

例題 5.5

R では、ネットワーク解析を行うためのパッケージ `igraph` が提供されている。このパッケージを用いて、図 5.21 で示したような 3 種類のネットワークを描画せよ。また、それぞれのネットワークについて、図 5.24 で示した 3 種類の中心性を調べよ。

解答例 (図 5.21 で示したような 3 種類のネットワークの描画)

igraph パッケージのインストールとロードは、以下のコマンドで実行可能です。

```
install.packages("igraph")  
library(igraph)
```

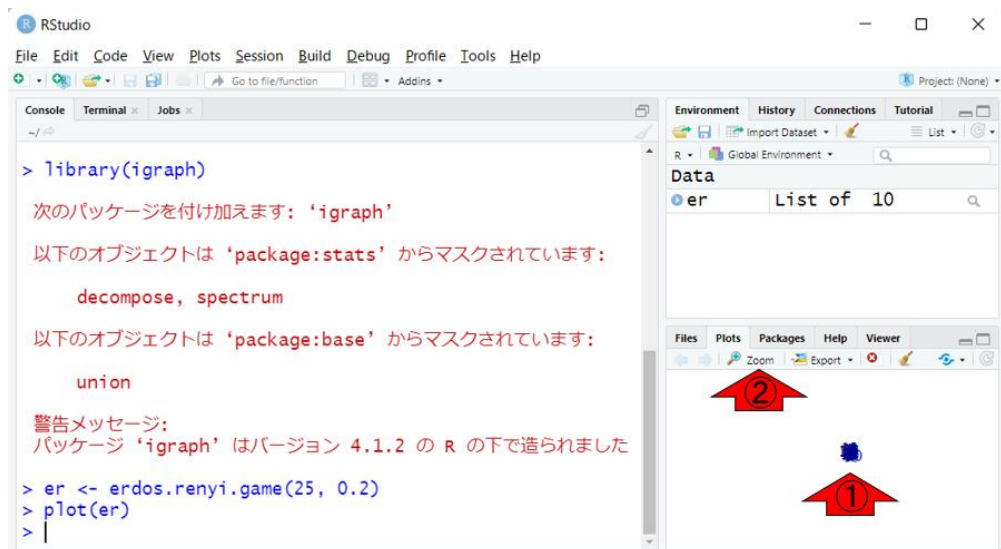
(a) ランダムネットワーク (エルデシュ・レニイモデル) の描画

例えば、頂点数 25、辺確率 0.2 のランダムネットワークは、`erdos.renyi.game` 関数を用いて以下のコマンドで作成することができます。

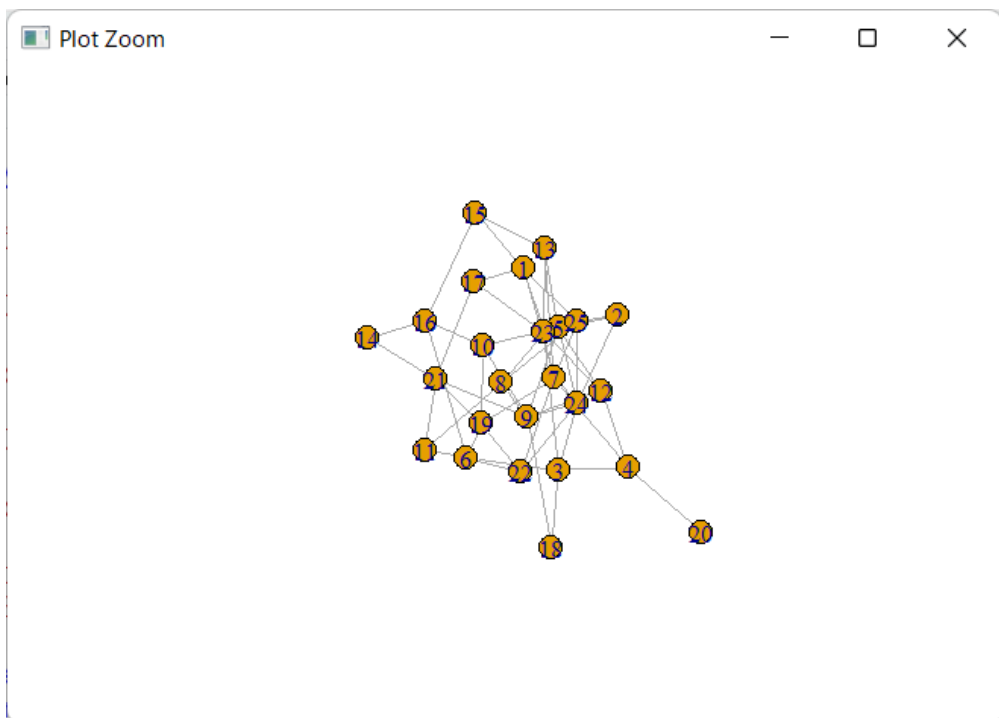
```
er <- erdos.renyi.game(25, 0.2)  
plot(er)
```

最初のコマンドは、ランダムネットワークを頂点数 25、辺確率 0.2 という情報を入力として与えて、`erdos.renyi.game` 関数を用いて作成した結果を、`er` というオブジェクトに保存せよという命令です。ここではオブジェクトの名前を `er` としましたが、これはランダムネットワークがエルデシュ・レニイモデルであることを念頭に置いています。次のコマンドは、作成したオブジェクト `er` を入力として与えて、`plot` 関数を用いて描画せよという命令です。

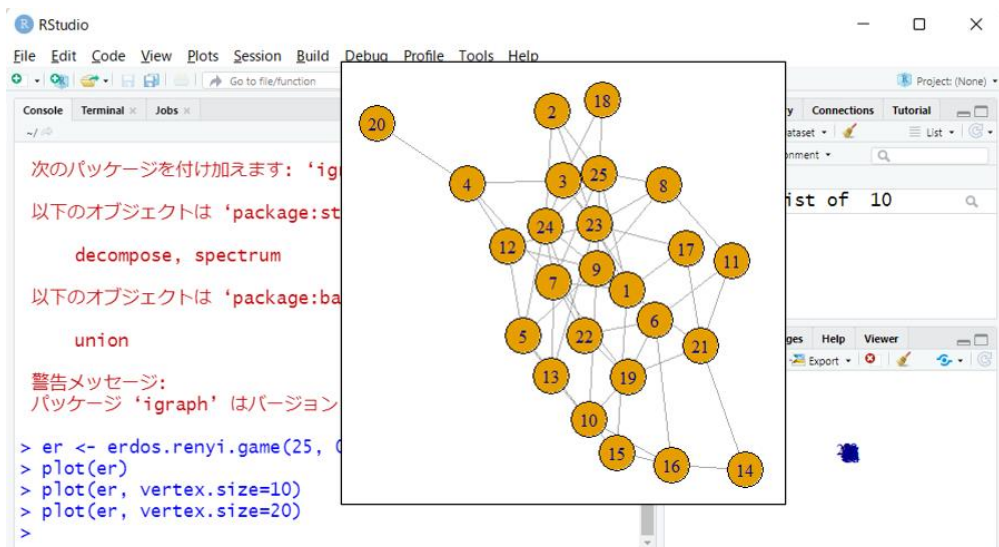
RStudio 上での実行結果のスクリーンショット(スクショ)を以下に示します。



