Samsara Intern Assignment

Austin Mac

December 15, 2019

Abstract

This project involving Samsara GPS data was conducted to analyze trends among defective GPS units. With this information, we provide recommendations to the Samsara team in order to deduce the root cause of defective units and ultimately reduce the number of defective units deployed. Fields in this data set include: date and time, input SN, total time, signal-to-noise ratio (SNR), detect OK time, GPS signal test PASS time, and time to first fix (TTFF). An unusually high TTFF was found in defective units, as well as a cluster of defective units found around April 3rd to April 4th. Based off of the analysis below, recommendations include: recalling a specific subset of gateway units, investigating weather conditions, and investigating possible user error.

Definitions

- Signal-to-noise ratio (SNR): Measure of signal strength in decibels (dB). Higher values indicate a higher SNR and a stronger signal.
- Time to first fix (TTFF): Amount of time in seconds required for a GPS to acquire enough signal to accurately provide navigation.

Data Processing

Given log files of GPS data for unique dates and times, the first task is to aggregate the data into a singular file for processing. This is done with the following bash command.

```
GPS Test Data austinmac$ cat LI5-1834871M/PASS/* LI5-1
834411M/PASS/* LI5-1843681M/PASS/* LI5-1843682M/PASS/*
LI5-1834230M/PASS/* LI5-1843685M/PASS/* > AggData.txt
```

Once all the logs are concatenated into one file, a Python script (see appendix) is used to parse the aggregate log file and format the data into a CSV for processing.

Analysis

First, a scatterplot matrix is created in order to observe any interesting correlations between fields, where defective units are coded red, and non-defective units are coded grey. Note, SNR3-SNR8 are ommitted from the matrix for the sake of parsimony.

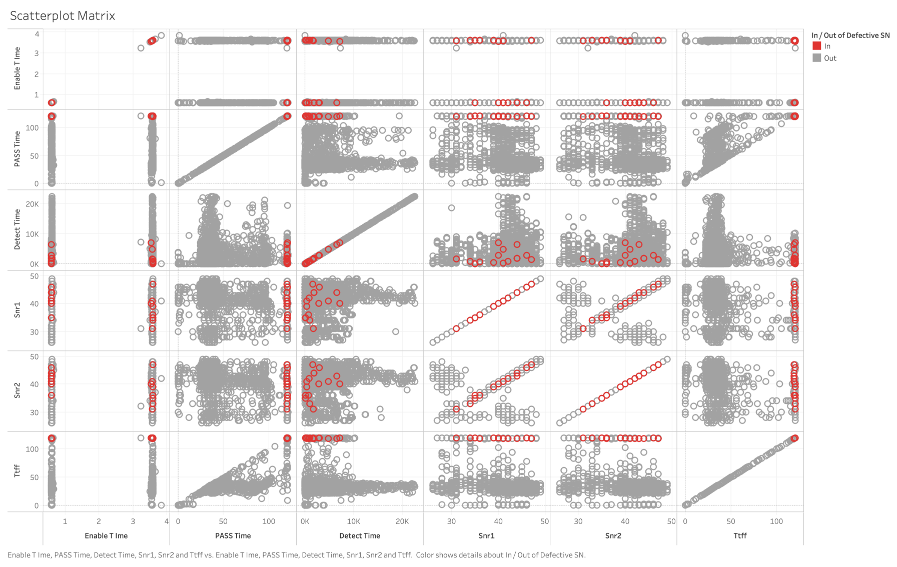

Figure 1: Scatterplot Matrix

From the plot, it appears all of the defective units have a TTFF near the maximum range of possible TTFF values. Additionally, they all exhibit maximal PASS times and minimal Detect times. Further observation reveals that the observations tend to have Enable times which cluster around ≈ 0.6 and ≈ 3.6. From this initial plot, the fields do not appear to have any obvious relationship to each other, besides the fact that PASS time is greater than or equal to TTFF. Further analysis illustrates a Pearson correlation coefficient of 0.740142333, showing that there exists a strong, positive, linear relationship between PASS time and TTFF. There do not appear to be any obvious differences in SNR between defective and non-defective units.

