Graph Measurements Part 1

Load libraries

```
library(dplyr)
library(igraph)
library(ggplot2)
library(tidygraph)
library(networkD3)
library(visNetwork)
library(knitr) # For table rendering
```

Graph object

```
df = read.table("./data/data.tsv", header = T)
veccol = c(rep("pink",5), rep("light blue",6))
g = graph_from_data_frame(df)
```

Graph measurements

- adhesion (edge connectivity) or cohesion (vertex connectivity)
- assortativity
- diameter
- girth
- radius
- mutual_count
- edge density
- automorphisms
- unconn_count
- size
- reciprocity
- mean_dist

Size

Counts the number of edges in the graph

```
gsize(g)
```

```
## [1] 32
```

Order

Order (number of vertices) of a graph

```
gorder(g)
```

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

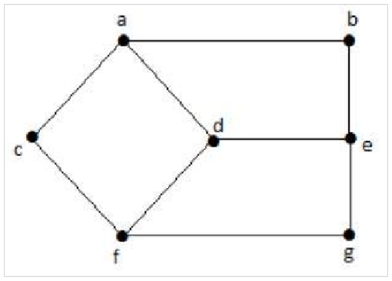
Eccentricity

The eccentricity of a vertex is its shortest path distance from the farthest other node in the graph.

Eccentricity of a Vertex

The maximum distance between a vertex to all other vertices is considered as the eccentricity of vertex.

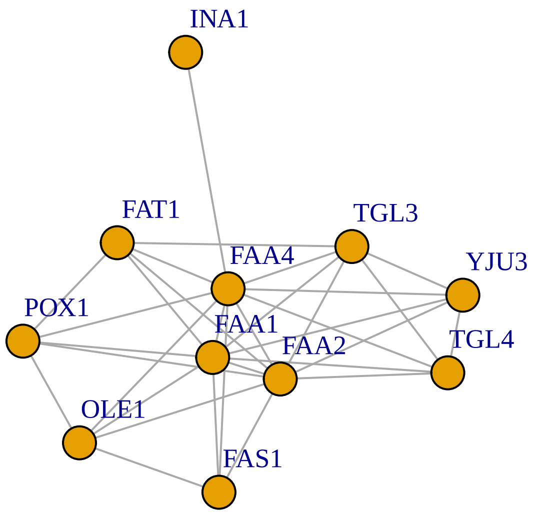
The distance from a particular vertex to all other vertices in the graph is taken and among those distances, the eccentricity is the highest of distances.



eccentricity of ‘a’ is 3.

- The distance from ‘a’ to ‘b’ is 1 (‘ab’),
- from ‘a’ to ‘c’ is 1 (‘ac’),
- from ‘a’ to ‘d’ is 1 (‘ad’),
- from ‘a’ to ‘e’ is 2 (‘ab’-‘be’) or (‘ad’-‘de’),
- from ‘a’ to ‘f’ is 2 (‘ac’-‘cf’) or (‘ad’-‘df’),
- from ‘a’ to ‘g’ is 3 (‘ac’-‘cf’-‘fg’) or (‘ad’-‘df’-‘fg’).

```
plot(g,edge.arrow.mode=0,vertex.label.cex=0.8,vertex.label.dist=3)
```



```
eccentricity(g)
```

```
## POX1 FAA1 TGL3 TGL4 FAA4 FAS1 FAA2 YJU3 OLE1 FAT1 INA1
##    2    2    2    2    1    2    2    2    2    2    2
```

Diameter

A network diameter is the longest geodesic distance (length of the shortest path between two nodes) in the network.