右下に見えている①が描画結果です。これだと小さすぎて見づらいですが、② Zoom ボタンを押せば、①のネットワークを拡大した図が以下のような感じで見られます。確かに指定した通り頂点数が 25 個あることが分かります。



頂点（ノード）のサイズを大きくしたい場合は、`plot` 関数実行時に `vertex.size` オプションで与える数値を大きくすればよいです。例えば、以下は `vertex.size=10` や `20` を与えて `plot` 関数を実行した結果のスクショです。

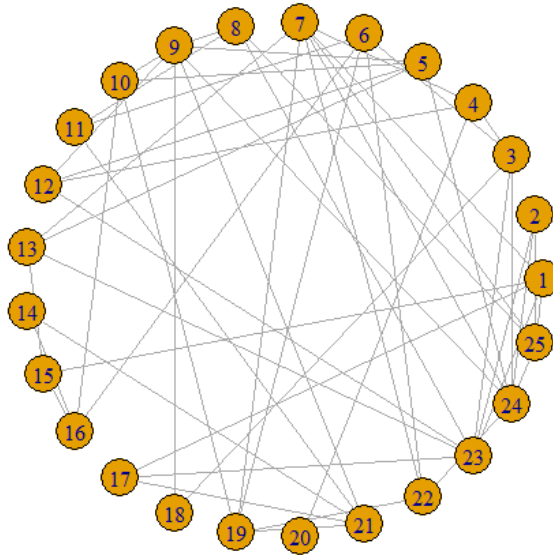


多少縮尺は異なりますが、`vertex.size=20` で実行し Zoom ボタンで見たネットワーク図だと、頂点の○の中にある数値がはっきり判読できることが分かります。注意点としては、一見すると異なるネットワーク図のように見えることです。し

かし、例えばこの図の場合は、②⑩は④と、そして⑭は⑮および⑯とリンクが張られている点と同じであることがわかるように、ネットワークのトポロジーは同じです。つまり、ランダムにレイアウトされているだけです。

上記のネットワーク図は、図 5.21a と本質的に同じである点にも注意してください。図 5.21a と同様にサークル状のネットワーク図にしたい場合は、以下のように `layout` オプションで `layout_in_circle` を与えて実行すればよいです。

```
plot(er, vertex.size=15, layout=layout_in_circle)
```



ここでは実行結果のネットワーク図しか示しませんが、確かに図 5.21a と似た結果が得られていることがわかります。ここまでの、同じトポロジーをもったネットワークでも、様々な見せ方があることがわかります。

(b) スモールワールドネットワークの描画

頂点数 $5 \times 5 = 25$ で、1 個となりの頂点とリンクした格子を、確率 0.1 でランダムに置き換えるスモールワールドネットワークは、`sample_smallworld` 関数を用いて以下のコマンドで作成することができます。

```
sw <- sample_smallworld(dim=2, size=5, nei=1, p=0.1)
plot(sw, vertex.size=15, layout=layout_in_circle)
```

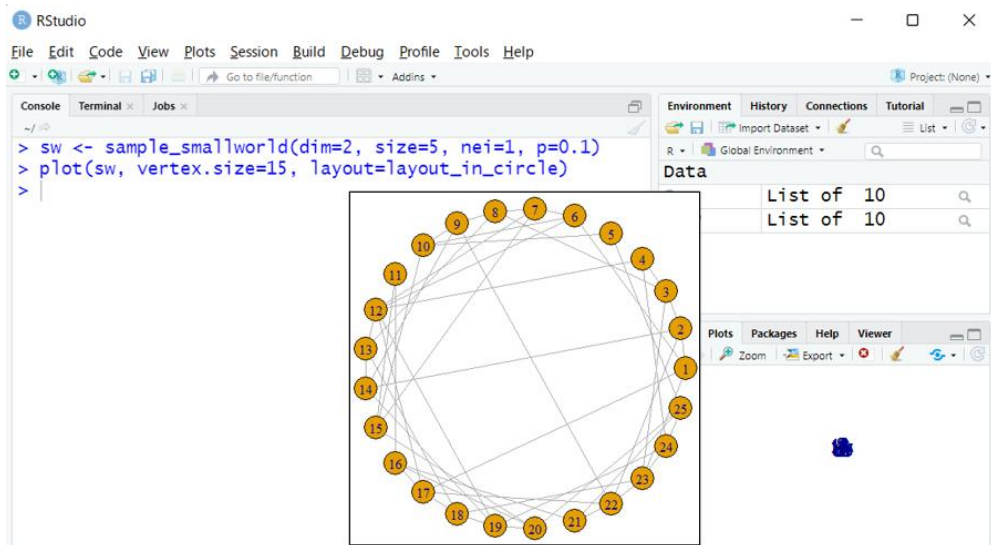


図 5.21b と似た図が描けていることがわかります。

(c) スケールフリー・ネットワークの描画

頂点を追加するたび、既にある頂点の次数の 2 乗に比例する割合で辺を 3 本ずつ足していく頂点数 25 のスケールフリー・ネットワークは、`sample_pa` 関数を用いて以下のコマンドで作成することができます。

```
sf <- sample_pa(n=25, power=2, m=3, directed=F)
plot(sf, vertex.size=15)
```

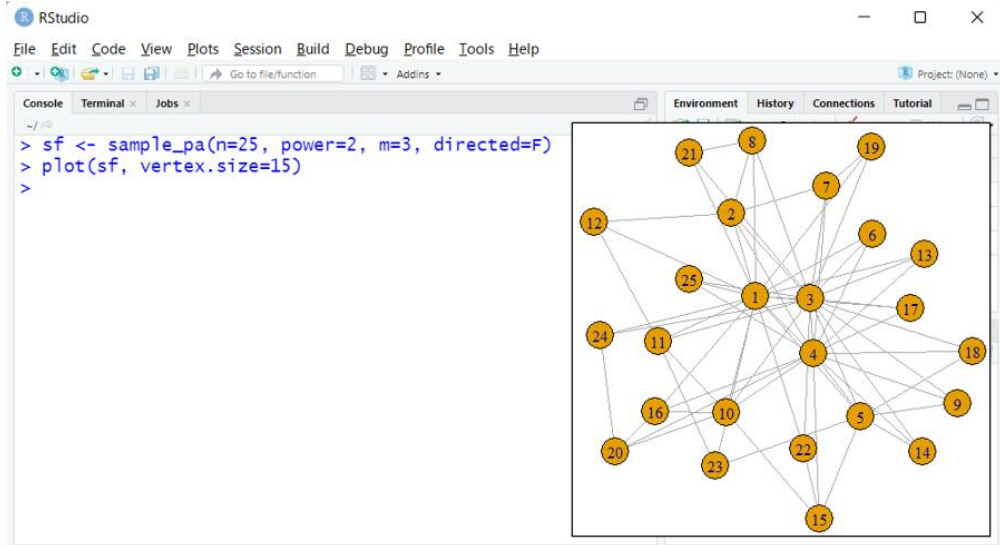
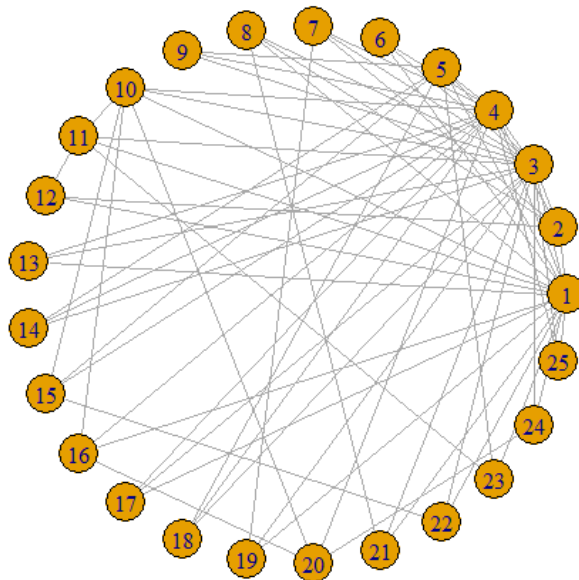


