navigation file path

Functions that can read RINEX data are provided in the Reader module, and they can read RINEX observation files and navigation files. They both require the path to the RINEX data to be entered as an argument and return a new json formatted data called LITE RINEX.

```
version 2
type G
receiver_type [3, 'TRIMBLE 5700        ']
antenna_type [4, 'TRM39105.00         ']
MARKER_NAME [8, 'BM01003012']
MARKER_NUMBER [9, 'BM01003012']
APPROX POSITION XYZ [-3173543.0196, 4134173.2686, 3666275.4904]
TIME OF FIRST OBS ['2013-8-12-8-10-30.0000000']
END OF HEADER 45
PRNS ['G01', 'G03', 'G06', 'G07', 'G08', 'G09', 'G11', 'G13', 'G14',
ObsTypes ['L1', 'C1', 'L2', 'P2', 'D1']
```

Figure 3 Output of the oheader function

Figure 3 shows the output of the oheader function. This contains some of the most important information in the header section in both categories, such as the version of the RINEX file, information about the type of observation recorded, etc. The second type of information is the marker name, receiver type, etc. The most important feature of this type of information is that it can be edited according to the user's needs, and it can be seen that this type of information is stored in a list, which is because the number of rows where these information are located is not fixed, so the line number is stored in the first item of the list for the purpose of modifying the information later on in the original file.

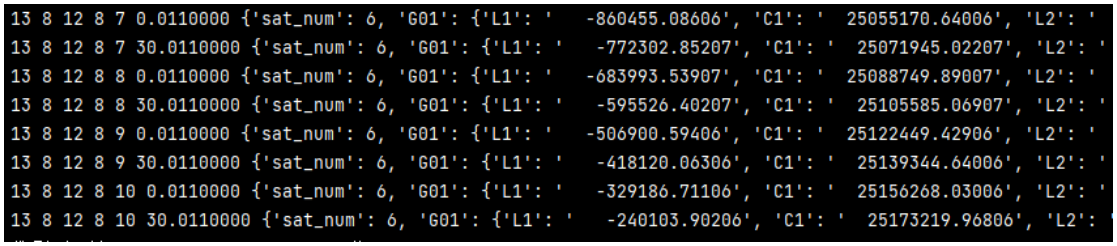


Figure 4 Output of the observations function

As shown in figure 4, the same logic applies to the translation of the logged portion of the observations. PyRINEX provides the ability to translate RINEX observations and GPS navigation files into LITE RINEX format. This is extremely helpful for a number of studies that aim to process GNSS observation data.

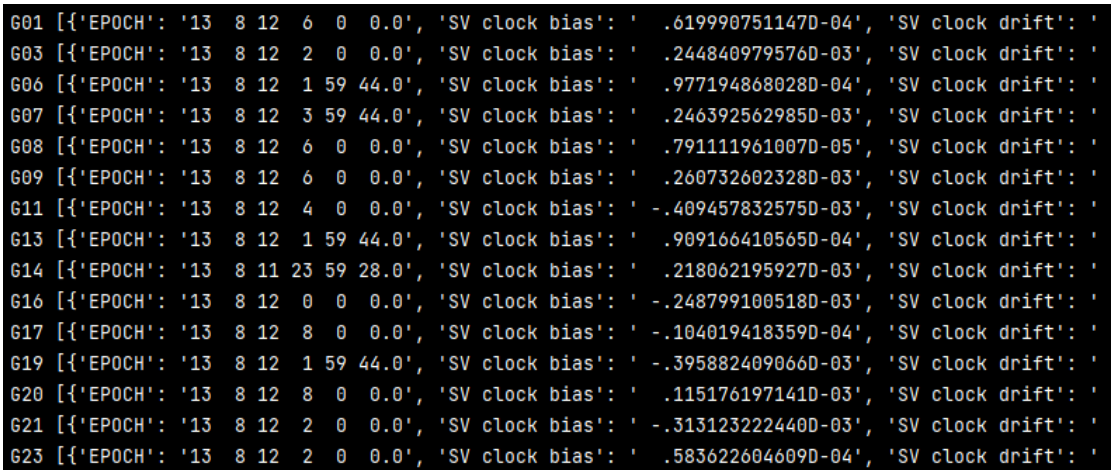
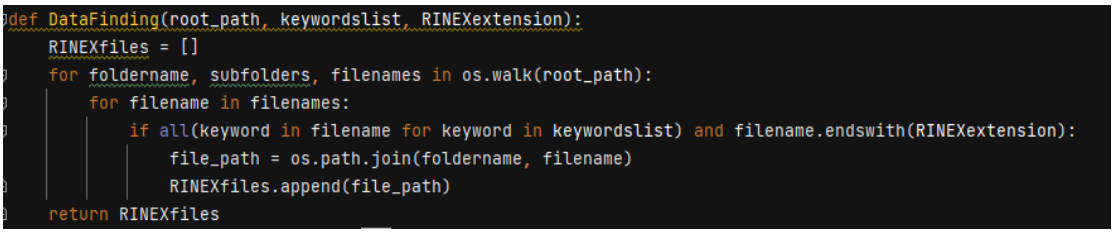


