



SOCIAL NETWORK ANALYSIS (SNA)

Siapa aja bisa membuat SNA menggunakan R dan Gephi

Workshop: Data Science for All

Ujang Fahmi, M.P.A

eppofahmi@gmail.com

2020. 03. 04

Editor: Canggih Puspo Wibowo, M.Eng.

Abstract

Jika Anda ingin lebih memahami jaringan sosial, jaringan informasi, atau bahkan jaringan saraf otak kita, maka Anda perlu mengetahui ilmu jaringan! Ini akan menunjukkan analisis jaringan menggunakan beberapa paket R, termasuk dplyr, ggplot2, igraph, ggraph dan juga visNetwork. Anda akan berperan sebagai Analis Interpol dan menyelidiki jaringan teroris di belakang pemboman kereta Madrid pada tahun 2004. Setelah mengikuti kursus ini, Anda akan dapat menganalisis jaringan apa pun dengan sentralitas dasar dan langkah-langkah kesamaan serta menciptakan visualisasi jaringan yang indah dan interaktif.

List of Tables

1	Contoh tabel dengan nama kolom yang menggunakan spasi	13
2	Contoh data untuk menentukan nodes dan edges	16
3	Data untuk network analisis berdasarkan mention Twitter	16

List of Figures

1	Visualisasi network topik pembicaraan sebuah isu di Twitter	2
2	Tampilan awal Rstudio	4
3	Gambar hasil dari sintak plot()	8
4	Ilustrasi degree centrality	20
5	Ilustrasi betweeness centrality	21
6	Ilustrasi closeness centrality	22
7	Ilustrasi eigenvector centrality	23
8	Ilustrasi PageRank centrality	24
9	Ilustrasi Hasil Visualisasi Modularity	25
10	Ilustrasi Cara Mengimpor Data	27
11	Contoh twit untuk menentukan nodes dan edges network	35
12	Hasil visualisasi network di Gephi	41
13	Komunitas dalam jejaring	46
14	Tampilan awal gephi	47
15	Tampilan saat impor spreadsheet	49
16	Tampilan saat menentukan jenis network	50
17	Tampilan saat import file graph	51
18	Plugins yang tersedia untuk gephi	52
19	Hasil visualisasi network dasar	54
20	Hasil visualisasi atribut eigencentrality	55
21	Hasil visualisasi ggraph dasar	57
22	Hasil visualisasi dengan besar nodes yang berbeda	59
23	Tampilan awal setelah impor data	61
24	Hasil dan referensi modularity yang digunakan Gephi	62
25	Hasil penghitungan modularity secara visual	63
26	Hasil penghitungan centrality secara visual	64
27	Memunculkan nama nodes dalam visual network	65
28	Hasil layout force atlas2	66
29	10 Kelompok dengan jumlah anggota terbanyak	68

Contents

List of Tables	II
List of Figures	III
1 Pengantar	1
2 Pengenalan R dan <i>Network Analysis</i> (2 Jam)	3
2.1 Latihan 1	3
2.2 Menggunakan R	4
2.2.1 Bagian Console/Terminal/R Markdown	4
2.2.2 Bagian Environment/History/Connection	7
2.2.3 Bagian File/Plots/Packages/Help/Viewer	7
2.3 Latihan 2	10
2.4 Menulis Skrip di R	10
2.5 Latihan 3	14
2.6 Konsep dan Fungsi Network Analysis	15
2.7 Latihan 4	18
3 Memahami Centrality dan Modularity dalam <i>Network</i> (3 Jam)	19
3.1 Degree Centrality	19
3.2 Betweenness Centrality	20
3.3 Closeness centrality	21
3.4 Eigenvector Centrality	22
3.5 PageRank	23
3.6 Modularity	25
3.7 Latihan 5	25
4 Pre-processing untuk Network Analysis (3 Jam)	26
4.1 Impor dan Ekspor Data	27
4.2 Latihan 6	29
4.3 Memilih kolom	30
4.4 Latihan 7	31
4.5 Memabagi kolom	32

4.6	Latihan 8	33
4.7	Tokenisasi	33
4.8	Latihan 9	34
4.9	Nodes dan Edges	35
4.10	Latihan 10	41
5	Menggunakan Igraph dan Gephi (3 Jam)	42
5.1	Package igraph	43
5.2	Aplikasi Gephi	46
5.3	Latihan 11	52
6	Visualisasi Network (4 Jam)	53
6.1	Visualisasi network menggunakan R	53
6.1.1	Visualisasi dengan <code>plot()</code>	53
6.1.2	Visualisasi dengan <code>ggraph</code>	57
6.2	Latihan 12	60
6.3	Visualisasi network menggunakan Gephi	61
6.4	Latihan 13	69
7	Penutup dan Referensi	69

1 Pengantar

Workshop **Social Network in R** ini ini bertujuan untuk memberikan bekal pengetahuan, keterampilan dan pengalaman langsung melakukan analisis social network kepada peserta. Pelatihan ini membutuhkan kurang lebih 15 Jam secara intensif. Untuk kelancaran pelatihan, peserta **direkomendasikan** telah mengenal bahasa pemrograman R atau minimal telah:

1. Memasang (*install*) R dan R Studio
2. Menginstall package `tidyvers`, `tidytext`, `igraph`, `ggraph`, `rgexf`
3. Menginstall aplikasi gephi (bisa didapatkan di: <https://gephi.org/users/download/>)

Pelatihan ini direkomendasikan bagi yang pernah atau telah menguasai dasar-dasar menggunakan R. Tapi, bagi yang baru belajar pun tetap bisa diikuti, dengan syarat bisa menyediakan waktu lebih banyak untuk memahami konsep dan penggunaan sintaks, khususnya yang ada dalam beberapa bagian awal materi ini. Pada akhir pelatihan, peserta diharapkan telah mampu untuk:

1. Menggunakan R
2. Mencari bantuan saat menghadapi error
3. Mengerti konsep network analysis
4. Mengetahui fitur-fitur dalam network
5. Mampu melakukan pre-processing untuk network analysis
6. Mampu menggunakan Gephi
7. Mampu memberikan interpretasi hasil network analysis
8. Pernah melalui proses network analysis dari awal hingga akhir yaitu mendapatkan data, melakukan pre-processing, analysis, dan interpretasi

Hasil akhir dari pelatihan ini, dapat dirangkum untuk dapat membuat dan menginterpretasikan visualisasi seperti Gambar 1. Di mana dalam gambar tersebut, bukan hanya ada Username, tapi juga beberapa kata yang menghubungkan antar kelompok username dalam warna yang berbeda.

Data untuk latihan dapat diambil dari: <https://drive.google.com/drive/folders/1cOnF2ireDSuIStDKKeALyZUK8O87D042?usp=sharing>

