wrangle_bt

February 10, 2016

1 Import data

In [1]:
```
import pandas as pd
f = pd.read_csv('../data/BLUETH_20150826.filtered.BT', header=None, names=['Site', 'Unix Time',
f.head()
```

Out[1]:

	Site	Unix Time	Anonymized Bluetooth ID
0	2425	1440547215	893E907D22081264BAA5F9D43B94F81A
1	2425	1440547227	F3C7B29393C21E9A1E4C322452E29A44
2	2425	1440547246	893E907D22081264BAA5F9D43B94F81A
3	2425	1440547246	F3C7B29393C21E9A1E4C322452E29A44
4	2409	1440547246	9D7FE2EB2B16BB56F44B73C77CE1DC1E

2 Sort and Group

Collect together vehicles based on their bluetooth ID. Sort by time.

In [2]:
```
f_sorted = f.sort_values(by=['Anonymized Bluetooth ID', 'Unix Time'])
f_sorted[0:10]
```

Out[2]:

	Site	Unix Time	Anonymized Bluetooth ID
4264	2409	1440568449	0008DAF8E65F92CC7A8B0F1D8A755A92
4277	2409	1440568467	0008DAF8E65F92CC7A8B0F1D8A755A92
4287	2409	1440568487	0008DAF8E65F92CC7A8B0F1D8A755A92
15901	2409	1440578240	000B1865B7FAA931B56B92C344F6B56B
15931	2409	1440578265	000B1865B7FAA931B56B92C344F6B56B
15946	2409	1440578277	000B1865B7FAA931B56B92C344F6B56B
55924	2409	1440611032	000B1865B7FAA931B56B92C344F6B56B
57247	2425	1440612288	000B1865B7FAA931B56B92C344F6B56B
57334	2425	1440612369	000B1865B7FAA931B56B92C344F6B56B
57349	2425	1440612381	000B1865B7FAA931B56B92C344F6B56B

In [3]:
```
f_sorted.dtypes
```

Out[3]:
```
Site                        int64
Unix Time                   int64
Anonymized Bluetooth ID    object
dtype: object
```

In [4]:
```
loops = 0
for i in f_sorted.iterrows():
    print (i)
    loops += 1
    if loops >= 3:
        break
```

```
(4264, Site                                                  2409
Unix Time                                          1440568449
Anonymized Bluetooth ID    0008DAF8E65F92CC7A8B0F1D8A755A92
Name: 4264, dtype: object)
(4277, Site                                                  2409
Unix Time                                          1440568467
Anonymized Bluetooth ID    0008DAF8E65F92CC7A8B0F1D8A755A92
Name: 4277, dtype: object)
(4287, Site                                                  2409
Unix Time                                          1440568487
Anonymized Bluetooth ID    0008DAF8E65F92CC7A8B0F1D8A755A92
Name: 4287, dtype: object)
```

```
In [5]: f_groups = f_sorted.groupby(['Anonymized Bluetooth ID'])
        f_groups
```

```
Out[5]: <pandas.core.groupby.DataFrameGroupBy object at 0x7faab255b210>
```

```
In [6]: loops = 0
        for bt_id, data in f_groups:
            print (bt_id)
            print (data)
            loops += 1
            if loops >= 3:
                break
```

```
0008DAF8E65F92CC7A8B0F1D8A755A92
      Site   Unix Time           Anonymized Bluetooth ID
4264  2409  1440568449  0008DAF8E65F92CC7A8B0F1D8A755A92
4277  2409  1440568467  0008DAF8E65F92CC7A8B0F1D8A755A92
4287  2409  1440568487  0008DAF8E65F92CC7A8B0F1D8A755A92
000B1865B7FAA931B56B92C344F6B56B
       Site   Unix Time           Anonymized Bluetooth ID
15901  2409  1440578240  000B1865B7FAA931B56B92C344F6B56B
15931  2409  1440578265  000B1865B7FAA931B56B92C344F6B56B
15946  2409  1440578277  000B1865B7FAA931B56B92C344F6B56B
55924  2409  1440611032  000B1865B7FAA931B56B92C344F6B56B
57247  2425  1440612288  000B1865B7FAA931B56B92C344F6B56B
57334  2425  1440612369  000B1865B7FAA931B56B92C344F6B56B
57349  2425  1440612381  000B1865B7FAA931B56B92C344F6B56B
57369  2425  1440612398  000B1865B7FAA931B56B92C344F6B56B
57396  2425  1440612420  000B1865B7FAA931B56B92C344F6B56B
57430  2425  1440612444  000B1865B7FAA931B56B92C344F6B56B
57470  2425  1440612481  000B1865B7FAA931B56B92C344F6B56B
000B95CE5869E922AA49D5D45CBE3326
       Site   Unix Time           Anonymized Bluetooth ID
37538  2425  1440595677  000B95CE5869E922AA49D5D45CBE3326
37659  2425  1440595775  000B95CE5869E922AA49D5D45CBE3326
```

```
In [7]: sample_veh = f_groups.get_group('000B1865B7FAA931B56B92C344F6B56B')
        sample_veh
```

```
Out[7]:        Site   Unix Time           Anonymized Bluetooth ID
       15901  2409  1440578240  000B1865B7FAA931B56B92C344F6B56B
       15931  2409  1440578265  000B1865B7FAA931B56B92C344F6B56B
```

```
15946  2409  1440578277  000B1865B7FAA931B56B92C344F6B56B
55924  2409  1440611032  000B1865B7FAA931B56B92C344F6B56B
57247  2425  1440612288  000B1865B7FAA931B56B92C344F6B56B
57334  2425  1440612369  000B1865B7FAA931B56B92C344F6B56B
57349  2425  1440612381  000B1865B7FAA931B56B92C344F6B56B
57369  2425  1440612398  000B1865B7FAA931B56B92C344F6B56B
57396  2425  1440612420  000B1865B7FAA931B56B92C344F6B56B
57430  2425  1440612444  000B1865B7FAA931B56B92C344F6B56B
57470  2425  1440612481  000B1865B7FAA931B56B92C344F6B56B
```