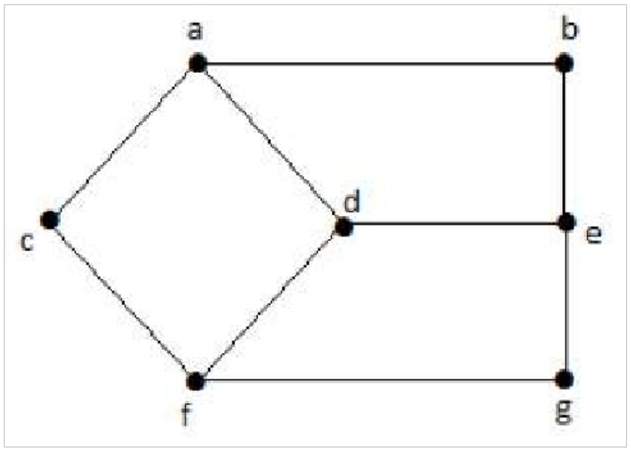
In igraph,

- diameter() returns the distance
- get_diameter() returns the nodes along the first found path of that distance.

Diameter of a Graph

The maximum eccentricity from all the vertices is considered as the diameter of the Graph G. The maximum among all the distances between a vertex to all other vertices is considered as the diameter of the Graph G.

In the graph, d(G) = 3; which is the maximum eccentricity.



eccentricity

e(b) = 3

e(c) = 3

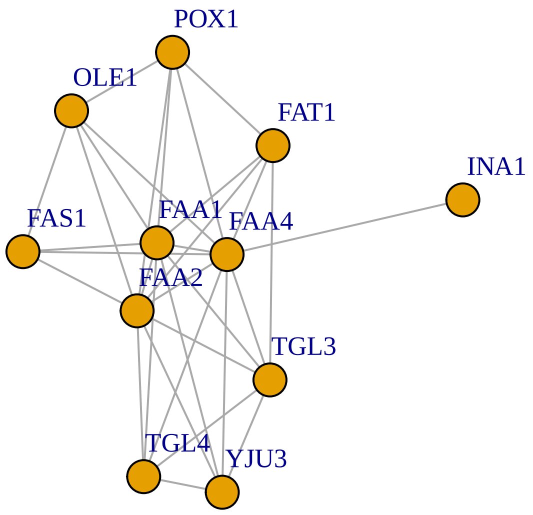
e(d) = 2

e(e) = 3

e(f) = 3

e(g) = 3

```
plot(g,edge.arrow.mode=0,vertex.label.cex=0.8,vertex.label.dist=3)
```



```
diameter(g)
```

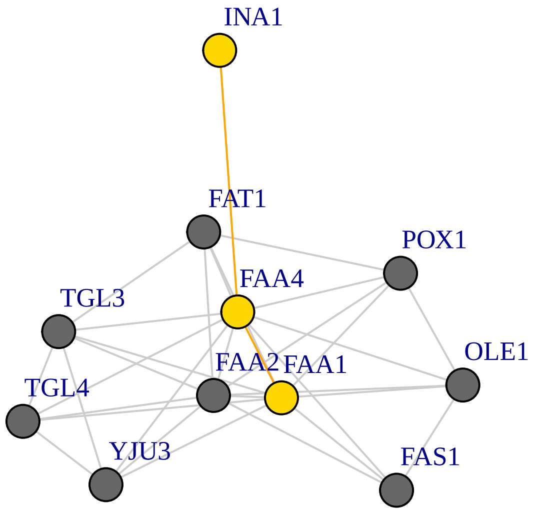
```
## [1] 2
```

```
diam <- get_diameter(g, directed=T)
diam
```

```
## + 3/11 vertices, named, from 043695d:
## [1] FAA1 FAA4 INA1
```

Color nodes along the diameter

```
vcol <- rep("gray40", vcount(g))
vcol[diam] <- "gold"
ecol <- rep("gray80", ecount(g))
ecol[E(g, path=diam)] <- "orange"
plot(g,
     vertex.color=vcol,
     edge.color=ecol,
     edge.arrow.mode=0,
     vertex.label.cex=0.8,
     vertex.label.dist=3
     )
```