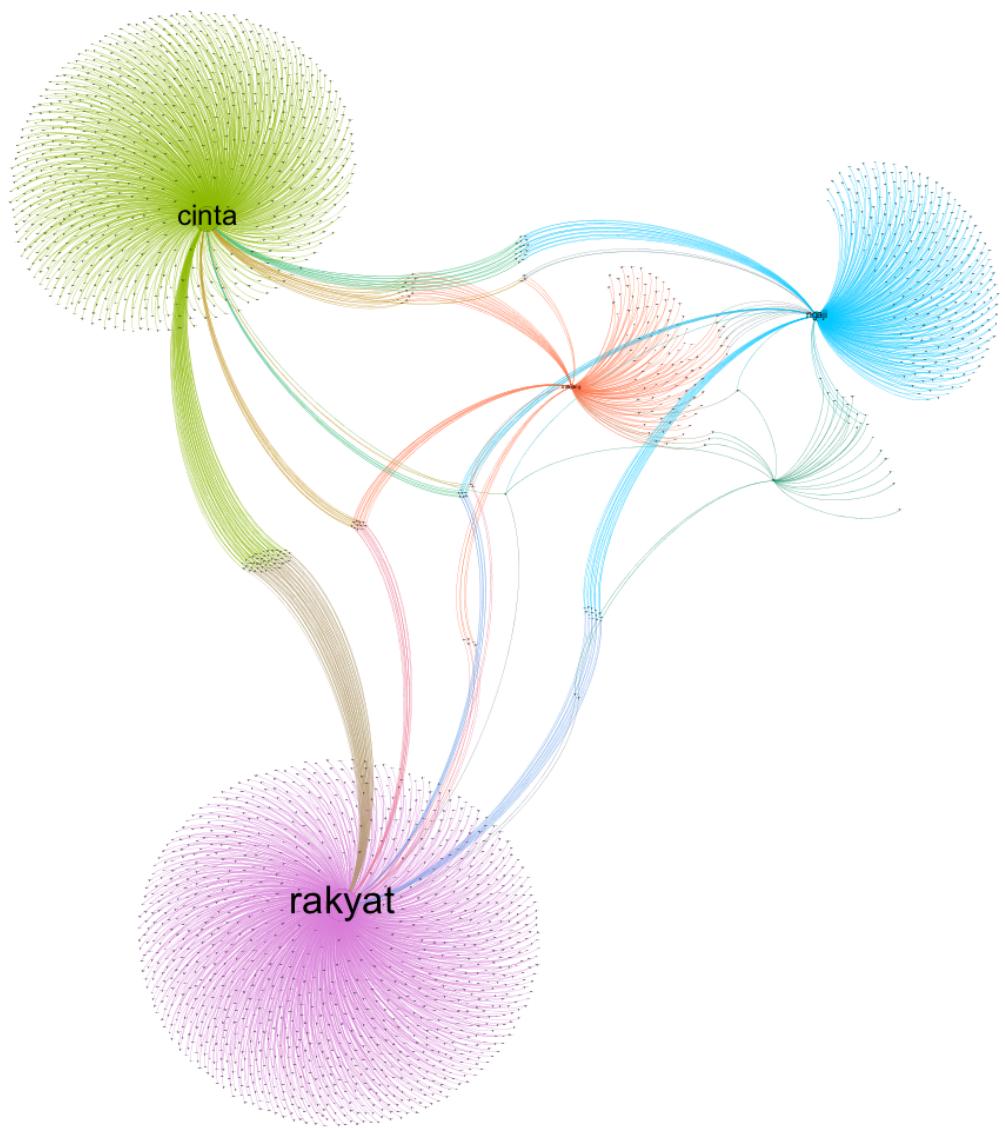


Figure 1: Visualisasi network topik pembicaraan sebuah isu di Twitter

2 Pengenalan R dan *Network Analysis* (2 Jam)

R merupakan sebuah bahasa pemrograman statistik yang bisa didapatkan secara gratis (*open source*). Dalam proses perkembangannya, R bukan hanya banyak digunakan oleh analis untuk analisis statistik, tapi juga untuk kebutuhan lain seperti analisis teks dan jaringan (*network*). Hal tersebut tidak terlepas dengan adanya packages atau library yang saat ini sudah berjumlah puluhan ribu.

Package atau **library** merupakan kumpulan fungsi yang bisa dibuat oleh siapa saja dan setelah di publikasikan juga bisa digunakan oleh siapa saja. Fungsi-fungsi yang terdapat di dalam sebuah package atau library hanya membutuhkan diinstall sekali oleh pengguna. Namun untuk menggunakannya, pengguna perlu menjalankan skrip untuk memanggil nama packagenya, yaitu sintak **library(...)**.

Beberapa dari kita, mungkin masih bingung dengan adanya R dan Rstudio yang menjadi pra-syarat untuk mengikuti pelatihan ini. Secara sederhana, R bisa dikatakan sebagai bahasanya, sementara Rstudio bisa dianggap sebagai alat untuk menggunakan bahasa tersebut lebih mudah digunakan sebagai alat komunikasi manusia dengan komputer atau biasa disebut sebagai *Integrated Development Environment* (IDE). Sebagian besar pengguna R banyak menggunakan Rstudio dalam proses analisisnya, sama halnya dengan pelatihan ini, dimana sepenuhnya akan menggunakan Rstudio.

2.1 Latihan 1

Berdasarkan penjelasan pada **Bagian 1**, jawablah pertanyaan berikut:

1. Apa fungsi R dan Rstudio?
2. Apa yang dimaksud dengan **Package** atau **library**?
3. Apa sintak yang bisa digunakan untuk memanggil sebuah **package**?

2.2 Menggunakan R

Hal pertama yang harus dilakukan untuk bisa menggunakan R adalah dengan mengunduh dan memasangnya di komputer atau laptop. Selanjutnya, jika ingin menggunakan IDE maka kita perlu juga untuk mengunduh dan memasang Rstudio. Setelah kedua hal tersebut dilakukan maka perangkat kita sudah siap digunakan untuk melakukan latihan dan atau analisis secara langsung. Seperti telah disebutkan diawal kita akan menggunakan Rstudio. Oleh karena itu, cobalah untuk menjalankan Rstudio anda, dan seharusnya anda akan menemukan aplikasi seperti tampak pada [Gambar Tampilan awal Rstudio](#).

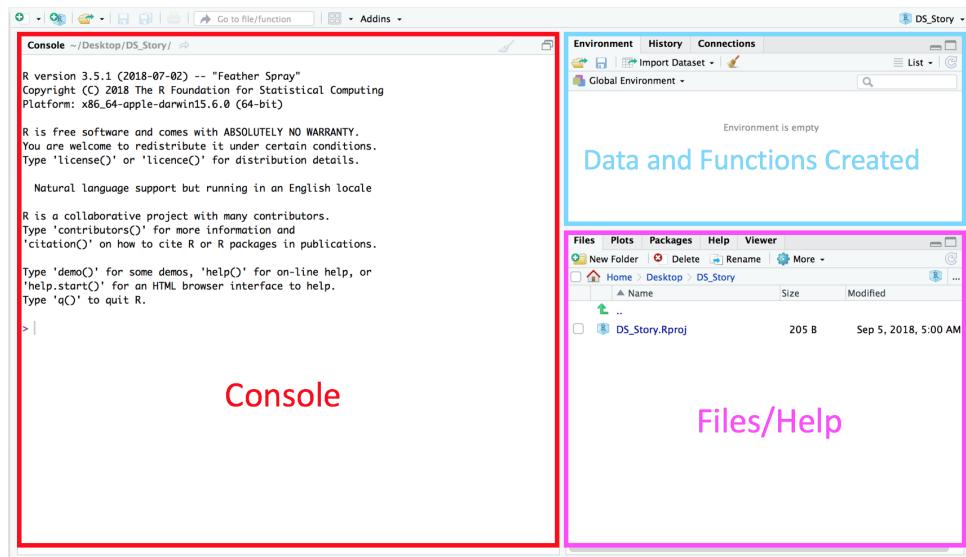


Figure 2: Tampilan awal Rstudio

Pada saat awal membuka rstudio, terdapat 3 bagian yang terbuka, yaitu bagian **console**, **data and function**, dan **files/help**. Namun setelah kita mulai menulis skrip, akan ada satu bagian lagi. Jadi, secara umum akan ada 4 bagian dalam Rstudio yang akan dihadapi saat melakukan analisis menggunakan R. Berikut adalah penjelasan tentang bagian-bagian yang perlu diperhatikan dalam menggunakan Rstudio.

