Network Coding Examples for Homework 6 (STA 352/652; MTH 359)

```
library(igraph)
```

Graph setup

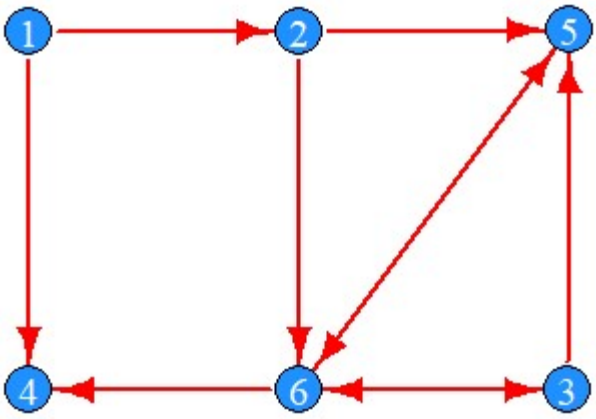
```
gd<-graph.formula(1-+2,1-+4,2-+5,3-+5,3++6,2-+6,6-+4,5++6)

igraph.options(vertex.color="dodgerblue",vertex.size=20,
vertex.label.cex=1.25,vertex.label.color="white",
edge.color="red",edge.arrow.size=1,edge.width=2)
```

Reordering nodes

```
A<-get.adjacency(gd)
A<-A[order(rownames(A)),order(rownames(A))]
gd<-graph_from_adjacency_matrix(A)
```

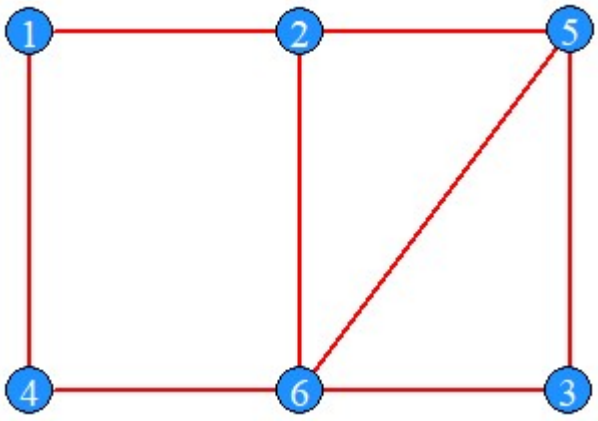
Moving nodes and plotting

```
tkplot(gd)
L<-tk_coords(6,norm=TRUE)*1.5;L[,1]<-L[,1]*1.5
plot(gd,layout=L,rescale=FALSE)sss
```



The undirected network

```
gu<-as.undirected(gd)
plot(gu,layout=L,rescale=FALSE)
```



Network Coding Examples for Homework 6 (STA 352/652; MTH 359)

The undirected network

```
g<-gu
```

The number of edges and nodes

```
vcount(g)
[1] 6
ecount(g)
[1] 8
V(g)
+ 6/6 vertices, named, from aa46fc7:
[1] 1 2 3 4 5 6
E(g)
+ 8/8 edges from aa46fc7 (vertex names):
[1] 1--2 1--4 2--5 3--5 2--6 3--6 4--6 5--6
```

Degrees

```
degree(g)
1 2 3 4 5 6
2 3 2 2 3 4
sort(degree(g))
1 3 4 2 5 6
2 2 2 3 3 4
degree_distribution(g)
[1] 0.0000000 0.0000000 0.5000000 0.3333333 0.1666667
```

Average degrees of near neighbours

```
knn(g)$knn
  1   2   3   4   5   6
2.5 3.0 3.5 3.0 3.0 2.5
neighborhood(g,order=1,nodes=1,mindist=1)
[[1]]
+ 2/6 vertices, named, from aa46fc7:
[1] 2 4
lapply(neighborhood(g,order=1,mindist=1),degree,g=g)
[[1]]
2 4
3 2
[[2]]
1 5 6
2 3 4
[[3]]
5 6
3 4
...
[[6]]
2 3 4 5
3 2 2 3

sapply(lapply(neighborhood(g,order=1,mindist=1),degree,g=g), mean)
[1] 2.5 3.0 3.5 3.0 3.0 2.5
```

