velo_correction

February 26, 2018

1 A. Etude statistique

1.1 1. Collecte

```
In [1]: import pandas as pa

In [2]: T=pa.read_csv('/Users/fredericbro/Documents/Lycee(henrimoissan)/Prof (henri-moissan)/Ann

In [3]: T.head()
```

Out[3]:

	Lat	Long	Month	Day	Year	Hour	Min	Sec	Altitude (m)	\
0	43.268042	-0.389441	7	14	2008	13	5	35	1260.739990	
1	43.267662	-0.389503	7	14	2008	13	5	39	1261.036011	
2	43.267255	-0.389569	7	14	2008	13	5	43	1261.286987	
3	43.266802	-0.389627	7	14	2008	13	5	47	1260.990967	
4	43.266185	-0.389709	7	14	2008	13	5	52	1260.671997	

	GPS
0	(43.268042, -0.389441)
1	(43.267662, -0.389503)
2	(43.267255, -0.389569)
3	(43.266802, -0.389627)
4	(43.266185, -0.389709)

1.2 2. Distance parcourue

1.2.1 a) Construction de la colonne des distances parcourues au fur et à mesure

```
In [4]: import geopy.distance as gd

In [5]: T['Distance']=pa.Series()

In [6]: T.loc[0,'Distance']=0

In [7]: for k in range(1,len(T)):
            T.loc[k,'Distance']=gd.distance(T.loc[k-1,'GPS'],T.loc[k,'GPS']).kilometers
```

1.2.2 b) Construction de la colonne des distances cumulées

```
In [8]: T['Distance_Cum']=T['Distance'].cumsum()

In [9]: T.head()
```

Out[9]:

	Lat	Long	Month	Day	Year	Hour	Min	Sec	Altitude (m)	\
0	43.268042	-0.389441	7	14	2008	13	5	35	1260.739990	
1	43.267662	-0.389503	7	14	2008	13	5	39	1261.036011	
2	43.267255	-0.389569	7	14	2008	13	5	43	1261.286987	
3	43.266802	-0.389627	7	14	2008	13	5	47	1260.990967	
4	43.266185	-0.389709	7	14	2008	13	5	52	1260.671997	

	GPS	Distance	Distance_Cum
0	(43.268042, -0.389441)	0.000000	0.000000
1	(43.267662, -0.389503)	0.042516	0.042516
2	(43.267255, -0.389569)	0.045533	0.088049
3	(43.266802, -0.389627)	0.050547	0.138597
4	(43.266185, -0.389709)	0.068870	0.207467

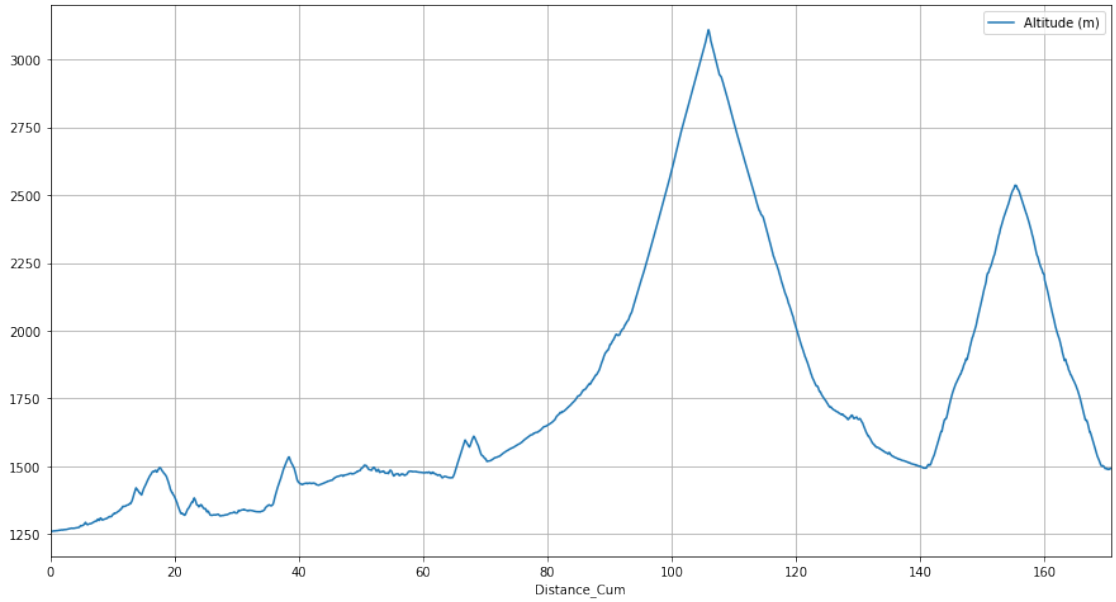
1.2.3 c) Diagramme Altitude en fonction de Distance cumulée

```
In [10]: T.plot(x='Distance_Cum',y='Altitude (m)',grid=True,figsize=(15,8))

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x115e9e6a0>

In [11]: import pylab as pl

In [12]: pl.show()
```



