

```

1  * Advanced Derivatives Assignment 1 (Jack, Zaid, Nick, Asim, Dikshant)
2
3  * Reproduction of Hull and White Results
4
5  clear all
6  set more off
7
8  *log using Version1, replace
9
10 cd "C:\Users\JackE\Documents\Jack\Jack's Work\UoT MFE\Semester 2\Advanced
    Derivatives\Assignment 1"
11 use "C:\Users\JackE\Documents\Jack\Jack's Work\UoT MFE\Semester 2\Advanced
    Derivatives\Assignment 1\Assignment1_V2.dta"
12
13 *summary stats
14 summarize spx_level strike_price impl_volatility
15
16 *generate variables
17 gen yearmonth = ym(year, month)
18 gen yearmonth1 = yearmonth-527
19
20 gen sqrt_T = sqrt(lifeopt/360)
21 gen change_S = (n_spx - spx_level)
22 gen change_F = (n_midmarket - midmarket)
23 gen delta_sq = delta^2
24
25 gen m1 = (vega/sqrt_T)
26 gen m2 = (change_S/spx_level)
27 gen multiplier = m1*m2
28
29 gen x1 = multiplier
30 gen x2 = multiplier * delta
31 gen x3 = multiplier * delta_sq
32 gen y = (change_F - (delta*change_S))
33
34 tostring datel, replace format(%20.0f)
35 gen date stata = date(datel, "YMD")
36
37 gen round_delta = round(delta, 0.1)
38
39 * CALLS
40
41 gen call x1 = .
42 gen call_x2 = .
43 gen call_x3 = .
44 gen yhat_call = .
45 gen window1 = .
46
47 *have 105 months
48
49 local j = 1
50 forvalues i = 1(1)105{
51 quietly reg y x1 x2 x3 if iscall ==1 & inrange(yearmonth1, 0+`i',35+`i'), noconstant
52 quietly replace call_x1 = _b[x1] if call_x1 ==. & inrange(yearmonth1, 0+`i',35+`i') & iscall
    ==1
53 quietly replace call_x2 = _b[x2] if call_x2 ==. & inrange(yearmonth1, 0+`i',35+`i') & iscall
    ==1
54 quietly replace call_x3 = _b[x3] if call_x3 ==. & inrange(yearmonth1, 0+`i',35+`i') & iscall
    ==1
55 quietly predict yhat
56 quietly replace yhat_call = yhat if yhat_call ==. & inrange(yearmonth1, 0+`i',35+`i')&
    iscall ==1
57 quietly drop yhat
58 quietly replace window1 = `j' if window1 ==. & inrange(yearmonth1, 0+`i',35+`i') & iscall ==1
59 local j = `j'+1
60 }
61
62 preserve
63
64 *collpase to get a_hat, b_hat, c_hat for each regression

```

```

65 collapse (mean) call_x1 call_x2 call_x3 date_stata, by (window1)
66 gen call_minusx2 = (-1)*call_x2
67 format date_stata %td
68
69 gen a_call = call_x1
70 gen minusb_call = call_minusx2
71 gen c_call = call_x3
72
73 *graph
74 line a_call minusb_call c_call date_stata, xtitle("Date") title("Call Parameters")
75 graph save Call.gph, replace
76
77 restore
78
79 * PUTS
80
81 gen put_x1 = .
82 gen put_x2 = .
83 gen put_x3 = .
84 gen yhat_put = .
85 gen window2 = .
86
87 local j = 1
88 forvalues i = 1(1)105{
89 quietly reg y x1 x2 x3 if iscall ==0 & inrange(yearmonth1, 0+`i',35+`i'), noconstant
90 quietly replace put_x1 = _b[x1] if put_x1 ==. & inrange(yearmonth1, 0+`i',35+`i') & iscall
==0
91 quietly replace put_x2 = _b[x2] if put_x2 ==. & inrange(yearmonth1, 0+`i',35+`i') & iscall
==0
92 quietly replace put_x3 = _b[x3] if put_x3 ==. & inrange(yearmonth1, 0+`i',35+`i') & iscall
==0
93 quietly predict yhat
94 quietly replace yhat_put = yhat if yhat_put ==. & inrange(yearmonth1, 0+`i',35+`i') & iscall
==0
95 quietly drop yhat
96 quietly replace window2 = `j' if window2 ==. & inrange(yearmonth1, 0+`i',35+`i') & iscall ==0
97 local j = `j'+1
98 }
99
100 preserve
101
102 *collapse to find parameters
103 collapse (mean) put x1 put x2 put x3 date stata, by (window2)
104 format date_stata %td
105
106 gen a_put = put_x1
107 gen b_put = put_x2
108 gen c_put = put_x3
109
110 *graph
111 line a_put b_put c_put date_stata, xtitle("Date") title("Put Parameters")
112 graph save Put.gph, replace
113
114 restore
115
116 gr combine Call.gph Put.gph
117 graph save Combined.gph, replace
118
119 **End PART 1
120
121 *ERRORS
122 gen mv_error_call = (y - yhat_call)^2 if iscall ==1
123 gen bs_error_call = y^2 if iscall ==1
124
125 gen mv_error_put = (y - yhat_put)^2 if iscall ==0
126 gen bs_error_put = y^2 if iscall ==0
127
128 *calls
129
130 by round_delta, sort: egen george = total( mv_error_call) if iscall==1