3 Segments

Track travel time between sequentially visited sites

```
In [8]: def segments(df):
            """
            Convert ordered table of visited sites into segments between adjacent nodes.
            dataframe -- site, time, bluetooth_id
            """
            results = []
            last_row = None
            for index, row in df.iterrows():
                if last_row is not None and row["Site"] != last_row["Site"]:
                    segment = (last_row["Anonymized Bluetooth ID"],
                               last_row["Site"],
                               row["Site"],
                               last_row["Unix Time"],
                               row["Unix Time"])
                    results.append(segment)
                last_row = row
            return results

        segments(sample_veh)
```

```
Out[8]: [('000B1865B7FAA931B56B92C344F6B56B', 2409, 2425, 1440611032, 1440612288)]
```

```
In [9]: results = []
        for bt_id, data in f_groups:
            for segment in segments(data):
                results.append(segment)
        all_segments = pd.DataFrame(results,
                                    columns=('Anonymized Bluetooth ID', 'Site A', 'Site B', 'Time A', ''
```

```
In [10]: all_segments.head()
```

```
Out[10]:              Anonymized Bluetooth ID  Site A  Site B      Time A      Time B
          0  000B1865B7FAA931B56B92C344F6B56B    2409    2425  1440611032  1440612288
          1  001504BE590593C444A53BBF36BB5766    2425    2409  1440565264  1440565930
          2  004D460B07F5065FF49550C0BC617D80    2425    2409  1440564072  1440564657
          3  0059EC8C0CD1D799286B4D15B654FD1D    2425    2409  1440603784  1440604792
          4  0086FDBDB7E30152DAFE1368A2920A0B    2425    2409  1440570201  1440570776
```

4 Filter Direction

Consider only outbound/westbound traffic originating from site 2409, traveling to site 2425

In [11]:
```
inbound = all_segments[all_segments["Site A"] == 2409]
```

In [12]:
```
inbound = inbound.copy()
inbound.head()
```

Out[12]:

	Anonymized Bluetooth ID	Site A	Site B	Time A	Time B
0	000B1865B7FAA931B56B92C344F6B56B	2409	2425	1440611032	1440612288
8	009977651C772AC7A2235A46A2CE1E72	2409	2425	1440598995	1440599881
9	00A8D3A8D3193707A47512E332A04B8E	2409	2425	1440549802	1440550407
11	00AA2C5E7B0FC69467FCA798179BC41F	2409	2425	1440587741	1440588539
13	00E5C469BA14439C1591B02D64F9AE9E	2409	2425	1440606426	1440608448

5 Cacluate Travel Time

In [13]:
```
travel_time = inbound["Time B"] - inbound["Time A"]
inbound["Travel Time"] = travel_time
```

In [14]:
```
inbound.head()
```

Out[14]:

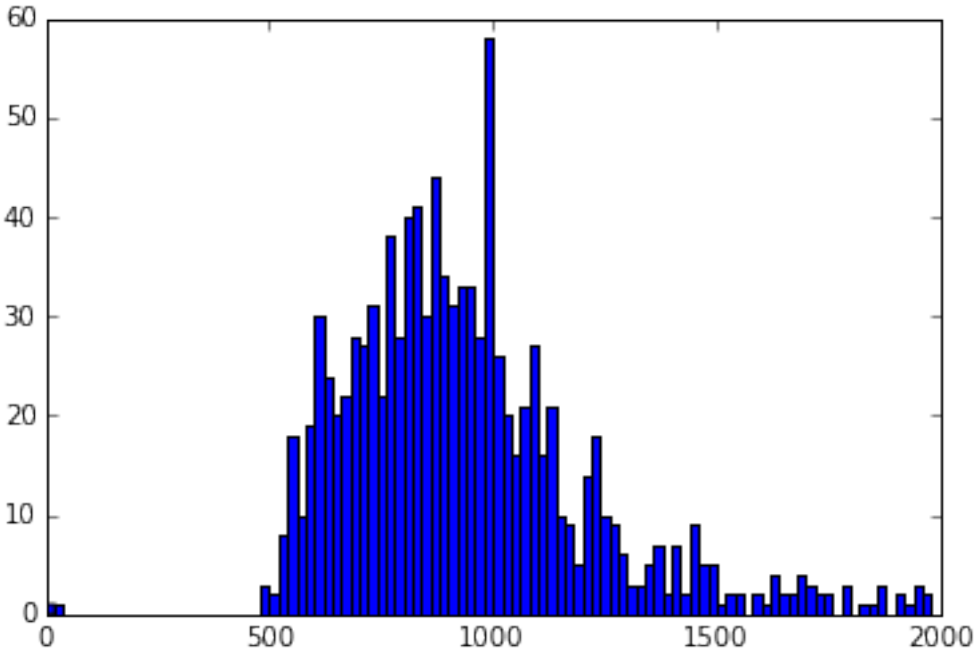
	Anonymized Bluetooth ID	Site A	Site B	Time A	Time B	\
0	000B1865B7FAA931B56B92C344F6B56B	2409	2425	1440611032	1440612288	
8	009977651C772AC7A2235A46A2CE1E72	2409	2425	1440598995	1440599881	
9	00A8D3A8D3193707A47512E332A04B8E	2409	2425	1440549802	1440550407	
11	00AA2C5E7B0FC69467FCA798179BC41F	2409	2425	1440587741	1440588539	
13	00E5C469BA14439C1591B02D64F9AE9E	2409	2425	1440606426	1440608448	

	Travel Time
0	1256
8	886
9	605
11	798
13	2022

6 Plot Vehicle Travel Times

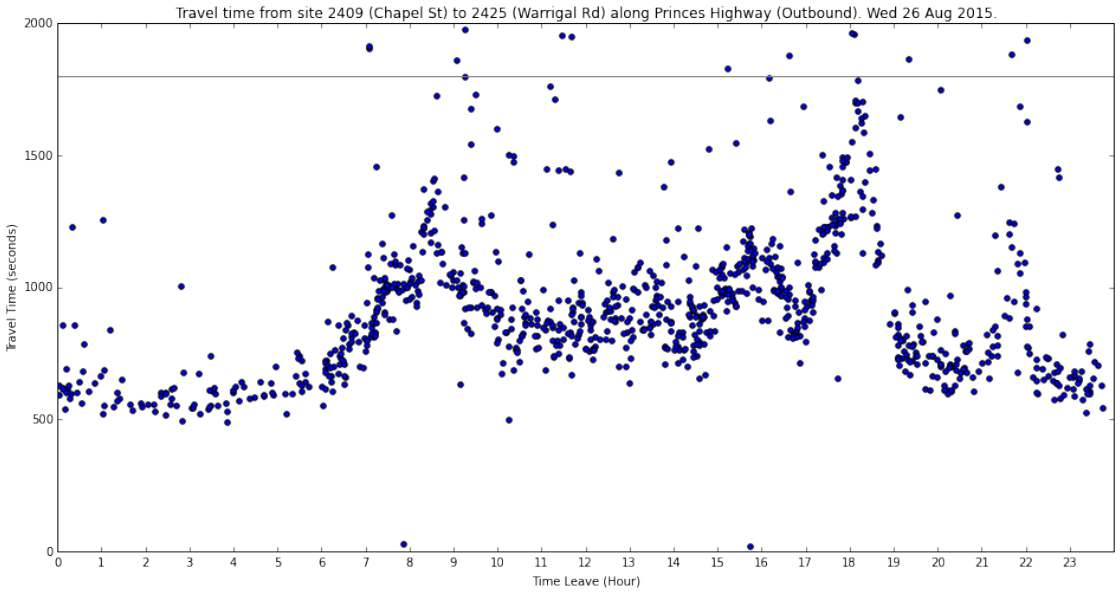
In [15]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

tt = list(travel_time)
bins = np.linspace(0, 2000, 101)
plt.hist(tt, bins=bins)
plt.show()
```