2.2.1 Bagian Console/Terminal/R Markdown

Console akan menampilkan hasil atau respons komputer terhadap skrip yang dijalankan. Pada bagian ini pula kita bisa mengidentifikasi error yang terjadi pada skrip yang dibuat. Console juga bisa digunakan untuk menulis skrip. Misalnya dengan menulis $32+24$ lalu

menekan enter pada bagian console akan menampilkan hasilnya, yaitu: [1] 56. Angka satu di sana merupakan urutan dari skrip yang dijalankan. Sementara hasilnya adalah 56.

```
32+24
```

```
# [1] 56
```

Pada bagian console ini pula kita bisa mendapatkan informasi apakah skrip yang dijalankan masih atau sudah selesai dieksekusi. Cobalah perhatikan bagian `console` pada saat menjalankan skrip berikut:

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

Pada bagian pojok kanan atas `console` akan ada indikator merah yang menyala-nyala. Jika tanda merah tersebut belum sepenuhnya berhenti, artinya masih ada proses yang berjalan dari skrip terakhir yang dieksekusi. Selain itu, seperti telah disebutkan sebelumnya, pada bagian `console` ini pula kita bisa mendapatkan informasi tentang error atau kesalahan yang terjadi. Untuk memahminya, cobalah untuk menjalankan beberapa skrip berikut di `console` Anda satu persatu.

```
library(paket)
print(datanya)
data pertama = 1 + 1
```

Saat dijalankan, ketiga skrip diatas akan menghasilkan keterangan error sebagai berikut:

1. Error in library(paket) : there is no package called 'paket'
2. Error in print(datanya) : object 'datanya' not found
3. Error: unexpected symbol in "data pertama"

Tiga hal di atas merupakan informasi yang menentukan keberlanjutan proses penggunaan R untuk melakukan apapun yang diinginkan. Error pertama menyebutkan bahwa tidak adaka `package` yang bernama `paket`. Artinya, `package` yang kita panggil menggunakan sintak `library()` tersebut tidak ada atau belum terinstall di R. Solusinya adalah kita

harus menginstallnya terlebih dahulu. Error kedua menyebutkan bahawa objek datanya tidak ditemukan. Artinya, objek tersebut tidak ada di R yang kita jalankan. Solusi yang bisa tempuh adalah dengan membuat atau mengimpor objek datanya terlebih dahulu kedalam R. Sementara ketereangan error terakhir menyebutkan bahwa ada simbol yang tidak diharapkan.

Simbol yang tidak diharapkan ini disebabkan oleh adanya **spasi** diantara kata **data** dengan **pertama**. Padahal skrip tersebut ditujukan untuk membuat objek R, sedangkan tatacara penulisan objek di R tidak mengenal spasi. Oleh karena itu, solusi untuk menghadapi error tersebut adalah dengan menghilangkan spasi, misalnya dengan mengganti **data pertama** menjadi **datapertama** atau **dataPertama** seperti skrip berikut.

```
dataPertama = 1 + 1  
print(dataPertama)
```

```
# [1] 2
```

Seperti dapat dilihat pada skrip di atas. Kita menghilangkan spasi dan mengganti p dengan P untuk membedakan kata. Dengan kata lain, dengan memperhatikan output error yang terjadi, kita juga bisa menggunakan informasi yang muncul sebagai solusi. Jika hal tersebut tidak cukup, maka kita bisa mencarinya di mesin pencarian atau di forum-forum seperti yang ada di stackoverflow, github, kaggle, dan sejenisnya.

Secara umum, **Console** merupakan bagian yang akan paling banyak digunakan pada saat menggunakan R. Namun, pada bagian ini juga ada 2 bagian lainnya, yaitu terminal. Satu bagian lainnya adalah **RMarkdown** yang akan muncul ketika kita membuat file dengan ekstensi **.rmd**. Terminal, bisa digunakan sama seperti fungsi **Terminal** pada Mac atau **Command Line** pada Windows. Sementara **RMarkdown** akan memunculkan skrip yang dijalankan melalui file dengan ekstensi **.rmd** atau sama dengan fungsi **Console** tapi khusus untuk **rmarkdown**.

2.2.2 Bagian Environment/History/Connection

Bagian **Environment** berfungsi untuk melihat hasil dari skrip yang kita buat sebagai sebuah objek R baik berupa data maupun sebuah fungsi. Misalnya, pada saat kita menjalankan skrip **dataPertama**, yaitu `dataPertama = 1 + 1`, maka **dataPertama** dianggap sebagai sebuah objek. Dengan kata lain, **dataPertama** akan tersimpan pada bagian **Environment**. Oleh karena itu pula, pada bagian console hanya akan menunjukkan `dataPertama = 1 + 1` seperti dapat dilihat pada skrip berikut.

```
dataPertama = 1 + 1
```

Hasilnya, karena **dataPertama** sudah dianggap sebagai objek, maka kita bisa menjalankan sintak lainnya, seperti `print(dataPertama)`. Namun, jika objek sintak `print()` tersebut tidak ada, maka kita akan mendapatkan error, yang menyebutkan objek print tidak ditemukan.

Hal lain yang juga terdapat pada bagian ini adalah **History** dan **Connection**. **History** akan menunjukkan skrip apa saja yang telah dijalankan dalam satu sesi penggunaan R. Satu sesi penggunaan dimulai pada saat kita membuka R. Sedangkan bagian **Connection** bisa digunakan untuk menyambungkan R dengan aplikasi lain, seperti hadoop, spark dan aplikasi sejenis lainnya untuk menyimpan data. Terakhir, dengan menggunakan **Rstudio**, kita juga bisa berkolaborasi dengan orang lain dengan menggunakan git, misalnya dengan menggunakan **gitlab** atau **github**.

2.2.3 Bagian File/Plots/Packages/Help/Viewer

Pada bagian ini, kita akan menemukan beberapa hal yang memiliki fungsi berbeda-beda. Berikut adalah penjelasan fungsi beberapa bagian tersebut.

1. File

Bagian **File** secara otomatis akan menuju pada folder di mana kita menyimpan proyek analisis menggunakan R. Cara membuat proyek akan dibahas pada materi tentang **Menulis Skrip di R**. Pada menu ini pula, kita akan bisa melihat data, skrip, plot, dan hal lain

yang kita simpan dalam folder yang sama.

2. Plots dan Viewer

Menu **Plots** pada bagian ini akan menunjukkan gambar yang dihasilkan oleh sintak atau fungsi untuk membuat plot, misalnya sintak `plot(...)`. Contohnya adalah **Gambar hasil dari sintak `plot()`** yang dihasilkan dari skrip `plot(mtcars)` (Coba jalankan sintak tersebut pada **Console**). Di mana gambar tersebut akan muncul pada bagian **Plots** dan bisa di perbesar dengan **Zoom** dan atau disimpan menggunakan menu **Export**.

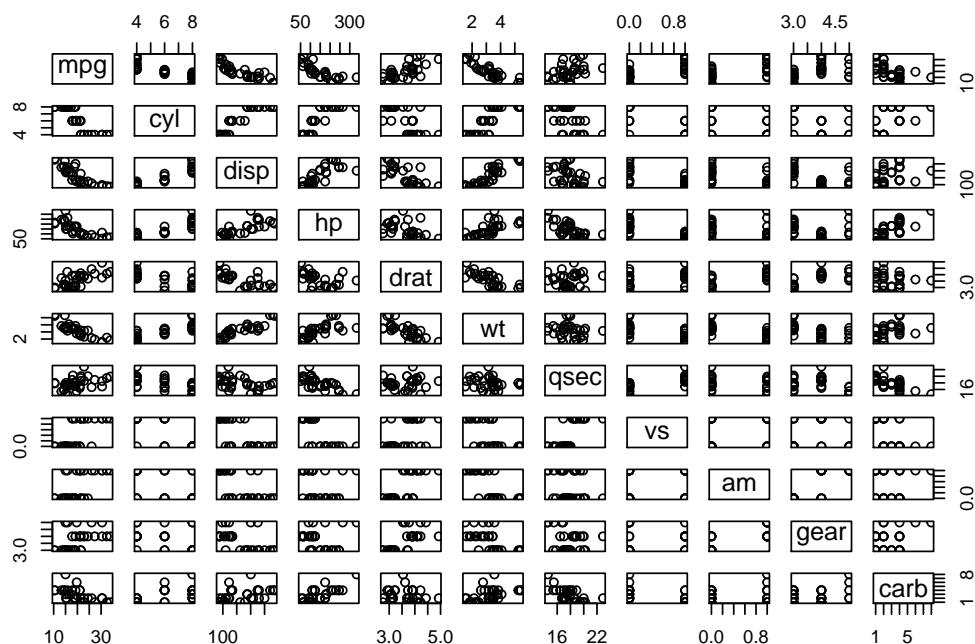


Figure 3: Gambar hasil dari sintak `plot()`

Sama seperti menu **Plots**, viewer juga berfungsi untuk menampilkan hasil sintak visualisasi. Perbedaannya, pada menu viewer akan menampilkan hasil visualiasi dari non-base sintak atau sintak yang bisa digunakan tanpa perlu memanggil **library** terlebih dahulu. Sementara **Viewer** akan menampilkan hasil visualisasi skrip selain non-base. **Viewer** ini bisa menampilkan bukan hanya plot, tapi juga visualisasi html. Berikut ini adalah contohnya.

```
library(ggplot2)  
library(tidyverse)
```

```
mtcars %>%  
  select(cyl, disp) %>%  
  ggplot(aes(x = cyl, y = disp)) +  
  geom_point()
```