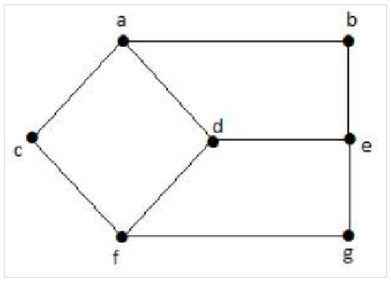


Radius

The eccentricity of a vertex is its shortest path distance from the farthest other node in the graph. The smallest eccentricity in a graph is called its radius. The eccentricity of a vertex is calculated by measuring the shortest distance from (or to) the vertex, to (or from) all vertices in the graph, and taking the maximum.

Radius of a Connected Graph

The minimum eccentricity from all the vertices is considered as the radius of the Graph G. The minimum among all the maximum distances between a vertex to all other vertices is considered as the radius of the Graph G



eccentricity

e(b) = 3

e(c) = 3

e(d) = 2

e(e) = 3

e(f) = 3

e(g) = 3

In the graph r(G) = 2, which is the minimum eccentricity for 'd'.

```
eccentricity(g)
```

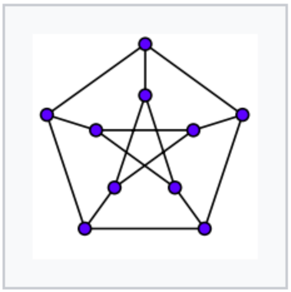
```
## POX1 FAA1 TGL3 TGL4 FAA4 FAS1 FAA2 YJU3 OLE1 FAT1 INA1
##    2    2    2    2    1    2    2    2    2    2    2
```

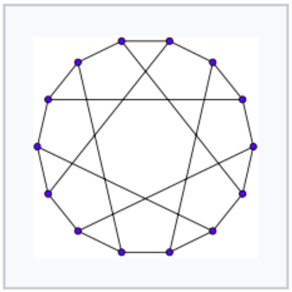
```
radius(g)
```

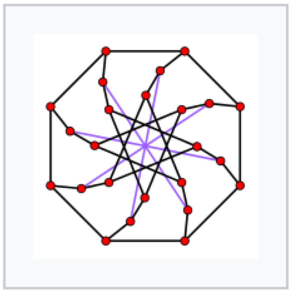
```
## [1] 1
```

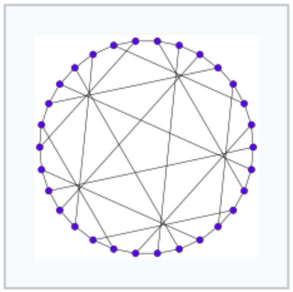
Girth

The girth of a graph is the length of the shortest circle in it. In graph theory, the girth of a graph is the length of a shortest cycle contained in the graph.[1] If the graph does not contain any cycles (i.e. it's an acyclic graph), its girth is defined to be infinity.[2] For example, a 4-cycle (square) has girth 4. A grid has girth 4 as well, and a triangular mesh has girth 3. A graph with girth four or more is triangle-free. https://en.wikipedia.org/wiki/Girth_(graph_theory)

