**Debugger variables**

**Unit of analysis**

* Bug-month (*where a month is 28 contiguous days*)

**Bug’s ID**

BugID

MonthID

**Bug’s outcome**

Bug status in current month *(at the beginning of the current month)*

* ‘ongoing’ (0)
* ‘resolved/verified’ as ‘duplicate (1)/fixed (2)/wontfix (3)/etc… (n)’
* (*but maybe it's possible to have multiple statuses in one month? Should be mindful of this.*)
* resolution(*not asked for, but there*)

**Bug’s characteristics**

Bugzilla

* N\_months since bug reported
  + **NOTE:** This is calculated as the difference in days between the date this was reported and the first date of the current bugmonth. We divide the result by 28 and round down. (We also keep the fractional result internally, and use it for certain calculations later)
* Importance
* N\_dependencies
* N\_blocks
* N\_duplicates
* Platform
* Product
* Assigned (Yes = 1; no = 0)

Assignee

* N\_bugs **(that had any history events with the assignee)** prior month/past monthly average/cumulative **(avg/cumulative acts over distinct bugs. e.g. If we have   
  January: Bug1, Bug2  
  February: Bug2, Bug3  
  Then February will have n\_bugs\_prior\_month = 2  
  March will have n\_bugs\_prior\_month = 2  
  and March will have n\_bugs\_cumulative = 3 (NOT 4)  
  Averaging over the debugger's “age” wrt when they entered the bug network (not the IRC network)**
* N\_history events focal bug prior month/past monthly average/cumulative
* N\_history events other bugs prior month/past monthly average/cumulative
* N\_IRC member links prior month/past monthly average/cumulative
* N\_IRC messages prior month/past monthly average/cumulative (directed/undirected)
* Constraint prior month/past monthly average *(IRC network only)* **(averaging over all earlier graphs that assignee appears in might be a bit funny, since we have overlapping windows of months, e,g, Jan1-Jan28, Jan14-Feb11, Jan28-.... So instead average over *non-overlapping* previous graphs. i.e. Walk backwards 28 days at a tim`e until we fall off the end)**
* Closeness prior month/past monthly average *(IRC network only)*
* Clustering prior month/past monthly average *(IRC network only)*
* Indegree prior month/past monthly average *(IRC network only)*
* Outdegree prior month/past monthly average *(IRC network only)*
* Betweenness prior month/past monthly average *(IRC network only)*
* Effective size prior month/past monthly average *(IRC network only)*
* Efficiency prior month/past monthly average [(1/k(k-1)(1/dij + … + dik + … + djk), where k = n\_alters, d = distance between alters (d = 0 for disconnected alters)] *(IRC network only)*
* Alter churn (number of different alters from prior month) prior month/ past monthly average/cumulative *This doesn't say IRC network only, but I'm going to calculate it that way for now, just because it's simpler to do that way. If I need to, I can go back and do it with the full graph.*
* Effective size churn (number of different disconnected alters from prior month) prior month/past monthly average/cumulative *(IRC network only)*
* *(NEW)*N\_reported bugs prior month/past monthly average/cumulative **Note about this: it's hard to describe a debugger's “age” for the purposes of this variable. We could look at the first bug history event for the assignee, but this might actually be after the first time they reported a bug (since this is not encoded as a bug history event right now... maybe it should be? In any case, for expediency, I'm going to measure assignee age wrt the very first day of the first month overall, just for the purposes of this variable. I can go back and change this later if necessary. (see bugmonth\_variables:enrich\_assignee\_lastbandaid)**

Network (**These are all calculated with respect to the focal bug's vertex)**

* Constraint prior month/past monthly average
* Closeness prior month/past monthly average
* Clustering prior month/past monthly average
* Effective size prior month /past monthly average (number of disconnected debuggers)
* Efficiency prior month/past monthly average
* Effective size churn prior month/past monthly average/cumulative (number of different disconnected debuggers from prior month)

**Bug’s debuggers**

Debuggers

* N\_debuggers **(involved in history events with this bug)** prior month/past monthly average/cumulative
* Debugger churn (number of different debuggers contributing history events from prior month) prior month/ past monthly average/cumulative
* (NEW) N\_reported bugs prior month (average/variance)/past monthly average/cumulative **NOT CALCULABLE**

Bugs and chat

* N\_bugs to which debuggers contribute history events prior month (average/variance)/past monthly average/cumulative *Average of averages? WoG: yes*
* N\_history events focal bug prior month (average/variance)/ past monthly average/cumulative **(NOTE: Stop walking backwards when we hit the focal bug's reported date)**
* N\_history events other bugs prior month (average/variance)/ past monthly average/cumulative
* N\_IRC member links prior month (average/variance)/ past monthly average/cumulative *(presumably this is equal to indegree + outdegree in the irc network)*
* N\_IRC messages prior month (average/variance)/past monthly average/cumulative (directed/undirected) (*presumably meaning messages sent, rather than received)*