Skrip di atas menggunakan fungsi dari ggplot untuk membuat visualisasi. Untuk bisa menjalankan skrip di atas, kita perlu terlebih dahulu menginstall packages `tidyverse`. Hal tersebut bisa digunakan dengan menggunakan sintaks `install.packages("nama_packages")`.

3. Packages

Pada menu **Package** kita bisa melihat daftar package atau library yang sudah diinstall. Setelah kursor diarahkan pada menu tersebut, kita bisa melihat nama, deskripsi, dan versi library masing-masing. Pada menu ini pula, kita bisa meng-uninstall sebuah package dengan menekan tombol x di menu tersebut.

Untuk menginstall sebuah packages kita bisa menggunakan fungsi atau sintak `install.packages(...)`. Selain itu, kita juga bisa menginstall melalui menu Tools pada bagian atas dengan urutan **Tools** > **Install Packages** > **Nama Package**. Di sini kita bisa memilih untuk menginstall langsung dari cran atau menginstall package yang sudah di unduh terkebih dahulu. Selain itu, parameter terakhir pada saat menginstall package adalah dependency yang secara umum lebih baik dibiarkan tercentang.

4. Help

Menu Help digunakan untuk melihat bantuan yang bisa didapatkan di R. Bantuan di sini pada umumnya terkait dengan sintak atau fungsi dari sebuah paket. Di mana penjelasan tentang sebuah fungsi bisa didapat dengan menggunakan sintak `?namafungsi`. Misalnya:

```
?ggplot  
?select
```

Dengan menjalankan skrip diatas, menu help akan menampilkan dokumentasi terkait

dengan `ggplot` dan `select`. Di mana kedua fungsi tersebut merupakan bagian dari fungsi yang ada dalam packages di library tidyverse. Dengan membaca dokumentasi tersebut, kita bisa menggunakan fungsi dengan sesuai dan diharapkan tidak terjadi error.

2.3 Latihan 2

Berdasarkan penjelasan pada [bagian Menggunakan R](#), jawablah pertanyaan-pertanyaan berikut:

1. Sebutkan dan jelaskan 3 bagian utama yang ada pada saat kita pertama kali membuka Rstudio beserta dengan fungsinya.
2. Pada bagian apakah kita bisa mendapatkan informasi pada saat terjadi error?
3. Apa yang akan ada dibagian `Console` dan `Environment` pada saat kita menjalankan skrip berikut?

```
jml_siswa = 10 + 17
```

4. Lengkapi skrip dibawah ini untuk membuat objek di R dengan nama `dataKedua`.

```
- - - = 78 * 17
```

5. Sintak apa yang bisa digunakan untuk mengetahui dokumentasi dari sebuah fungsi yang ada dalam sebuah packages di R?
6. Jelaskan perbedaan menu `Plots` dan `Viewer` di Rstudio.

2.4 Menulis Skrip di R

Sebelum kita mulai belajar menulis skrip, kita akan terlebih dahulu mengatur tempat dimana skrip tersebut akan disimpan. Di R, hal tersebut disebut dengan project. Project adalah nama lain dari folder tempat di mana kita menyimpan semua hal terkait dengan analisis yang akan dilakukan. Pembuat folder khusus ini menjadi penting jika kita akan menggunakan analisis lain atau menganalisis merupakan pekerjaan rutin.

Untuk membuat project di R bisa dimulai dengan mengikuti langkah-langkah berikut:

1. Pilih **File > New Project > New Directory**
2. Buat nama Directory/Folder
3. Pilih folder tempat menyimpan directory yang dibuat
4. Tekan/pilih **Create Project**

Dengan mengikuti empat langkah di atas, kita telah membuat satu folder baru di dalam komputer kita. Di folder tersebutlah nantikan kita akan menyimpan semua file terkait dengan proyek analisis yang kita kerjakan. Ketika kita membuat proyek baru, sebaiknya kita membuat directory baru agar filenya tidak tercampur dan mudah untuk melakukan penelusuran.

Setelah kita memiliki proyek di R, kita sudah siap untuk mulai menulis skrip atau kode-kode dalam R yang bisa dibaca oleh komputer untuk melakukan analisis data. Untuk memulai menulis skrip, kita harus terlebih dahulu membuat file baru. File umum yang bisa dibuat di R ataupun Rstudio adalah **.R** dan **.rmd**. Pada kesempatan kali ini, kita akan terlebih dahulu mempelajari cara menulis skrip di file **.R** (dot R). Untuk file **.R** (dot R) kita bisa mengetik langkah berikut:

File > New File > R Script

Langkah di atas akan memberikan kita sebuah halaman baru bernama **Untitled1**, setelahnya kita bisa menyimpannya terlebih dahulu dengan menekan **Cmd+Save** atau **Ctr+Save**. Berilah nama file tersebut, misalnya **latihan1**. File **latihan1.R** tersebut secara otomatis telah tersimpan dalam folder proyek yang dibuat di awal.

Sebelum mulai menulis, terdapat beberapa hal mendasar yang perlu diketahui terlebih dahulu, yaitu:

1. Tanda Pagar (#);

Tanda Pagar (#) digunakan untuk memberikan komentar. Artinya, setiap skrip yang diawali dengan tanda tersebut tidak akan dibaca sebagai sebuah perintah oleh komputer. Contohnya bisa dilihat pada skrip berikut.

```
# Memanggil library tidyverse  
library(tidyverse)
```

Pada saat kita mengawali tulisan dengan tanda pagar, umumnya akan menghasilkan tulisan miring (*italic*) atau berwarna beda dengan tulisan lainnya. Seperti contoh di atas, kalimat *Memanggil library tidyverse* tercetak miring, sementara `library(tidyverse)` tercetak biasa. Artinya, skrip diatas, hanya memberikan perintah untuk memanggil package `tidyverse` dengan sintak `library()`. Fungsi menulis komentar di sini, adalah untuk memberikan penjelasan bagi kita sendiri, atau mungkin orang lain yang akan menggunakan skrip yang kita tulis.

Dalam beberapa kasus, penggunaan dan penulisan keterangan sangat dianjurkan dan bahkan lebih banyak dan panjang dibandingkan dengan skripnya sendiri. Hal ini, untuk memudahkan bukan hanya orang lain, tapi juga kita sendiri ketika suatu waktu harus membuka kembali skrip yang sudah lama kita buat.

2. Tanda Arah (<-) atau Sama Dengan (=)

Tanda `<-` atau `=` memiliki arti yang sama, yaitu untuk memasukkan sebuah perintah menjadi sebuah objek. Objek yang dimaksud di sini adalah sesuai yang akan muncul di bagian environment setelah skrip itu dijalankan atau dieksekusi. Contoh penggunaannya adalah sebagai berikut:

```
data_ke_1 <- 30  
data_ke_2 = 80  
  
data_ke_3 <- data_ke_1 / data_ke_2 * 100  
  
data_ke_1 > data_ke_2 # lebih dari  
data_ke_1 >= data_ke_2 # lebih dari atau sama dengan  
  
data_ke_1 < data_ke_2 # kurang dari  
data_ke_1 <= data_ke_2 # kurang dari atau sama dengan
```

```

data_ke_1 == data_ke_2 # sama dengan
data_ke_1 != data_ke_2 # tidak sama dengan

```

Skrip diatas, pada saat yang bersamaan menunjukkan: (1) Kemampuan R sebagai kalkulator; dan (2) Cara menulis dan membuat objek di R. Untuk mempermudah memahaminya, kita bisa membaca skrip dari sisi kanan. Sehingga ketika kita membuat sebuah objek, maka nama di sisi kiri tanda = atau <- merupakan nama objek dan bisa di cek pada bagian environment, dan yang ada di sisi kanan merupakan isinya.

