**Kinetic Signals Up Time– How To Guide**

**Gracen Alleman, Intern 7/25/2024**

The **Kinetic Signals Up Time** Power BI dashboard is used to sort and visualize data coming from the **Kinetic Signals alarm report**. It shows an *average percentage of* *down time* and it shows the *total duration of down time* from the inputted time period form **Kinetic Signals**. It has the ability to sort by *division* and *device number*.

* **Retrieving data:** Log into Kinetic signals using your NCID and go to Recent Alarms. Download for your preferred time frame and check to make sure that “active” and “acknowledged” are neutral.
* Once the data is updated you can load in the data into the **Kinetics Python file**.

>import csv

>with open("Alarm\_Report-2024-07-15\_15-47-14.csv", 'r', encoding='utf-8') as file:

>reader = csv.reader(file, delimiter=',', quotechar='"', quoting=csv.QUOTE\_MINIMAL)

>data = [row for row in reader]

This is where we enter into our first round of data cleaning. The way that the data is parsed by Pandas or other csv readers creates errors’, the data is not reading quoted commas correctly. This seems to be the case because not all text columns are quoted and there is a leading space on all fields. This document continues to work with the broken data, but I realize that if one could fix the parsing problems then data wrangling would not be needed and allow for fewer opportunities for errors.

The first round of data cleaning involves converting the data into a Pandas data frame, promoting the first row to be column headers, and renaming the headers.

* Convert the data to Pandas.

>df = pd.DataFrame(data)

* Promote the first row as headers and delete the first row.

>df.columns = df.iloc[0]

>df = df[1:]

* Rename the headers to delete the leading space and create names for the last four columns, which are named *col1* through *col4* respectively.

> df.columns = ['Received', 'Cleared', 'Severity', 'Name', 'Index', 'Device', 'Intersection', 'Group', 'Description', 'Active', 'Acknowledged', 'Comment', 'col1', 'col2', 'col3', 'col4']

The data has predictable quoted commas which is what is causing the column runoff. It gets run through a data wrangling function to check to see how far out that data runs off the page and sees if the data is the category of “time drift” or “Database Upload Failed”. Sometimes the data in columns, *devices*, and *intersections* also produce commas. The *description* column is not used nor the *intersection* column because it is a copy of Device. This data will be written over by data that might be used, but it is safe to do so because the original is not being used.

>for c, row in df.iterrows():

>if pd.notna(row['col2']):

>if row['Name']==" Time Drift":

>df.at[c, 'Group'] = df.at[c, 'Active']

>df.at[c,'Description'] =df.at[c,'col1']

>df.at[c, 'Active'] =df.at[c,'col2']

>df.at[c,'Acknowledged']=df.at[c,'col3']

>df.at[c,'Comment'] = df.at[c,'col4']

>box.append(df.loc[c]) …

The next step is data cleaning for the now wrangled data. The columns *Active* and *Acknowledged* are converted to Boolean, *col1* through *col4* are dropped, and time is converted to from local time GMT-2 to UTC time.

* *Active* and *Acknowledged* are mapped to true and false with their leading spaces, along with null values mapped to false, and then converted to Boolean.

>df1['Acknowledged'] = df1['Acknowledged'].map({' true':True,' false':False,' ':False})

>df1['Active'] = df1['Active'].map({' true':True,' false':False,' ':False})

* The temporary columns are dropped.

>df2=df1.drop(['col1','col2','col3','col4'],axis=1)

* The time columns *Received* and *Cleared* are converted to Date Time.

>df2['Cleared'] = pd.to\_datetime(df2['Cleared'], format= " %m/%d/%Y %I:%M:%S %p", errors='coerce')

>df2['Cleared'] = df2['Cleared'].dt.tz\_localize('Etc/GMT-2')

>df2['Cleared'] = df2['Cleared'].dt.tz\_convert('UTC')

The next step is data manipulation, where the *duration* column is added and columns indicating the *Division* and the *Device* number are also added.

* The *Duration* column is made.

>df2['Duration'] = df2['Cleared'] - df2['Received']

* The function *extract numbers* is called to sort out the division ID and the ID of the intersection.

