GNSS Measurement Analysis | ED18B027 Report

Created	@Apr 16, 2021 9:46 PM
Status	
Tag	

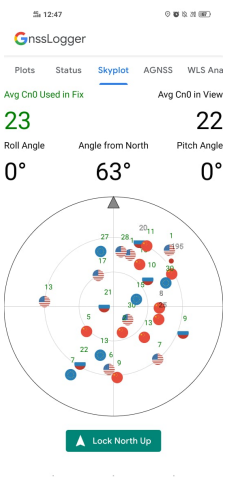
Task 1: Data Collection

Datalogs were collected for 10 minutes using the GNSSLogger Android App at three different locations.

Initially when I collected data using my phone, I was getting very bad results especially indoors. My phone was not able to maintain stable connection for even 2 GPS satellites indoors. I was only able to get results when I am very close to a window/door.

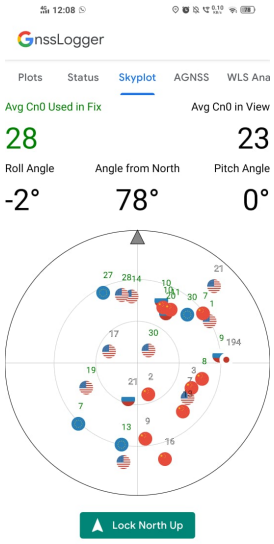
I was able to connect to much much more satelittes than the outdoor case, even for the partial-sky case when I tried with different phone (*newer model*).

Outdoor (Terrace)



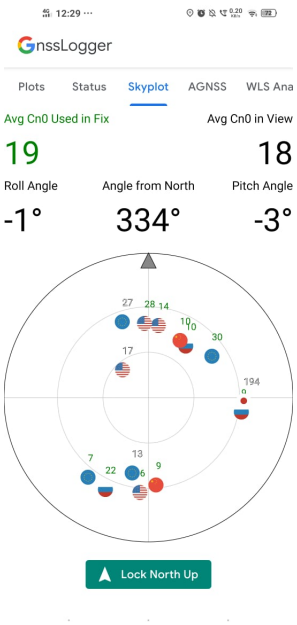
ID	GNSS	Freq	C/N0	Used	Azim	Elev
1		L1	17.2	Y	21	45
6		L1	30.5	Y	35	184
7		L1	18.4	Y	30	140
13		L1	19.6	Y	30	274
14		L1	20.7	Y	43	18
17		L1	19.3	Y	60	334
28		L1	23.8	Y	42	8
30		L1	19.6	Y	73	133
19		L1			56	277
10		G1	21.9	Y	52	52
22		G1	17.2	Y	32	210
9		G1	17.4	Y	25	111
20		G1	10.1	Y	30	22
21		G1	16.1	Y	82	268
2		B1I	19.6	Y	66	161
5		B1I	22.7		58	227
7		B1I	19.9	Y	32	61
8		B1I	12.9	Y	50	92
9		B1I	25.0	Y	28	177
10		B1I	15.8	Y	47	32
11		B1I	17.2	Y	30	29
13		B1I	18.3	Y	52	135
25		B1I	22.8		49	106
23		B1I			61	357

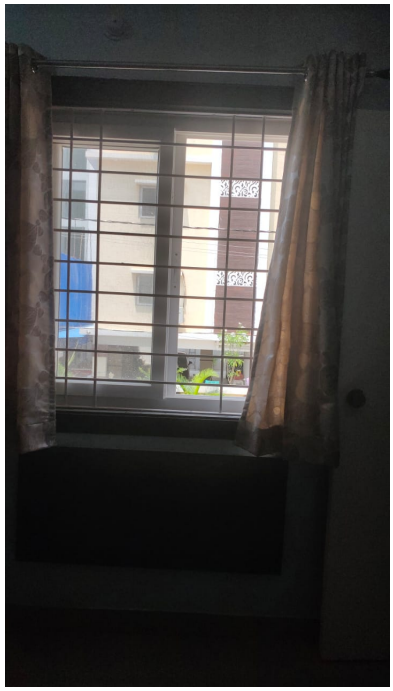
Partial-Sky (Balcony)