Figure 5 Output of the navigations function

Figure 5 shows the output of the navigations function, each satellite represented by a key will correspond to a list as a value, this is because it is possible for a satellite to be logged multiple times in the navigation file and the list will store them together.

DataManagement



```
def DataFinding(root_path, keywordslist, RINEXextension):
    RINEXfiles = []
    for foldername, subfolders, filenames in os.walk(root_path):
        for filename in filenames:
            if all(keyword in filename for keyword in keywordslist) and filename.endswith(RINEXextension):
                file_path = os.path.join(foldername, filename)
                RINEXfiles.append(file_path)
    return RINEXfiles
```

Figure 6 DataFinding function in DataManagement module

The DataFinding function implements the function of retrieving and filtering the RINEX data under a certain path. The function is implemented based on Python's os library, which is used to interact with the operating system, including file and directory operations. To use this function the user needs to enter the specified root directory, a list of keywords to filter its subfolders, and a extension representing the type of RINEX data file. After that, it will traverse all the files under the path, and then determine whether it meets the conditions, it is worth noting that even if the input

extension is "08o", some files with "08O" as extension will still be output in the result list because the RINEX standard format does not specify the extension case.

```
def DataCleaning(RINEX_FILES,ReceiverLibraryPath, AntennaLibraryPath, newfolder_root):
    fieldnames = ["origin path", "version", "station", "non English", "origin marker", "origin rec", "origin ant",
                  "new path", "marker", "longitude", "latitude", "rec type", "ant type"]
```

Figure 7 DataCleaning function in DataManagement module

In this function, you need to input the root path of the RINEX data that you want to be processed, the path of the ReceiverLibrary and AntennaLibrary, and the path that you want to output after data cleaning. The basic logic for PyRINEX to modify the four aforementioned errors is similar.

The DataCleaning function in DataManagement provide automatic errata for LITE RINEX after reading. It should be noted that for receiver type and antenna type corrections, the CSV files ReceiverLibrary and AntennaLibrary need to be read first, and PyRINEX will store the incorrect spelling and correct spelling as keys and values, respectively, as dictionaries in Python after reading them. After that, PyRINEX will check if there is a key in dictonray when it reads the corresponding line of the two contents, and if there is, it will replace it with the corresponding value, so that it can correct the specific contents in this way. The two CSV files can be freely edited by the user, which makes the processing of the data more customizable. CSV should like as shown in figure 8.