図 5.21c と同様のネットワーク図にしたい場合は、以下のように実行します。

```
plot(sf, vertex.size=15, layout=layout_in_circle)
```



この図を眺めると、①や③や④の頂点（ノード）が比較的リンクが多い、つまり多くの辺（エッジ）が張られていることがなんとなくわかります。

解答例 (図 5.24 で示した 3 種類の中心性を調べる)

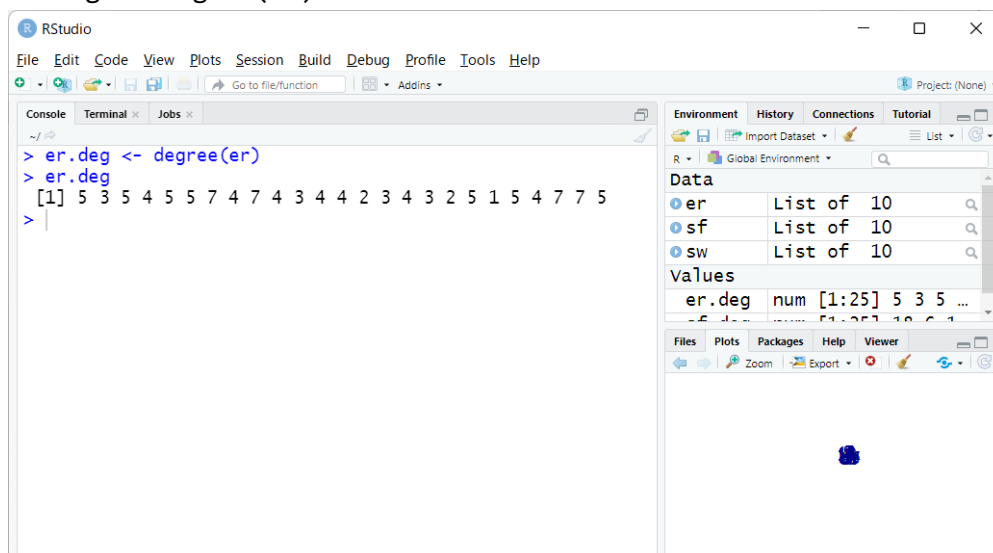
直前で示した「スケールフリー・ネットワークのサークル図」で、①や③や④がネットワークの中心っぽいことを議論しています。図 5.24 で示した 3 種類の中心性を調べますが、先ほどまでに得られた (a) ランダムネットワークのオブジェクト `er`、(b) スモールワールドネットワークのオブジェクト `sw`、(c) スケールフリー・ネットワークのオブジェクト `sf` が存在するという前提でコマンドを示します。次数中心は `degree` 関数、近接中心は `closeness` 関数、媒介中心は `betweenness` 関数で調べることができます。

(a) ランダムネットワーク (エルデシュ・レニイモデル) の中心性

次数中心

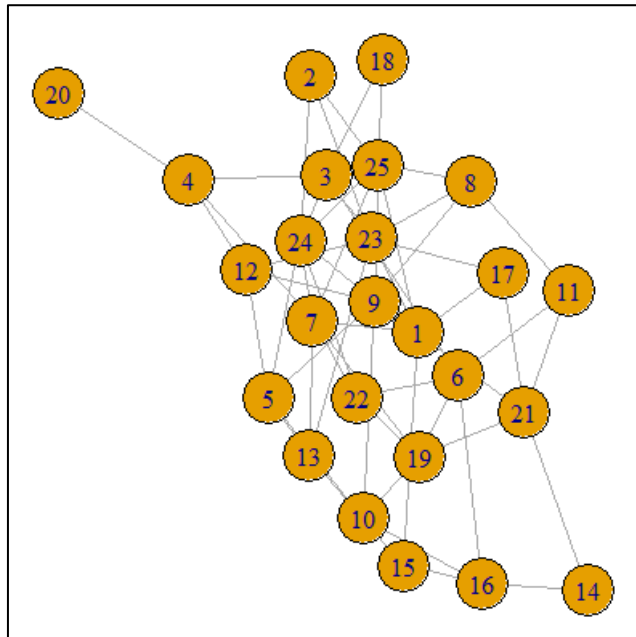
オブジェクト `er` を入力として、まずは次数中心を調べます。`degree` 関数を用いて以下のように実行することで、頂点 (ノード) ごとのリンク数 (つまり次数) の計算結果が `er.deg` というオブジェクトに保存されます。

```
er.deg <- degree(er)
```