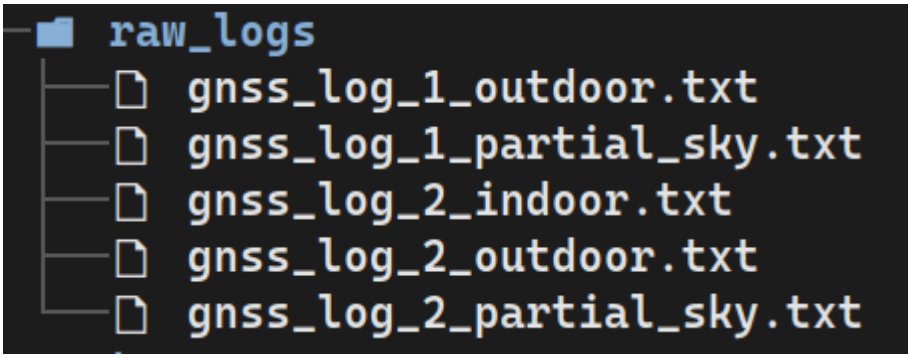
ID	GNSS	Freq	C/N0	Used	Azim	Elev
1		L1	18.9		30	60
6		L1	27.2	Y	18	188
7		L1	11.7		46	125
14		L1	37.7	Y	42	355
17		L1	16.0		68	292
19		L1	24.6	Y	49	244
21		L1	7.3		12	45
28		L1	41.0	Y	40	347
30		L1	27.8	Y	78	42
10		G1	28.8	Y	42	21
9		G1	31.5	Y	31	85
20		G1	23.5	Y	49	30
21		G1	13.4		63	194
2		B1I	16.2		66	161
3		B1I	18.2		47	115
5		B1I	16.6		58	227
7		B1I	18.6	Y	31	53
8		B1I	21.5		42	104
9		B1I	12.8		35	174
10		B1I	29.6	Y	46	25
11		B1I	34.8	Y	45	32
13		B1I	12.3		42	137
16		B1I	15.5		18	164
23		B1I			47	345

Indoor (Facing Window)





The logfiles are stored in the folder **raw_logs** with files having 1 corresponding to the first phone and 2 with the second phone.



Task 2: CDF

Each logfile was first parsed in "**parse.py**" using **AWK** in the ***Bash Shell*** to create temporary parsed files in ***./tmp*** which contains only the Fix and Status lines with only those columns required. (*parsing removes non-GPS and non-UsedInFix satellites*). These files are distroyed after running the program.

The code related to plotting CDF can be found in "**distribution_per_fix.py**". Each latitude and longitude value was extracted and mean value was found. A *numpy* implimentation of ***haversine*** was used which I found here. Radius of earth used here was **6371** km instead of that used in the link. The groundtruth

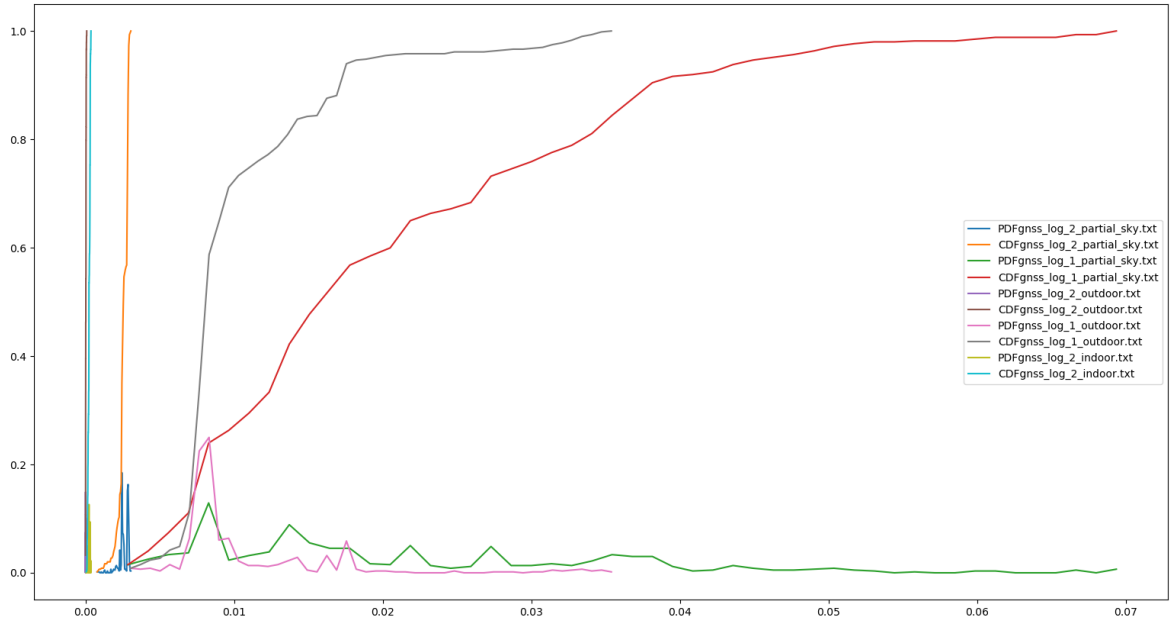
vector was taken as mean latitude and longitude repeated as many times as there were fixes.

