Biomass Prediction All Data (Moving Window)_two_vars

May 24, 2017

This script tiles the study area into $400m^2$ cells and computes a biomass prediction for each cell using the regression parameters developed in the script All Data Biomass Regression.

```
In [1]: %matplotlib inline
        import pandas, numpy, math
        from os import system
        import time
        from matplotlib import pyplot as plt
        from IPython.display import display
        import matplotlib
        import warnings
        warnings.filterwarnings('ignore')
        from numpy import log as ln
```

Import first and last return textfiles.

```
In [2]: firstfile = r"D:\Users\jkelly\first_all\first_all_combined.txt"
        first = pandas.read_table(firstfile, header=0, delimiter=" ")
        system('say table finished') #only works ok Mac
```

```
Out[2]: 1
```

This function computes only first-return mean and first-return 50th percentile cutoff for each grid cell, the explanatory variables identified in the prior script. These are then combined with the regression parameters from the previous script to predict a per-cell biomass estimate on the 0.04 ha grid cells.

```
In [26]: #read parameters from stored textfile
         f = open('parameters_all_two.csv', 'r')
         with f:
             intercept = float(f.readline())
             b1 = float(f.readline())
             b2 = float(f.readline())
         f.close()
         print (intercept, b1, b2)
```

```
0.466532587579 -0.20488443893 0.144928149721
```

```
In [27]: def heightstats(SERIES, INT, B1, B2):

             MEAN = numpy.mean(SERIES)
             H50 = numpy.percentile(SERIES, 50)
             #these values are copied from biomass regression script
             #intercept = 0.466532
             #b1 = -0.2048844
             #b2 = 0.1449281

             BIO = numpy.exp(INT + B1*ln(H50) + B2*MEAN)
             outlist = [MEAN, H50, round(BIO,2)]

             return(outlist)
```

The next section sets up the moving window loop. It first builds a pandas dataframe with column names and then sorts on x values in order to work efficiently in left-to-right columns across the study area. It also identifies the extent coordinates and Δ, the increment for cell size (here 20m x 20m).

```
In [5]: df = pandas.DataFrame(first)
        df.columns = ['Xcoord', 'Ycoord', 'Zcoord', 'Angle']
        df1 = df.sort_values(by=['Xcoord'], ascending=True)
        df1.index = range(1,len(df1) + 1)

        print("df = " + str(len(df)))

        xmin0 = 604790.00
        xmax0 = 610380.37
        delta = 20
        ymin0 = 6596136.30
        ymax0 = 6600916.67

df = 188881266


In [6]: df1.head()

Out[6]:       Xcoord      Ycoord  Zcoord  Angle
        1  604917.28  6598933.61    0.11    -19
        2  604917.28  6598933.70    0.09    -19
        3  604917.28  6598933.59    0.07    -19
        4  604917.28  6598933.11    0.01    -19
        5  604917.28  6598933.62    0.11    -19
```

The next section uses Δ to create lists of x and y block endpoint coordinates.

```
In [7]: ylist = []
        # from max to min by 20s
        for i in range(int(ymin0),  int(math.ceil(ymax0)), int(delta)):
```

```
        ylist.append(i)
    xlist = []
    for i in range(int(xmin0),  int(math.ceil(xmax0)), int(delta)):
        xlist.append(i)

    len(xlist),len(ylist)
```

Out[7]: (280, 240)

The next section loops over x values. It isolates a column by slicing off a Δ-unit-wide xstrip, which it then sorts by y values in order to work in blocks from bottom to top, chopping off each Δ-unit-wide yblock and calculating that block's lidar metrics and predicted biomass.