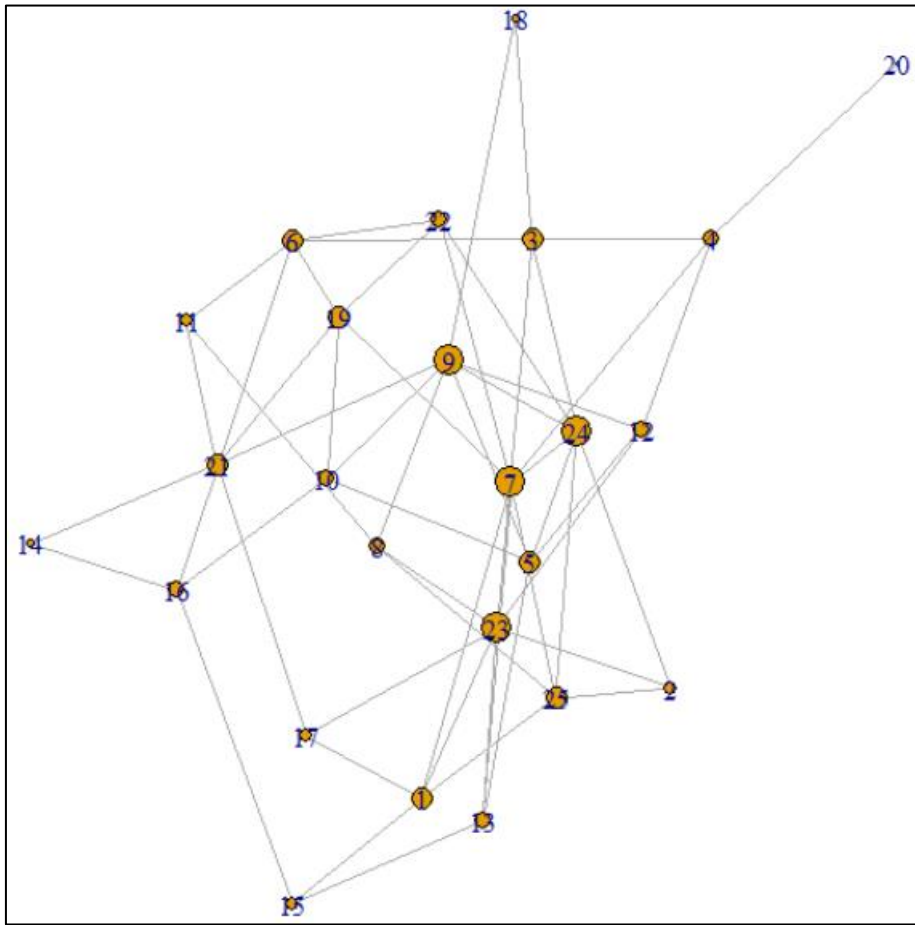
上記のスクリーンショットは、`er.deg` の中身の表示まで行っています。1 から 7 までの範囲の数値が計 25 個存在しますが、これは頂点数 25 としてランダムネットワークを生成したものだからです。以下でも改めて示すこのランダムネットワークのトポロジーを見ることで、なぜこのような結果になるかが理解できます。例えば、頂点⑩はリンクが頂点④のみなので、`er.deg` の 20 番目の要素が 1 になっているのです。また、頂点②はリンクが③・④・⑤の 3 つなので、`er.deg` の 2 番目の要素が 3 になっているのです。さらに、頂点④は、リンクが③・⑦・⑫・⑩の 4 つなので、`er.deg` の 4 番目の要素が 4 になっているのです。図 5.24 でも示しているように、「次数が最大のものを選ぶ指標」が次数中心なので、この

場合は、「最大次数 7 をもつ計 4 つの頂点 (⑦と⑨と㉓と㉔)」が答えということになります。確かに下記のネットワーク図でも中心付近に存在していることがわかります。



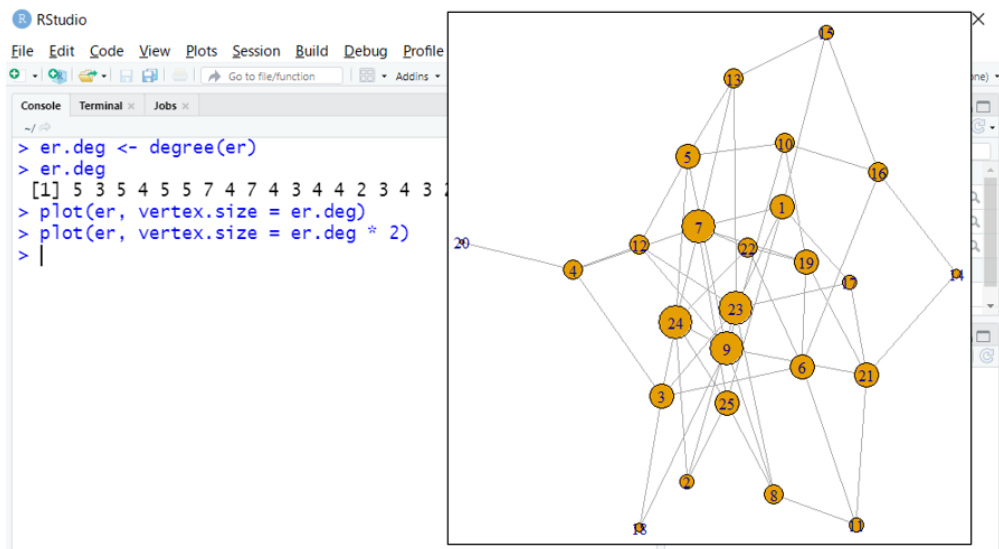
尚、ネットワーク図を描画する際に、次数計算結果である `er.deg` の情報を `plot` 関数実行時の `vertex.size` オプションの引数として与えることで、次数が大きい頂点ほどサイズを大きくすることができます。実際に、さきほど答えとして導き出した「最大次数 7 をもつ計 4 つの頂点 (⑦と⑨と㉓と㉔)」が一番大きく表示されていることがわかります。

```
plot(er, vertex.size = er.deg)
```

頂点全体を 2 倍にしたい場合は、例えば以下のように `vertex.size` オプションのところで、`er.deg * 2` とやれば「最大次数 7 をもつ計 4 つの頂点 (⑦と⑨と⑬と⑭)」の大きさが $7 * 2 = 14$ になります。

```
plot(er, vertex.size = er.deg * 2)
```

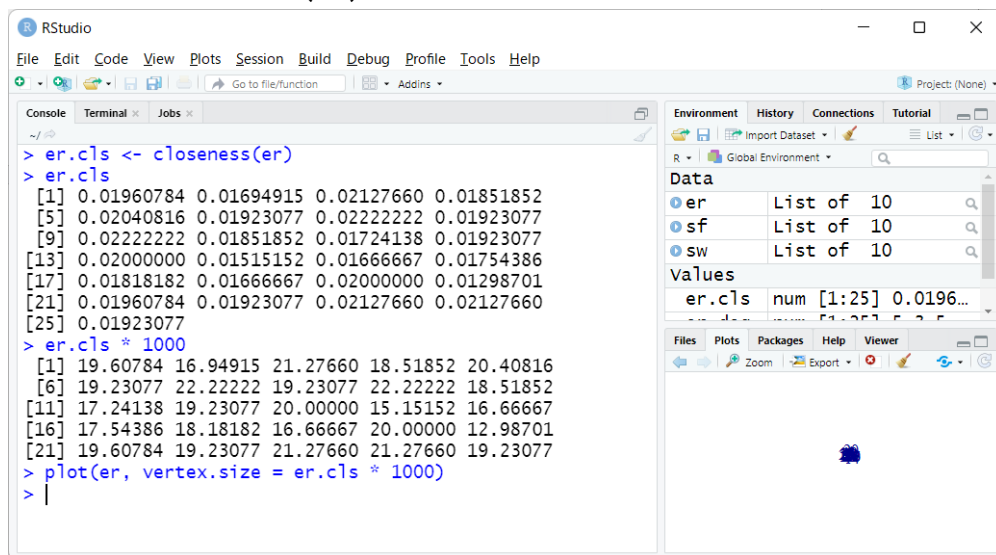


こんな具合で、`plot` 関数実行時の `vertex.size` オプションで与える数値の範囲を任意に変更することで、中心性指標の値が全体として小さい近接中心のような場合でも、視覚的に判読可能な大きさに調整することができます。