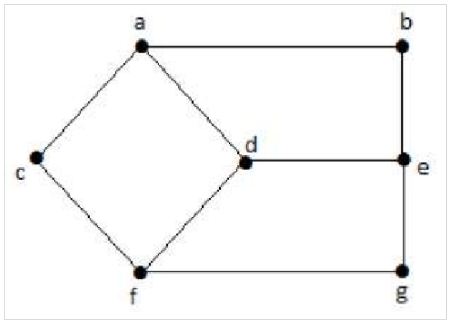

The Petersen graph has a girth of 5


The Heawood graph has a girth of 6


The McGee graph has a girth of 7


The Tutte–Coxeter graph (*Tutte eight cage*) has a girth of 8

Girth



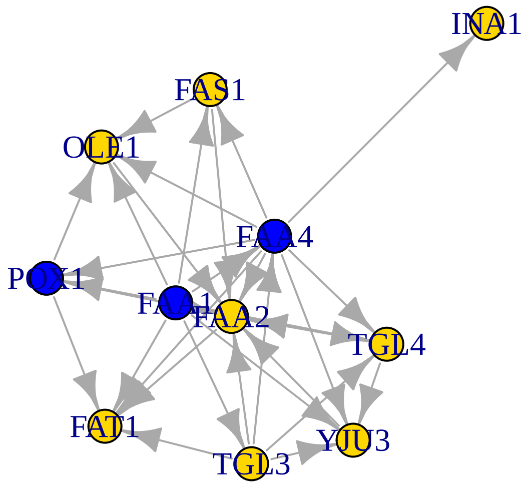
The number of edges in the shortest cycle of 'G' is called its Girth.

In the example graph, the Girth of the graph is 4, which we derived from the shortest cycle a-c-f-d-a or d-f-g-e-d or a-b-e-d-a.

```
set.seed(7)
gir = girth(g)
print(gir)
```

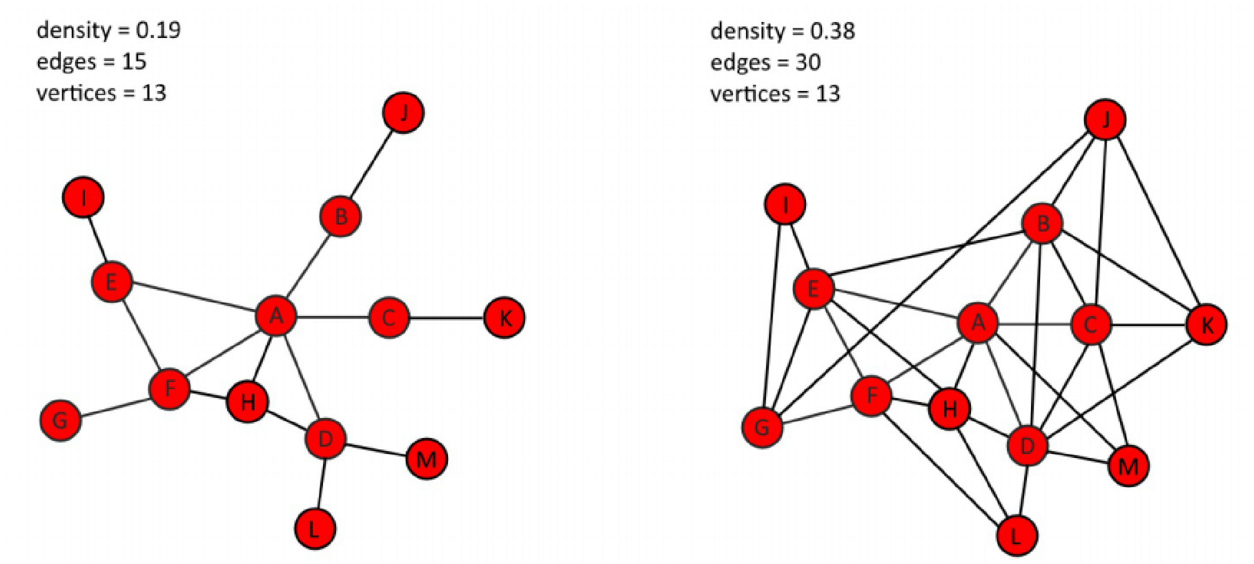
```
## $girth
## [1] 3
##
## $circle
## + 3/11 vertices, named, from 043695d:
## [1] FAA1 POX1 FAA4
```

```
veccol = rep("gold", vcount(g))
veccol[which(names(V(g)) %in% names(gir$circle))]="blue"
plot(g, vertex.color = veccol)
```



