# Cleaning GPS data collected from Catoctin park hikers

In this porject I intend to clean a dataset collected between July 31st and August 20th from 121 GPS divices that were distributed among the hikers in Catoktin Mountain Park, MD. During this period, every hiker was given a GPS divice in the beginning of her hike at the visiting center and was asked to return the divice at the end of her hike, at the same location (i.e. visitors' center). The data collected from these GPS is not consistent as many GPS holders have occasionally turned off the divice or have drived long distances while holding these divices. Since we are looking into their hiking behaviors we are only interested in hiking trajectories and therefor will need to exclude undesired trajectories. Since the hikers have paused at the visitors' center for relatively long time, we also would like to exclude the data collected during that time since these data points significantly affect the average speed and time spent. Therefore we are looking at the following steps:

- Remove data points located at the visiting center
- Remove data points collected while driving by filtering high speed path segments
- Identify spots where the hiker turns of his/her diviceRemove path segments where the time interval or distance between two consecutive points is larger than one minute and 180 meters, respectively.
- Remove tracks with less than 50 points (shorter than roughly 10 minutes hikes)
- Remove extraneous points. These points include those points where more than one point has been collected from a certain location at a certain time
- Remove points that are not in the Catoctin Mountain Park Area.

# Accessing the data stored in shapefiles

In [1]:
```
# open folders
import os
path = "C:/Users/sur216/Desktop/GPS Tracks_ALL"
folders = os.listdir (path)[1:]
dbf_path = []
for n,f in enumerate(folders):
        if not "GPS " in f: continue
        n_path = path + "/"+f
        print  f
        for j in os.listdir(n_path):
            if "dbf" in j:dbf_path.append(str(n_path + "/" +j))
```

```
GPS - Tracks 8-3-16
GPS - Tracks 8-4-2016
GPS Tracks - 7-31-16
GPS Tracks - 8-1-16
GPS Tracks - 8-10-2016
GPS Tracks - 8-11-2016
GPS Tracks - 8-13-2016
GPS Tracks - 8-17-2016
GPS Tracks - 8-19-2016
GPS Tracks - 8-20-2017
GPS Tracks - 8-5-2016
GPS Tracks - 8-6-2016
GPS Tracks - 8-7-2016
GPS Tracks - 8-9-2016
GPS Tracks- 8-14-2016
GPS Tracks- 8-16-2016
```

We will next read through dbf files and record everything in python lists. As shown below, this data includes 22 columns.

In [2]:
```
# open shapefiles and save the data into python lists
from dbfread import DBF
import pandas as pd
records = []
for n, i in enumerate(dbf_path):
        idr = i.split("/")[-1][:-4]
        if not "pt" in idr:continue
        for record in DBF(i):
            record.update({"id":idr})
            records.append(record)
columns = []
for n,r in enumerate(records):
    if n==1:
        for j in r: columns.append(j)
print columns
```

```
[u'type', u'tident', u'ident', u'Latitude', u'Longitude', u'y_proj', u'x_pro
j', u'comment', u'new_trk', u'new_seg', u'display', u'color', u'altitude',
 u'depth', u'temp', u'time', u'model', u'filename', u'ltime', u'desc', u'lin
k', 'id']
```

In [3]:
```
# convert the lists to Pandas DataFrame. We have 22 columns and 128930 rows
recs = []
for n,r in enumerate(records):
        lst = []
        for c in columns :
            lst.append(r[c])
        recs.append(lst)
df = pd.DataFrame(recs)
df.columns = columns
print "table size :",df.shape
```

```
table size : (128930, 22)
```

In [4]:
```
print len(df["id"].unique())
```

```
42
```

# Creating new ids

As we can see this data includes 42 unique GPS ids, however, recall that these GPS divices have been assigned to different people at different dates. Therefore, we will need to create a new ID that considers both the date and the GPS divice. We will call this "new_id". We can see that we have 146 unique IDs now.

In [5]:
```
# create new id for each GPS track based on data and file name
new_id = ["/".join(["-".join(list(i.split(" ")[0].split("/")[1:]))]+[j]) for
i,j in zip(df['time'].tolist(),df['id'].tolist())]
```

In [6]:
```
df['new_id'] = new_id
```

In [7]:
```
print len(df["new_id"].unique())
```

```
146
```

# Excluding the points nearby the visitor center

As discussed earlier, we wanted to exclude the points collected from the visitor center. To this end, I difined a 46X63 meters rectangle around the visitor center and excluded every point that was inside this square.