3. Spasi dan tanda Dollar (\$)

Sebagian besar bahasa pemrograman tidak menggunakan spasi. Ada banyak alasan yang membuatnya tidak digunakan. Namun, yang perlu dipahami di sini dalam konteks R adalah, spasi tidak digunakan atau dilarang digunakan untuk menamai sebuah objek. Artinya, semua objek di R harus ditulis tanpa spasi. Misalnya, kita akan menamai objek kita `data_ke_1`, maka hal yang harus ditulis adalah `datake1` atau `data_Ke1`.

Selain untuk nama objek, walaupun dianjurkan untuk tidak menggunakan, spasi tetap masih bisa digunakan. Misalnya untuk menamai sebuah kolom dan atau isi sebuah kolom. Seperti yang bisa dilihat pada tabel di bawah ini.

Table 1: Contoh tabel dengan nama kolom yang menggunakan spasi

Title	Type	Start airing	Starting season
Fullmetal Alchemist: Brotherhood	TV	2009-4-5	Spring
Kimi no Na wa.	Movie	2016-8-26	-
Gintama°	TV	2015-4-8	Spring
Steins;Gate 0	TV	2018-4-12	Spring
Steins;Gate	TV	2011-4-6	Spring
Ginga Eiyuu Densetsu	OVA	1988-1-8	-

Tabel 1 menunjukkan bahwa nama kolom tetap bisa menggunakan spasi. Tapi untuk mengaksesnya, kita tidak bisa hanya dengan nama kolom tersebut secara langsung. Untuk mengakses konten dari sebuah kolom yang menggunakan spasi memerlukan tanda petik. Misalnya, tabel di atas bernama `daten`. Maka untuk mengakses kolom `Title` kita hanya perlu menggunakan nama data + tanda dollar (\$) + nama kolom (`daten$title`). Tapi untuk mengakses kolom `Start airing` kita bisa mengaksesnya dengan menambahkan tanda ('') pada nama kolom seperti contoh berikut:

```
daten$`Start airing`
```

Sampai bagian ini, kita sudah mempelajari beberapa hal untuk bisa mulai menulis sebuah skrip R menggunakan Rstudio. Namun sebelum melanjutkan ke bagian berikutnya, cobalah jawab beberapa pertanyaan pada bagian [latihan 3](#).

2.5 Latihan 3

Berdasarkan penjelasan pada bagian [Menulis Skrip di R](#), jawablah pertanyaan-pertanyaan berikut:

1. Sebutkan urutan membuat proyek analisis di R/Rstudio
2. Apakah fungsi tanda = sama dengan <-?
3. Apa yang ada di sisi sebelah kiri dan sebelah kanan tanda <- atau =?
4. Tuliskan tanda yang menunjukkan lebih dari, lebih dari atau sama dengan, kurang dari, kurang dari atau sama dengan, sama dengan, dan tidak sama dengan.
5. Cobalah untuk menjalankan skrip berikut:

```
install.packages("devtools")
devtools::install_github("eppofahmi/keData")
library(keData)

daten <- keData::dataAnime
```

6. Berdasarkan skrip di atas, di dalam environment kita sudah memiliki sebuah objek

dengan nama `daten`. Lengkapi skrip berikut untuk mengakses kolom ke-7 (Starting season).

```
dataku <- daten$...
```

2.6 Konsep dan Fungsi Network Analysis

Network analisis sebenarnya bukan hal baru sebagai sebuah metode analisis. Dalam konteks ilmu sosial, sosiolog seperti Georg Simmel and Émile Durkheim pernah menulis tentang pentingnya mempelajari pola hubungan antar aktor sosial. Secara umum, ilmuwan sosial telah menggunakan istilah *social network* sejak awal abad 20. Walaupun dalam kenyataannya baru dalam beberapa tahun terakhir saja Social Network Analysis (SNA) lebih banyak dikenal publik.

SNA dapat didefinisikan sebagai sebuah proses investigasi struktur sosial melalui jejaring dan teori graph. Jejaring (*Network*) tersebut mengkarakteristikkan struktur terkoneksi yang selanjutnya biasa disebut **nodes** (bisa berupa aktor individu, masyarakat, atau sesuai di dalam jejaring) dan garis, penghubung, atau koneksi yang menghubungkan antar nodes yang selanjutnya biasa disebut sebagai **edges**.

Oleh karena itu, secara sederhana SNA dapat diartikan sebagai sebuah analisis terhadap keberadaan **nodes** dan **edges** yang menghubungkannya. Untuk itu pula, sebagai tahap pertama dalam proses pembuatan network yang harus diketahui pertama adalah apa atau siapa **nodes** dan bagaimana hubungannya atau **edges**nya. Sebagai ilustrasi, perhatikanlah contoh Twit pada Tabel 2. Di mana dalam tabel tersebut terdapat dua kolom yang menunjukkan pengirim, yaitu kolom `username` dan kolom `text`, yang berisi kalimat serta text lain yang bisa dijadikan sebagai salah satu indikator username lain, yaitu tanda @.

Berdasarkan data yang terdapat dalam Tabel 2 tersebut, kita bisa membuat sebuah analisis sosial network dengan memanfaatkan keberadaan Username Twitter yang dimention dengan pengirimnya. Username Twitter selalu ditandai dengan adanya awalan tanda @. Artinya, tanda tersebut bersifat konsisten. Oleh karena itu pula tanda @ bisa dijadikan sebagai parameter dalam melakukan pre-processing datanya nanti.

Table 2: Contoh data untuk menentukan nodes dan edges

username	text
@Interna_Bali	H-7! Bali Allergy Immunology Update 2019. @blogdokter @BaleBengong @UdayanaUniv @oppie85 @depri_bali @fk @KemenkesRI @kolegiumipd @Ariatesta @wira_n @iMaiMacHaN @aryad_p @yubai. PIC @winadharhestipic.twitter.com/r8rvY60iUF – at Ruang Teater Widya Sabha Usadha Fakultas Kedokteran Universitas Udayana
@HafizYusaryaha	I seriously considering some type of punishment for someone encouraging other people not to be vaccinated, especially if that someone is a public figure. I mean, seriously, it's a global threat. It doesn't have any rational bg whatsoever. #antivaxx #VaccinesWork @KemenkesRI

Berdasarkan Tabel 2, kita bisa membuat skenario untuk analisis network misalnya: Username dari kolom `username` dan `text` dianggap sebagai **Nodes** dan **Edges** atau link yang menghubungkan keduanya adalah mention. Oleh karena itu, username @HafizYusaryaha -> (baca: @ke-1 terhubung dengan @ke-2) @KemenkesRI. Lebih lengkapnya bisa dilihat pada Tabel 3. Di mana username dalam kolom text diubah menjadi satu token di kolom dengan mention.