The distance errors from the "groundtruth" was then divided into **50** bins and bundled together using **np.histogram** (*nearby errors were counted and put in a single bucket*). These error counts were then normalised so that the total summation is 1 (= *total propability*), which is also the PDF for the errors. Then, **np.cumsum** was used to calculate the CDF.

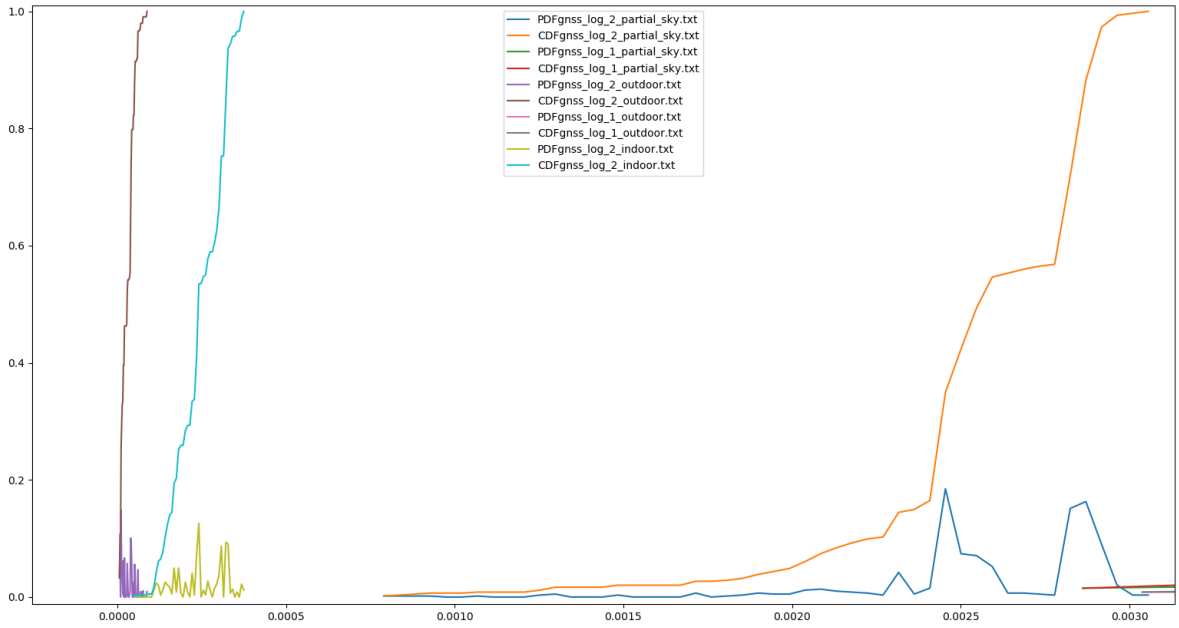
Observations

From the CDF one can look at its slope and decide its variance. Low variance would mean higher slope. It is expected that reading outside have less error than readings inside and also have less variance.

But the readings show something a bit different, the below graph is an overlay of all the CDF's:



- Firstly, errors due to the first phone is much larger than the second phone as expected and variance outdoor is lower than variance for partial case, also as expected. Readings indoor couldn't be recorded implies infinite error can be assumed.
- *Zoomed:*



- The errors and the variance for the outdoor case is lower than those of the indoor case just as expected. But the partial sky case, doesn't follow as expected. This maybe because, from the place where the readings were taken, there was an apartment right across and hence multipath readings might have been captured. So instead of a gaussian distribution there was a bi-modal distribution, one from direct line of sight for satellites in front of me and the other from reflection from the building for the satellites behind me. This can also be observed in the PDF of the partial case as shown above.

Task 3: Status corresponding to Fix

From observing the UnixTime for Statuses and Fixes it can be concluded that a Fix is written first, then corresponding statuses one for each GPS Satellite used.