1.3 3. Temps

1.3.1 a) Colonne du temps exprimé en heure

```
In [13]: T['Temps']=T['Hour']+T['Min']/60+T['Sec']/3600
```

1.3.2 b) Liste des temps entre deux enregistrements

```
In [14]: L=[0]+[T.ix[k+1,'Temps']-T.ix[k,'Temps'] for k in range(4549)]
```

```
/Users/fredericbro/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: DeprecationWar
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  """Entry point for launching an IPython kernel.
```

```
In [15]: T['Delta_time']=pa.Series(L)
```

```
In [16]: T.head()
```

```
Out[16]:          Lat      Long  Month  Day  Year  Hour  Min  Sec  Altitude (m)  \
         0  43.268042 -0.389441      7   14  2008    13    5   35   1260.739990
         1  43.267662 -0.389503      7   14  2008    13    5   39   1261.036011
         2  43.267255 -0.389569      7   14  2008    13    5   43   1261.286987
         3  43.266802 -0.389627      7   14  2008    13    5   47   1260.990967
         4  43.266185 -0.389709      7   14  2008    13    5   52   1260.671997

                          GPS  Distance  Distance_Cum      Temps  Delta_time
         0  (43.268042, -0.389441)  0.000000      0.000000  13.093056    0.000000
         1  (43.267662, -0.389503)  0.042516      0.042516  13.094167    0.001111
         2  (43.267255, -0.389569)  0.045533      0.088049  13.095278    0.001111
         3  (43.266802, -0.389627)  0.050547      0.138597  13.096389    0.001111
         4  (43.266185, -0.389709)  0.068870      0.207467  13.097778    0.001389
```

1.4 4. Calcul de la vitesse

1.4.1 a) Vitesse entre 2 enregistrements

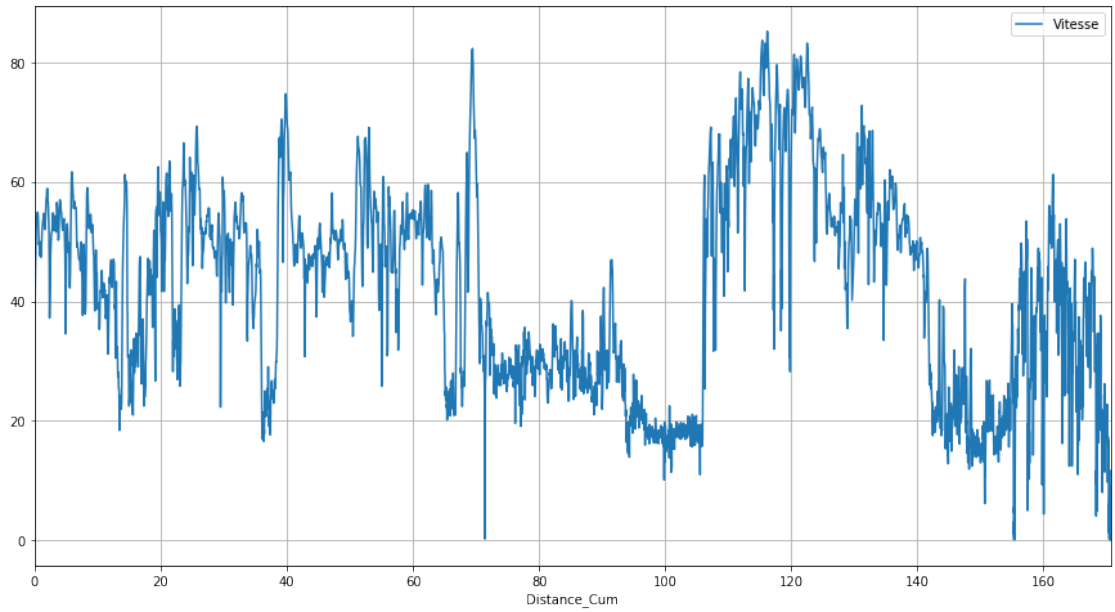
```
In [17]: T['Vitesse']=pa.Series([0]+[(T.ix[k,'Distance'])/(T.ix[k,'Delta_time']) for k in range(
```