Network **(Note: I'm only averaging over debuggers who ever appear in the IRC network, rather than counting dbs who never appear as zeros throughout)**

* Constraint prior month (average/variance)/past monthly average *(IRC network only)*
* Closeness prior month (average/variance)/past monthly average *(IRC network only)*
* Clustering prior month (average/variance)/past monthly average *(IRC network only)*
* Indegree prior month (average/variance)/past monthly average *(IRC network only)*
* Outdegree prior month (average/variance)/past monthly average *(IRC network only)*
* Betweenness prior month (average/variance)/past monthly average *(IRC network only)*
* Effective size prior month (average/variance)/past monthly average *(IRC network only)*
* Efficiency prior month (average/variance)/past average *(IRC network only)*
* Alter churn (number of different alters from prior month) prior month (total/variance)/past monthly average/cumulative
* Effective size churn (number of different disconnected alters from prior month) prior month (total/variance)/past monthly average /cumulative *(IRC network only)* I'm assuming that a “disconnected alter” is an alter without connections to any other alters

**Bug context**

Bugs

* N\_unresolved bugs prior month **(# of bugs that were still unresolved at the end of the previous month)**
* N\_active bugs prior month (i.e., with history events) **(# of bugs that had at least one history event during the previous month)**
* N\_reported bugs prior month **(# of bugs whose reported date fell in the previous month)**
* N\_resolved/fixed bugs prior month **(# of bugs that went from unresolved to resolved sometime during the previous month) Note: for the sake of simplicity, we're using the single resolution date associated with each bug, and ignoring the possibility that a bug was resolved, then reopened, then resolved again.**

Debuggers, chat and events

* N\_debuggers prior month **(# of debuggers who sent or received IRC messages or touched a bug sometime during the previous month)**
* N\_IRC members prior month **(# of debuggers who sent or received IRC messages during the previous month)**
* N\_history events prior month
* N\_IRC chat messages prior month (directed/undirected)

Network

* Network diameter prior month *(IRC network only)*
* Network average path length prior month *(IRC network only)*
* Network density prior month *(IRC network only)*
* Network clustering prior month *(IRC network only)*

**Network Variable Definitions**

NOTE: Those marked with \* are computable via in igraph in Python. So, all but two; and I found code for effective size (URL below).

**\*Betweenness**

The betweenness of node i is: sum j≠i≠k (σjk (i) /σjk), where j and k are nodes other than i in the network, σjk is the number of shortest paths from j to k, and σjk (i) is the number of shortest paths from j to k that pass through i.

The betweenness value for each node i is normalized by dividing by the number of node pairs excluding i: (N-1)(N-2)/2, where N is the total number of nodes in the connected graph that to which i belongs.

**\*Closeness**

Closeness of a node is defined by the inverse of the average shortest path length and is computed as follows: 1/avg(L(n,m)), where L(n,m) is the length of the shortest path between two nodes n and m.

**\*Clustering Coefficient**

Clustering of a node i is defined as: 2ei/(ki(ki-1)), where ki is the number of connections of i and ei is the number of connected pairs between all connections of i.

**\*Constraint**

The constraint of node i's ego network V[i], is defined for directed and valued graphs,

C[i] = sum( [sum( p[i,j] + p[i,q] p[q,j], q in V[i], q≠i,j )]^2, j in V[i], j≠i)

for a graph of N nodes, where proportional tie strengths are defined as

p[i,j]=(a[i,j]+a[j,i]) / sum(a[i,k]+a[k,i], k in V[i], k≠ i),

a[i,j] are elements of A and the latter being the graph adjacency matrix. For isolated vertices, constraint is undefined.

**\*Density**

The density of a graph is the number of ties divided by the number of pairs. The number of pairs is [N(N-1)]/2.

**\*Diameter**

Diameter of a network is the length of the longest path between connected nodes.

**\*Indegree/outdegree**

In directed networks, the indegree of a node i is the number of incoming links and the outdegree is the number of outgoing links.

**Effective size**

Effective size of a node is the number of alters the node has, minus the average number of ties that each alter has to other alters: n – 2t/2, where n is the number of alters, and t is the number of ties among them.

\*Effective size is not a function in igraph, but here is code to compute it: [http://stackoverflow.com/questions/7969825/creating-new-measures-in-igraph/7970408#7970408](#7970408)

**Efficiency**

Efficiency of node i is the average of the inverse of the distances between the nodes linked **(directly)** with i:

[(1/k(k-1))sum(1/L(n,m), where k = number of nodes linked to i and L(n,m) is the length of the shortest path between two nodes n and m, that are linked to i.

I'm going to make the executive decision to leave this one out for now and only do it at the end if I have time.