近接中心

オブジェクト `er` を入力として、近接中心を調べます。`closeness` 関数を用いて以下のように実行することで、頂点（ノード）ごとの全ての他頂点への最短経路を全て計算し、その総和で頂点数を割り算した結果が `er.cls` というオブジェクトに保存されます。

```
er.cls <- closeness(er)
```



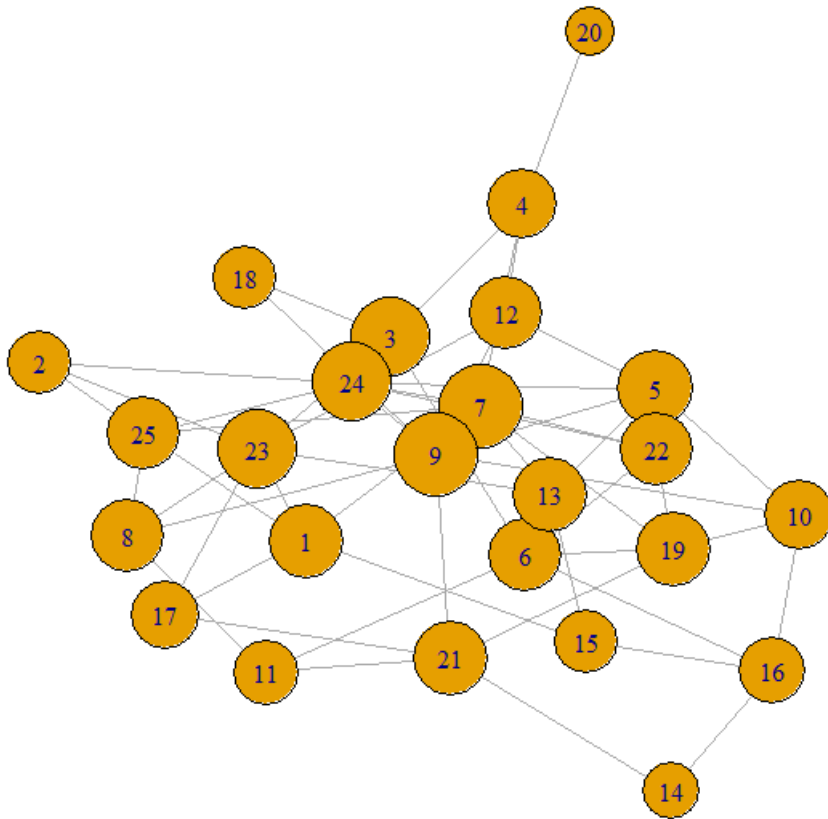
The screenshot shows the RStudio interface. The console window displays the following R code and its output:

```
> er.cls <- closeness(er)
> er.cls
[1] 0.01960784 0.01694915 0.02127660 0.01851852
[5] 0.02040816 0.01923077 0.02222222 0.01923077
[9] 0.02222222 0.01851852 0.01724138 0.01923077
[13] 0.02000000 0.01515152 0.01666667 0.01754386
[17] 0.01818182 0.01666667 0.02000000 0.01298701
[21] 0.01960784 0.01923077 0.02127660 0.02127660
[25] 0.01923077
> er.cls * 1000
[1] 19.60784 16.94915 21.27660 18.51852 20.40816
[6] 19.23077 22.22222 19.23077 22.22222 18.51852
[11] 17.24138 19.23077 20.00000 15.15152 16.66667
[16] 17.54386 18.18182 16.66667 20.00000 12.98701
[21] 19.60784 19.23077 21.27660 21.27660 19.23077
> plot(er, vertex.size = er.cls * 1000)
> |
```

The Environment pane on the right shows the `er.cls` object as a numeric vector of length 25, with the first value being 0.01960784.

上記のスクリプトは、`er.cls` オブジェクト取得後に、まず中身の表示を行っています。本文中でも説明したとおり、ネットワークにおけるいずれの頂点にも近い頂点が大きな値をとります。さきほどの次数中心では、答えが「最大次数7をもつ計4つの頂点（⑦と⑨と⑳と㉔）」となりましたが、**近接中心では最大値0.0222222をもつ2つの頂点（⑦と⑨）が答えだと判断します。**

しかしながら、この `er.cls` の数値をそのまま頂点のサイズとしてネットワーク図を描画しても、全体として非常に小さな頂点サイズになってしまい重要な頂点を視覚的に判別することができません。それゆえ、上記のスクリプトでも `er.cls * 1000` などとして、表示上妥当な数値範囲をアドホックに探り、その結果を `plot` 関数実行時の `vertex.size` オプションの引数として与えます。以下は、上記スクリプトの最後のコマンド実行結果として得られたネットワーク図です。

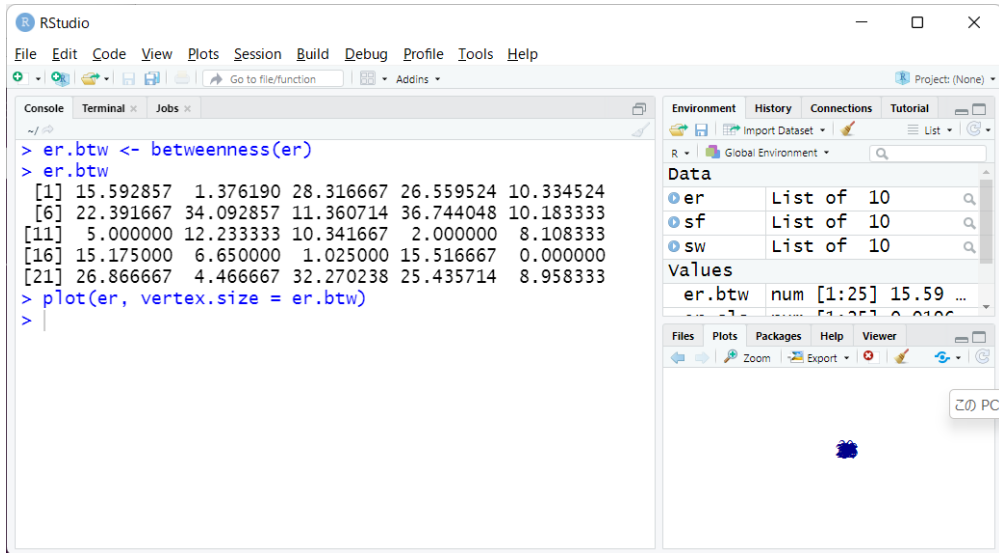


既に「最大値 0.0222222 をもつ 2 つの頂点 (⑦と⑨) が答え」だとわかってはいますが、これらの頂点は視覚的にもネットワーク図の中心付近に位置しており妥当であることがわかります。

媒介中心

オブジェクト `er` を入力として、媒介中心を調べます。`betweenness` 関数を用いて以下のように実行することで、自分以外の頂点間における最短経路にどれだけ含まれるかを計算した結果が `er.btw` というオブジェクトに保存されます。