1.4.2 b) Diagramme vitesse en fonction de distance cumulée

```
In [18]: T.plot(x='Distance_Cum',y='Vitesse',grid=True,figsize=(15,8))
```

```
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1191822e8>
```

```
In [19]: pl.show()
```



1.5 5. Pente

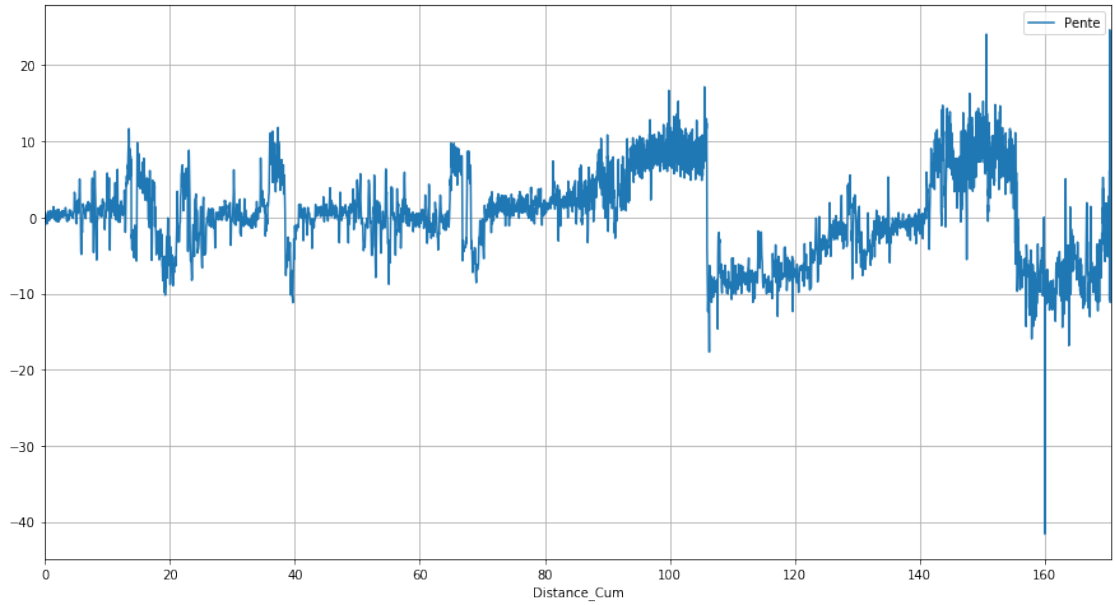
```
In [20]: L=[0]+[(T.ix[k+1,'Altitude (m)']-T.ix[k,'Altitude (m)'])/T.ix[k+1,'Distance']*0.1 for k
```

```
In [21]: T['Pente']=pa.Series(L)
```

```
In [22]: T.plot(x='Distance_Cum',y='Pente',grid=True,figsize=(15,8))
```

```
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x119311b00>
```

```
In [23]: pl.show()
```