TRIMBLE 4000	?	TRIMBLE 4000S
TRIMBLE4700	?	TRIMBLE 4700
TRIMBLE5700	?	TRIMBLE 5700
TRIMBLE5800	?	TRIMBLE 5800
TRIMBLE5800II	?	TRIMBLE 5800
TPS GB1000	?	TPS GB-1000
TRIMBLENETR9	?	TRIMBLE NETR9

Figure 8 Example of ReceiverLibrary.csv/AntennaLirary.csv

For the protection of raw data, the new RINEX file after data cleaning will be written to a specified new path, user needs to specify the root path of the output. After that, PyRINEX will use the mkdir function in the os library to create a new folder with the corresponding "doy" name and write it to it, and then when there are RINEX data observed on the same date that are cleansed by the data, they will also be written to this folder, which can help to organize a large amount of unorganized RINEX data. The reporter CSV file is shown in figure 9.

origin path	version	station	non l	origin r	origin rec	origin ant	new path	marker	longitude	latitude	rec type	ant type	filename
F:\gpsdata\01	2		yes	7182	TPS GB100	TPSPG-A1	F:\RINEX07	7182	128.3502	37.56502	TPS GB-10	TPSPG-A1	TRUE
F:\gpsdata\01	2		yes	7183	TPS GB100	TPSPG-A1	F:\RINEX07	7183	128.2721	37.52085	TPS GB-10	TPSPG-A1	TRUE
F:\gpsdata\01	2		yes	7174	TPS GB100	TPSPG_A1	F:\RINEX07	7174	128.1715	37.72236	TPS GB-10	TPSPG_A1	TRUE
F:\gpsdata\01	2		yes	7175	TPS GB100	TPSPG_A1	F:\RINEX07	7175	128.0185	37.67045	TPS GB-10	TPSPG_A1	TRUE
F:\gpsdata\01	2		yes	7171	TPS GB100	TPSPG_A1	F:\RINEX07	7171	127.9244	37.68351	TPS GB-10	TPSPG_A1	TRUE
F:\gpsdata\01	2		yes	7169	TPS GB100	TPSPG_A1	F:\RINEX07	7169	127.9106	37.70805	TPS GB-10	TPSPG_A1	TRUE
F:\gpsdata\01	2		yes	7179	TPS GB100	TPSPG_A1	F:\RINEX07	7179	128.3068	37.74973	TPS GB-10	TPSPG_A1	TRUE
F:\gpsdata\01	2		yes	7180	TPS GB100	TPSPG-A1	F:\RINEX07	7180	128.4648	37.6714	TPS GB-10	TPSPG-A1	TRUE
F:\gpsdata\01	2		yes	7184	TPS GB100	TPSPG-A1	F:\RINEX07	7184	128.4596	37.49302	TPS GB-10	TPSPG-A1	TRUE
F:\gpsdata\01	2		yes	7181	TPS GB100	TPSPG-A1	F:\RINEX07	7181	128.4373	37.60671	TPS GB-10	TPSPG-A1	TRUE
F:\gpsdata\01	2			169	TRIMBLE 5	TRM39105.(	F:\RINEX07	169	127.9087	37.70846	TRIMBLE 5	TRM39105.(	TRUE
F:\gpsdata\01	2			170	TRIMBLE 5	TRM39105.(	F:\RINEX07	170	127.8543	37.70401	TRIMBLE 5	TRM39105.(	TRUE
F:\gpsdata\01	2			171	TRIMBLE 5	TRM39105.(	F:\RINEX07	171	127.9263	37.64545	TRIMBLE 5	TRM39105.(	TRUE
F:\gpsdata\01	2			174	Unknown!	TRM39105.(	F:\RINEX07	174	128.1738	37.72003	TRIMBLE 5	TRM39105.(	TRUE

QualityCheck

Table 2 Functions in QualityCheck module

Functions	Discription
plot (filename, title, gps serises, epochs, dataset, type, column)	Plot function for azi_ele(path), ION_MP(opath) and cycleslip(path)
SatelliteSignalPlot(path)	Output the signal plot
azi_ele(path)	Calculate the azimuth and elevation for GPS satellite
ION_MP(opath)	Calculate the multipath and ionospheric delay
cycleslip(path)	Calculate cycle slip
QualityCheck(path)	Perform all quality checks on a given RINEX data
batchQC(rootpath, keywords_list, extension)	Performs a quality check on RINEX data under a path, the principle is the same as the DataFinding function.