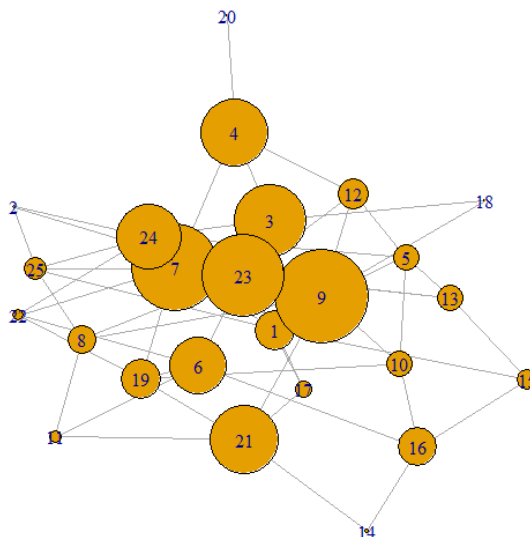
```
er.btw <- betweenness(er)
```



```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Project: (None)
Environment History Connections Tutorial
R Global Environment
Data
er List of 10
sf List of 10
sw List of 10
Values
er.btw num [1:25] 15.59 ...
Files Plots Packages Help Viewer
この PC
```

```
> er.btw <- betweenness(er)
> er.btw
[1] 15.592857 1.376190 28.316667 26.559524 10.334524
[6] 22.391667 34.092857 11.360714 36.744048 10.183333
[11] 5.000000 12.233333 10.341667 2.000000 8.108333
[16] 15.175000 6.650000 1.025000 15.516667 0.000000
[21] 26.866667 4.466667 32.270238 25.435714 8.958333
> plot(er, vertex.size = er.btw)
>
```

上記のスクショは、`er.btw` オブジェクト取得後に、中身の表示を行っています。本文中でも説明したとおり、前述の2つの中心性指標と同様に、大きな数値の頂点ほど中心性が高いと判断します。次数中心では「最大次数7をもつ計4つの頂点(⑦と⑨と⑬と⑭)」、近接中心では「最大値0.0222222をもつ2つの頂点(⑦と⑨)」でしたが、媒介中心では「最大値36.744048をもつ頂点⑨」のみが答えだと判断します。

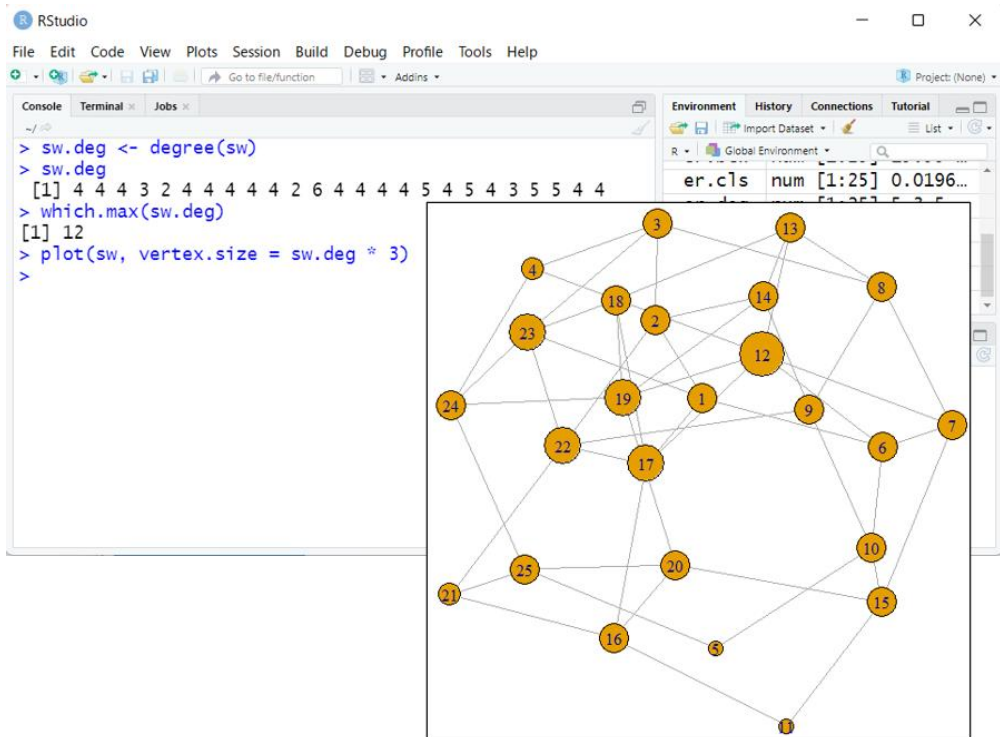


(b) スモールワールドネットワークの中心性

次数中心

オブジェクト `sw` を入力として、次数中心を調べます。`degree` 関数を用いて以下のように実行することで、頂点（ノード）ごとのリンク数（つまり次数）の計算結果が `sw.deg` というオブジェクトに保存されます。

```
sw.deg <- degree(sw)
```

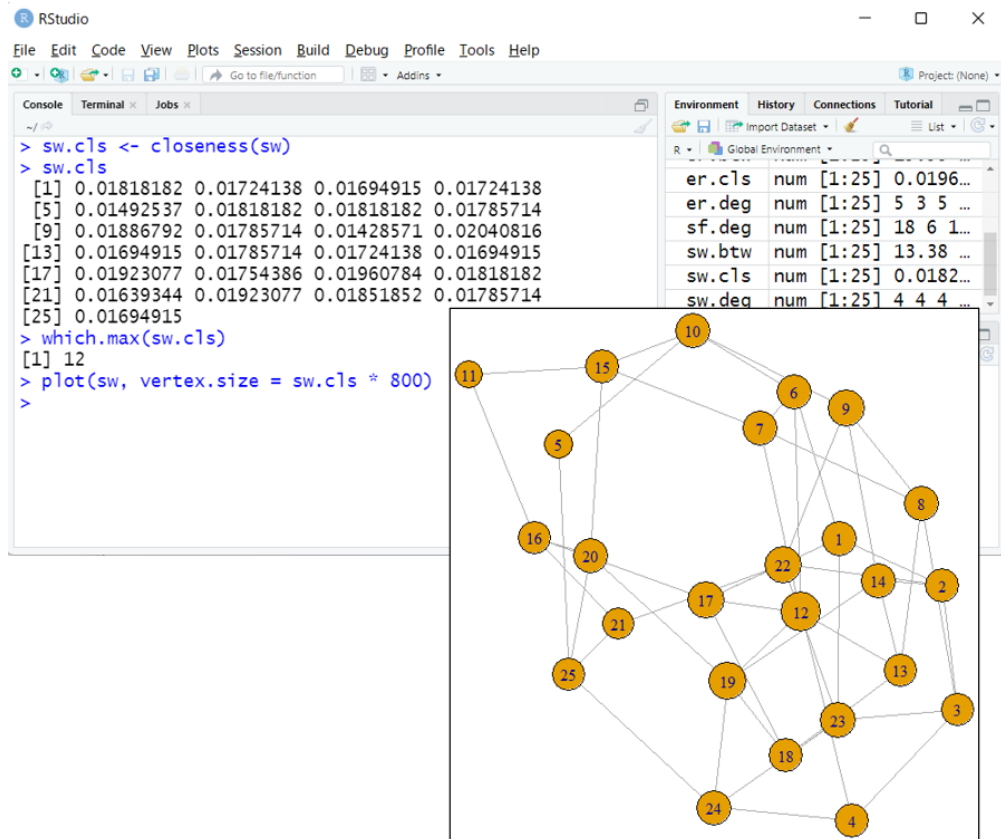


上記のスクリプトは、`sw.deg` オブジェクト取得後に、まず中身の表示を行っています。計 25 個の要素からなる数値ベクトルなので、最大値をもつ頂点を見つけるのは比較的簡単ではありますが。しかし、`sw.deg` という数値ベクトルを入力として `which.max` 関数を実行すれば、最大値をもつ要素番号を表示してくれます。この場合は 12 が表示されていますが、**頂点番号12**が次数中心であることを意味しています。この結果は、ネットワーク図からも確認できます。