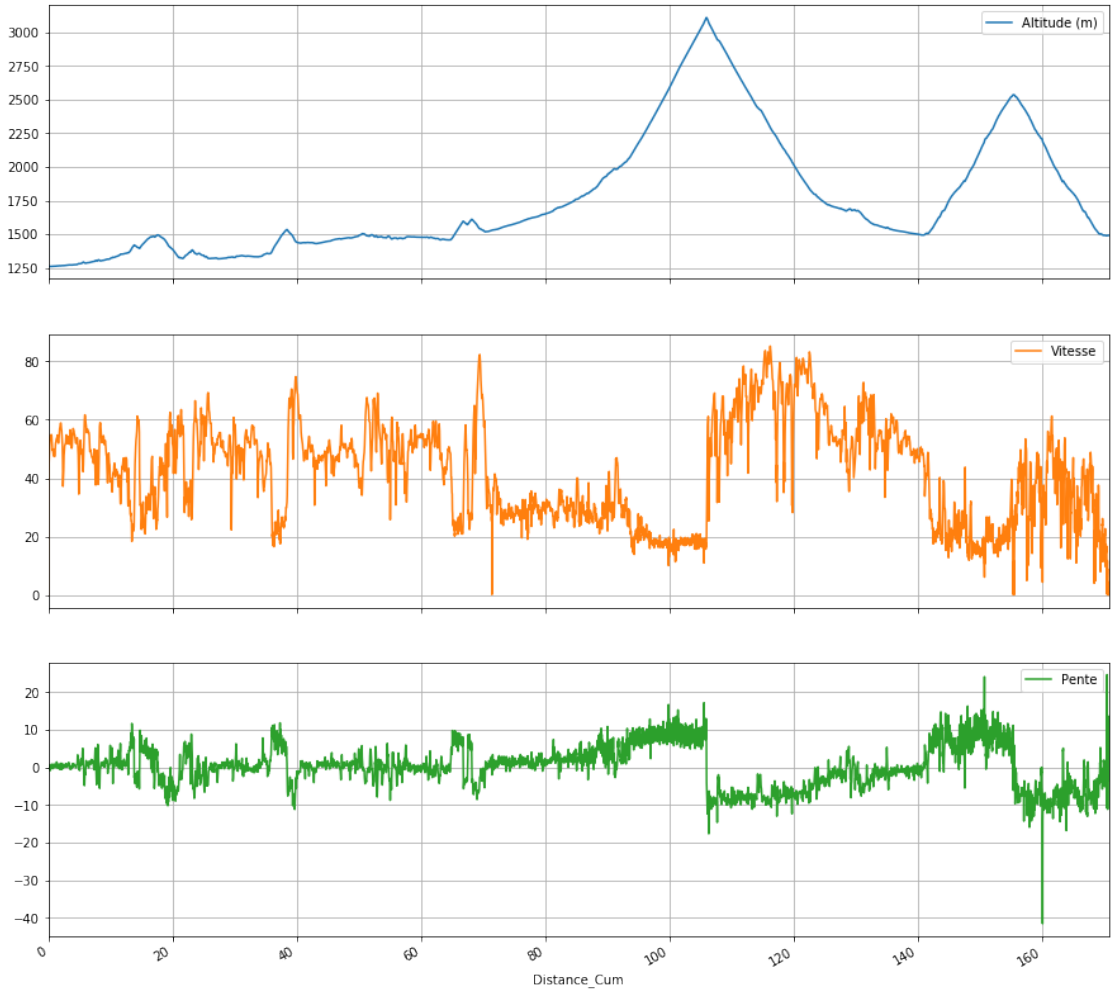


1.6 graphes superposés

```
In [24]: T.plot(x='Distance_Cum',y=['Altitude (m)','Vitesse','Pente'],subplots=True,grid=True,fi
```

```
Out[24]: array([<matplotlib.axes._subplots.AxesSubplot object at 0x118f34a58>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x1195ab438>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x1192a72b0>], dtype=object)
```

```
In [25]: pl.show()
```