The plot function is responsible for visualizing the results of the quality check. It is worth noting, however, that if the user wishes to make changes to the function then the code for the parameters should not be changed, as these parameters are not actually entered by the user.

```
gps_prn = prns[1:]
name = os.path.basename(opath)
cloumn = math.ceil(len(prns)/18)
plot(opath[:-4] + "MP1_plot.jpg", name[:-4] + " MP1 plot", gps_prn, epochs, MP1MP2, 0,cloumn)
plot(opath[:-4] + "MP2_plot.jpg", name[:-4] + " MP2 plot", gps_prn, epochs, MP1MP2, 1,cloumn)
plot(opath[:-4] + "ION_plot.jpg", name[:-4] + " ION plot", gps_prn, epochs, MP1MP2, 2,cloumn)
plot(opath[:-4] + "IOD_plot.jpg", name[:-4] + " IOD plot", gps_prn, epochs, MP1MP2, 3,cloumn)
```

Figure 10 Examples of using the plot function

As shown in Figure 10, this function will be called in the rest of the functions of the QUALITY CHECK, whose parameters are determined by these functions, such as COLUMN determines how many columns the satellites should be written in the legend.

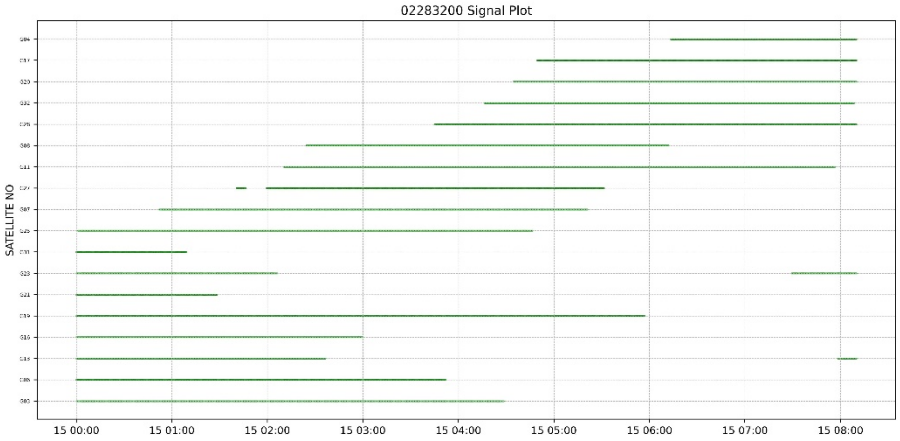


Figure 11 Examples output of the SatelliteSignalPlot function

In the SatelliteSignalPlot function, a visualization of the satellite models received by the receiver in each time slot is provided. The function outputs a schematic diagram, which allows the user to visualize the type of satellites received during each time period and, more importantly, to know which satellites have had interruptions in the reception of their signals, which means that it is possible that poor observing conditions have triggered difficulties in the reception of the signals. Figure 11 shows schematic output of the SatelliteSignalPlot function.