近接中心

オブジェクト `sw` を入力として、近接中心を調べます。`closeness` 関数を用いて以下のように実行することで、頂点（ノード）ごとの全ての他頂点への最短経路を全て計算し、その総和で頂点数を割り算した結果が `sw.cls` というオブジェクトに保存されます。

```
sw.cls <- closeness(sw)
```

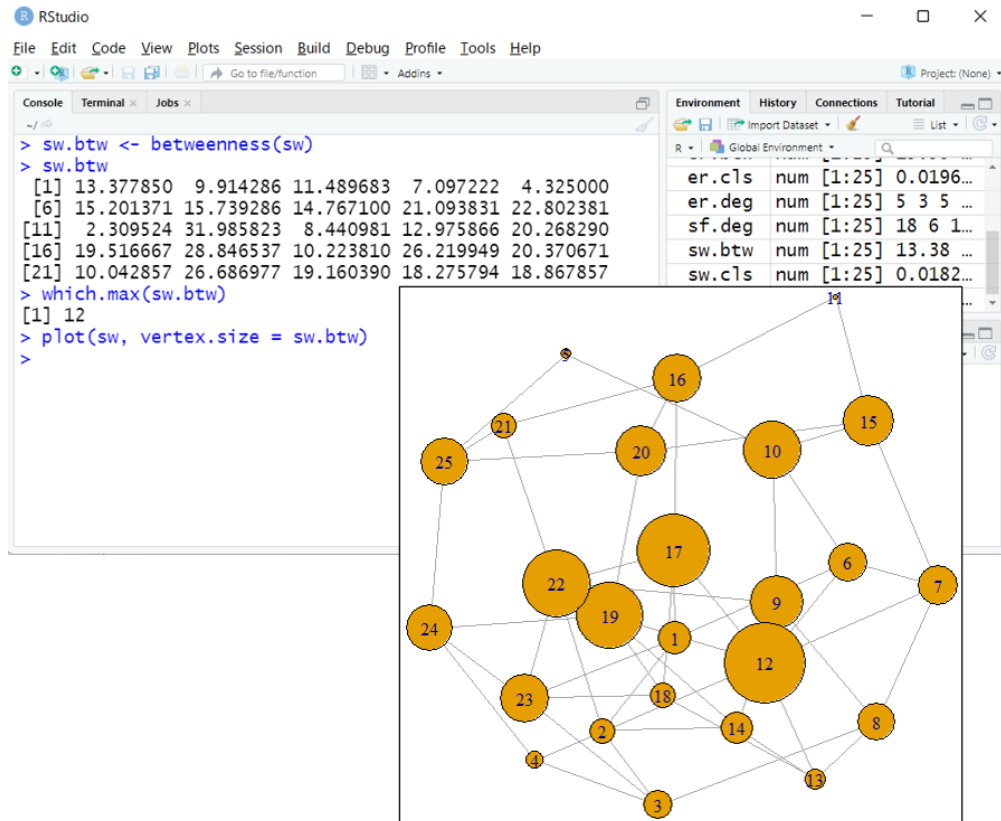


次数中心の結果と同様、**頂点番号⑫**が**近接中心**であることがわかります。この結果は、ネットワーク図からもなんとなくですが確認できます。

媒介中心

オブジェクト `sw` を入力として、媒介中心を調べます。`betweenness` 関数を用いて以下のように実行することで、自分以外の頂点間における最短経路にどれだけ含まれるかを計算した結果が `sw.btw` というオブジェクトに保存されます。

```
sw.btw <- betweenness(sw)
```



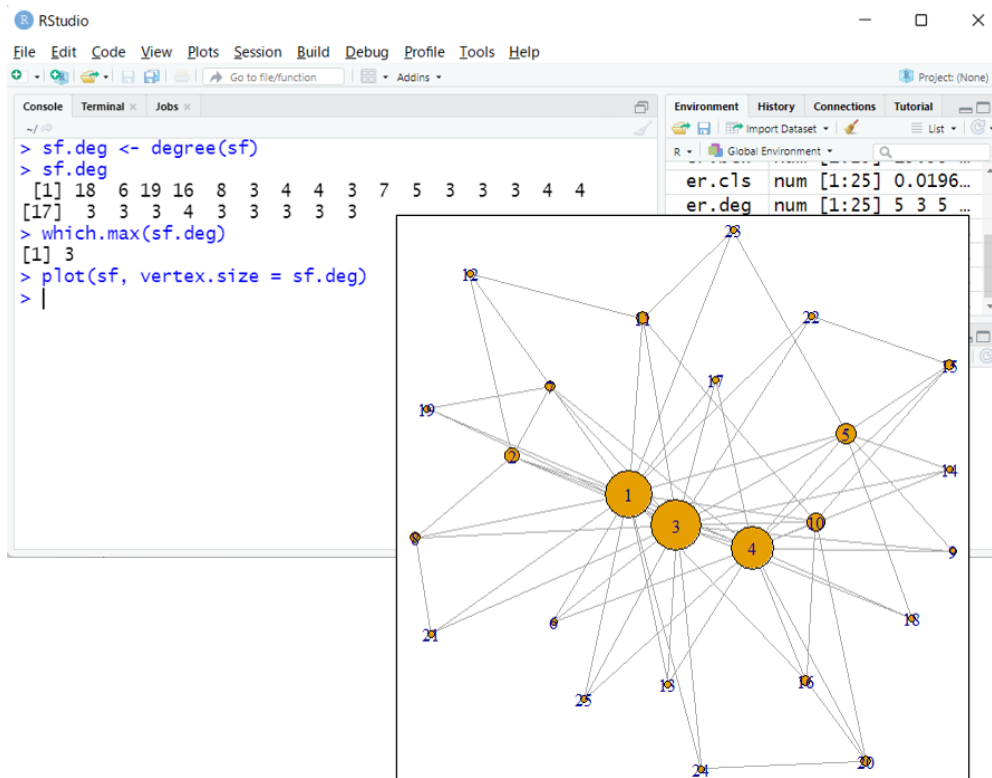
次数中心および近接中心の結果と同様、**頂点番号⑫**が媒介中心であることがわかります。この結果は、ネットワーク図からもなんとなくですが確認できます。