Network Coding Examples for Homework 6 (STA 352/652; MTH 359)

Shortest path distances and diameter

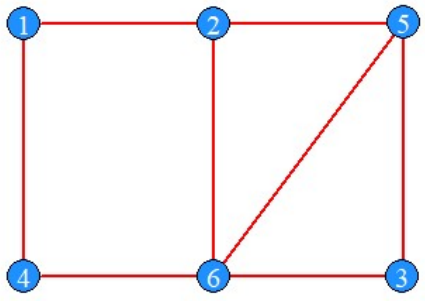
```
distances(g)
  1 2 3 4 5 6
1 0 1 3 1 2 2
2 1 0 2 2 1 1
3 3 2 0 2 1 1
4 1 2 2 0 2 1
5 2 1 1 2 0 1
6 2 1 1 1 1 0
diameter(g)
[1] 3
which(shortest.paths(g)==diameter(g),arr.ind=TRUE)
  row col
3   3   1
1   1   3
```

Closeness centralities

```
closeness(g)
        1         2         3         4         5         6
0.1111111 0.1428571 0.1111111 0.1250000 0.1428571 0.1666667

apply(distances(g),1,sum)
1 2 3 4 5 6
9 7 9 8 7 6
1/apply(distances(g),1,sum)
        1         2         3         4         5         6
0.1111111 0.1428571 0.1111111 0.1250000 0.1428571 0.1666667
```

Betweenness

```
betweenness(g)
        1         2         3         4         5         6
0.5000000 2.1666667 0.0000000 0.8333333 0.8333333 3.6666667

edge_betweenness(g)
[1] 3.666667 2.333333 2.833333 1.833333 2.833333 3.166667 4.333333 2.000000
E(g)
+ 8/8 edges from 77cc4d2 (vertex names):
[1] 1--2 1--4 2--5 3--5 2--6 3--6 4--6 5--6
```



Network Coding Examples for Homework 6 (STA 352/652; MTH 359)

Cliques

```
table(sapply(cliques(g),length))
1 2 3
6 8 2
cliques(g,min=3,max=3)
[[1]]
+ 3/6 vertices, named, from 77cc4d2:
[1] 3 5 6
[[2]]
+ 3/6 vertices, named, from 77cc4d2:
[1] 2 5 6
```

Adding an edge

```
gtemp<-g
gtemp[2,3]<-1
cliques(gtemp,min=4,max=4)
[[1]]
+ 4/6 vertices, named, from 0ba0c18:
[1] 2 3 5 6

max_cliques(gtemp,min=2,max=2)
[[1]]
+ 2/6 vertices, named, from 0ba0c18:
[1] 1 4
[[2]]
+ 2/6 vertices, named, from 0ba0c18:
[1] 1 2
[[3]]
+ 2/6 vertices, named, from 0ba0c18:
[1] 4 6
```

Edge density

```
edge_density(g)
[1] 0.5333333
ecount(g)/choose(vcount(g),2)
[1] 0.5333333
```

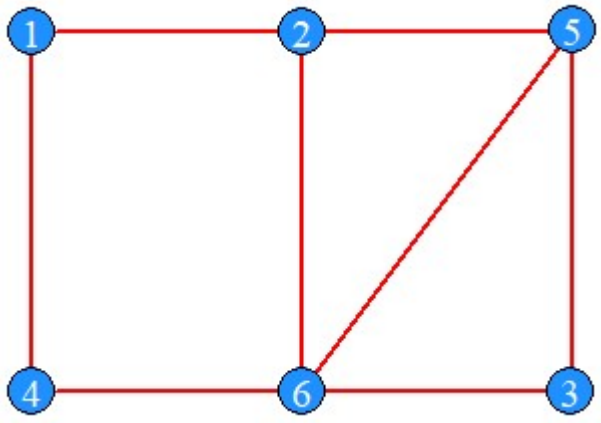
Clustering coefficients

```
transitivity(g)
[1] 0.4

choose(degree(g),2)
1 2 3 4 5 6
1 3 1 1 3 6

sum(choose(degree(g),2))
[1] 15

triad_census(g)
 [1] 1 0 8 0 0 0 0 0 0 0 9 0 0 0 0 2
```