Edge density

The proportion of present edges from all possible edges in the network. This denotes how dense a graph is. A dense graph is a graph in which the number of edges is close to the maximal number of edges. The opposite, a graph with only a few edges, is a sparse graph.



```
# Ignore the edges connecting same node. i.e. edge between node A to A
# max edge: N*(N-1) where N is total nodes
# if e is total edge, then edge density is e/N*(N-1)
edge_density(g, loops=F)
```

```
## [1] 0.2909091
```

```
# Include the edges connecting same node. i.e. edge between node A to A
# max edge: N*N where N is total nodes
# if e is total edge, then edge density is e/N*N
edge_density(g, loops=T)
```

```
## [1] 0.2644628
```

```
ecount(g)/(vcount(g)*(vcount(g)-1))
```

```
## [1] 0.2909091
```

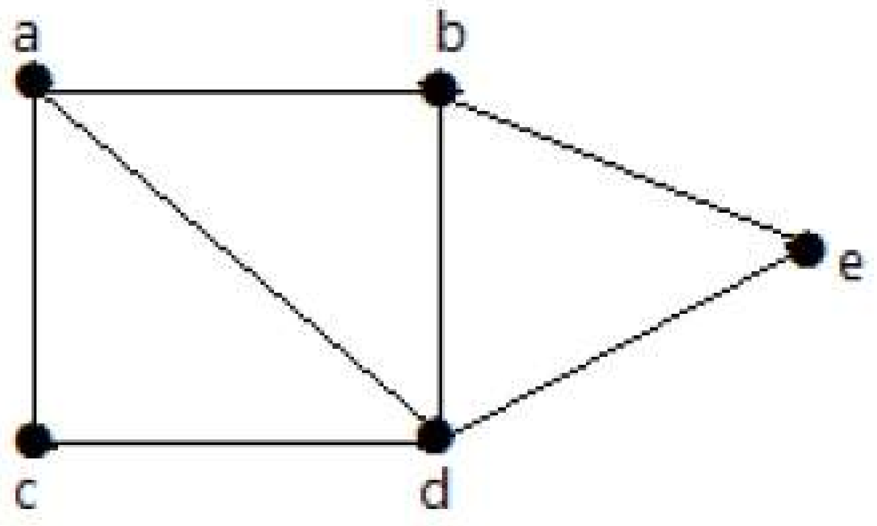
Group Adhesion or Cohesion

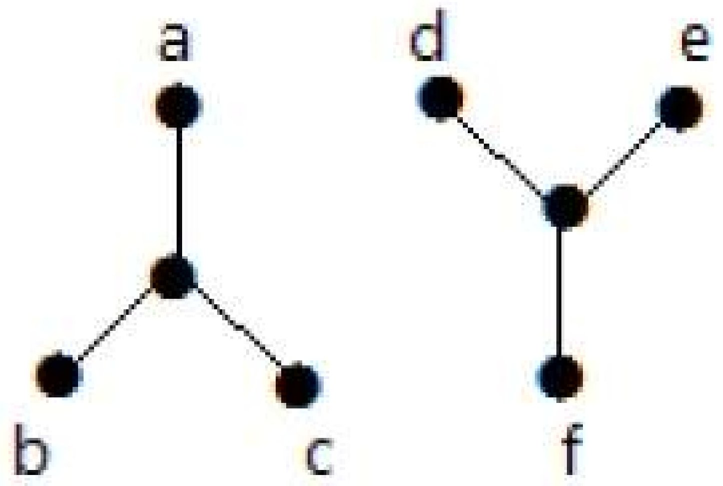
Group Adhesion (Edge connectivity) or Cohesion (Vertex connectivity)

What we call cohesion is the contribution made by (adding or subtracting) individual members of a group, together with their ties, to holding it together. What we call adhesion (edge cohesion) is the contribution made to holding a group together—keeping membership constant—by (adding or subtracting) ties between its members. https://onlinelibrary.wiley.com/doi/10.1111/0081-1750.00098

Connectivity of a graph: A graph is said to be connected if there is a path between every pair of vertex. From every vertex to any other vertex, there should be some path to traverse. That is called the connectivity of a graph. A graph with multiple disconnected vertices and edges is said to be disconnected. https://www.tutorialspoint.com/graph_theory/graph_theory_connectivity.htm

Connected graph





Disconnected graph

In the above example, traversing from vertex ‘a’ to vertex ‘f’ is not possible because there is no path between them directly or indirectly. Hence it is a disconnected graph.

