Biomass Prediction All Data (Moving Window) three 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 [36]: %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 [37]: firstfile = r"D:\Users\jkelly\first_all\first_all_combined.txt"
         first = pandas.read_table(firstfile, header=0, delimiter=" ")
         lastfile = r"D:\Users\jkelly\last_all\last_all_combined.txt"
         last = pandas.read_table(lastfile, header=0, delimiter=" ")
         system('say table finished')
```

```
Out[37]: 1
```

This function computes only last-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 biomass estimate on the 0.04 ha grid cells.

```
In [46]: def heightstats(L_SERIES, F_SERIES, INT, B1, B2, B3):

             L_MEAN = numpy.mean(L_SERIES)
             H50 = numpy.percentile(F_SERIES, 50)


             BIO = numpy.exp(INT + B1*ln(L_MEAN) + B2*ln(H50) + B3*H50)
             outlist = [L_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 20x20).

```
In [39]: df = pandas.DataFrame(first)
         df.columns = ['Xcoord', 'Ycoord', 'Zcoord', 'Angle']
         dfF = df.sort_values(by=['Xcoord'], ascending=True)
         dfF.index = range(1,len(dfF) + 1)
         df = pandas.DataFrame(last)
         df.columns = ['Xcoord', 'Ycoord', 'Zcoord', 'Angle']
         dfL = df.sort_values(by=['Xcoord'], ascending=True)
         dfL.index = range(1,len(dfL) + 1)
         print("dfF = " + str(len(dfF)))
         print("dfL = " + str(len(dfL)))
         xmin0 = 604790.00
         xmax0 = 610380.37
         delta = 20
         ymin0 = 6596136.30
         ymax0 = 6600916.67

dfF = 188881266
dfL = 41918199


In [40]: dfL.head()

Out[40]:       Xcoord      Ycoord  Zcoord  Angle
         1  604924.18  6598334.93    0.01      3
         2  604924.20  6598335.88    0.01      3
         3  604924.24  6598332.00    0.01      3
         4  604924.27  6598329.08    0.01      3
         5  604924.29  6598328.11    0.01      3
```

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

```
In [41]: 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[41]: (280, 240)
```

Dimensions arrays to hold statistics.

```
In [65]: f = open('parameters_all.csv', 'r')
         with f:
             intercept = float(f.readline())
             b1 = float(f.readline())
             b2 = float(f.readline())
             b3 = float(f.readline())

         f.close()
         print (intercept, b1, b2, b3)
```

```
0.691575840756 0.134929275669 -0.318404481456 0.128437207872
```

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 [47]: statslist=[] ## Good to here

         for xmin in xlist:
             xmax = xmin + delta

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

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

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

             for ymin in ylist:
                 ymax = ymin + delta

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

                 if len(yblockF) < 10: continue
                 elif len(yblockF) >= 10:
```

```
                    blocklist = heightstats(yblockL.Zcoord, yblockF.Zcoord, interc
                    anglemean = numpy.mean(yblockF.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[47]: 1
```

Converts statistics list to pandas dataframe and writes to csv.

```
In [61]: #statsarray = numpy.array(statslist)
         outdf = pandas.DataFrame(statslist)
         columnlist = ['L_mean', 'F_h50', 'Biomass_est', 'Angle',
                       "Xcenter", "Ycenter"]
         outdf.columns = columnlist
         outdf.Biomass_est = outdf.Biomass_est.apply(pandas.to_numeric)
         outdf.to_csv('trial_run_1.csv')
```

```
In [67]: outdf[0:100]
```

```
Out[67]:
```

	L_mean	F_h50	Biomass_est	Angle	Xcenter	Ycenter
0	1.708929	12.940	5.01	20.000000	604920.0	6597886.0
1	1.649508	7.530	2.95	19.074380	604920.0	6597906.0
2	2.048000	9.000	3.47	18.620690	604920.0	6597926.0
3	0.863462	8.250	2.88	17.989247	604920.0	6597946.0
4	0.215000	8.310	2.40	17.020833	604920.0	6597966.0
5	0.625429	7.545	2.60	16.352273	604920.0	6597986.0
6	0.493529	2.960	1.88	15.720430	604920.0	6598006.0
7	0.572083	5.050	2.12	14.961905	604920.0	6598026.0
8	0.240000	5.430	1.93	14.000000	604920.0	6598046.0
9	0.890000	6.260	2.45	13.341772	604920.0	6598066.0
10	1.175769	11.845	4.25	12.702128	604920.0	6598086.0
11	3.719259	17.610	9.18	12.000000	604920.0	6598106.0
12	7.320303	17.050	9.46	11.053097	604920.0	6598126.0
13	4.144545	16.060	7.86	10.269231	604920.0	6598146.0
14	7.368462	18.130	10.67	9.600000	604920.0	6598166.0
15	4.234231	20.090	12.32	8.895652	604920.0	6598186.0
16	5.710909	19.025	11.38	8.000000	604920.0	6598206.0
17	7.990588	18.020	10.65	7.161017	604920.0	6598226.0
18	5.662917	15.765	7.94	6.434426	604920.0	6598246.0
19	4.478718	17.300	9.10	5.702479	604920.0	6598266.0
20	1.459487	10.630	3.88	4.957265	604920.0	6598286.0
21	4.641591	15.190	7.27	4.000000	604920.0	6598306.0
22	6.033276	16.110	8.32	3.211679	604920.0	6598326.0

