Project1

Ari Boyarsky, Isaac Gritz, Clare Lohrmann, Sameer Rau, Abdul Sheikhnureldin

October 3, 2016

Setup:

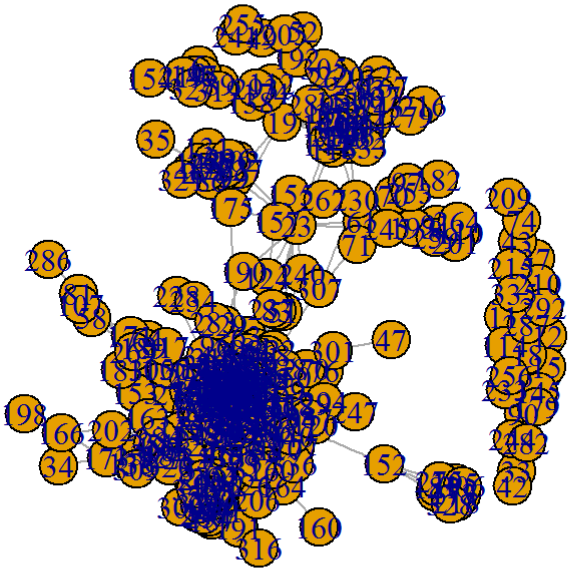
```
#clean workspace
rm(list = ls())

#require igraph
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
#read edges csv for nodeid 0
graph.edges <- read.csv(file = 'facebook/0.edges', sep=" ", header=FALSE, col.names=c("source",
"target"))

#get edges
graph.e <- graph.edges - min(graph.edges) + 1
#get nodes
graph.n <- data.frame(id=seq(max(graph.e) - min(graph.e) + 1))

graph.n$group <- 1

#import into igraph
g = graph.data.frame(graph.e, directed=FALSE, vertices=graph.n)

#simplify code
g2 = simplify(g)

#plot
plot(g2)
```



Question 3:

Explore with some functions from ppt

```
#get verticies
V(g2)
```

```
## + 347/347 vertices, named:
##   [1] 1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17
##  [18] 18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
##  [35] 35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51
##  [52] 52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68
##  [69] 69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85
##  [86] 86  87  88  89  90  91  92  93  94  95  96  97  98  99  100 101 102
## [103] 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
## [120] 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136
## [137] 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153
## [154] 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170
## + ... omitted several vertices
```

```
#get edges
edges(g2)
```

```
## [[1]]
## IGRAPH UN-- 347 2519 --
## + attr: name (v/c), group (v/n)
## + edges (vertex names):
##  [1] 1--48  1--53  1--54  1--73  1--88  1--92  1--119 1--126 1--133 1--194
## [11] 1--236 1--280 1--299 1--315 1--322 1--346 2--20  2--115 2--116 2--149
## [21] 2--226 2--312 2--326 2--333 2--343 3--9   3--25  3--26  3--67  3--72
## [31] 3--85  3--122 3--142 3--170 3--188 3--200 3--228 3--274 3--280 3--283
## [41] 3--323 4--78  4--152 4--181 4--195 4--218 4--273 4--275 4--306 4--328
## [51] 5--87  5--122 5--156 5--158 5--169 5--180 5--187 5--204 5--213 5--235
## [61] 5--315 5--316 6--89  6--95  6--147 6--219 6--319 7--22  7--31  7--38
## [71] 7--65  7--87  7--103 7--129 7--136 7--168 7--213 7--246 7--291 7--304
## + ... omitted several edges
##
## attr(,"class")
## [1] "igraph.edge"
```

```
#find vertex attributes, did not load attributes so we should not expect much here
vertex_attr(g2)
```

```
## $name
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"
##  [12] "12"  "13"  "14"  "15"  "16"  "17"  "18"  "19"  "20"  "21"  "22"
##  [23] "23"  "24"  "25"  "26"  "27"  "28"  "29"  "30"  "31"  "32"  "33"
##  [34] "34"  "35"  "36"  "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"
##  [45] "45"  "46"  "47"  "48"  "49"  "50"  "51"  "52"  "53"  "54"  "55"
##  [56] "56"  "57"  "58"  "59"  "60"  "61"  "62"  "63"  "64"  "65"  "66"
##  [67] "67"  "68"  "69"  "70"  "71"  "72"  "73"  "74"  "75"  "76"  "77"
##  [78] "78"  "79"  "80"  "81"  "82"  "83"  "84"  "85"  "86"  "87"  "88"
##  [89] "89"  "90"  "91"  "92"  "93"  "94"  "95"  "96"  "97"  "98"  "99"
## [100] "100" "101" "102" "103" "104" "105" "106" "107" "108" "109" "110"
## [111] "111" "112" "113" "114" "115" "116" "117" "118" "119" "120" "121"
## [122] "122" "123" "124" "125" "126" "127" "128" "129" "130" "131" "132"
## [133] "133" "134" "135" "136" "137" "138" "139" "140" "141" "142" "143"
## [144] "144" "145" "146" "147" "148" "149" "150" "151" "152" "153" "154"
## [155] "155" "156" "157" "158" "159" "160" "161" "162" "163" "164" "165"
## [166] "166" "167" "168" "169" "170" "171" "172" "173" "174" "175" "176"
## [177] "177" "178" "179" "180" "181" "182" "183" "184" "185" "186" "187"
## [188] "188" "189" "190" "191" "192" "193" "194" "195" "196" "197" "198"
## [199] "199" "200" "201" "202" "203" "204" "205" "206" "207" "208" "209"
## [210] "210" "211" "212" "213" "214" "215" "216" "217" "218" "219" "220"
## [221] "221" "222" "223" "224" "225" "226" "227" "228" "229" "230" "231"
## [232] "232" "233" "234" "235" "236" "237" "238" "239" "240" "241" "242"
## [243] "243" "244" "245" "246" "247" "248" "249" "250" "251" "252" "253"
## [254] "254" "255" "256" "257" "258" "259" "260" "261" "262" "263" "264"
## [265] "265" "266" "267" "268" "269" "270" "271" "272" "273" "274" "275"
## [276] "276" "277" "278" "279" "280" "281" "282" "283" "284" "285" "286"
