iGraph

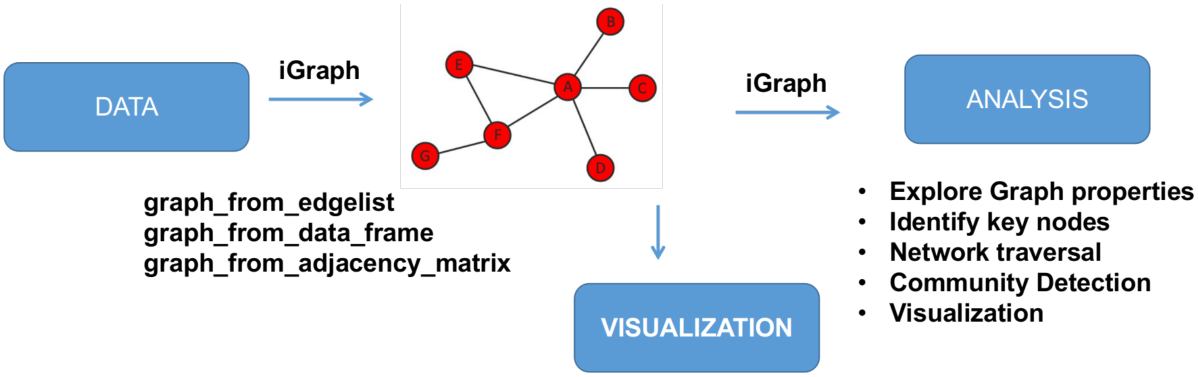
Install libraries

```
install.packages(c("igraph","dplyr","ggplot2","tidygraph","networkD3","visNetwork"))
```

Load libraries

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

Overall theme



DATA

iGraph

graph_from_edgelist
graph_from_data_frame
graph_from_adjacency_matrix

iGraph

ANALYSIS

- Explore Graph properties
- Identify key nodes
- Network traversal
- Community Detection
- Visualization

VISUALIZATION

Data to be used in the study

```
df = read.table("./data/data.tsv", header = T)
kable(head(df))
```

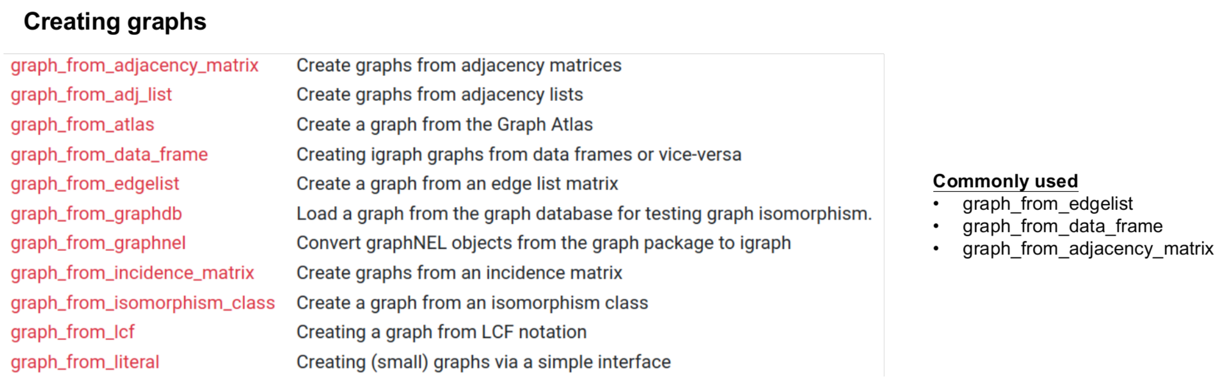
node1	node2
POX1	FAA2
FAA1	POX1
TGL3	YJU3