Network Coding Examples for Homework 6 (STA 352/652; MTH 359)

#[A,B,C, the empty graph. (1)
#A<->B, C (8)
#A<->B<->C. (9)
#A<->B<->C, A<->C, the complete graph. (2)]

```
6/15
[1] 0.4
```

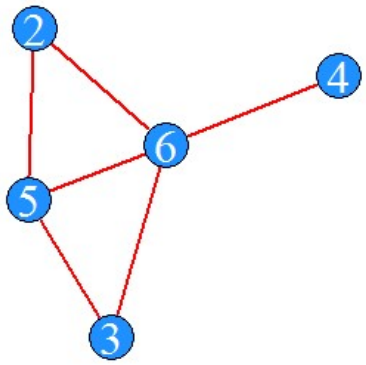



```
transitivity(g,type="local")
```

```
        1         2         3         4         5         6
0.0000000 0.3333333 1.0000000 0.0000000 0.6666667 0.3333333
```

An ego graph

```
ge<-make_ego_graph(gu,nodes=6)[[1]]
(ecount(ge)-vcount(ge)+1)/choose(vcount(ge)-1,2)
[1] 0.3333333
plot(gE)
```



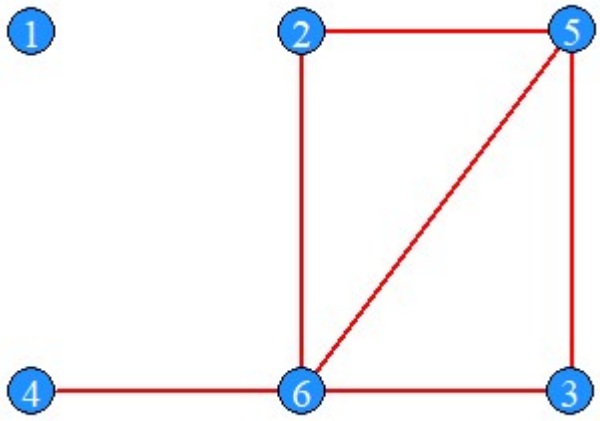
Network Coding Examples for Homework 6 (STA 352/652; MTH 359)

Vertex connectivity

```
vertex_connectivity(g)
[1] 2
components(g-vertices(c(2,6)))
$membership
1 3 4 5
1 2 1 2
$csize
[1] 2 2
$no
[1] 2
```

Edge connectivity

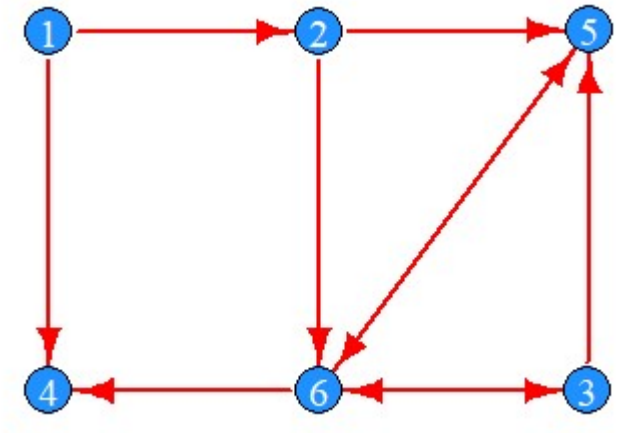
```
edge_connectivity(g)
[1] 2
E(g)
+ 8/8 edges from 77cc4d2 (vertex names):
[1] 1--2 1--4 2--5 3--5 2--6 3--6 4--6 5–6
components(g-edges(1:2))
$membership
1 2 3 4 5 6
1 2 2 2 2 2
$csize
[1] 1 5
$no
[1] 2

plot(g-edges(1:2),layout=L,rescale=FALSE)
```