Since the relationship between defective units and TTFF appears to be of interest, side-by-side plots of TTFF against defective units and non-defective units are created, where red indicates defective units, and blue indicates non-defective units.

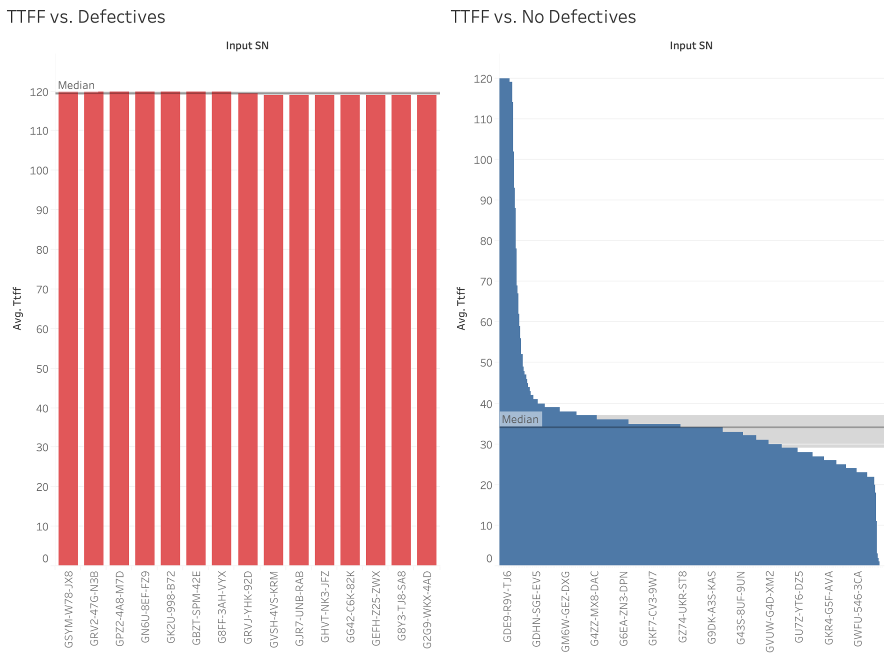

Figure 2: TTFF

The median average TTFF of defective units is 119.5 seconds, while the median for non-defective units is much smaller at 34 seconds. Since 120 seconds seems to be the maximum TTFF value for this data set, it appears that TTFF measurement caps at 120 seconds, suggesting that units with TTFF over 120 seconds may simply be logged as 120 seconds. Perhaps these defective units never fix in the first place. Observation of the plot of non-defective units illustrates a handful of units which have TTFF values of 120 seconds as well. This seems highly unusual, given that the majority of non-defective units have TTFF values roughly $\frac{1}{3}$ that amount.

Verifying that these unusual observations are outliers involves calculating externally studentized residuals from a fitted model, which is omitted from this report. However, we can observe the distribution of TTFF values.

