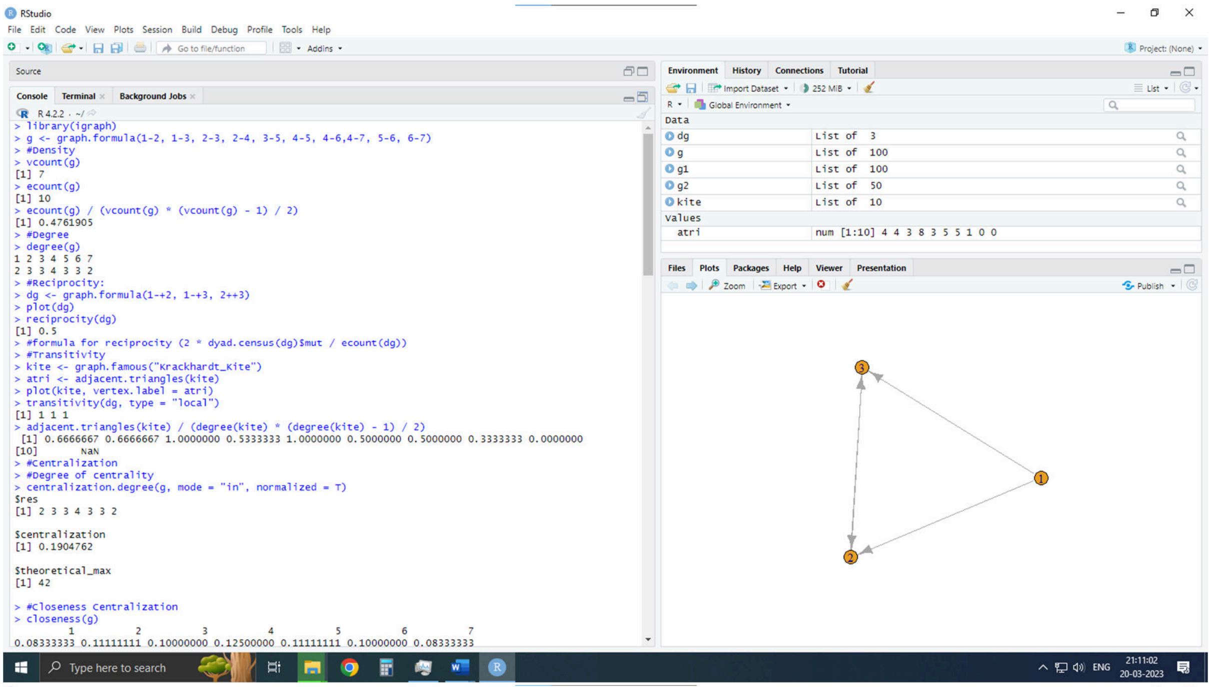
Practical 3: Compute the following node level measures: (i) Density; (ii) Degree; (iii) Reciprocity; (iv) Transitivity; (v) Centralization; (vi) Clustering.

Code:

```
# Load the igraph library
library(igraph)

# Create a graph object 'g'
g <- graph.formula(1-2, 1-3, 2-3, 2-4, 3-5, 4-5, 4-6,4-7, 5-6, 6-7)

# Density
# Number of vertices
vcount(g)

# Number of edges
ecount(g)

# Density of the graph
ecount(g) / (vcount(g) * (vcount(g) - 1) / 2)

# Degree
degree(g)

# Reciprocity:
# Create a directed graph 'dg'
dg <- graph.formula(1-+2, 1-+3, 2++3)

# Plot the directed graph 'dg'
plot(dg)

# Reciprocity of the directed graph 'dg'
reciprocity(dg)

# Formula for reciprocity
(2 * dyad.census(dg)$mut / ecount(dg))

# Transitivity
# Create a famous graph 'kite'
kite <- graph.famous("Krackhardt_Kite")

# Find the adjacent triangles in the 'kite' graph
atri <- adjacent.triangles(kite)

# Plot the 'kite' graph with vertex labels as adjacent triangles
plot(kite, vertex.label = atri)
```

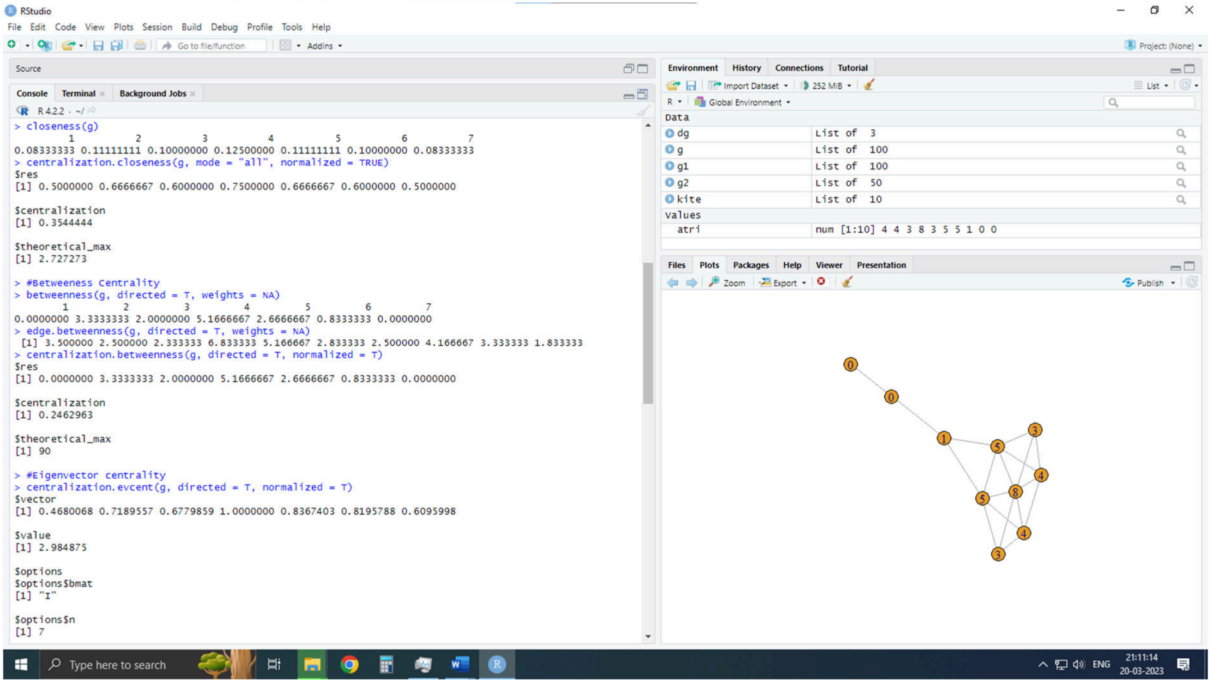
```r
# Local transitivity of the directed graph 'dg'
transitivity(dg, type = "local")

# Proportion of adjacent triangles to all possible triangles in the 'kite'
graph
adjacent.triangles(kite) / (degree(kite) * (degree(kite) - 1) / 2)

# Centralization
# Degree of centrality
centralization.degree(g, mode = "in", normalized = T)

# Closeness Centralization
closeness(g)
centralization.closeness(g, mode = "all", normalized = TRUE)

# Betweeness Centrality
betweenness(g, directed = T, weights = NA)
edge.betweenness(g, directed = T, weights = NA)
centralization.betweenness(g, directed = T, normalized = T)

# Eigenvector centrality
centralization.evcent(g, directed = T, normalized = T)