>df2 = extract\_numbers(df2)

* The *extract numbers* function works by calling the function *extract number,* which is a re.search in an instance where it looks for two numbers followed by a dash and then four more numbers.

>if isinstance(text, str):

>match = re.search(r'(\d{2})-(\d{4})', text)

>if match:

>return match.group(1), match.group(0)

* The function *extract numbers* calls the extract number function looking at columns *Device* and *Intersection*. If it matches the division and device it gets added to their respective columns.

> df.loc[mask, 'Division'], df.loc[mask, 'Device#'] = zip(\*df.loc[mask, 'Intersection'].apply(lambda x: extract\_number(x)))

* If device name is not found in the columns *Intersection* or *Device* then the column *Group* is checked for just the division code.

>if isinstance(text, str):

>match = re.search(r'D(\d+)', text)

The main Alarm report can now be exported to csv, where it will be loaded into Power BI.

>df2.to\_csv('Alarm Report.csv', sep=',')

From the alarm report data frame, the *Device#, Duration* and *Name* columns were filtered into a new data frame. The duration of the comm failures is summed up per *Device* and made into a new data frame where *Device* is a unique column and *Total Durati*on is the other column. Data manipulation is done where the third and fourth columns are added: one for *division* and one for *percentage of total duration of comm failure*. Data is converted to Time Delta and the communication report is exported as a csv.

* Columns from *df2* are used to create a new data frame with columns *Device#, Duration*, and *Name*.

>if 'Device#' in row.index:

>tohist.append(df2.loc[c, ['Device#','Duration','Name']])

>tohist = pd.DataFrame(tohist)

* For every *Device* with “Comm Failure” the Duration is summed up into *total duration* and made into two columns of a new data frame.

>for name in tohist['Device#'].unique():

>total\_duration = tohist.loc[(tohist['Device#'] == name) & (tohist['Name'] == ' Comm Failure'), 'Duration'].sum()

>tohist1[name] = total\_duration

>tohist2 = pd.DataFrame(list(tohist1.items()), columns=['Device#', 'Total\_Duration'])

* A new column is added: *Division*, which uses a similar method as before but using a str extract function.

>tohist2['Division'] = tohist2['Device#'].str.extract(r'^(\d{2})')

* *Total duration* is converted to time delta.

>tohist2['Total\_Duration'] = pd.to\_timedelta(tohist2['Total\_Duration'], errors='coerce')

* Then a percentage of time is calculated using the function *calculate\_time\_difference* which calculates the total time in seconds of the total data frame. The original data frame *df2* is what this function calculations are done on. Then the total time in seconds from *calculate\_time\_difference* is used to multiply by 100 and then divides by the total duration row, to get a preset of the total time.

>tohist2['percentage\_of\_time'] = tohist2['Total\_Duration'].dt.total\_seconds() / calculate\_time\_difference(df2) \* 100

>df\_sorted = df.sort\_values(by=time\_column)

>first\_time = df\_sorted.iloc[0][time\_column]

>last\_time = df\_sorted.iloc[-1][time\_column]

>time\_difference = last\_time - first\_time

>total\_seconds = time\_difference.total\_seconds()

* Null values are taken out and the data frame is written as a csv.

>tohist2 = tohist2.drop(tohist2[tohist2['Division'].isna()].index)

>tohist2.to\_csv(‘Comm Failure’.csv', sep=',')

The data from Python can now be loaded into Power BI. Make sure that when you were writing csv your Python working directory is the same as your Power BI working directory. Then in Power BI refresh both data sets which should load in the new version of the data, as long as they share the same name. Alternatively, you could have Power BI on an automatic refresh as new versions get outputted from Python.

**Next steps for Kinetic signals up time dashboard**: creating a permanent pulling of data and refreshing of Power BI and adding more to Power BI to create a more complete story. Also sorting out the data loading error might be a good bug to fix on account of the bug’s size. I would also recommend setting up a storage system for the data and holding on to it in chunks of quarters of a year and then averaging the *total down time* over a defined time chunk of a quarter.