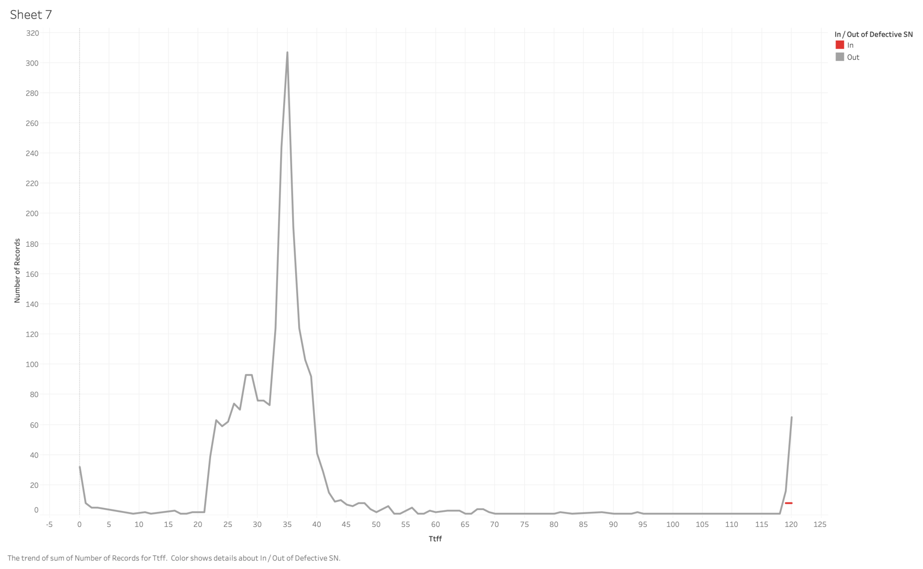

Figure 3: Distribution of TTFF

The TTFF values of defective units are high and are concentrated around 120. This is an area of interest, especially since these observations do not follow the trend of TTFF centered around 35 seconds.

We can also perform time analysis on the set, given date and time data for defective and non-defective units. Below is a plot of TTFF against date and time, where defective units are coded red and non-defective

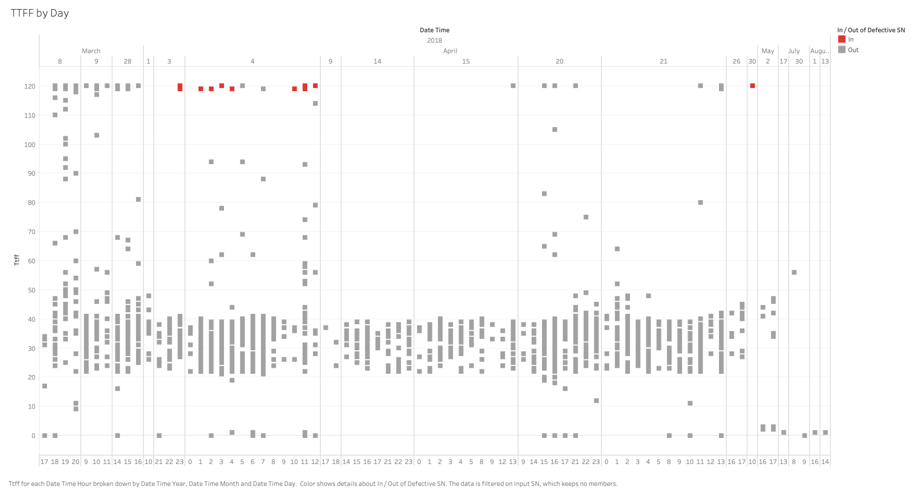

Figure 4: TTFF by Day

From the plot, it is evident that the majority of defective units were logged on April 3-4 between the hours of 23 and 12. This observation is quite intriguing since the defective units seem to be concentrated around this time (with the exception of one unit), hence another area of interest. Furthermore, high TTFF observations support the inference from Figure 2.

Recommendations

From analysis of the above plots, recommendations are as follows:

1. **Recall 95% of gateways with TTFF $\geq$ 120 which were tested between April 3, 2018 at 11:00pm and April 4, 2018 at 12:00pm**. Figures 2 and 3 illustrate anomalies in the TTFF values of gateway units. From the data trend, it appears TTFF values should be centered around 35 seconds; however, Figure 3 illustrates a small peak where a handful of gateway units appear to have unusually high TTFF values. We do not wish to recall 95% of all the gateway units which have high TTFF values, as this would create a logistical nightmare and cause unneccessary supply strain, so we investigate only the gateway units tested between April 3rd and April 4th. These dates are chosen because Figure 3 illustrates an unusual clustering of defective units around these dates and times. With further investigation of recalled units, we can perform appropriate hypothesis tests with more data and perform quality assurance checks with engineers to unearth any production/hardware errors.