Some corner cases were considered when counting fixes and satellites:

- **Missing Statuses**, implies 2 Fixes one after the other which leads to 0 satellites used for fix.
- **Missing Fixes**, implies 2 different Status groups one after the other which leads to large number of satelittes (*around 20*)

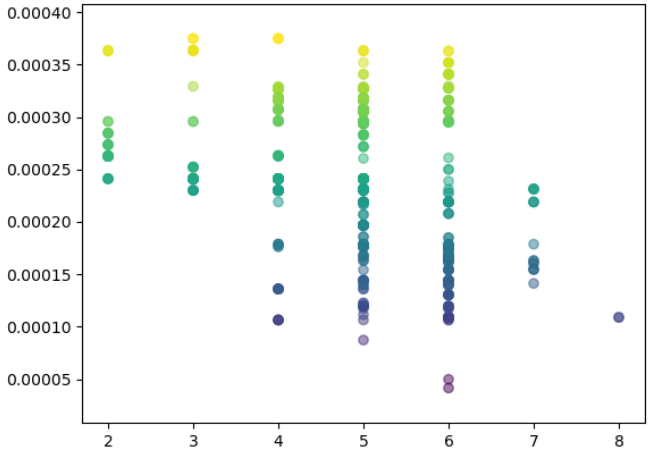
This is fixed by considering UnixTime of Fix. Sometimes though, 2 different Status groups have the same UnixTime, hence it is made sure that svid's don't

repeat again without a fix in between *(assuming that the second group has the same satellites given they occured instantly one after the other)*

Median number of satellites used was then calculated.

Distance Errors vs No. of Satellites used

A scatter plot was constructed correlating **distance errors** and **number of satellites used**. For a givn number of satellites, more the error more warmer the color.

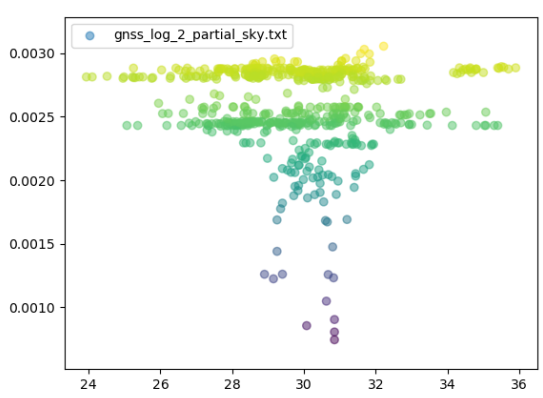


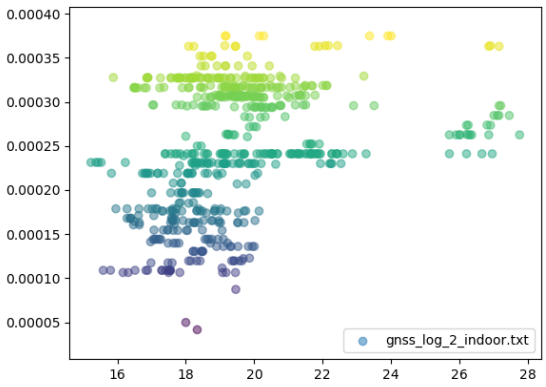
Observation

In general the graphs show that more the number of satellites less is the error. But, I feel that finding meaning from this graph would not tell us the full picture. Since, most of the time we are going to have a certain number of satellites hence statistically we would get more variance and readings only for some number of satellites. For the low satellite case it might just so happen that the satellites are distributed perfectly and you are in a perfect location that gives very less error and adding two more and removing one (net one increase) may disturb this symmetry. Similar but opposite argument for the high number of satellite case may result in high error (alligned satellites). Hence, only if many number of readings are considered for each case will we be able to make a proper judgement.

Distance Errors vs Average SNR per Fix

Average SNR was calculated similar to no.of satellites. A scatter plot correlating **distance errors** and **average SNR** was constructed. Similar to before more the error more warmer in the color.





Observations:

- Similar to before most of the time a certain SNR is recieved. But more can be deduced out of this plot.
- SNR does not have a major effect on the error. But we expect that more the SNR the lower is the error. This maybe because after a certain SNR the error in reading the message packet is almost negligible compared to the errors caused due to time drift and physical position and alignment of satellites. If SNR is very low then maybe it would start showing more errors. (*The application might even reject such readings*)
- The error distribution can be observed here as it is not dependent on SNR. For the case of partial-sky the bi-modal property of the error can be neatly observed. It is like looking at the PDF from a different perspective. If "looked" from the error axis it is like looking at the PDF of the errors.
- A look at the indoor case showes that SNR almost always never crosses a certain value which makes sense because it is a closed room and signal is weak. For low values of SNR the errors has a lot of variance as we can have more satellites with the same SNR giving a low error or less satellites giving more error. *(connectivity fluctuates more as we are inside)*
- The above 2 examples show that this can also be used as a fingerprint for a location!