In [28]:
```
statslist=[] ## Good to here

 for xmin in xlist:
     xmax = xmin + delta

     # cut x-strip between current xmin and xmax
     xstrip = df1.loc[lambda df2:(df2.Xcoord >= xmin)
                      & (df2.Xcoord < xmax), :]


     if len(xstrip) == 0: continue
     #print("*** xstrip centered at " + str(xmin + (delta/2)) +
     #      " has " + str(len(xstrip)) +" elements")

     # sort x-strip by y
     xstrip = xstrip.sort_values(by=['Ycoord'], ascending=True)
     xstrip.index = range(1,len(xstrip) + 1)


     for ymin in ylist:
         ymax = ymin + delta

         # cut y-block between current ymin and ymax
         yblock = xstrip.loc[lambda df3:(df3.Ycoord >= ymin)
                             & (df3.Ycoord < ymax), :]


         if len(yblock) < 10: continue
         elif len(yblock) >= 10:


             blocklist = heightstats(yblock.Zcoord, intercept, b1, b2)
             anglemean = numpy.mean(yblock.Angle)
             blocklist.append(anglemean)
             blocklist.append(xmin + delta/2)   # append block center coords
             blocklist.append(ymin + delta/2)
```

```
                    statslist.append(blocklist)

        system('say process finished') #only works on mac
```

```
Out[28]: 1
```

Converts statistics list to pandas dataframe and writes to csv.

```
In [29]: #statsarray = numpy.array(statslist)
         outdf = pandas.DataFrame(statslist)
         columnlist = ['F_mean', 'F_h50', 'Biomass_est', 'Angle',
                       "Xcenter", "Ycenter"]
         outdf.columns = columnlist
         outdf.Biomass_est = outdf.Biomass_est.apply(pandas.to_numeric)
         outdf.to_csv('prediction_from_Fmean_h50.csv')
```

```
In [30]: outdf[0:100]
```

Out[30]:

	F_mean	F_h50	Biomass_est	Angle	Xcenter	Ycenter
0	10.452517	12.940	4.29	20.000000	604920.0	6597886.0
1	7.523223	7.530	3.14	19.074380	604920.0	6597906.0
2	7.088448	9.000	2.84	18.620690	604920.0	6597926.0
3	7.494194	8.250	3.07	17.989247	604920.0	6597946.0
4	7.611042	8.310	3.11	17.020833	604920.0	6597966.0
5	7.158977	7.545	2.97	16.352273	604920.0	6597986.0
6	3.957097	2.960	2.27	15.720430	604920.0	6598006.0
7	5.404095	5.050	2.50	14.961905	604920.0	6598026.0
8	5.166882	5.430	2.38	14.000000	604920.0	6598046.0
9	6.118987	6.260	2.66	13.341772	604920.0	6598066.0
10	11.421915	11.845	5.03	12.702128	604920.0	6598086.0
11	15.937586	17.610	8.92	12.000000	604920.0	6598106.0
12	16.075487	17.050	9.16	11.053097	604920.0	6598126.0
13	15.020962	16.060	7.96	10.269231	604920.0	6598146.0
14	17.823130	18.130	11.66	9.600000	604920.0	6598166.0
15	19.959826	20.090	15.56	8.895652	604920.0	6598186.0
16	17.289386	19.025	10.68	8.000000	604920.0	6598206.0
17	16.958644	18.020	10.30	7.161017	604920.0	6598226.0
18	15.248033	15.765	8.26	6.434426	604920.0	6598246.0
19	16.304298	17.300	9.44	5.702479	604920.0	6598266.0
20	9.731880	10.630	4.03	4.957265	604920.0	6598286.0
21	13.296885	15.190	6.27	4.000000	604920.0	6598306.0
22	15.385693	16.110	8.39	3.211679	604920.0	6598326.0
23	5.110612	0.010	8.59	2.510204	604920.0	6598346.0
24	0.058649	0.010	4.13	1.783784	604920.0	6598366.0
25	0.039091	0.010	4.12	0.987013	604920.0	6598386.0
26	0.056282	0.010	4.13	0.000000	604920.0	6598406.0
27	0.048701	0.010	4.13	0.000000	604920.0	6598426.0
28	0.052308	0.010	4.13	-0.423077	604920.0	6598446.0
29	0.054091	0.010	4.13	-1.170455	604920.0	6598466.0

```
..         ...     ...          ...         ...       ...         ...
70  18.919430  19.605        13.45    8.000000  604940.0  6598206.0
71  19.435365  20.190        14.40    7.192067  604940.0  6598226.0
72  18.290385  18.950        12.36    6.443320  604940.0  6598246.0
73  16.363265  17.180         9.54    5.714579  604940.0  6598266.0
74  15.364139  16.090         8.36    4.974257  604940.0  6598286.0
75  14.193680  15.900         7.08    4.002165  604940.0  6598306.0
76  13.356021  15.500         6.30    3.263830  604940.0  6598326.0
77   6.361469   1.880         3.52    2.536082  604940.0  6598346.0
78   0.929740   0.010         4.69    1.762082  604940.0  6598366.0
79   0.119017   0.010         4.17    1.000000  604940.0  6598386.0
80   0.052194   0.010         4.13    0.012658  604940.0  6598406.0
81   0.066939   0.010         4.14    0.000000  604940.0  6598426.0
82   0.080000   0.010         4.14   -0.405063  604940.0  6598446.0
83   0.058755   0.010         4.13   -1.160643  604940.0  6598466.0
84   0.043843   0.010         4.12   -2.000000  604940.0  6598486.0
85   0.056176   0.010         4.13   -2.831933  604940.0  6598506.0
86   6.466391   0.960         4.10   -3.694190  604940.0  6598526.0
87   7.961212   7.840         3.31   -4.286501  604940.0  6598546.0
88   3.402508   0.190         3.67   -5.000000  604940.0  6598566.0
89   3.160332   0.080         4.23   -5.981550  604940.0  6598586.0
90   0.396195   0.010         4.34   -6.668142  604940.0  6598606.0
91   0.067443   0.010         4.14   -7.365297  604940.0  6598626.0
92   0.042461   0.010         4.12   -8.020942  604940.0  6598646.0
93   0.049200   0.010         4.13   -9.000000  604940.0  6598666.0
94   0.058406   0.010         4.13   -9.729084  604940.0  6598686.0
95   0.040275   0.010         4.12  -10.376471  604940.0  6598706.0
96   0.051847   0.010         4.13  -11.058559  604940.0  6598726.0
97   0.585000   0.080         2.91  -12.000000  604940.0  6598746.0
98   0.043381   0.010         4.12  -12.661905  604940.0  6598766.0
99   0.022396   0.010         4.11  -13.364583  604940.0  6598786.0

[100 rows x 6 columns]
```

```
In [31]: outdf1 = outdf.replace([numpy.inf, -numpy.inf],
                                numpy.nan).dropna(subset=['Biomass_est'], how="all'
```

These are the total biomass estimate and the overall biomass-per-hectare for the 2170.5 hectare study area:

```
In [32]: total = round(outdf1['Biomass_est'].sum(), 3)
```

```
In [33]: print ("total biomass in Mg, average biomass per hectare")
         "{:,}".format(total), "{:,}".format(round(total/2170.5, 3))
```

```
total biomass in Mg, average biomass per hectare
```

```
Out[33]: ('222,707.62', '102.607')
```

```
In [ ]:

In [34]: df.mean()

Out[34]: Xcoord    6.076169e+05
         Ycoord    6.598451e+06
         Zcoord    8.214472e+00
         Angle     1.024749e+00
         dtype: float64

In [35]: outdf1.mean()

Out[35]: F_mean         7.518051e+00
         F_h50          7.802198e+00
         Biomass_est    4.102487e+00
         Angle          1.413942e+00
         Xcenter        6.076454e+05
         Ycenter        6.598487e+06
         dtype: float64

In [ ]:
```