## [287] "287" "288" "289" "290" "291" "292" "293" "294" "295" "296" "297"
## [298] "298" "299" "300" "301" "302" "303" "304" "305" "306" "307" "308"
## [309] "309" "310" "311" "312" "313" "314" "315" "316" "317" "318" "319"
## [320] "320" "321" "322" "323" "324" "325" "326" "327" "328" "329" "330"
## [331] "331" "332" "333" "334" "335" "336" "337" "338" "339" "340" "341"
## [342] "342" "343" "344" "345" "346" "347"
##
## $group
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [71] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [106] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [141] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [176] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [211] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [246] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [281] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [316] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
#check if the graph is simple (does not have loops or multiple edges between verticies)
is.simple(g2)
```

```
## [1] TRUE
```

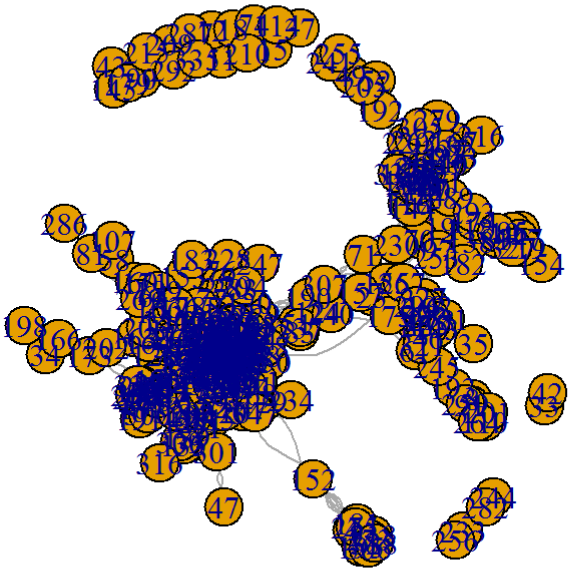
```
degree(g2, "198")
```

```
## 198
##   1
```

```
degree(g2)
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  16   9  16   9  12   5  19   7  56   9   0   0  30  14   0   8  12   0
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  15  14  64  10  16  15  68  67   4  12  12  16  22   5   1   1   1  10
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##   0   8  14  43  23   1   0   5  11   4   1  21   3  10   6   1  30   7
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  16  77  14   3  18   7   2  25   5   6  11  14  75   8   9   1   2  23
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##   9   0  13   2   5   8  11  22   2  33   6  12  13   5  12  19   7   1
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
##   7  20   7  21   5   8   2  48  12   8  18   5  15  31  13   7   2  12
## 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
##  36   4  13   2  39   0  20  16   5  35  61   3  11  62  17   3   3   6
## 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
##  15  27   6  15   6  15  17  18   9  21  15   1   8  10  27  42  11  14
## 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
##   1   9   5  19  13  10   6   4   1   1   2  11   2  24  13   1  24   7
## 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   5   2  10   3   6  10  37  45   7  40   5   3  16  13  10  12   2  19
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
##   9   2   1  17  25  43  15  47   6   3   2   4   4  18   8  12  15   1
## 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
##  46  56   3   3  56  21   1   3   2   6   0   0  29  17  38  16   0   1
## 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
##   7   8   5   3   7  10  26  27   9  13  14   2   5   8  20  24   1   1
## 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
##   4  36   6  22  58   2   1  23   7   1   4  13   2  20  23   4  13  64
## 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
##   2  16   1   1  17  14   7   7  37   3   6   4  26  17   1  10   5   3
## 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
##  72  44   8  13   9  17  64   9   1  42  15   1   4  15  46   1   0   3
## 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
##   3  13  35   0   2   2   9   6  24  10  19   6   2  19  20  54   1   8
## 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
##   3  23   9  12   6  25  36  12  55   1   6  10   7  20   2  71  38  25
## 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
##  38  18   3   8  29  15  19  42   7  27   0   2   8   6  26   5  11  33
## 343 344 345 346 347
##  17   8  15  26   6
```

```
astrocollab <- upgrade_graph(g)
plot(astrocollab)
```