```
23  4.535833   0.010        10.62   2.510204  604920.0  6598346.0
24       NaN   0.010          NaN   1.783784  604920.0  6598366.0
25       NaN   0.010          NaN   0.987013  604920.0  6598386.0
26       NaN   0.010          NaN   0.000000  604920.0  6598406.0
27       NaN   0.010          NaN   0.000000  604920.0  6598426.0
28       NaN   0.010          NaN  -0.423077  604920.0  6598446.0
29       NaN   0.010          NaN  -1.170455  604920.0  6598466.0
..       ...     ...          ...        ...       ...        ...
70  6.472672  19.605        12.36   8.000000  604940.0  6598206.0
71  7.279588  20.190        13.41   7.192067  604940.0  6598226.0
72  5.837090  18.950        11.32   6.443320  604940.0  6598246.0
73  5.920509  17.180         9.32   5.714579  604940.0  6598266.0
74  4.432748  16.090         7.96   4.974257  604940.0  6598286.0
75  5.549938  15.900         8.04   4.002165  604940.0  6598306.0
76  4.851095  15.500         7.56   3.263830  604940.0  6598326.0
77  4.390476   1.880         2.54   2.536082  604940.0  6598346.0
78  0.010000   0.010         4.65   1.762082  604940.0  6598366.0
79       NaN   0.010          NaN   1.000000  604940.0  6598386.0
80       NaN   0.010          NaN   0.012658  604940.0  6598406.0
81       NaN   0.010          NaN   0.000000  604940.0  6598426.0
82       NaN   0.010          NaN  -0.405063  604940.0  6598446.0
83       NaN   0.010          NaN  -1.160643  604940.0  6598466.0
84       NaN   0.010          NaN  -2.000000  604940.0  6598486.0
85       NaN   0.010          NaN  -2.831933  604940.0  6598506.0
86  4.967000   0.960         2.84  -3.694190  604940.0  6598526.0
87  3.478871   7.840         3.36  -4.286501  604940.0  6598546.0
88  1.261613   0.190         3.58  -5.000000  604940.0  6598566.0
89  0.230000   0.080         3.70  -5.981550  604940.0  6598586.0
90  0.110000   0.010         6.43  -6.668142  604940.0  6598606.0
91       NaN   0.010          NaN  -7.365297  604940.0  6598626.0
92       NaN   0.010          NaN  -8.020942  604940.0  6598646.0
93       NaN   0.010          NaN  -9.000000  604940.0  6598666.0
94       NaN   0.010          NaN  -9.729084  604940.0  6598686.0
95       NaN   0.010          NaN -10.376471  604940.0  6598706.0
96       NaN   0.010          NaN -11.058559  604940.0  6598726.0
97       NaN   0.080          NaN -12.000000  604940.0  6598746.0
98       NaN   0.010          NaN -12.661905  604940.0  6598766.0
99       NaN   0.010          NaN -13.364583  604940.0  6598786.0

[100 rows x 6 columns]
```

```
In [68]: 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 [69]: total = round(outdf1['Biomass_est'].sum(), 3)
```

```
In [70]: 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[70]: ('212,177.71', '97.755')

In [71]: dfL.mean()

Out[71]: Xcoord    6.075622e+05
         Ycoord    6.598420e+06
         Zcoord    2.485655e+00
         Angle     1.475276e+00
         dtype: float64

In [72]: dfF.mean()

Out[72]: Xcoord    6.076169e+05
         Ycoord    6.598451e+06
         Zcoord    8.214472e+00
         Angle     1.024749e+00
         dtype: float64

In [73]: outdf1.mean()

Out[73]: L_mean         2.162664e+00
         F_h50          8.501369e+00
         Biomass_est    4.261966e+00
         Angle          1.756524e+00
         Xcenter        6.075992e+05
         Ycenter        6.598487e+06
         dtype: float64

In [ ]:

In [ ]:
```