Table 3: Data untuk network analisis berdasarkan mention Twitter

username	mention
@HafizYusaryaha	@KemenkesRI
@Interna_Bali	@blogdokter
@Interna_Bali	@BaleBengong
@Interna_Bali	@UdayanaUniv

@Interna_Bali	@oppiet85
@Interna_Bali	@depri_bali
@Interna_Bali	@fk
@Interna_Bali	@KemenkesRI
@Interna_Bali	@kolegiumipd
@Interna_Bali	@Ariatesta
@Interna_Bali	@wira_n
@Interna_Bali	@iMaiMacHaN
@Interna_Bali	@aryad_p
@Interna_Bali	@yubai
@Interna_Bali	@winadharmestipic

Tabel 3 menunjukkan cara mempersiapkan data untuk melakukan analisis network. Jika kembali ke Tabel 2, pada baris pertama kita melihat di sana terdapat beberapa usrename. Oleh karena itu, pada Tabel 3 kita menemukan bahwa @Interna_Bali memention beberapa nama lain dalam bari yang berbeda. Sementara bari kedua di Tabel 2, hanya ada satu username yang di mention. Oleh karena itu pula, hanya ada @HafizYusaryahya sekali di (satu baris) di Tabel 3.

Dua kolom inilah yang menjadi dasar awal kita melakukan analisis *social network*. Untuk itu, setiap pre-processing yang dilakukan tahap awalnya adalah membuat dua kolom tersebut, di mana ada kolom sumber dan kolom target. Dalam konteks data Twitter yang didefinisikan di atas, kolom sumber berasal dari username yang membuat postingan. Sementara targetnya adalah username yang di mention dalam tubuh twitnya. Data dengan dua kolom seperti Tabel 3 ini juga dapat disebut sebagai **Adjacency List**.

Setelah memiliki **Adjacency List**, kita sudah bisa menggunakan Gephi untuk melakukan analisis network. Di mana kita bisa melakukan pemetaan tentang urgensi masing-masing aktor atau nodes dalam sebuah jaringan dan juga mendeteksi komunitas yang ada di dalamnya.

Secara umum SNA digunakan untuk memetakan interaksi antar aktor, seperti telah disebut sebelumnya. Sementara secara praktis, misalnya di sektor publik, SNA dapat digunakan

untuk melihat dan merancang strategi komunikasi pemimpin, analisis keterlibatan individu atau kelompok dalam penggunaan penggunaan media, serta pemecahan masalah berbasis masyarakat dengan mendeteksi aktor-aktor penting di dalamnya berdasarkan struktur dalam network.

Hal yang tidak mungkin kita abaikan dalam sektor publik terkait dengan pemanfaatan SNA adalah untuk melakukan tracking ancaman terhadap negara. Misalnya untuk mendeteksi aktor teror yang selama ini telah dipraktikkan oleh negara maju, misalnya Amerika Serikat melalui The National Security Agency (NSA), yang telah menggunakan SNA untuk memetakan jejaring organisasi dan aktor terorisme melalui data yang didapat dari media sosial, call detail records (CDRs), dan beberapa sumber lain.

Di sektor swasta, SNA juga biasanya dimanfaatkan untuk: mendukung interaksi dengan customer, analisis informasi sistem pengembangan. Tidak kalah pentingnya, sektor swasta juga bisa dan banyak yang menggunakan SNA untuk mendukung pemasaran, divisi bisnis intelejen dengan menganalisis media sosial.

Beberapa contoh di atas mungkin dilakukan karena dengan SNA, kita bisa mengetahui posisi dan peran penting sebuah nodes atau aktor. Selain itu, dengan SNA pula kita bisa mendapatkan informasi tentang pola hubungan yang terjadi. Misalnya terkait dengan jejaring terorisme, NSA bisa dengan mudah menarget orang yang menjadi penghubung atau memiliki betweenness centrality tinggi untuk memutus alur informasi. Sementara, sektor publik juga bisa melakukan pendekatan langsung terhadap aktor penting di masyarakat berdasarkan SNA. Dengan demikian, pemerintah bisa mendengar lebih jelas tentang apa yang diinginkan oleh masyarakat dan bisa memutuskan sebuah tindakan yang tepat.

Untuk itu pula, pada bagian selanjutnya kita akan mempelajari fitur-fitur yang ada dalam sebuah jaringan dan membahas beberapa diantaranya yang paling umum dan sering digunakan.

2.7 Latihan 4

Berdasarkan penjelasan pada [bagian Konsep Network Analysis](#), jawablah pertanyaan-pertanyaan berikut:

1. Jelaskan pengertian Social Network Analysis?
2. Buatlah skenario network dari Twitter dengan skema retweet. Tentukan siapa yang jadi nodes sumber dan targetnya.
3. Buatlah contoh **Adjacency List**.
4. Sebutkan beberapa fungsi SNA yang anda ketahui?

3 Memahami Centrality dan Modularity dalam *Network* (3 Jam)

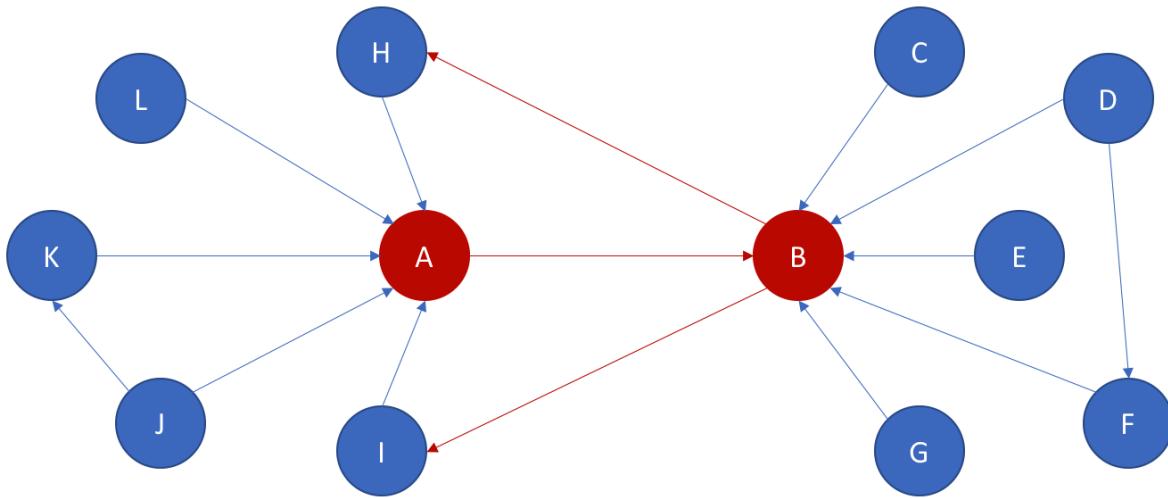
Centrality (*centrality measurement*) di dalam network merupakan salah satu parameter pengukuran utama dan paling sering digunakan. Centrality dapat digunakan untuk mendapatkan informasi misalnya tentang posisi sebuah nodes. Apakah ia bisa menjadi penghubung, atau bahkan influencer terhadap nodes lainnya.

Selain itu, salah satu yang juga cukup menarik dalam analisis jejaring adalah modularity, di mana hasil akhirnya dapat digunakan sebagai salah satu parameter utama untuk mendeteksi keberadaan komunitas di dalam jejaring. Beberapa parameter yang ada di dalam jejaring tersebut akan dibahas dalam beberapa bab berikut.

3.1 Degree Centrality

Degree centrality menunjukkan jumlah koneksi yang dimiliki sebuah node. Jika jejaringnya terarah atau biasa disebut *directed network* maka terdapat dua parameter degree centrality, yaitu *indegree* dan *outdegree*. *Indegree* adalah sebuah perhitungan yang menunjukkan jumlah koneksi yang mengarah pada sebuah nodes. Sementara *outdegree* menunjukkan jumlah koneksi dari sebuah nodes yang mengarah pada nodes-nodes lain. Namun dalam beberapa kasus, degree centrality juga dapat berupa jumlah antara *indgree* dan *outdegree*.

Sebagai ilustrasi coba perhatikan Gambar 4, di mana dalam gambar tersebut ada nodes-nodes yang saling terhubung. Berdasarkan Ilustrasi tersebut kita bisa ketahui nodes mana/apa yang memiliki nilai degree centrality tertinggi.



A Indegree centrality = 5
 A Outdegree centrality = 1
 A Degree centrality = ?

B Indegree centrality = 6
 B Outdegree centrality = 2
 B Degree centrality = ?