node1	node2
TGL4	YJU3
TGL3	TGL4
FAA4	POX1

```
dim(df)
```

```
## [1] 32  2
```

iGraph Functions to create graph



Creating graphs

graph_from_adjacency_matrix	Create graphs from adjacency matrices
graph_from_adj_list	Create graphs from adjacency lists
graph_from_atlas	Create a graph from the Graph Atlas
graph_from_data_frame	Creating igraph graphs from data frames or vice-versa
graph_from_edgelist	Create a graph from an edge list matrix
graph_from_graphdb	Load a graph from the graph database for testing graph isomorphism.
graph_from_graphnel	Convert graphNEL objects from the graph package to igraph
graph_from_incidence_matrix	Create graphs from an incidence matrix
graph_from_isomorphism_class	Create a graph from an isomorphism class
graph_from_lcf	Creating a graph from LCF notation
graph_from_literal	Creating (small) graphs via a simple interface

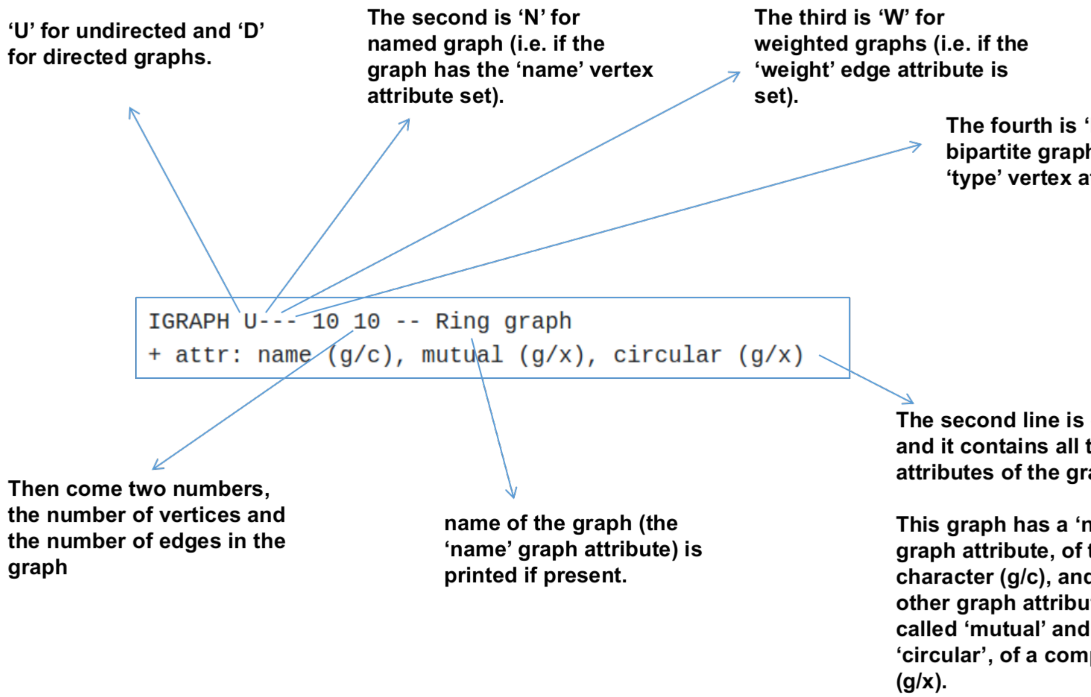
Commonly used

- graph_from_edgelist
- graph_from_data_frame
- graph_from_adjacency_matrix

Graph from a data frame

```
g = graph_from_data_frame(df)
print(g)
```

```
## IGRAPH fd8bfd4 DN-- 11 32 --
## + attr: name (v/c)
## + edges from fd8bfd4 (vertex names):
##  [1] POX1->FAA2 FAA1->POX1 TGL3->YJU3 TGL4->YJU3 TGL3->TGL4 FAA4->POX1
##  [7] POX1->FAT1 FAA1->FAT1 FAA4->FAS1 FAA1->FAS1 FAA4->FAT1 FAS1->FAA2
## [13] FAA4->OLE1 FAA1->FAA4 FAA1->OLE1 FAA2->FAT1 FAA1->TGL4 TGL3->FAA4
## [19] FAA1->TGL3 FAS1->OLE1 FAA1->YJU3 YJU3->FAA2 FAA4->YJU3 POX1->OLE1
## [25] FAA4->INA1 FAA1->FAA2 FAA4->FAA2 FAA4->TGL4 OLE1->FAA2 TGL3->FAT1
## [31] TGL4->FAA2 TGL3->FAA2
```



```
IGRAPH U--- 10 10 -- Ring graph
+ attr: name (g/c), mutual (g/x), circular (g/x)
```