# Clustering
# Create two graphs 'g1' and 'g2'
g2 <- barabasi.game(50, p = 2, directed = F)
g1 <- watts.strogatz.game(1, size = 100, nei = 5, p = 0.05)

# Combine the two graphs 'g1' and 'g2'
g <- graph.union(g1, g2)

# Simplify the combined graph 'g'
g <- simplify(g)

# Plot the simplified graph 'g'
plot(g)
```

OUTPUT



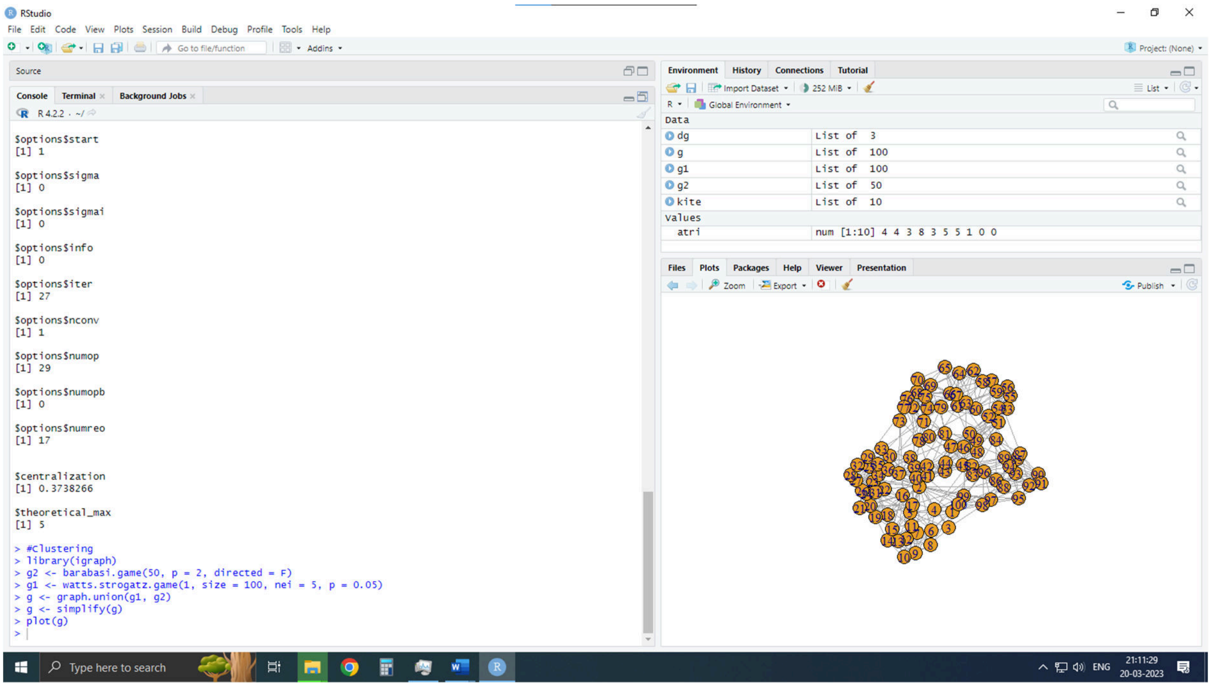
```
> library(igraph)
> g <- graph.formula(1-2, 1-3, 2-3, 2-4, 3-5, 4-5, 4-6,4-7, 5-6, 6-7)
> #Density
> vcount(g)
[1] 7
> ecount(g)
[1] 10
> ecount(g) / (vcount(g) * (vcount(g) - 1) / 2)
[1] 0.4761905
> #Degree
> degree(g)
1 2 3 4 5 6 7
2 3 3 4 3 3 2
> #Reciprocity:
> dg <- graph.formula(1-+2, 1-+3, 2++3)
> plot(dg)
> reciprocity(dg)
[1] 0.5
> #formula for reciprocity (2 * dyad.census(dg)$mut / ecount(dg))
> #Transitivity
> kite <- graph.famous("Krackhardt_Kite")
> atri <- adjacent.triangles(kite)
> plot(kite, vertex.label = atri)
> transitivity(dg, type = "local")
[1] 1 1 1
> adjacent.triangles(kite) / (degree(kite) * (degree(kite) - 1) / 2)
 [1] 0.6666667 0.6666667 1.0000000 0.5333333 1.0000000 0.5000000 0.5000000 0.3333333 0.0000000
[10]       NaN
> #Centralization
> #Degree of centrality
> centralization.degree(g, mode = "in", normalized = T)
$res
[1] 2 3 3 4 3 3 2

$centralization
[1] 0.1904762

$theoretical_max
[1] 42

> #Closeness Centralization
> closeness(g)
         1          2          3          4          5          6          7 
0.08333333 0.11111111 0.10000000 0.12500000 0.11111111 0.10000000 0.08333333
```

```
Data
dg      List of  3
g       List of  100
g1      List of  100
g2      List of  50
kite    List of  10
Values
atri    num [1:10] 4 4 3 8 3 5 5 1 0 0
```



```
> closeness(g)
         1          2          3          4          5          6          7 
0.08333333 0.11111111 0.10000000 0.12500000 0.11111111 0.10000000 0.08333333