2. **Investigate weather conditions and environmental factors after April 3-4.** The clustering of defective units around these dates is very strange and warrants further investigation. Although unlikely, the batch of gateway units tested during this date range may all have been deployed in the same geographic location afterward. Failure of these gateway units may be a result of environmental factors contributing to a weak signal. Contact customers who had faulty gateway units and obtain shipping data. Then analyze weather data on the appropriate shipping route and determine if unusual weather patterns contributed to failure of gateway units.

3. **Investigate user error in the batch of gateway units tested April 3-4.** A possible cause of defects includes user error. If the batch of gateways tested between April 3rd and April 4th was purchased by customer who configured the units incorrectly, it may explain the block of defective units. If possible, obtain any user complaint reports and determine how the user configured the unit. Else, send an engineer to ensure correct installation of gateways.

Appendix

```
raw = open("test.txt", "r")
GPSdata = open("GPSdata2.csv", "a")

headers = [
    "DateTime,",
    "Input SN,",
    "Detect OK Time,",
    "Enable GPS Module OK Time,",
    "TTFF,",
    "SNR1,",
    "SNR2,",
    "SNR3,",
    "SNR4,",
    "SNR5,",
    "SNR6,",
    "SNR7,",
    "SNR8,",
    "GPS Signal Test PASS Time,",
    "Total Time\n"
    ]
for header in headers:
    GPSdata.write(header)
SNRCount = 0
for line in raw.readlines():
    # DateTime
    if line[:4] == "2018":
        GPSdata.write(line[:19])
        GPSdata.write(",")

    # Input SN
    if line[20:28] == "Input SN":
        GPSdata.write(line[31:43])
        GPSdata.write(",")

    # Detect OK Time
    if line[:6] == "Detect":
        end = line.index("sec.")
        GPSdata.write(line[22:end - 1])
        GPSdata.write(",")

    # Enable GPS Module OK Time
    if line[:21] == "Enable GPS Module OK!":
        end = line.index("sec.")
        GPSdata.write(line[33:end - 1])
        GPSdata.write(",")

    # TTFF
    if line[:4] == "TTFF":
        end = line.index("(s)")
        GPSdata.write(line[7:end - 1])
        GPSdata.write(",")

    # SNR1
    if line[:4] == "SNR1":
        GPSdata.write(line[7:9])
        GPSdata.write(",")
        SNRCount += 1

    # SNR2
    if line[:4] == "SNR2":
        GPSdata.write(line[7:9])
        GPSdata.write(",")
        SNRCount += 1

    # SNR3
    if line[:4] == "SNR3":
        GPSdata.write(line[7:9])
        GPSdata.write(",")
        SNRCount += 1

    # SNR4
    if line[:4] == "SNR4":
        GPSdata.write(line[7:9])
        GPSdata.write(",")
        SNRCount += 1

    # SNR5
    if line[:4] == "SNR5":
        GPSdata.write(line[7:9])
        GPSdata.write(",")
        SNRCount += 1

    # SNR6
    if line[:4] == "SNR6":
        GPSdata.write(line[7:9])
        GPSdata.write(",")
        SNRCount += 1

    # SNR7
    if line[:4] == "SNR7":
        GPSdata.write(line[7:9])
        GPSdata.write(",")
        SNRCount += 1

    # SNR8
    if line[:4] == "SNR8":
        GPSdata.write(line[7:9])
        GPSdata.write(",")
        SNRCount += 1


    # GPS Signal Test PASS Time
    if line[:21] == "GPS Signal Test PASS!":
        for i in range(0,8-SNRCount):
            GPSdata.write(",")
        end = line.index("sec.")
        GPSdata.write(line[33:end - 1])
        GPSdata.write(",")
        SNRCount = 0

    # Total Time
    if line[:5] == "Total":
        GPSdata.write(line[11:])

raw.close()
GPSdata.close()
```