‘U’ for undirected and ‘D’ for directed graphs.

The second is ‘N’ for named graph (i.e. if the graph has the ‘name’ vertex attribute set).

The third is ‘W’ for weighted graphs (i.e. if the ‘weight’ edge attribute is set).

The fourth is ‘
bipartite graph
‘type’ vertex a

Then come two numbers, the number of vertices and the number of edges in the graph

name of the graph (the ‘name’ graph attribute) is printed if present.

The second line is
and it contains all t
attributes of the gra

This graph has a ‘n
graph attribute, of t
character (g/c), and
other graph attribu
called ‘mutual’ and
‘circular’, of a comp
(g/x).

Example of graph object

The description of an igraph object starts with up to four letters:

- D or U, for a directed or undirected graph
- N for a named graph (where nodes have a name attribute)
- W for a weighted graph (where edges have a weight attribute)
- B for a bipartite (two-mode) graph (where nodes have a type attribute)

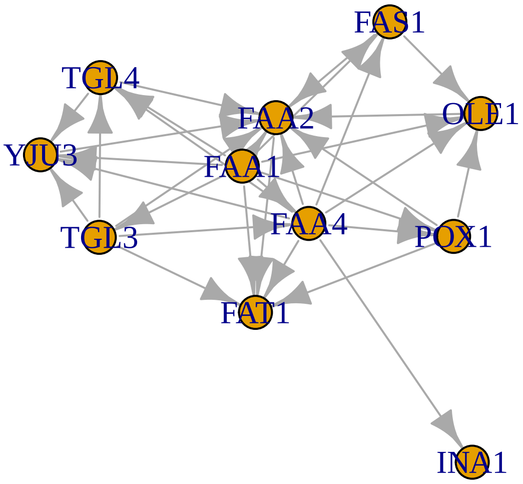
The description also lists node & edge attributes, for example:

- (g/c) - graph-level character attribute
- (v/c) - vertex-level character attribute
- (e/n) - edge-level numeric attribute

So we can see from first line, graph object g is Directed, Named with 11 nodes and 32 edges. Second line shows that nodes have one attribute (name of type character (g/c))

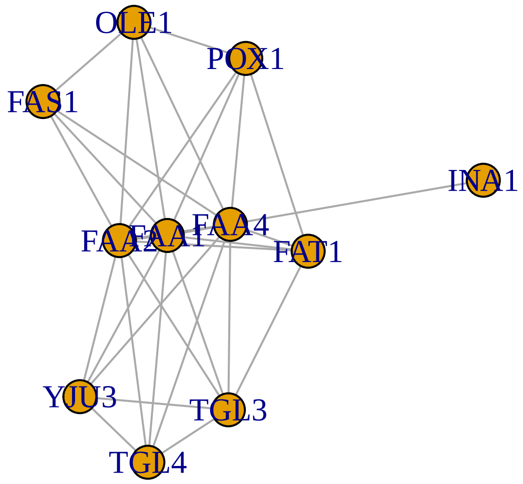
Plot graph using plot()

```
plot(g)
```