In [8]:
```
# identifying those points that fall near the visitors' center
df2 = df[(df['Latitude']<39.63435507) & (df['Latitude']>39.63378507) & (df['L
ongitude']<-77.44983899) & (df['Longitude']>-77.45038231)]
```

In [9]:
```
#df3 is the sets of points without those located at the visiting center
df3 = pd.merge(df, df2, how='outer', indicator=True)
df3 = df3[df3["_merge"]=="left_only"]
df3 = df3.iloc[:,0:23]
```

# Calculating speed and distance

In the next step we will calculate the speed between every two consecutive points. This step is important as this will provide the basis for filtering the points. The distance between every two points has been calculated by latitude, longitude and altitude. Also, we will calculate and store the change in elevation, planar distance and time difference in separate columns.

In [10]:
```
#convert to timestamp format
df3["time"] = pd.to_datetime(df3["time"])
```

In [11]:
```
# calculate distances, time steps and speeddf4
from geopy.distance import vincenty
import math
import datetime as dt
import time
tracks = df3['new_id'].unique()
df4 = pd.DataFrame()
for n,t in enumerate(tracks):
    q = df3[df3["new_id"]==t]
    q = q.sort(['time'], ascending=True)
    time = q["time"].tolist()
    lons = q["Longitude"].tolist()
    lats = q["Latitude"].tolist()
    alts = q["altitude"].tolist()
    lat_lng =zip(lats,lons)
    dist_horiz = ["NA"]
    dist_vert = ["NA"]
    tot_dist = ["NA"]
    time_dif = ["NA"]
    speed = ["NA"]
    for n,p in enumerate(lat_lng):
        if not n==0:
            y = vincenty(lat_lng[n], lat_lng[n-1]).meters
            z = alts[n]-alts[n-1]
            t = time[n]-time[n-1]
            t = t.total_seconds()
            d = math.sqrt(y**2+z**2)
            dist_horiz.append(y)
            dist_vert.append(z)
            tot_dist.append(d)
            time_dif.append(t)
            if not t == 0:
                speed.append(d/t)
            else: speed.append("NA")
    q['dist_horiz'] = dist_horiz
    q['dist_vert'] = dist_vert
    q['time_dif'] = time_dif
    q['tot_dist'] = tot_dist
    q['speed'] = speed
    df4 = df4.append(q)
```

```
C:\Users\sur216\Anaconda2\lib\site-packages\ipykernel\__main__.py:10: FutureW
arning: sort(columns=....) is deprecated, use sort_values(by=.....)
```

In [222]:
```
df4.to_csv("GPS_compiled.csv", index = False)
```

As we can see below, the time interval between concecutive points is not steady. Although the majority of points have been collected within 10 seceonds intervals (104699 points). We can also see that some data points has been collected with 0 seconds intervals. If we print these points out we can see that the distance between thes epoints are also 0. Therefore we will simply remove these from our dataframe.

In [12]:
```
ls = [int(i) for i in df4[df4["tot_dist"].isin(["NA"])==False]['time_dif'].un
ique()]
ls2 = [int(i) for i in df4[df4["tot_dist"].isin(["NA"])==False]['time_dif']]
ls.sort()
from collections import defaultdict
fq= defaultdict( int )
for t in ls2:
    fq[t] += 1
print fq
```

```
defaultdict(<type 'int'>, {0: 3554, 1: 20, 2: 14, 3: 11, 4: 5, 5: 3, 6: 3, 7:
3, 8: 5, 9: 4944, 10: 104699, 11: 4961, 12: 11, 3085: 1, 14: 2, 15: 6, 16: 2,
17: 3, 530: 1, 19: 1, 20: 40, 21: 8, 22: 4, 23: 2, 24: 1, 29: 3, 30: 22, 31:
 1, 32: 2, 10785: 1, 18: 1, 550: 1, 40: 14, 42: 2, 13319: 1, 44: 1, 7726: 1,
 47: 1, 49: 1, 50: 11, 51: 2, 54: 1, 570: 1, 60: 15, 16959: 1, 64: 1, 65: 1,
 68: 1, 69: 1, 70: 11, 1614: 1, 13: 2, 80: 4, 2132: 1, 100: 2, 90: 4, 9231:
 1, 94: 1, 9828: 1, 619: 1, 108: 1, 109: 1, 110: 3, 111: 1, 119: 1, 9336: 1,
 122: 1, 127: 2, 130: 2, 107: 1, 132: 1, 12808: 1, 4716: 1, 140: 1, 14479: 1,
660: 1, 150: 3, 1178: 1, 160: 2, 679: 1, 20649: 1, 170: 2, 1195: 1, 3954: 1,
 180: 1, 187: 1, 189: 1, 1733: 1, 200: 1, 34: 1, 2256: 1, 120: 1, 2774: 1, 89
20: 1, 220: 2, 7391: 1, 229: 1, 3821: 1, 7037: 1, 16113: 1, 7412: 1, 250: 1,
 1790: 1, 261: 1, 1801: 1, 1290: 1, 14722: 1, 270: 3, 795: 1, 5916: 1, 797:
 1, 293: 1, 10538: 1, 300: 2, 3380: 1, 3893: 1, 311: 1, 314: 1, 318: 1, 1464
5: 1, 860: 1, 370: 1, 15736: 1, 893: 1, 2946: 1, 390: 1, 400: 1, 17498: 1, 41
2: 1, 2117: 1, 1955: 1, 420: 1, 9640: 1, 23985: 1, 13748: 1, 13754: 1, 4540:
 1, 1469: 1, 963: 1, 460: 1, 2467: 1, 78: 1, 8271: 1, 8673: 1, 2021: 1, 1328
7: 1, 490: 1, 491: 1, 16366: 1, 4591: 1, 9713: 1, 500: 2})
```