```

```

131 by round_delta, sort: egen george2 = total( bs_error_call) if iscall==1
132 egen george3 = total(mv_error_call) if iscall ==1
133 egen george4 = total(bs_error_call) if iscall==1
134
135 *puts
136 by round_delta, sort: egen frank = total( mv_error_put) if iscall ==0
137 by round_delta, sort: egen frank2 = total( bs_error_put) if iscall ==0
138 egen frank3 = total(mv_error_put) if iscall ==0
139 egen frank4 = total(bs_error_put) if iscall ==0
140
141 *calculate gains
142 gen gain_call = (1-(george/george2))*100 if iscall ==1
143 gen gain_put = (1-(frank/frank2))*100 if iscall ==0
144
145 gen call_all = (1-(george3/george4))*100 if iscall ==1
146 gen put_all = (1-(frank3/frank4))*100 if iscall ==0
147 egen call_tot = mean(call_all) if iscall ==1
148 egen put_tot = mean(put_all) if iscall ==0
149
150
151 *put calculated GAINS in matrix
152 local r = 100
153 local column Call_Delta Gain(%) Put_Delta Gain(%)
154 matrix A = J(10,4,.)
155 matrix colnames A=`column'
156 matrix A[1,1] = 0.1
157 matrix A[2,1] = 0.2
158 matrix A[3,1] = 0.3
159 matrix A[4,1] = 0.4
160 matrix A[5,1] = 0.5
161 matrix A[6,1] = 0.6
162 matrix A[7,1] = 0.7
163 matrix A[8,1] = 0.8
164 matrix A[9,1] = 0.9
165 matrix A[10,1] = `r'
166
167 matrix A[1,3] = -0.9
168 matrix A[2,3] = -0.8
169 matrix A[3,3] = -0.7
170 matrix A[4,3] = -0.6
171 matrix A[5,3] = -0.5
172 matrix A[6,3] = -0.4
173 matrix A[7,3] = -0.3
174 matrix A[8,3] = -0.2
175 matrix A[9,3] = -0.1
176 matrix A[10,3] = `r'
177
178 preserve
179 collapse (mean) call_tot
180 mkmat call_tot, matrix(yay)
181 matrix A[10,2] = yay[1,1]
182 restore
183 preserve
184 collapse (mean) put_tot
185 mkmat put_tot, matrix(yay1)
186 matrix A[10,4] = yay1[1,1]
187 restore
188
189 preserve
190 bysort round_delta: keep if _n==1
191 mkmat gain_call, matrix(call)
192 mkmat gain_put, matrix(put)
193
194 local i = 2
195 matrix A[1,`i'] = call[10,1]
196 matrix A[2,`i'] = call[11,1]
197 matrix A[3,`i'] = call[12,1]
198 matrix A[4,`i'] = call[13,1]
199 matrix A[5,`i'] = call[14,1]
200 matrix A[6,`i'] = call[15,1]

```

```
201  matrix A[7,`i'] = call[16,1]
202  matrix A[8,`i'] = call[17,1]
203  matrix A[9,`i'] = call[18,1]
204
205
206
207  local i = 4
208  matrix A[1,`i'] = put[1,1]
209  matrix A[2,`i'] = put[2,1]
210  matrix A[3,`i'] = put[3,1]
211  matrix A[4,`i'] = put[4,1]
212  matrix A[5,`i'] = put[5,1]
213  matrix A[6,`i'] = put[6,1]
214  matrix A[7,`i'] = put[7,1]
215  matrix A[8,`i'] = put[8,1]
216  matrix A[9,`i'] = put[9,1]
217
218  matlist A
219
220  restore
221
222  *log close
223  *translate Version1.smcl Version1.pdf
224
225
226
```