Make a undirected graph

```
g1 = graph_from_data_frame(df, directed = F);
plot(g1)
```



```
print(g1)
```

```
## IGRAPH feac5f3 UN-- 11 32 --
## + attr: name (v/c)
## + edges from feac5f3 (vertex names):
##  [1] POX1--FAA2 POX1--FAA1 TGL3--YJU3 TGL4--YJU3 TGL3--TGL4 POX1--FAA4
##  [7] POX1--FAT1 FAA1--FAT1 FAA4--FAS1 FAA1--FAS1 FAA4--FAT1 FAS1--FAA2
## [13] FAA4--OLE1 FAA1--FAA4 FAA1--OLE1 FAA2--FAT1 FAA1--TGL4 TGL3--FAA4
## [19] FAA1--TGL3 FAS1--OLE1 FAA1--YJU3 FAA2--YJU3 FAA4--YJU3 POX1--OLE1
## [25] FAA4--INA1 FAA1--FAA2 FAA4--FAA2 TGL4--FAA4 FAA2--OLE1 TGL3--FAT1
## [31] TGL4--FAA2 TGL3--FAA2
```

Node and Edge details

Node details

```
# Get nodes
# Get nodes
V(g)
```

```
## + 11/11 vertices, named, from fd8bfd4:
##  [1] POX1 FAA1 TGL3 TGL4 FAA4 FAS1 FAA2 YJU3 OLE1 FAT1 INA1
```

```r
# Total nodes
vcount(g)
```

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

```r
# Get vertices name
V(g)$name
```

```
##  [1] "POX1" "FAA1" "TGL3" "TGL4" "FAA4" "FAS1" "FAA2" "YJU3" "OLE1" "FAT1"
## [11] "INA1"
```

Edge details

```r
# Get edges
E(g)
```

```
## + 32/32 edges from fd8bfd4 (vertex names):
##  [1] POX1->FAA2 FAA1->POX1 TGL3->YJU3 TGL4->YJU3 TGL3->TGL4 FAA4->POX1
##  [7] POX1->FAT1 FAA1->FAT1 FAA4->FAS1 FAA1->FAS1 FAA4->FAT1 FAS1->FAA2
## [13] FAA4->OLE1 FAA1->FAA4 FAA1->OLE1 FAA2->FAT1 FAA1->TGL4 TGL3->FAA4
## [19] FAA1->TGL3 FAS1->OLE1 FAA1->YJU3 YJU3->FAA2 FAA4->YJU3 POX1->OLE1
## [25] FAA4->INA1 FAA1->FAA2 FAA4->FAA2 FAA4->TGL4 OLE1->FAA2 TGL3->FAT1
## [31] TGL4->FAA2 TGL3->FAA2
```

```r
# Edge count
ecount(g)
```

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

```r
# Edges
head( E(g)[[]] )
```

```
## + 6/32 edges from fd8bfd4 (vertex names):
## [1] POX1->FAA2 FAA1->POX1 TGL3->YJU3 TGL4->YJU3 TGL3->TGL4 FAA4->POX1
```