1.7 6. Résumés statistiques

```
In [26]: T1=T[['Altitude (m)','Vitesse','Pente']]
```

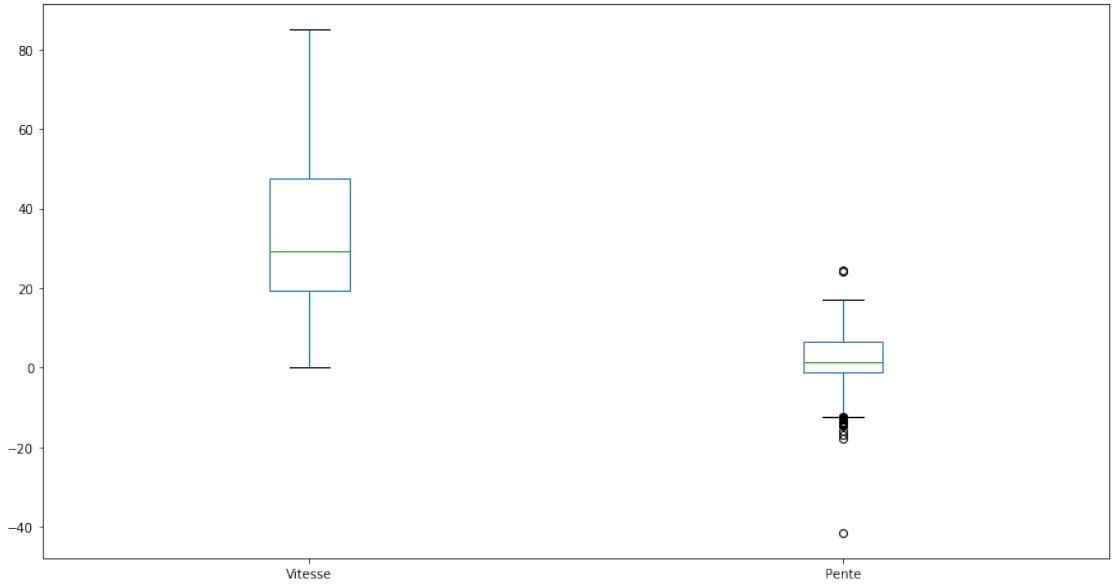
```
In [27]: T1.describe()
```

```
Out[27]:        Altitude (m)      Vitesse        Pente
         count   4550.000000  4549.000000  4550.000000
         mean    1833.591110    33.166030     1.790231
         std      469.866663    17.565651     5.686708
         min     1260.061035     0.000000   -41.573634
         25%     1481.927216    19.273823    -1.011493
         50%     1647.755982    29.210893     1.348505
         75%     2159.779175    47.724124     6.542558
         max     3109.891113    85.222119    24.639696
```

```
In [28]: T1[['Vitesse','Pente']].plot(kind='box',figsize=(15,8))
```

```
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x11993c860>
```

```
In [29]: pl.show()
```