In [13]:
```
ls = [int(i) for i in df4[df4["tot_dist"].isin(["NA"])==False]['tot_dist'].un
ique()]
ls2 = [int(i) for i in df4[df4["tot_dist"].isin(["NA"])==False]['tot_dist']]
ls.sort()
from collections import defaultdict
fq= defaultdict( int )
for t in ls2:
    fq[t] += 1
print fq
```

```
defaultdict(<type 'int'>, {0: 23982, 1: 9085, 2: 5784, 3: 5469, 4: 5469, 5: 5
388, 6: 5867, 7: 6115, 8: 6704, 9: 6997, 10: 7103, 11: 6757, 12: 5697, 13: 45
46, 14: 3377, 15: 2302, 16: 1562, 17: 1097, 18: 716, 19: 500, 20: 430, 21: 32
2, 22: 286, 23: 206, 24: 174, 25: 150, 26: 113, 27: 117, 28: 120, 29: 78, 30:
70, 31: 62, 32: 70, 33: 39, 34: 47, 35: 60, 36: 28, 37: 34, 38: 38, 39: 33, 4
0: 27, 41: 31, 42: 33, 43: 27, 44: 37, 45: 26, 46: 25, 47: 25, 48: 15, 49: 3
3, 50: 22, 51: 29, 52: 23, 53: 18, 54: 14, 55: 20, 56: 14, 57: 15, 58: 14, 5
9: 13, 60: 10, 61: 16, 62: 16, 63: 13, 64: 12, 65: 21, 66: 17, 67: 16, 68: 9,
69: 9, 70: 9, 71: 11, 72: 11, 73: 13, 74: 8, 75: 9, 76: 1, 77: 8, 78: 4, 79:
 9, 80: 4, 81: 7, 82: 7, 83: 9, 84: 5, 85: 8, 86: 3, 87: 7, 88: 5, 89: 9, 90:
8, 91: 8, 92: 8, 93: 8, 94: 11, 95: 15, 96: 8, 97: 11, 98: 8, 99: 8, 100: 13,
101: 8, 102: 9, 103: 9, 104: 9, 105: 9, 106: 9, 107: 8, 108: 8, 109: 3, 110:
 11, 111: 11, 112: 12, 113: 7, 114: 6, 115: 8, 116: 6, 117: 4, 118: 6, 119:
 3, 120: 7, 121: 11, 122: 9, 123: 7, 124: 9, 125: 8, 126: 9, 127: 6, 128: 5,
 129: 12, 130: 5, 131: 5, 132: 15, 133: 11, 134: 7, 135: 5, 136: 6, 137: 5, 1
38: 7, 139: 6, 140: 9, 141: 10, 142: 11, 143: 11, 144: 10, 145: 11, 146: 5, 1
47: 9, 148: 7, 149: 6, 150: 3, 151: 9, 152: 3, 153: 7, 154: 8, 155: 5, 156:
 9, 157: 5, 158: 4, 159: 6, 160: 12, 161: 8, 162: 5, 163: 3, 164: 7, 165: 5,
 166: 1, 167: 5, 168: 6, 169: 3, 170: 6, 171: 6, 172: 7, 173: 5, 174: 3, 175:
7, 176: 4, 177: 2, 178: 8, 179: 3, 180: 4, 181: 1, 182: 3, 183: 5, 184: 4, 18
5: 2, 186: 3, 187: 4, 188: 1, 190: 1, 191: 1, 192: 2, 705: 1, 195: 2, 197: 1,
200: 2, 201: 1, 202: 3, 203: 1, 204: 1, 205: 2, 206: 1, 207: 1, 208: 1, 209:
 1, 210: 1, 213: 2, 215: 1, 217: 1, 218: 1, 222: 1, 223: 1, 225: 1, 227: 3, 2
29: 1, 234: 1, 245: 1, 349: 1, 767: 1, 256: 1, 3843: 1, 262: 1, 271: 1, 1498:
1, 306: 1, 3385: 1, 1674: 1, 3390: 1, 3400: 1, 3408: 1, 3409: 1, 193: 1, 341
8: 1, 3420: 1, 3421: 1, 3422: 1, 3424: 1, 3432: 1, 361: 1, 3436: 1, 3645: 1,
 3441: 1, 3442: 1, 3439: 1, 374: 1, 3453: 1, 3466: 1, 3467: 1, 3389: 1, 1092:
1, 411: 1, 3486: 1, 3488: 1, 3491: 1, 3497: 2, 3498: 1, 3500: 1, 3506: 1, 344
3: 1, 1974: 1, 3511: 1, 3515: 1, 3518: 1, 672: 1, 1558: 1, 673: 1, 3540: 1, 3
546: 1, 5098: 1, 668: 1, 1524: 1, 505: 1, 529: 1, 426: 1, 510: 1})
```