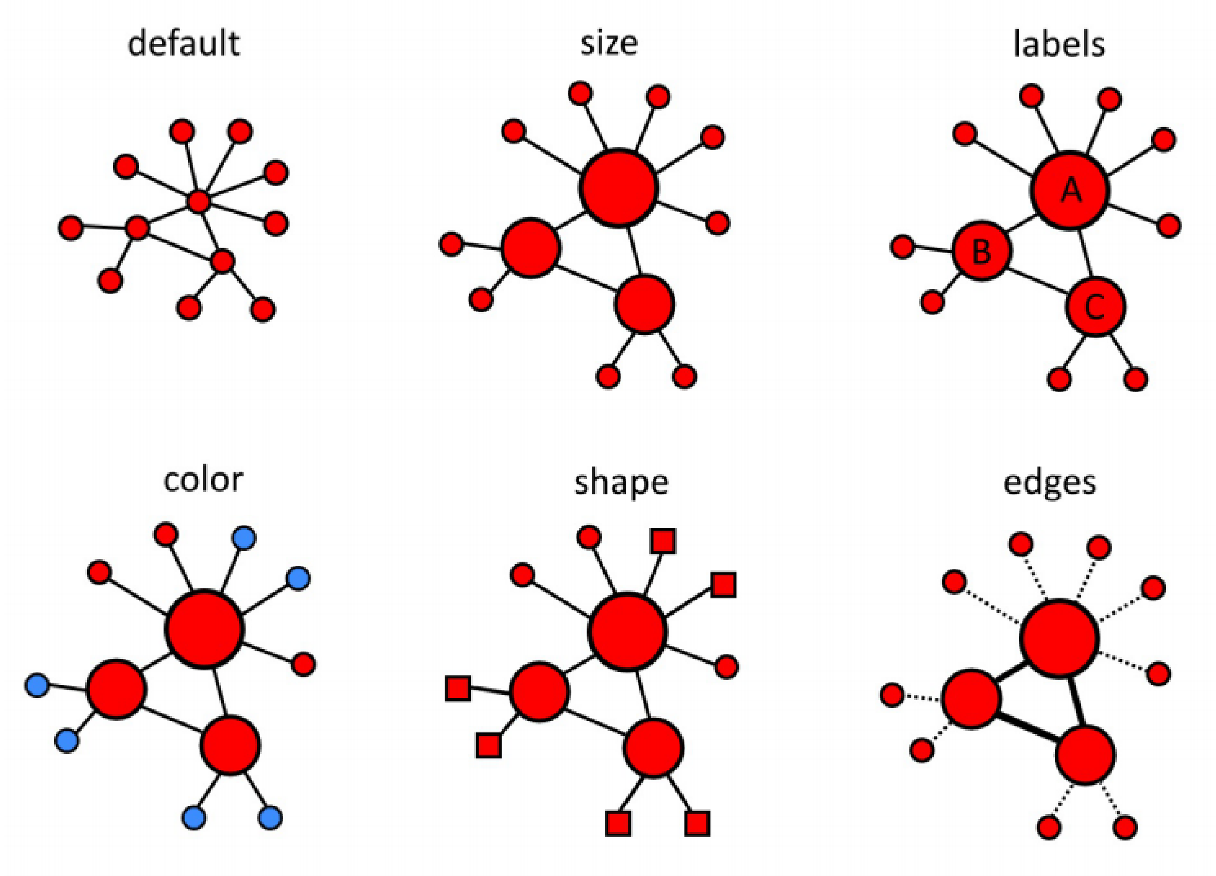
Plot Parameters

Vertex parameters

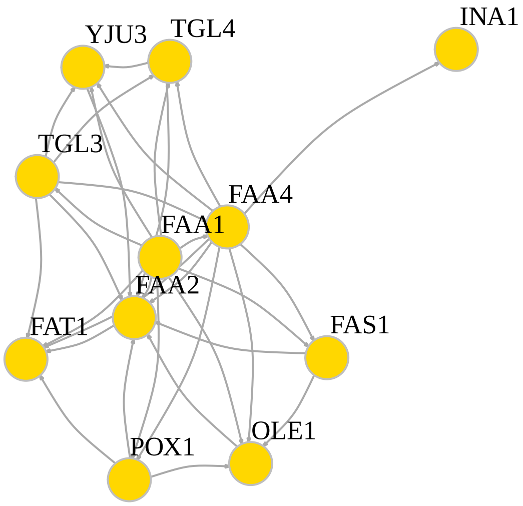
- **vertex.color** Node color
- **vertex.frame.color** Node border color
- **vertex.shape** One of “none”, “circle”, “square”, “csquare”, “rectangle”, “crectangle”, “vrectangle”, “pie”, “raster”, or “sphere”
- **vertex.size** Size of the node (default is 15)
- **vertex.size2** The second size of the node (e.g. for a rectangle)
- **vertex.label** Character vector used to label the nodes
- **vertex.label.family** Font family of the label (e.g.“Times”, “Helvetica”)
- **vertex.label.font** Font: 1 plain, 2 bold, 3, italic, 4 bold italic, 5 symbol
- **vertex.label.cex** Font size (multiplication factor, device-dependent)
- **vertex.label.dist** Distance between the label and the vertex
- **vertex.label.degree** The position of the label in relation to the vertex,where 0 right, “pi” is left, “pi/2” is below, and “-pi/2” is above