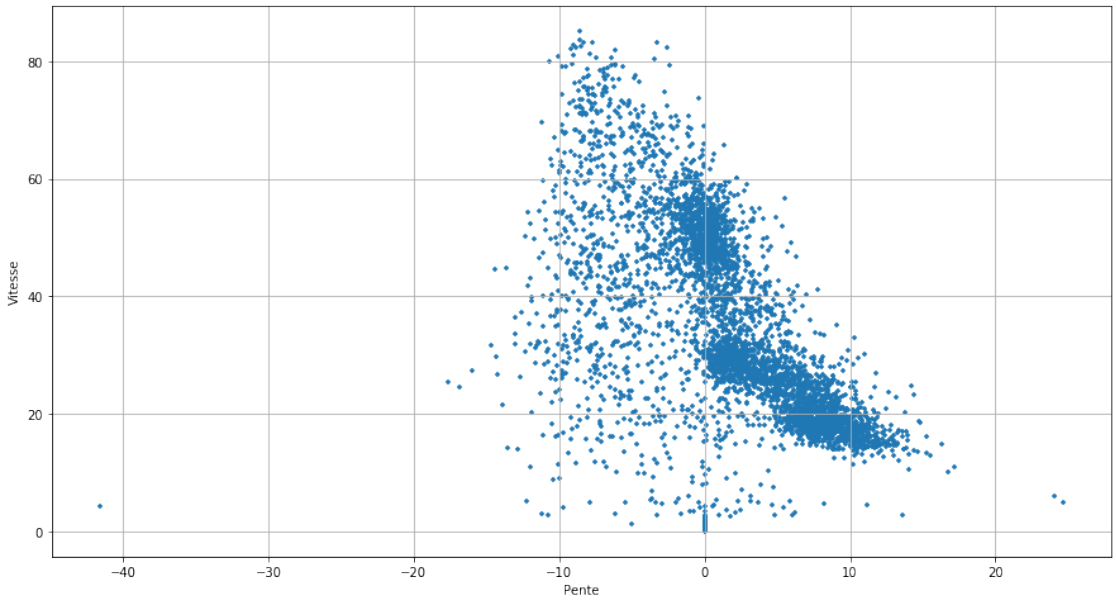


1.8 7. Vitesse en fonction de la pente

```
In [30]: T.plot(kind='scatter',x='Pente',y='Vitesse',marker='+',grid=True,figsize=(15,8))
```

```
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x11f49fb00>
```

```
In [31]: pl.show()
```



2 B. Représentation carte

```
In [32]: import folium
         from folium import plugins

In [33]: moy_Lat=T['Lat'].mean()
         moy_Long=T['Long'].mean()
         moy_Lat,moy_Long

Out[33]: (43.03225092373612, -0.021162988791208634)

In [34]: carte = folium.Map(location=[moy_Lat,moy_Long],tiles='Stamen Terrain', zoom_start=13)

In [35]: folium.Marker(location=[T.ix[0,'Lat'],T.ix[0,'Long']],
                       popup='Départ',
                       icon=folium.Icon(color='green',icon='cloud')).add_to(carte)

Out[35]: <folium.map.Marker at 0x11f6b7438>

In [36]: folium.Marker(location=[T.ix[4549,'Lat'],T.ix[4549,'Long']],
                       popup='Arrivée',
                       icon=folium.Icon(color='blue',icon='cloud')).add_to(carte)

Out[36]: <folium.map.Marker at 0x11f6c6f98>

In [37]: Loc=[(T.ix[k,'Lat'],T.ix[k,'Long']) for k in range(4550)]

In [38]: folium.PolyLine(Loc,color='red',weight=2, opacity=1).add_to(carte)
```

```
Out[38]: <folium.features.PolyLine at 0x11f6b7eb8>

In [39]: carte

Out[39]: <folium.folium.Map at 0x11f6abef0>

In [40]: #carte.save('C:/Lycee(henrimoissan)/Prof (henri-moissan)/Annee 2017/philippe/Activite/V
```

2.0.1 3. Altitude

```
In [41]: T1=T[T['Altitude (m)']==max(T['Altitude (m)'])]

In [42]: T1

Out[42]:            Lat      Long  Month  Day  Year  Hour  Min  Sec  Altitude (m)  \
         2711  42.90839  0.145268      7   14  2008    16    9   49   3109.891113

                              GPS  Distance  Distance_Cum      Temps  Delta_time  \
         2711  (42.90839, 0.145268)  0.018818    105.935342  16.163611    0.001111

                Vitesse    Pente
         2711  16.936457  5.95228

In [43]: folium.Marker(location=[T.ix[2711,'Lat'],T.ix[2711,'Long']],
                       popup='Altitude maximale = 3110 m',
                       icon=folium.Icon(color='black',icon='info-sign')).add_to(carte)

Out[43]: <folium.map.Marker at 0x11f9f4898>

In [44]: #carte.save('C:/Lycee(henrimoissan)/Prof (henri-moissan)/Annee 2017/philippe/Activite/V

In [45]: carte

Out[45]: <folium.folium.Map at 0x11f6abef0>
```