Network Coding Examples for Homework 6 (STA 352/652; MTH 359)

The undirected network

```
plot(gd,layout=L,rescale=FALSE)
```



Directed dyad types

```
dyad.census(gd)
$mut
[1] 2
$asym
[1] 6
$null
[1] 7
```

Directed triad types

Triad_census {igraph} *Triad census, subgraphs with three vertices*

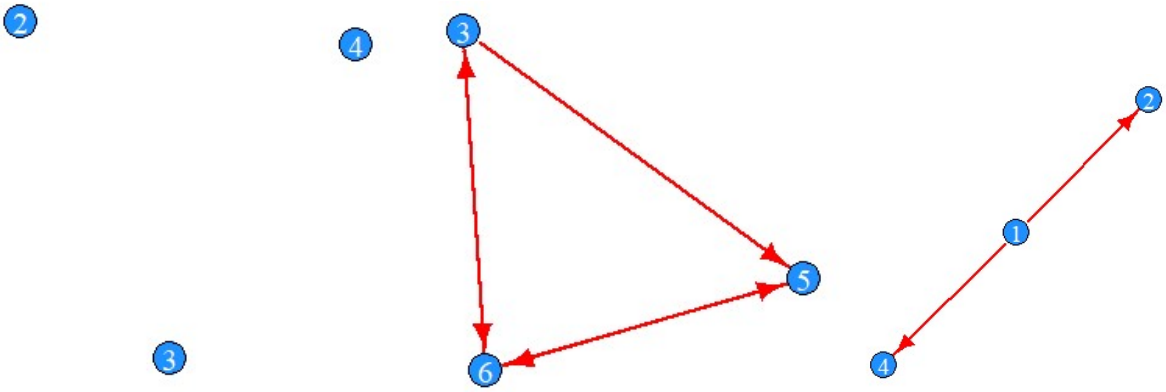
Description. This function counts the different subgraphs of three vertices in a graph. Triad census was defined by David and Leinhardt (Davis, J.A. and Leinhardt, S. (1972). The Structure of Positive Interpersonal Relations in Small Groups. In J. Berger (Ed.), Sociological Theories in Progress, Volume 2, 218-251. Boston: Houghton Mifflin.). Every triple of vertices (A, B, C) are classified into the 16 possible states: 003 -- A,B,C, the empty graph; 012 -- A->B, C, the graph with a single directed edge. 102 A<->B, C, the graph with a mutual connection between two vertices. 021D, A<-B->C, the out-star. 021U A->B<-C, the in-star. 021C A->B->C, directed line. 111D A<->B<-C. 111U A<->B->C. 030T A->B<-C, A->C. 030C A<-B<-C, A->C. 201 A<->B<->C. 120D A<-B->C, A<->C. 120U A->B<-C, A<->C. 120C A->B->C, A<->C. 210 A->B<->C, A<->C. 300 A<->B<->C, A<->C, the complete graph.

```
triad_census(gd)
[1] 1 6 2 1 2 3 1 2 0 0 0 1 0 0 1 0
```

Network Coding Examples for Homework 6 (STA 352/652; MTH 359)

Plotting subgraphs

```
plot(induced.subgraph(gd, c(2,3,4)))
plot(induced.subgraph(gd, c(3,5,6)))
plot(induced.subgraph(gd, c(1,2,4)))
```



Reciprocity

```
> reciprocity(g2,mode="default")
[1] 0.4
```

Strongly connected components

```
components(g2,mode="strong")
$membership
1 2 4 5 3 6
1 2 4 3 3 3
$csize
[1] 1 1 3 1
$no
[1] 4

sort(components(g2,mode="strong")$membership)
1 2 5 3 6 4
1 2 3 3 3 4
```