Edge parameters

- edge.color Edge color
- edge.width Edge width, defaults to 1
- edge.arrow.size Arrow size, defaults to 1
- edge.arrow.width Arrow width, defaults to 1
- edge.lty Line type, could be 0 or "blank", 1 or "solid", 2 or "dashed",3 or "dotted", 4 or "dotdash", 5 or "longdash", 6 or "twodash"
- edge.label Character vector used to label edges
- edge.label.family Font family of the label (e.g."Times", "Helvetica")
- edge.label.font Font: 1 plain, 2 bold, 3, italic, 4 bold italic, 5 symbol
- edge.label.cex Font size for edge labels
- edge.curved Edge curvature, range 0-1 (FALSE sets it to 0, TRUE to 0.5)
- arrow.mode Vector specifying whether edges should have arrows,
- possible values: 0 no arrow, 1 back, 2 forward, 3 both

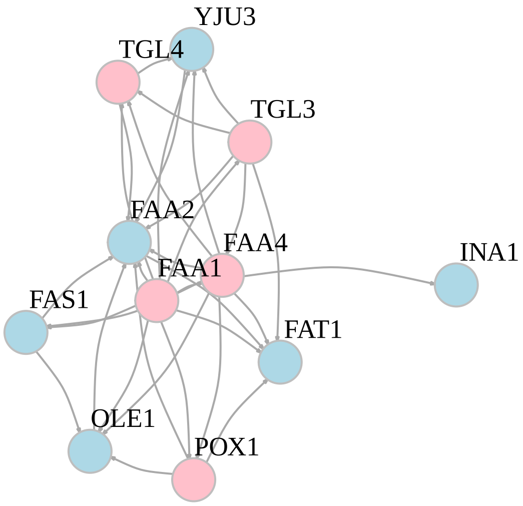


```
plot(g,
     edge.arrow.size=0.1,
     vertex.color="gold",
     vertex.size=20,
     vertex.frame.color="gray",
     vertex.label.color="black",
     vertex.label.cex=0.8,
     vertex.label.dist=3,
     edge.curved=0.2)
```



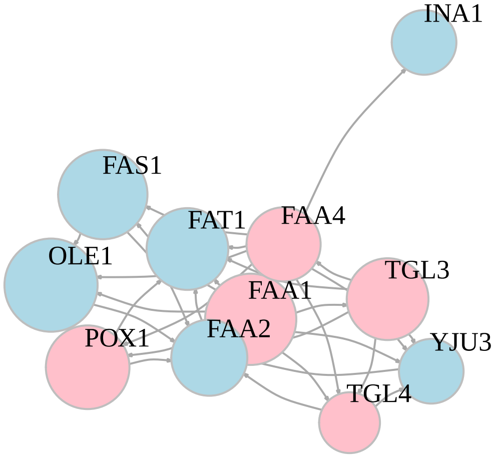
Color the vertices

```
# color vector of length = number of nodes
veccol = c(rep("pink",5), rep("light blue",6))
plot(g,
     edge.arrow.size=0.1,
     vertex.color=veccol,
     vertex.size=20,
     vertex.frame.color="gray",
     vertex.label.color="black",
     vertex.label.cex=0.8,
     vertex.label.dist=3,
     edge.curved=0.2)
```



Node size

```
# Randomly create a vector of size = total nodes
set.seed(123)
V(g)$size = sample(c(30:50),11, replace = T)
plot(g,
     edge.arrow.size=0.1,
     vertex.color=veccol,
     vertex.size=V(g)$size,
     vertex.frame.color="gray",
     vertex.label.color="black",
     vertex.label.cex=0.8,
     vertex.label.dist=3,
     edge.curved=0.2)
```