Figure 4: Ilustrasi degree centrality

Nodes A, berada diantara nodes C, D, E, F, G, dan nodes A. Sementara Nodes A juga berada diantara nodes I, J, K, L, M, dan H. Namun jika diperhatikan, nodes-nodes tersebut dihubungkan dengan garis dengan arah. Artinya, network ini berupa *directed network*. Oleh karena itu, degree centrality network ini ada tiga, yaitu inderee, outdegree, dan bisa degree centrality yang merupakan hasil penambahan IN dan OUT *degree*.

3.2 Betweenness Centrality

Centrality selanjutnya yang juga sering digunakan adalah betweenness centrality, yaitu sebuah centrality yang menunjukkan posisi sebuah nodes sebagai penghubung atau menjadi *bottleneck*. Nodes yang menjadi communication bottleneck di sini dianggap penting karena arus informasi dapat dipastikan melalui nodes tersebut. Parameter ini juga dapat digunakan untuk mengidentifikasi boundary spanners, yaitu orang atau node yang berperan sebagai penghubung (jembatan) antara dua komunitas. Betweenness centrality sebuah node dihitung dengan menjumlahkan semua shortest path yang mengandung node tersebut.

Sebagai ilustrasi, mari kita melihat dan menganggap persimpangan jalan sebagai sebuah node. Maka, semakin banyak jalan yang harus melewati persimpangan tersebut, maka semakin penting posisi persimpangan tersebut. Karena penting dan ramai, harga tanah atau ruko di daerah persimpangan pun kemudian menjadi lebih tinggi dan fasilitas disekitarnya pun kemungkinan lebih baik dibanding di jalan-jalan biasa. Hal tersebut terjadi karena jika, misalnya rambu lalu lintas di persimpangan mati, maka dampaknya akan dirasakan di jalan-jalan atau bahkan daerah lain.

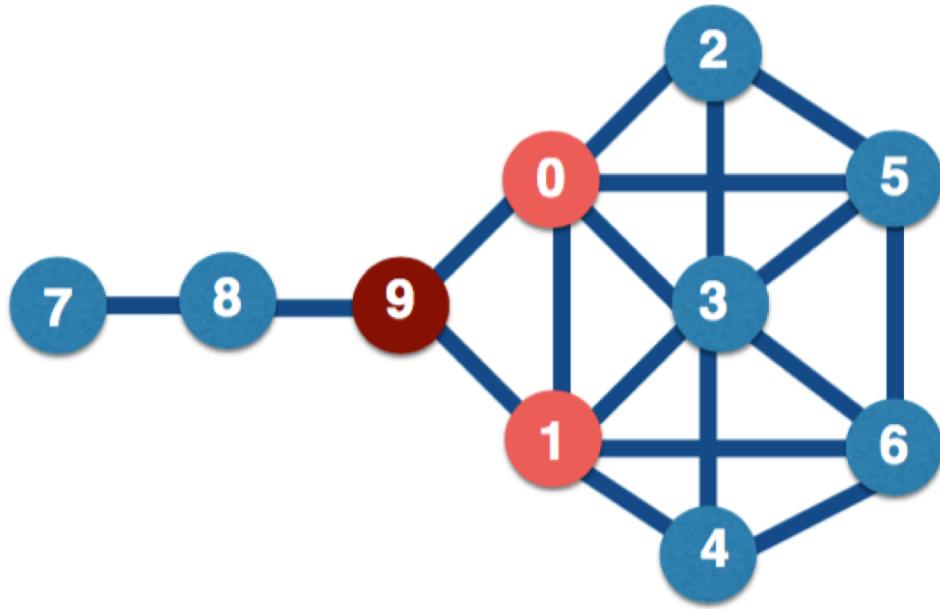


Figure 5: Ilustrasi betweenness centrality

Istilah lain, yang juga biasa digunakan untuk menggambarkan betweenness adalah jembatan. Dimana dari [Gambar 5](#) diketahui bahwa ada tiga nodes mungkin memiliki tingkat betweenness tertinggi, yaitu 0, 1, dan 3 yang menjadi jembatan nodes lain untuk terhubung. Jika tidak ada nodes tersebut tidak ada, maka kemungkinan network juga akan memiliki bentuk yang berbeda.

3.3 Closeness centrality

Closeness centrality menunjukkan jarak rata-rata antara node dengan semua node yang lain di jaringan. Ukuran ini menggambarkan kedekatan node sebuah dengan node lain. Semakin dekat, semakin terhubung nodes tersebut dengan lainnya. Jika nodes tersebut

merupakan orang, maka orang yang memiliki nilai closeness tinggi dapat disebut sebagai orang gaul yang kenal dengan banyak orang. Oleh karena itu, ia dapat menyebarkan informasi lebih cepat tanpa perantara pihak ketiga.

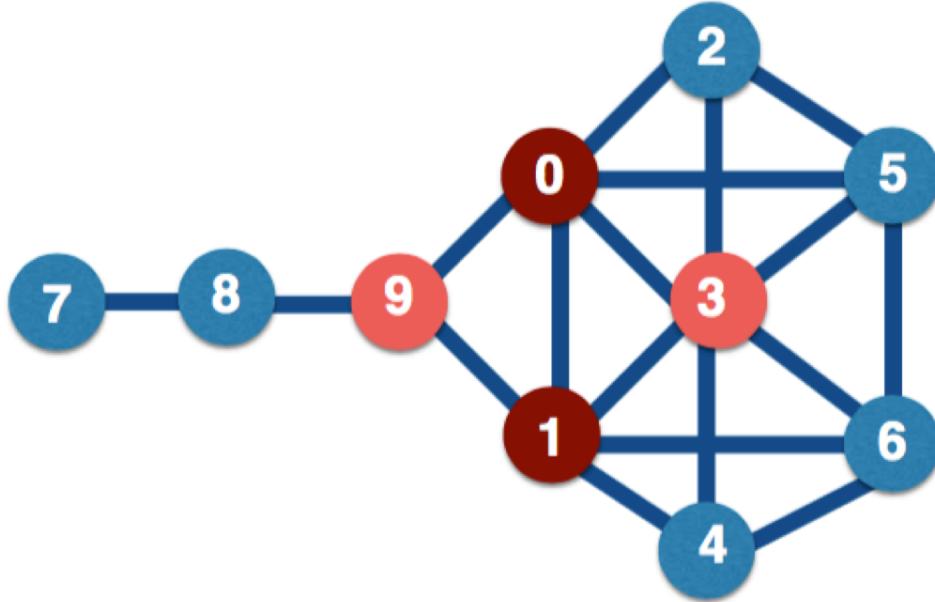


Figure 6: Ilustrasi closeness centrality

Dari Gambar [Gambar 6](#) diketahui bahwa nodes yang memiliki jarak terpendek dengan nodes lain adalah nodes 0, 9, 1, dan 3. Hal tersebut dikarenakan keempat nodes tersebut terhubung secara langsung dengan nodes lainnya. Coba bandingkan dengan, misalnya nodes 8, atau 2. Kedua nodes tersebut hanya terhubung dengan masing 2 dan 3. Sementara keempat nodes yang disebut diawal terhubung dengan lebih dari atau sama dengan 3 nodes lainnya.

3.4 Eigenvector Centrality

Eigenvector centrality menunjukkan bobot yang lebih tinggi pada node yang terhubung dengan node yang juga memiliki keterhubungan tinggi. Dapat dikatakan versi rekursif dari degree centrality. Selain itu, eigenvector juga menunjukkan nodes yang banyak terhubung dengan nodes lain yang juga memiliki pengaruh atau penting. Sebagai ilustrasi, cobalah perhatikan ilustrasi di Gambar 7, di mana angkanya menunjukkan nilai eigenvector nodes.