> centralization.closeness(g, mode = "all", normalized = TRUE)
$res
[1] 0.5000000 0.6666667 0.6000000 0.7500000 0.6666667 0.6000000 0.5000000

$centralization
[1] 0.3544444

$theoretical_max
[1] 2.727273

> #Betweeness Centrality
> betweenness(g, directed = T, weights = NA)
        1         2         3         4         5         6         7 
0.0000000 3.3333333 2.0000000 5.1666667 2.6666667 0.8333333 0.0000000
> edge.betweenness(g, directed = T, weights = NA)
 [1] 3.500000 2.500000 2.333333 6.833333 5.166667 2.833333 2.500000 4.166667 3.333333 1.833333
> centralization.betweenness(g, directed = T, normalized = T)
$res
[1] 0.0000000 3.3333333 2.0000000 5.1666667 2.6666667 0.8333333 0.0000000

$centralization
[1] 0.2462963

$theoretical_max
[1] 90

> #Eigenvector centrality
> centralization.evcent(g, directed = T, normalized = T)
$vector
[1] 0.4680068 0.7189557 0.6779859 1.0000000 0.8367403 0.8195788 0.6095998

$value
[1] 2.984875

$options
$options$bmat
[1] "I"

$options$n
[1] 7
```

```
Data
dg      List of  3
g       List of  100
g1      List of  100
g2      List of  50
kite    List of  10
Values
atri    num [1:10] 4 4 3 8 3 5 5 1 0 0
```



```
$options$start
[1] 1

$options$sigma
[1] 0

$options$sigmai
[1] 0

$options$info
[1] 0

$options$iter
[1] 27

$options$nconv
[1] 1

$options$numop
[1] 29

$options$numopb
[1] 0

$options$numreo
[1] 17


$centralization
[1] 0.3738266

$theoretical_max
[1] 5

> #Clustering
> library(igraph)
> g2 <- barabasi.game(50, p = 2, directed = F)
> g1 <- watts.strogatz.game(1, size = 100, nei = 5, p = 0.05)
> g <- graph.union(g1, g2)
> g <- simplify(g)
> plot(g)
> 
```

```
Data
dg          List of  3
g           List of  100
g1          List of  100
g2          List of  50
kite        List of  10
Values
  atri      num [1:10] 4 4 3 8 3 5 5 1 0 0
```