Question 4

Explore 10 functions

The fucntion `eccentricity()` calulcates the shortest path distance from the farthest other node in the graph, for each node.

```
#find eccentricity of verticies, we are using head here so the markdown file does not get too lo
ng
head(eccentricity(g2))
```

```
## 1 2 3 4 5 6
## 8 9 8 9 8 9
```

The function `dfs()`, preforms a depth-first search from the root node, in this case 1. This is an algorithm to traverse the graph. `bfs()` is another function that employs an algorthim, breadth-first search, this algorithm attempts to go to each edge.

```
#dfs with node 1 as root
dfs(g2, 1)
```

```
## $root
## [1] 0
##
## $neimode
## [1] "out"
##
## $order
## + 347/347 vertices, named:
##   [1] 1   48  30  9   3   25  21  13  26  40  29  16  82  56  55  67  10
##  [18] 142 31  7   22  87  5   122 45  104 98  59  118 65  203 50  109 119
##  [35] 27  54  53  24  57  80  88  73  126 260 160 170 62  96  133 141 38
##  [52] 117 77  231 106 169 103 136 120 247 277 39  69  105 148 113 66  134
##  [69] 132 72  165 121 206 178 186 123 200 75  85  156 188 79  185 271 63
##  [86] 261 172 128 150 64  100 163 173 34  166 198 202 329 130 191 204 92
## [103] 94  101 180 187 194 196 249 213 161 199 211 222 238 239 60  158 168
## [120] 129 291 208 285 146 313 84  51  23  61  193 8   91  110 259 201 245
## [137] 264 83  237 99  68  143 35  46  175 19  17  41  14  20  2   115 28
## [154] 116 140 137 32  93  111 44  312 144 149 162 326 214 230 70  71  307
## + ... omitted several vertices
##
## $order.out
## NULL
##
## $father
## NULL
##
## $dist
## NULL
```

```
#bfs with node 1 as root
bfs(g2, 1)
```

```
## $root
## [1] 1
##
## $neimode
## [1] "out"
##
## $order
## + 347/347 vertices, named:
##   [1] 1   48  53  54  73  88  92  119 126 133 194 236 280 299 315 322 346
##  [18] 30  57  80  130 180 199 203 271 302 320 330 332 24  94  101 146 191
##  [35] 196 204 213 242 248 249 254 260 266 317 27  313 329 25  331 21  56
##  [52] 106 164 178 186 303 336 187 9   13  26  39  50  64  67  79  82  98
##  [69] 100 105 109 113 122 125 128 132 148 150 163 170 171 176 185 188 189
##  [86] 200 208 217 229 239 252 261 269 277 285 297 318 323 324 325 339 342
## [103] 40  62  96  141 183 224 232 276 69  84  121 142 169 257 272 304 314
## [120] 3   31  117 153 231 290 291 300 5   7   87  136 158 211 223 246 288
## [137] 345 38  45  60  75  85  104 118 161 207 221 250 308 341 127 184 108
## [154] 173 72  134 172 212 222 238 265 298 29  55  59  65  268 270 334 338
## + ... omitted several vertices
##
## $rank
## NULL
##
## $father
## NULL
##
## $pred
## NULL
##
## $succ
## NULL
##
## $dist
## NULL
```