(c) スケールフリー・ネットワークの中心性

次数中心

オブジェクト `sf` を入力として、次数中心を調べます。`degree` 関数を用いて以下のように実行することで、頂点（ノード）ごとのリンク数（つまり次数）の計算結果が `sf.deg` というオブジェクトに保存されます。

```
sf.deg <- degree(sf)
```

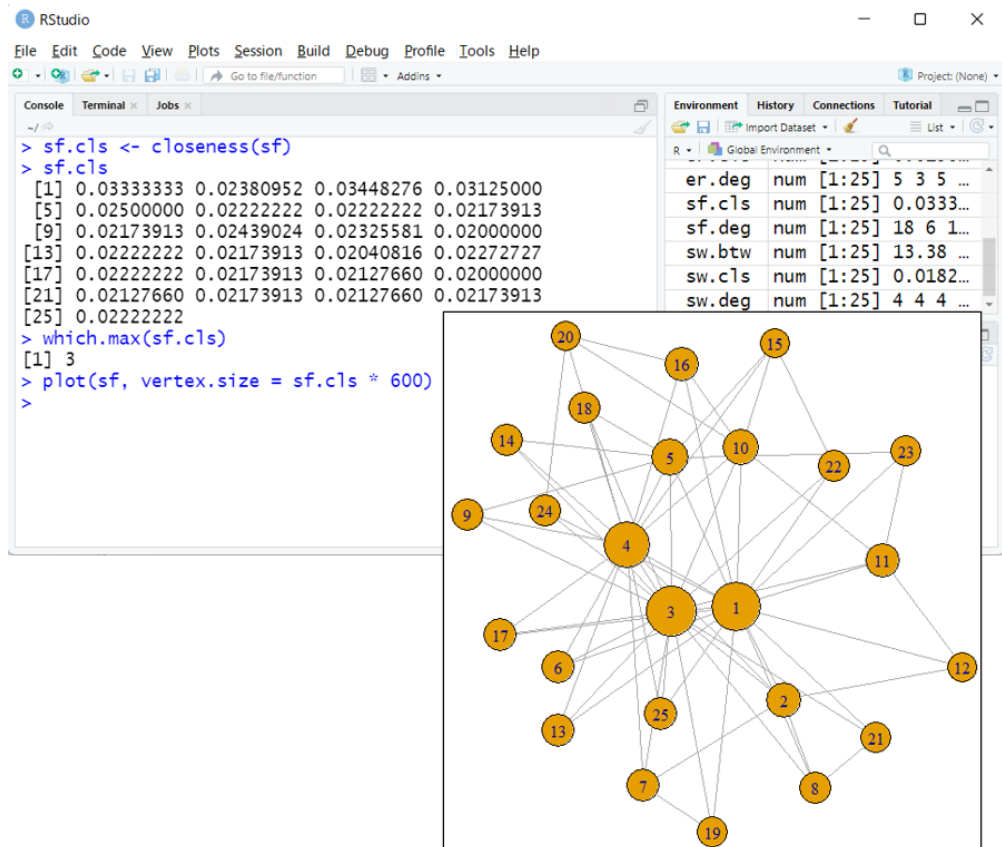


頂点番号③が次数中心であることがわかります。この結果は、ネットワーク図からも確認できます。

近接中心

オブジェクト `sf` を入力として、近接中心を調べます。`closeness` 関数を用いて以下のように実行することで、頂点（ノード）ごとの全ての他頂点への最短経路を全て計算し、その総和で頂点数を割り算した結果が `sf.cls` というオブジェクトに保存されます。

```
sf.cls <- closeness(sf)
```

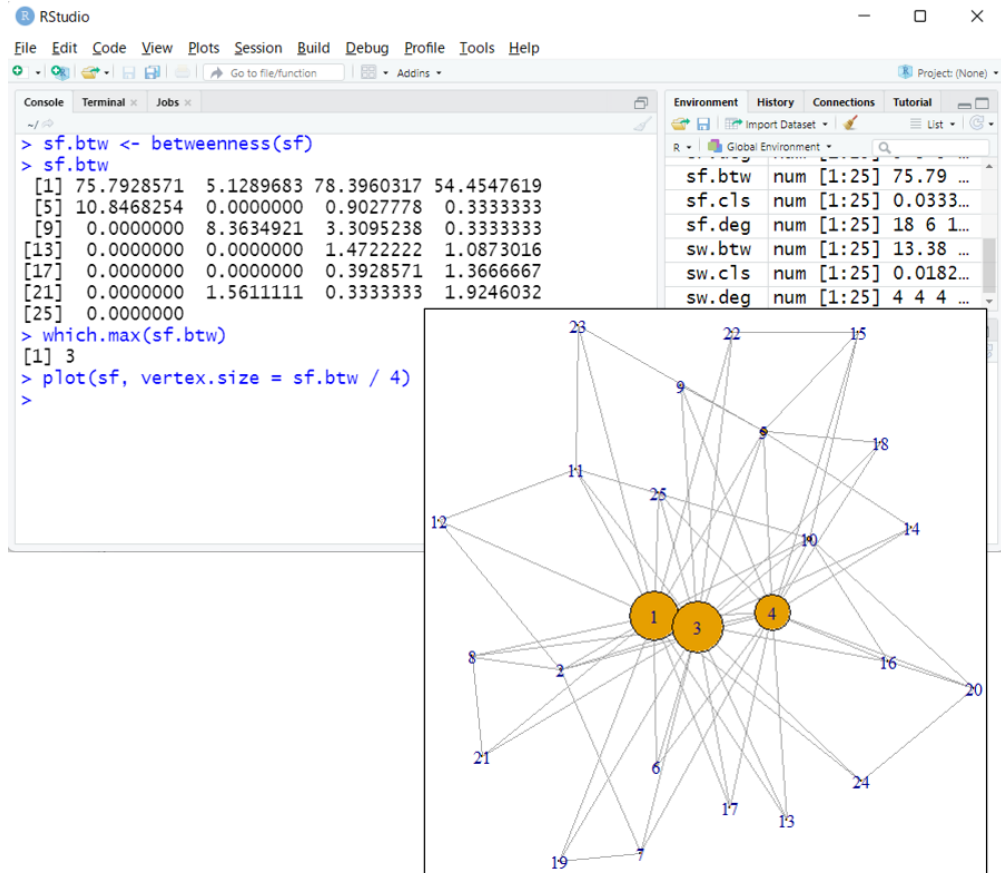


頂点番号③が近接中心であることがわかります。この結果は、ネットワーク図からも確認できます。

媒介中心

オブジェクト `sf` を入力として、媒介中心を調べます。`betweenness` 関数を用いて以下のように実行することで、自分以外の頂点間における最短経路にどれだけ含まれるかを計算した結果が `sf.btw` というオブジェクトに保存されます。

```
sf.btw <- betweenness(sf)
```



頂点番号③が媒介中心であることがわかります。この結果は、ネットワーク図からも確認できます。