In [14]:
```
#remove points with 0 time difference
df4 = df4[df4['time_dif'] !=0]
```

In [15]:
```
df4.shape
```

Out[15]: (115105, 28)

# filtering according to speed, time and total distance

We will now remove those tracks with less than 50 data points (less than 10 minutes hiking). After this step we will end up with 113 GPS tracks. We will also delet those points that show speeds higher than 3 m/s, the time difference between every two points is higher than 60 seconds and the distnce taken between tow points is higher than 180 meters.

In [16]:
```
# walked for at least 10 mins
lll = []
df5 = pd.DataFrame()
for n, t in enumerate(tracks):
    q = df4[df4["new_id"]==t]
    if not q.shape[0]<50: lll.append(q)
for d in lll:
    if not d.empty: df5 = pd.concat([df5,d], ignore_index=True)
```

In [17]:
```
df5 = df5[df5['speed']<3]
df5 = df5[df5['time_dif']<60]
df5 = df5[df5['tot_dist']<180]
df5.shape
```

Out[17]: (112809, 28)

In [18]:
```
len(df5['new_id'].unique())
```

Out[18]: 113

In [24]:
```
df5.to_csv("GPS_quasicleaned.csv", index = False)
```

# Using Arcmap for filtering

In the next step, we will use the arcmap software to reove the data points that are not close to the Catoctin park area. Occasionally, the GPS holders have drived to a nearby town or other areas without turning their GPS divices off. We are not interested in these data points and our cleaning process so far has not filtered these. To this end, we will use the Arcmap 10.4 software and manually remove these points. After this step, we will convert the DBF data to Pandas Dataframe:

In [19]:
```
from dbfread import DBF
import pandas as pd
records = []
dbf_path = "C:/Users/sur216/Desktop/GPS Tracks_ALL/all_compiled_projected.db
f"
for record in DBF(dbf_path):
            records.append(record)
print records[1]
```

```
OrderedDict([(u'type', u'TRACK'), (u'tident', u'03-AUG-16-05-1'), (u'ident',
 u'T1'), (u'Latitude', 39.63843433), (u'Longitude', -77.44265922), (u'y_pro
j', 39.63843433), (u'x_proj', -77.44265922), (u'comment', u''), (u'new_trk',
 u''), (u'new_seg', u''), (u'display', u''), (u'color', u''), (u'altitude', 4
36.58), (u'depth', 0.0), (u'temp', 0.0), (u'time', datetime.date(2016, 8,
 3)), (u'model', u''), (u'filename', u''), (u'ltime', u''), (u'desc_', u''),
 (u'link', u''), (u'id', u'05-1pt'), (u'new_id', u'08-03/05-1pt'), (u'dist_ho
riz', 6.88132158169), (u'dist_vert', 5.29), (u'time_dif', 11.0), (u'tot_dis
t', 8.67967088723), (u'speed', 0.789060989748)])
```