`count_triangles()` attempts to count how many triangles a node is a part of. A triangle, is a part of a graph that has a cycle with 3 nodes and 3 edges.

```
#count triangles a vertex is a part of
count_triangles(g2)
```

```
##   [1]  41  31  70  30  14   9  63  12 578  28   0   0 273  64   0  16  45
##  [18]   0  19  58 662  16   7  93 609 870   5  48  24  52  88  10   0   0
##  [35]   0  43   0   5  38 340 106   0   0  10  37   6   0  55   0  21  12
##  [52]   0 148   9  97 970  41   1 115  12   0 145  10  14  22  78 955  22
##  [69]  20   0   1 144  15   0  56   1   4  28  51 110   0 201  12  21  46
##  [86]  10  38  45  12   0  12 143  17 133   8   6   0 569  38  14 124  10
## [103]  51 271  48   5   1  50 317   6  38   1 380   0  92  74   2 303 482
## [120]   0  20 691  89   0   1  10  73 198  13  28  15  33  73 103  36  81
## [137]  49   0  27  39 183 437  29  60   0  12   9  61  58  21  13   3   0
## [154]   0   0  24   0  83  44   0 140  17   3   0  23   1  13  28 272 443
## [171]  11 337   1   2  42  53  32  24   0 125  30   0   0  80 150 428  88
## [188] 467  14   0   1   0   3 123  28  44  74   0 464 637   3   1 628 115
## [205]   0   0   1  12   0   0 232  90 187  61   0   0  16  28   6   3  17
## [222]  31 227 170  29  61  43   1   3   2  75 162   0   0   4 237  12 159
## [239] 658   0   0 142  11   0   3  41   0  86 150   4  58 775   0 107   0
## [256]   0  60  37  12   4 297   3  14   6 199 118   0  40   2   2 945 306
## [273]  28  38  30  98 657  27   0 380  74   0   2  69 422   0   0   2   2
## [290]  62 205   0   1   0  15  14 192  42 111  10   0 124 102 521   0  28
## [307]   1 108  36  30   5 114 298  21 432   0   5  19  12  81   1 675 321
## [324]  94 408  96   3  28  69  84  81 338  15 166   0   1  21   8  65   6
## [341]  40 291  80  10  63 160  11
```

`is.directed()` checks to see if a graph is directed. `is.weighted()` checks to see if a graph has weights. `is.connected()` checks to see if a graph is connected (i.e. has a path between every pair of vertices).

```
#is the graph directed
is.directed(g2)
```

```
## [1] FALSE
```

```
#is the graph weighted
is.weighted(g2)
```

```
## [1] FALSE
```

```
#is the graph connected
is.connected(g2)
```

```
## [1] FALSE
```

`head()` retrieves the first few results of a fucntion. `closeness()` computes the closeness cnetrality, how many steps to get to every other node from another node.

```
#get closeness centrality of vertices, we are using head here so the markdown file does not get
 too long
head(closeness(g2))
```

```
##            1            2            3            4            5
## 0.0001108402 0.0001047230 0.0001107052 0.0001050420 0.0001104362
##            6
## 0.0001028807
```

`count_multiple()` counts the multiple edges a node has, edges that have the same head and tail vertices.

```
#find loop edges, we are using head here so the markdown file does not get too long
head(count_multiple(g2))
```

```
## [1] 1 1 1 1 1 1
```

`components()` finds connected components of a graph.

```
#fine components of graph
components(g2)
```

```
## $membership
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##   1   1   1   1   1   1   1   1   1   1   2   3   1   1   4   1   1   5
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   6   1   1   1
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##   7   1   1   1   1   6   8   1   1   1   1   1   1   1   1   1   1   1
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##   1   9   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1  10
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
##   1   1   1   1   1  11   1   1   1   1   1   1   1   1   1   1   1   1
## 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
##  10   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1  10   1
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
##   1   1   1   1   1   1   1   1   1   1  12  13   1   1   1   1  14   1
## 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1  15   1
## 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
##   1   1   1   1   1   1   1   1   1  16   1   1   1   1   1   1   1   1
## 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
##   1   1   1  15   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
##   1   1   1   1   1   1   1   1   1   1   1  16   1   1   1   1  17   1
## 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
##   1   1   1  18   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
##   1   1   1   1   1   1   1   1   1   1   1  19   1   1   1   1   1   1
## 343 344 345 346 347
##   1   1   1   1   1
##
## $csize
##  [1] 324   1   1   1   1   2   1   1   1   3   1   1   1   1   2   2   1
## [18]   1   1
##
## $no
## [1] 19
```