Add vertex attributes

```
set.seed(123)
total_mutations = sample(x = 0:30, size = vcount(g))
set_vertex_attr(graph = g, name = "Mutation", value = total_mutations)
```

```
## IGRAPH fd8bfd4 DN-- 11 32 --
## + attr: name (v/c), size (v/n), Mutation (v/n)
## + edges from fd8bfd4 (vertex names):
##  [1] POX1->FAA2 FAA1->POX1 TGL3->YJU3 TGL4->YJU3 TGL3->TGL4 FAA4->POX1
##  [7] POX1->FAT1 FAA1->FAT1 FAA4->FAS1 FAA1->FAS1 FAA4->FAT1 FAS1->FAA2
## [13] FAA4->OLE1 FAA1->FAA4 FAA1->OLE1 FAA2->FAT1 FAA1->TGL4 TGL3->FAA4
## [19] FAA1->TGL3 FAS1->OLE1 FAA1->YJU3 YJU3->FAA2 FAA4->YJU3 POX1->OLE1
## [25] FAA4->INA1 FAA1->FAA2 FAA4->FAA2 FAA4->TGL4 OLE1->FAA2 TGL3->FAT1
## [31] TGL4->FAA2 TGL3->FAA2
```

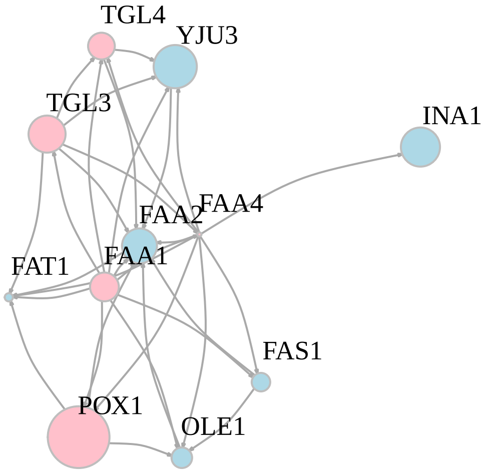
```
print(g)
```

```
## IGRAPH fd8bfd4 DN-- 11 32 --
## + attr: name (v/c), size (v/n)
## + edges from fd8bfd4 (vertex names):
##  [1] POX1->FAA2 FAA1->POX1 TGL3->YJU3 TGL4->YJU3 TGL3->TGL4 FAA4->POX1
##  [7] POX1->FAT1 FAA1->FAT1 FAA4->FAS1 FAA1->FAS1 FAA4->FAT1 FAS1->FAA2
## [13] FAA4->OLE1 FAA1->FAA4 FAA1->OLE1 FAA2->FAT1 FAA1->TGL4 TGL3->FAA4
## [19] FAA1->TGL3 FAS1->OLE1 FAA1->YJU3 YJU3->FAA2 FAA4->YJU3 POX1->OLE1
## [25] FAA4->INA1 FAA1->FAA2 FAA4->FAA2 FAA4->TGL4 OLE1->FAA2 TGL3->FAT1
## [31] TGL4->FAA2 TGL3->FAA2
```

```
# List of vertex attributes
vertex_attr(g)
```

```
## $name
##  [1] "POX1" "FAA1" "TGL3" "TGL4" "FAA4" "FAS1" "FAA2" "YJU3" "OLE1" "FAT1"
## [11] "INA1"
##
## $size
##  [1] 44 48 43 32 39 47 40 34 49 43 34
```

```
plot(g,
     edge.arrow.size=0.1,
     vertex.color=veccol,
     vertex.size=total_mutations,
     vertex.frame.color="gray",
     vertex.label.color="black",
     vertex.label.cex=0.8,
     vertex.label.dist=3,
     edge.curved=0.2)
```


TGL4
YJU3
TGL3
INA1
FAA2
FAA4
FAA1
FAT1
FAS1
POX1
OLE1