Adhesion (Edge connectivity)

- The **edge connectivity of a pair of vertices** (source and target) is the minimum number of edges needed to remove to eliminate all (directed) paths from source to target.
- The **edge connectivity lambda of the graph G** is the minimum number of edges that need to be deleted, such that the graph G gets disconnected.

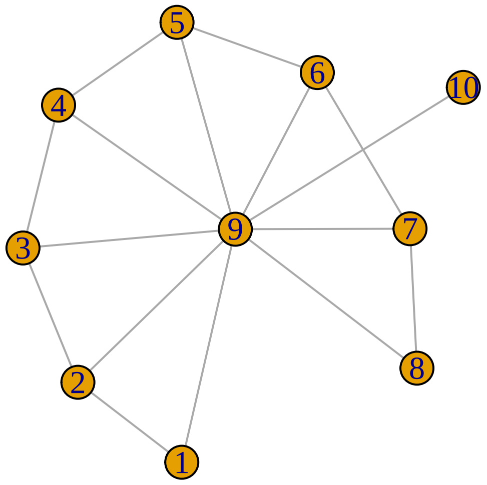
https://cp-algorithms.com/graph/edge_vertex_connectivity.html#toc-tgt-1

For example an already disconnected graph has an edge connectivity of 0

A connected graph with at least one bridge has an edge connectivity of 1

A connected graph with no bridges has an edge connectivity of at least 2.

```
g1 = graph.union(graph.ring(9), graph.star(10, c=9, mode="undirected"))
plot(g1)
```



```r
# we need to remove 2 edge (1-9 and 1-2) to make 1 disconnected from 6
edge_connectivity(g1, 1, 6)
```

```
## [1] 2
```

```r
# we need to remove 1 edge (9-10) to make 10 disconnected from 9
edge_connectivity(g1, 1, 10)
```

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

```r
# Therefore Graph's edge connectivity is 1
edge_connectivity(g1)
```

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

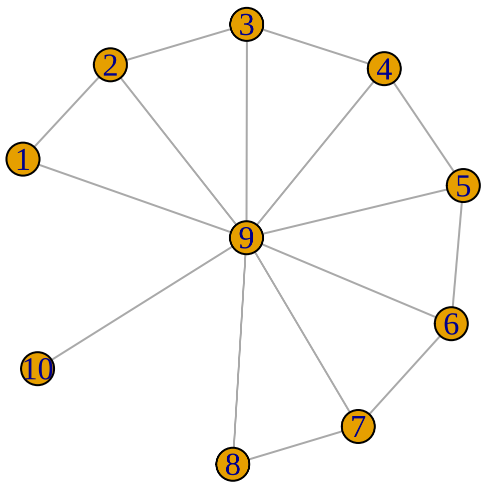
```r
# In our example, edge connectivity is 1 since minimum number of edges that need to be deleted to make
edge_connectivity(as.undirected(g1))
```

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

Cohesion (Vertex connectivity)

The vertex connectivity of two vertices (source and target) in a directed graph is the minimum number of vertices needed to remove from the graph to eliminate all (directed) paths from source to target. vertex_connectivity calculates this quantity if both the source and target arguments are given and they're not NULL. Note that this quantity is not defined if source and target are adjacent.

```
plot(g1)
```



```
vertex_connectivity(g1, 1, 6)
```

```
## [1] 2
```

```
vertex_connectivity(g1)
```

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