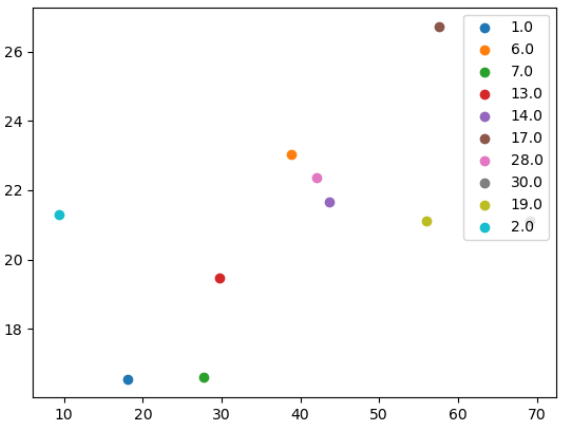
Task 4: Status corresponding to Satellite

This section of the code is written in "**distribution_per_satellite.py**". Each satellites information across all fixes is first collected in a dictionary using svid as key and numpy array as value. numpy array is used for easy computation of averages.

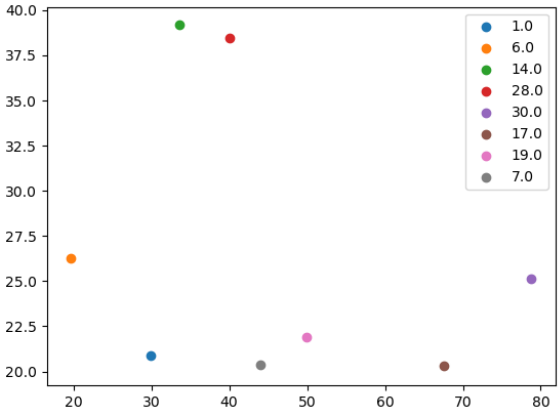
A table is drawn for each logfile as **stdout** to the terminal.

Average Elevation vs Average SNR

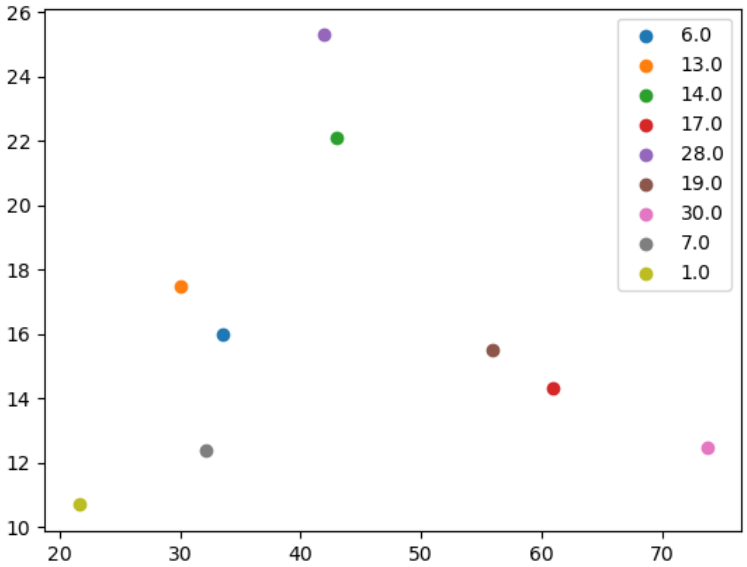
A scatter plot is constructed which shows different satellites in different colors.