Question 5

```
#determine central person
ac1 = alpha_centrality(g)
tail(sort(ac1),5)
```

```
##       153        44       343        19       138
##  6.747615  6.901731  7.802863  9.675359 20.350718
```

```
#longest path, distances creates the distance table, which.max picks out highest non-inf value
which.max(distances(g2))
```

```
## [1] 11
```

```
#largest clique, we are using head here so the markdown file does not get too long
head(largest.cliques(g))
```

```
## [[1]]
## + 15/347 vertices, named:
##  [1] 56  271 26  67  323 122 285 277 170 188 239 200 142 186 322
##
## [[2]]
## + 15/347 vertices, named:
##  [1] 56  271 26  67  323 122 285 277 170 188 239 200 25  186 322
##
## [[3]]
## + 15/347 vertices, named:
##  [1] 56  271 26  67  323 122 285 277 170 188 9   200 142 186 322
##
## [[4]]
## + 15/347 vertices, named:
##  [1] 56  271 26  67  323 122 285 277 170 188 9   200 25  186 322
##
## [[5]]
## + 15/347 vertices, named:
##  [1] 56  271 26  67  323 122 21  200 277 170 142 186 188 322 239
##
## [[6]]
## + 15/347 vertices, named:
##  [1] 56  271 26  67  323 122 21  200 277 170 142 186 188 322 9
```

```
# Hence, the largest click is 15

#ego, we are using head here so the markdown file does not get too long
head(ego(g2, gorder(g2), V(g2)))
```

```
## [[1]]
## + 324/347 vertices, named:
##   [1] 1   48  53  54  73  88  92  119 126 133 194 236 280 299 315 322 346
##  [18] 30  57  80  130 180 199 203 271 302 320 330 332 24  94  101 146 191
##  [35] 196 204 213 242 248 249 254 260 266 317 27  313 329 25  331 21  56
##  [52] 106 164 178 186 303 336 187 9   13  26  39  50  64  67  79  82  98
##  [69] 100 105 109 113 122 125 128 132 148 150 163 170 171 176 185 188 189
##  [86] 200 208 217 229 239 252 261 269 277 285 297 318 323 324 325 339 342
## [103] 40  62  96  141 183 224 232 276 69  84  121 142 169 257 272 304 314
## [120] 3   31  117 153 231 290 291 300 5   7   87  136 158 211 223 246 288
## [137] 345 38  45  60  75  85  104 118 161 207 221 250 308 341 127 184 108
## [154] 173 72  134 172 212 222 238 265 298 29  55  59  65  268 270 334 338
## + ... omitted several vertices
##
## [[2]]
## + 324/347 vertices, named:
##   [1] 2   20  115 116 149 226 312 326 333 343 14  41  44  111 162 214 17
##  [18] 19  28  137 140 144 192 220 262 23  151 243 305 310 93  230 337 279
##  [35] 289 89  112 138 174 175 227 293 319 32  167 49  52  205 51  61  83
##  [52] 99  124 155 190 237 240 245 267 216 70  71  97  253 307 6   95  147
##  [69] 219 327 46  68  86  102 143 177 225 263 277 278 296 321 131 241 255
##  [86] 25  31  84  193 157 170 96  229 222 8   91  259 182 40  154 35  9
## [103] 10  13  21  22  26  39  50  55  56  57  67  98  104 105 109 113 118
## [120] 119 120 122 130 132 134 142 159 161 168 169 172 186 188 200 203 204
## [137] 208 213 239 248 252 271 272 280 285 290 291 294 304 308 311 315 322
## [154] 323 324 325 332 334 339 342 3   65  69  72  73  76  79  88  94  103
## + ... omitted several vertices
##
## [[3]]
## + 324/347 vertices, named:
##   [1] 3   9   25  26  67  72  85  122 142 170 188 200 228 274 280 283 323
##  [18] 21  30  56  66  69  75  79  105 113 119 128 133 134 141 148 156 161
##  [35] 169 176 185 186 199 203 224 231 232 252 258 271 272 276 277 285 291
##  [52] 295 297 304 315 322 329 334 341 342 31  39  40  51  65  73  76  83
##  [69] 84  88  94  98  103 104 130 158 221 236 237 239 246 257 270 288 290
##  [86] 325 331 332 336 13  55  62  109 118 123 172 212 223 248 265 298 308
## [103] 313 10  45  82  136 196 213 261 303 324 344 345 132 165 5   235 251
## [120] 281 284 124 211 250 260 268 208 96  1   117 121 153 100 108 127 159
## [137] 184 197 48  178 59  60  63  207 222 238 50  27  54  64  125 146 150
## [154] 163 171 189 217 229 269 318 339 183 38  87  314 106 22  29  338 77
## + ... omitted several vertices
##
## [[4]]
## + 324/347 vertices, named:
##   [1] 4   78  152 181 195 218 273 275 306 328 339 7   22  31  38  65  87
##  [18] 103 119 129 136 146 148 158 161 168 246 260 277 291 313 315 322 334
##  [35] 340 347 213 304 308 185 324 21  25  51  67  83  84  109 122 142 200
##  [52] 237 252 280 338 117 141 178 248 332 13  82  118 203 261 297 314 5
##  [69] 56  98  169 172 211 271 285 323 1   9   26  27  39  48  50  54  64
##  [86] 79  100 105 113 125 128 132 150 163 170 171 176 188 189 199 208 217
## [103] 229 239 269 318 325 329 331 342 120 156 53  317 224 272 40  60  232
## [120] 242 62  258 223 126 160 206 10  55  57  104 130 134 159 175 186 204
## [137] 290 294 311 72  231 63  222 238 265 80  94  121 133 236 249 257 288
```

```
## [154] 302 303 345 30  45  75  85  88  207 221 250 276 341 66  123 165 96
## + ... omitted several vertices
##
## [[5]]
## + 324/347 vertices, named:
##   [1] 5   87  122 156 158 169 180 187 204 213 235 315 316 7   22  136 161
##  [18] 246 291 308 322 339 3   9   21  25  26  31  45  55  56  62  66  67
##  [35] 98  104 109 113 119 123 128 141 142 170 176 186 188 200 203 224 232
##  [52] 236 239 248 251 252 261 271 272 274 276 277 280 281 284 285 297 303
##  [69] 304 323 325 332 342 344 345 85  258 295 40  60  82  132 168 242 317
##  [86] 10  103 106 121 125 165 185 257 318 334 24  48  53  80  92  94  101
## [103] 194 196 249 254 266 299 302 346 57  130 329 330 30  96  178 231 324
## [120] 340 1   105 133 199 211 223 288 313 38  65  129 347 120 146 39  118
## [137] 314 72  208 172 338 75  88  126 207 221 250 260 290 341 148 152 228
## [154] 283 69  79  134 13  108 127 159 184 197 212 51  73  76  83  84  237
## + ... omitted several vertices
##
## [[6]]
## + 324/347 vertices, named:
##   [1] 6   89  95  147 219 319 19  327 154 17  41  112 115 138 174 175 214
##  [18] 227 289 293 312 343 111 116 137 140 144 310 326 14  20  28  44  93
##  [35] 151 226 230 243 337 2   149 192 220 262 23  46  68  86  99  102 143
##  [52] 177 225 263 277 278 296 321 333 131 162 305 279 32  167 70  71  97
##  [69] 253 307 216 49  52  205 51  61  83  124 155 190 237 240 245 267 35
##  [86] 9   10  13  21  22  25  26  31  39  40  50  55  56  57  67  98  104
## [103] 105 109 113 118 119 120 122 130 132 134 142 159 161 168 169 170 172
## [120] 186 188 200 203 204 208 213 239 248 252 271 272 280 285 290 291 294
## [137] 304 308 311 315 322 323 324 325 332 334 339 342 182 241 255 84  193
## [154] 157 96  229 222 8   91  259 3   30  66  69  72  75  79  85  128 133
## + ... omitted several vertices
```

```
# power centrality, we are using head here so the markdown file does not get too long
head(power_centrality(g, nodes = V(g)))
```

```
##           1           2           3           4           5           6
## -0.04147431  0.01926547 -0.68186769 -0.48702208 -0.53447243 -1.79416816
```