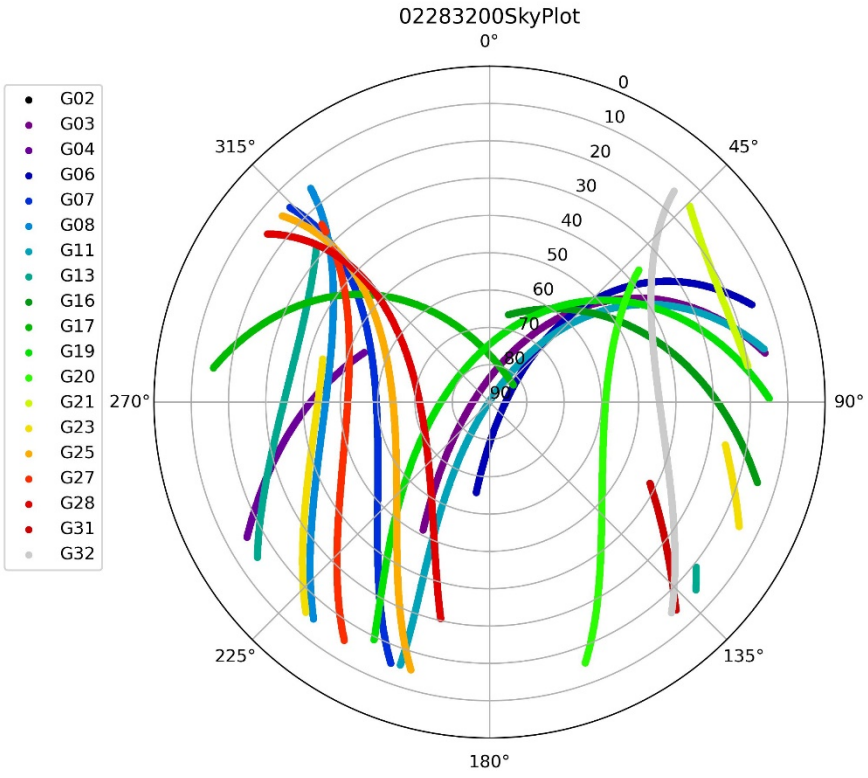


Figure 12 Examples output of the azi_ele function

When using the azi_ele function, you need to make sure that the RINEX data is internally logged with GPS satellite data and that the navigation file and the observation file are under the same path.

epoch	G01	G03	G06	G07	G08	G09	G11	G13	G14
13 8 12 0 0 30.0010000									[2.7231193335204305, 0.6197214451251352]
13 8 12 0 1 0.0010000									[2.7244568001278826, 0.6155936610163916]
13 8 12 0 1 30.0010000									[2.7257818460444057, 0.6114685133291546]
13 8 12 0 2 0.0010000									[2.7270945500551527, 0.6073460320514418]
13 8 12 0 2 30.0010000									[2.728394989348543, 0.6032262469118151]
13 8 12 0 3 0.0010000			[3.3499638990221654, 0.3854352735613723]						[2.729683239548119, 0.5991091873825859]
13 8 12 0 3 30.0010000			[3.349890485774989, 0.3893519244181958]						[2.730959374743607, 0.5949948826829812]
13 8 12 0 4 0.0010000			[3.3498231289198404, 0.3932741979847059]						[2.732223467521178, 0.5908833617822565]
13 8 12 0 4 30.0010000			[3.3497618094959027, 0.39720207952891445]						[2.7334755889929663, 0.5867746534027819]
13 8 12 0 5 0.0010000			[3.349706508559498, 0.4011355542435031]						[2.734715808825822, 0.5826687860230527]
13 8 12 0 5 30.0010000			[3.3496572071828656, 0.4050746072441165]						[2.735944195269368, 0.5785657878807001]
13 8 12 0 6 0.0010000			[3.349613886452906, 0.40901922356767506]						[2.7371608151833433, 0.5744656869754263]
13 8 12 0 6 30.0010000			[3.349576527469898, 0.41296938817066836]						[2.7383657340642706, 0.5703685110719238]
13 8 12 0 7 0.0010000			[3.349545111346179, 0.416925085927475]						[2.7395590160714662, 0.566274287702752]
13 8 12 0 7 30.0010000			[3.3495196192047985, 0.42088630162866625]						[2.7407407240524027, 0.5621830441711745]
13 8 12 0 8 0.0010000			[3.349500032178138, 0.424853019979318]						[2.74191091956745, 0.5580948075539658]
13 8 12 0 8 30.0010000			[3.349486331406492, 0.42882522559733355]						[2.7430696629140137, 0.5540096047041884]
13 8 12 0 9 0.0010000			[3.349478498036628, 0.4328029030117548]						[2.7442170131500756, 0.549927462253933]
13 8 12 0 9 30.0010000			[3.3494765132202984, 0.43678603666109006]						[2.745353028117168, 0.545848406617031]
13 8 12 0 10 0.0010000			[3.3494803581127215, 0.440774610891639]						[2.7464777644627825, 0.5417724639917361]
13 8 12 0 10 30.0010000			[3.3494900138710326, 0.44476860995582074]						[2.747591277662249, 0.5376996603633828]
13 8 12 0 11 0.0010000			[3.349505461652691, 0.4487680180105107]						[2.7486936220400704, 0.5336300215070113]
13 8 12 0 11 30.0010000			[3.3495266826138477, 0.45277281911538053]						[2.7497848507907543, 0.5295635729899685]
13 8 12 0 12 0.0010000			[3.349553657907684, 0.4567829972312356]						[2.750865015999147, 0.5255003401744867]