```
In [16]: import calendar
         plt.figure(figsize=(16,8))
         start_of_day = calendar.timegm((2015,8,26,0,0,0))
         plt.scatter(list((inbound["Time A"] - start_of_day)/3600), list(inbound["Travel Time"]))
         plt.title("Travel time from site 2409 (Chapel St) to 2425 (Warrigal Rd) along Princes Highway
         plt.ylabel("Travel Time (seconds)")
         plt.xlabel("Time Leave (Hour)")
         plt.xticks(np.arange(24))
         plt.xlim([0,24])
         plt.ylim([0,2000])
         plt.axhline(y=1800, color='grey') # threshold
         plt.show()
```

```
/home/asimmons/anaconda3/envs/python2/lib/python2.7/site-packages/matplotlib/collections.py:590: FutureW
  if self._edgecolors == str('face'):
```



7 Aggregate

Aggregate into 15 minute bins. Some vehicles stop along the way, take a longer route, or only pass the other site on the way back from their desitination. We eliminate these by only considering reasonable travel times, then taking the median value.

```
In [17]: # Filter extreme travel times
         inbound = inbound[inbound["Travel Time"] <= 1800]
```

```
In [18]: max(inbound["Travel Time"])
```

```
Out[18]: 1795
```

Experiment with timezones. Bluetooth times appear to be taken from the Unix Epoch, but need to be reinterpreted as AEST rather than GMT. (My guess is that the VicRoads clock was set by someone who didn't understand that the Unix clock should be set to UTC)

```
In [19]: #http://stackoverflow.com/questions/5956638/converting-python-datetime-to-timestamp-and-back-i
         import pytz
         import datetime
         min_time = min(inbound["Time A"])
         print min_time
         d = datetime.datetime.fromtimestamp(min_time, tz=pytz.utc)
         print d.isoformat()
         d2 = datetime.datetime.utcfromtimestamp(min_time)
         print d2.tzinfo
         d2 = pytz.timezone('Australia/Melbourne').localize(d2)

         #d2.tzinfo = pytz.utc
         print d2.isoformat()
```

```
1440547335
2015-08-26T00:02:15+00:00
None
2015-08-26T00:02:15+10:00
```

```
In [20]: import pytz
         import datetime

         # Times should be in GMT+0, but VicRoads has changed their clock to be in AEST
         tz = pytz.timezone('Australia/Melbourne')
         def parse_date(unix_time):
             d_utc = datetime.datetime.utcfromtimestamp(unix_time)
             # d_utc has no tzinfo. Re-interpret time as AEST.
             d_wrong = pytz.timezone('Australia/Melbourne').localize(d_utc)
             # Vicroads doesn't know what they're doing. So we need to copy their error.
             return d_wrong

         ts = pd.Series(list(inbound["Travel Time"]),
                        index=list([parse_date(t) for t in inbound["Time A"]]))
```

```
In [21]: ts.sort_index().tail()
```

```
Out[21]: 2015-08-26 23:31:56+10:00    652
         2015-08-26 23:33:30+10:00    718
         2015-08-26 23:39:34+10:00    704
         2015-08-26 23:44:23+10:00    628
         2015-08-26 23:45:24+10:00    542
         dtype: int64
```

```
In [22]: ts_resampled = ts.resample('15Min', how='median')

         # Index over entire day, even if some times are missing. Last 15 minutes usualy not present.
         rng = pd.date_range('2015-08-26 00:00:00+10:00', periods=24*4, freq='15Min')
         ts_resampled = pd.Series(ts_resampled, index=rng)

         # Fill in missing values
         ts_resampled = ts_resampled.fillna(method='pad')
```

```
In [23]: ts_resampled.tail()
```

```
Out[23]: 2015-08-26 22:45:00+10:00    626
         2015-08-26 23:00:00+10:00    653
         2015-08-26 23:15:00+10:00    609
         2015-08-26 23:30:00+10:00    678
         2015-08-26 23:45:00+10:00    542
         Freq: 15T, dtype: float64
```

8 Plot Aggregated Travel Times

```
In [24]: plt.figure(figsize=(16,8))
         plt.scatter(np.arange(len(ts_resampled)), ts_resampled.values)
         plt.title("Travel time from site 2409 (Chapel St) to 2425 (Warrigal Rd) along Princes Highway
         plt.ylabel("Travel Time (seconds)")
         plt.xlabel("Time Leave (15 min offset)")
         plt.xlim([0,95])
```

```
plt.ylim([0,2000])
plt.show()
```