In [107]:
```
# save to csv after arcmap edits
columns = []
for n,r in enumerate(records):
    if n==1:
        for j in r: columns.append(j)
print columns
recs = []
for n,r in enumerate(records):
        lst = []
        for c in columns :
            lst.append(r[c])
        recs.append(lst)
df = pd.DataFrame(recs)
df.columns = columns
print "table size :",df.shape
```

```
[u'type', u'tident', u'ident', u'Latitude', u'Longitude', u'y_proj', u'x_pro
j', u'comment', u'new_trk', u'new_seg', u'display', u'color', u'altitude',
 u'depth', u'temp', u'time', u'model', u'filename', u'ltime', u'desc_', u'lin
k', u'id', u'new_id', u'dist_horiz', u'dist_vert', u'time_dif', u'tot_dist',
 u'speed']
table size : (111748, 28)
```

# Aggregating the data

In the next step we will aggregate the values that we are interested in. We would like to have the average speed, total distance and total time spent. We will also add the amount of vertical distance climbed up and down separately.

In [108]:
```
sm = df.groupby(['new_id']).sum()
mn = df.groupby(['new_id']).mean()
mn_spd = mn.loc[:,['speed']]
mn_spd['new_id']= mn_spd.index
mn_vert = df.loc[:,['new_id','dist_vert']]
clm_up = mn_vert[mn_vert['dist_vert']>0]
clm_up = clm_up.groupby(['new_id']).sum()
clm_dwn = mn_vert[mn_vert['dist_vert']<0]
clm_dwn = clm_dwn.groupby(['new_id']).sum()
clm_dwn['climb_down'] = clm_dwn['dist_vert']
clm_up['climb_up'] = clm_up['dist_vert']
del clm_up['dist_vert']
del clm_dwn['dist_vert']
clm_up['new_id'] = clm_up.index
clm_dwn['new_id'] = clm_dwn.index
mn_vert['dist_vert'] = mn_vert['dist_vert'].abs()
mn_vert = mn_vert.groupby(['new_id']).mean()
mn_vert['mean_elevation'] = mn_vert['dist_vert']
mn_vert =  mn_vert.loc[:,['mean_elevation']]
```

In [109]:
```
up_dwn = clm_up.merge(clm_dwn)
sm = sm.loc[:,['dist_horiz','time_dif','tot_dist']]
sm['new_id']= sm.index
df1 = up_dwn.merge(sm, on = 'new_id')
df2 = mn_spd.merge(df1, on= 'new_id' )
df2.columns = ['average_speed','new_id','climb_up','climb_down','Total_planar
_distance','Total_time','Total_distance']
```

In [111]:
```
df2 = df2.reindex_axis(['new_id','average_speed','climb_up','climb_down','Tot
al_planar_distance','Total_time','Total_distance'], axis=1)
print df2.head(10)
```

```
          new_id  average_speed  climb_up  climb_down  Total_planar_distance
\
0  07-31/02-1_pt       0.587001   1059.33    -1047.80           12975.771458

1  07-31/03-1_pt       0.813220    697.47     -680.64            7780.278455

2  07-31/06-1_pt       0.784326    445.62     -450.91            5254.870909

3  07-31/07-1_pt       0.763597    502.31     -413.39            4151.602668

4  07-31/08-1_pt       0.877671    474.88     -447.01            6094.750247

5  07-31/08-2_pt       0.307855    712.23     -661.28            5064.454156

6  07-31/09-1_pt       0.865830   1105.10    -1053.19           12523.573990

7  07-31/10-2_pt       0.824399    298.44     -274.88            4166.763060

8  07-31/11-1_pt       0.792242    723.90     -780.14            8188.272212

9  07-31/13-1_pt       0.764995    667.61     -800.27            8598.491833


   Total_time  Total_distance
0     22810.0    13374.081864
1      9910.0     8056.900198
2      6910.0     5416.952194
3      5730.0     4383.761234
4      7070.0     6221.389793
5     17791.0     5473.632868
6     14890.0    12910.774230
7      5171.0     4261.997310
8     10720.0     8489.355263
9     11610.0     8878.036376
```

In [106]:
```
df2.to_csv("GPS_aggregated.csv", index = False)
```