Figure 13 Examples output CSV file of the azi_ele function

The CSV file output together with the azi_ele function is shown in Figure 12, with the first data in each cell representing the azimuth and the second data representing the elevation angle (It's all in arcs).

Note that both the CSV file and the charts are generated in the same path as the original RINEX data, which is the same for all subsequent quality check related functions.

```
epoch 13 8 12 0 0 30.0010000
G14 [2.7231193335204305, 0.6197214451251352]
G16 [4.381956806587844, 0.7518445094435549]
G29 [0.9795007528779629, 0.5891365465569922]
G31 [0.7771883872963753, 1.1030195802322884]
G32 [4.4978594096955, 0.3914340658522126]
```

Figure 15 Examples output Numpy array of the ION_MP function

At the same time, the output of this function is written to a list, where each element of the list is a dictionary as shown in Figure 15.

	G01	G02	G03	G04
EPOCH1	[0, 0, 0, 0]	[MP1, MP2, ION, IOD]	[MP1, MP2, ION, IOD]	[0, 0, 0, 0]
EPOCH2	[MP1, MP2, ION, IOD]	[MP1, MP2, ION, IOD]	[0, 0, 0, 0]	[MP1, MP2, ION, IOD]
EPOCH3	[0, 0, 0, 0]	[0, 0, 0, 0]	[0, 0, 0, 0]	[MP1, MP2, ION, IOD]
EPOCH4	[MP1, MP2, ION, IOD]	[MP1, MP2, ION, IOD]	[MP1, MP2, ION, IOD]	[MP1, MP2, ION, IOD]

Figure 15 Examples output Numpy array of the ION_MP function

For the two functions ION_MP and cycleslip, their most important feature is that they will output a Numpy array in addition to the CSV file and the graph, which can be convenient for the user to carry out subsequent calculations. The ordering of the output data in the third dimension of this three-dimensional array is shown in Figure 13 (The output of cycleslip has an index of 3 in the third dimension.).

The results of MP1 and MP2, as well as ION and IOD, are output to four different plots, but then written two by two to the same CSV file, where the order of precedence in each cell is the same as in Figure 15.

The cycleslip function is identical.

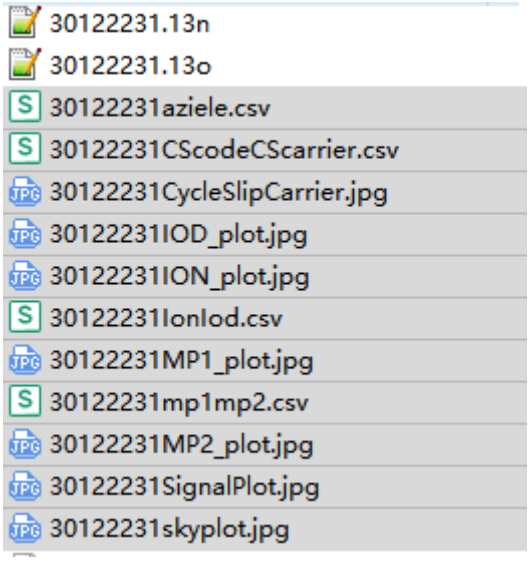


Figure 16 Example of output after using the QualityCheck function.

When using the QualityCheck function, it automatically runs all of the above quality check calculations at once and produces the result file shown in Figure 16 under the same path.

The operation mechanism of batchQC function is similar to that of DataFinding function, users only need to input keywords to batch process specific RINEX data under a folder.