Outdoor



Partial Sky



Indoor

Observation:

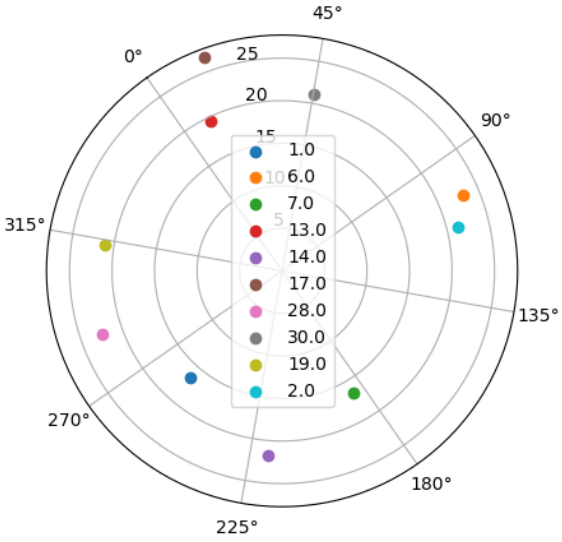
It can be observed that for the partial and indoor case the SNR is high at around 45 degrees and it make sense because:

- If elevation is too high, SNR goes down as there are 5 floors above me.

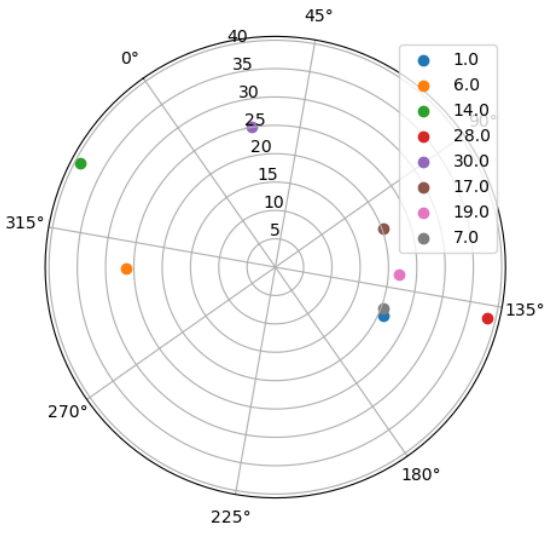
- If elevation is too low, SNR goes down as there would be many buildings in the line of sight.
- In the indoor case it is easier to penetrate 1 concrete wall than 5+ concrete walls.
- In the outdoor case the plot looks random and the SNR depends on other factors

Average Azimuth vs Average SNR

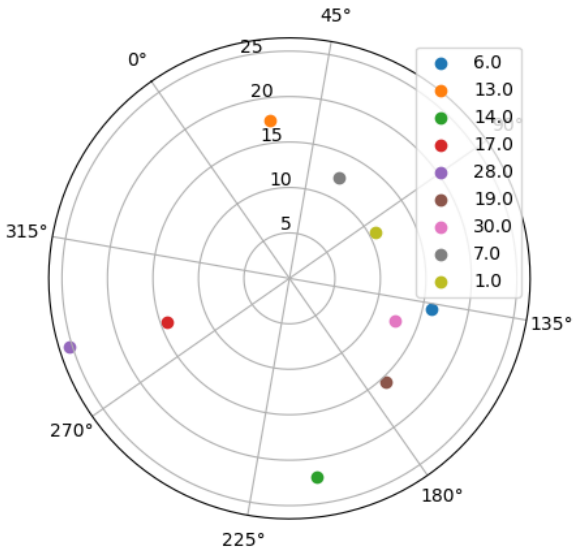
A polar scatter plot is constructed with each satellite represented by a different color. Since, clockwise rotation increases the value of theta for azimuth according to wikipedia, the theta direction is inverted. From inspection using various phone magnetometers (*initial reading only*) I have found the azimuth to be around 35 degrees. Hence, I rotated the graph by **35 degrees**. *(I tried using the groundtruth values in the website provided for finding azimuth but the values very too close to give an answer)*



Outdoor



Partial Sky



Indoor

Observation:

- For the outdoor case, the SNR seems to be not dependent on azimuth and only depends on the satellite.
- For the partial sky case, most of the readings seem to come from the upper half as expected.
- For the indoor case, the SNR is low for the top left quadrant and high for the rest as there is more open area on these sides.
- This said, I feel that of all the data the azimuth data seems to have more errors as they seem somewhat different from the "**skyplot**" shown in the app. *(Maybe they show fused values)*

Resources

- https://en.wikipedia.org/wiki/Azimuth
- Numpy Haversine: https://stackoverflow.com/questions/29545704/fast-haversine-approximation-python-pandas
- GNNSLogger App Android: https://play.google.com/store/apps/details?id=com.google.android.apps.location.gps.gnsslogger&hl=en_IN&gl=US
- GNSS API docs: https://developer.android.com/reference/android/location/GnssStatus