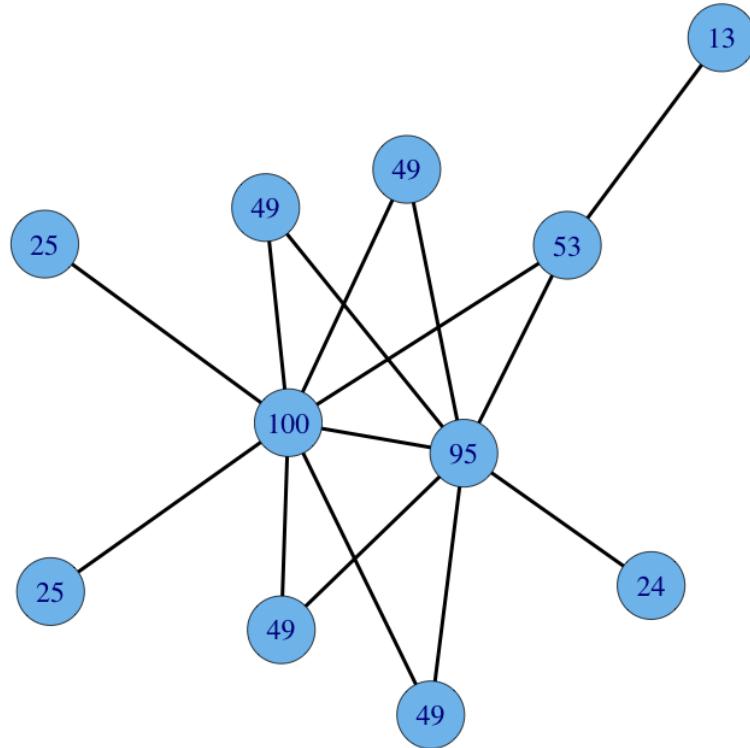


Figure 7: Ilustrasi eigenvector centrality

Berbeda dengan degree centrality, di mana nodes yang paling banyak terhubung dengan nodes lain akan memiliki nilai yang tinggi. Sementara, nodes yang memiliki eigenvector centrality tinggi belum tentu memiliki nodes terhubung yang banyak. Hal ini dikarenakan di dalam eigenvector kita juga menilai nodes apa yang terhubung, sehingga eigenvector juga dapat disebut sebagai nodes penting karena terhubung dengan nodes-nodes penting lainnya dalam *network*.

3.5 PageRank

PageRank dibuat oleh Google untuk menyesuaikan dan mengatasi masalah yang mungkin muncul dari Katz centrality. Di mana jika nodes dengan tingkat centrality tinggi terhubung dengan nodes lain, maka semua nodes yang terhubung dengan nodes tersebut juga akan

mendapatkan centrality tinggi. Padahal dalam banyak kasus, hal tersebut tidak berlaku karena belum tentu nodes yang saling terhubung sama-sama penting dalam sebuah network.

PageRank menggunakan tiga faktor berbeda dalam menentukan tingkat sebuah nodes, yaitu: (1) Jumlah tautan yang diterima; (2) Kecenderungan tautan dari penghubung; (3) Tingkat centrality dari nodes yang terhubung. Faktor pertama mungkin jelas, di mana semakin banyak nodes yang terhubung, semakin tinggi pula tingkat centrality nodesnya.

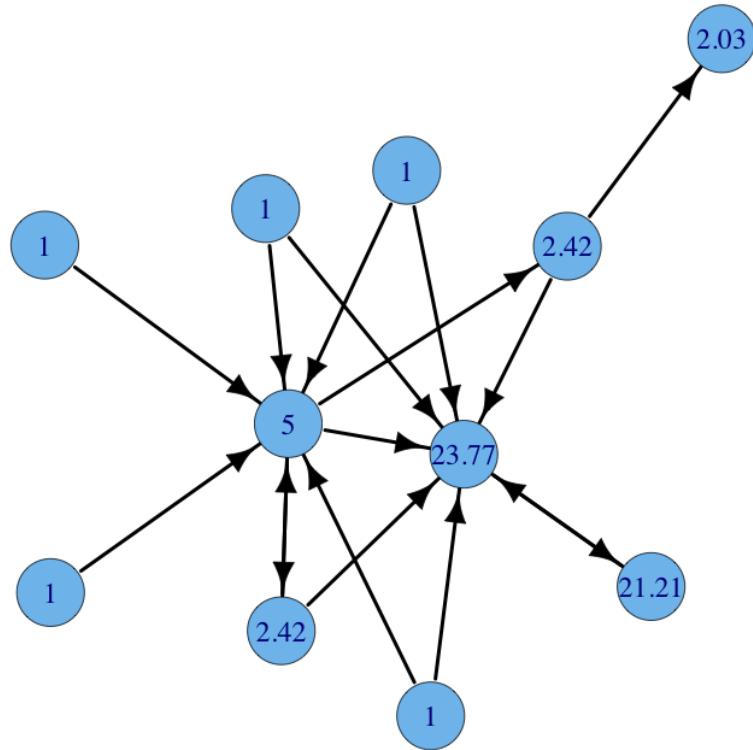


Figure 8: Ilustrasi PageRank centrality

PageRank dapat digunakan untuk jaringan yang berbentuk graph berarah (*directed network*). Prinsip yang digunakan adalah semakin penting sebuah node, maka semakin banyak node tersebut di-*refer* oleh node lain. Contohnya, situs seperti kompas.com punya peringkat yang tinggi karena banyak di-*refer* di page lain. Selain itu, jika ada sebuah blog yang misalnya, juga di-*refer* oleh kompas.com, maka peringkat blog ini juga akan naik.

3.6 Modularity

Parameter selanjutnya yang sering digunakan pada saat melakukan Social Network analisis adalah modularity yang menunjukkan kedekatan jarak sebuah nodes dengan nodes lainnya secara bersamaan. Oleh karena itu pula modularity sering digunakan untuk melihat keberadaan kelompok dalam sebuah jejaring.

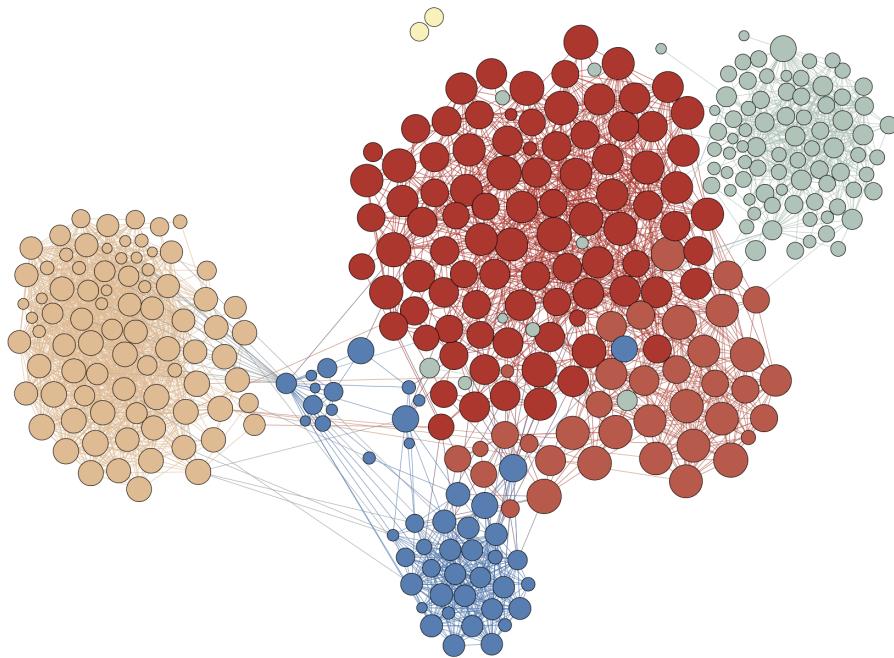


Figure 9: Ilustrasi Hasil Visualisasi Modularity

Gambar 9 dihasilkan menggunakan aplikasi gephi, di mana warna menunjukkan kelompok atau nodes yang memiliki kedekatan jarak. Sementara besaran nodes menunjukkan centrality dari masing-masing nodes. Berdasarkan Gambar 9 tersebut, setidaknya kita bisa mendapatkan dua informasi, yaitu terkait dengan kemungkinan jumlah komunitas atau kelompok serta nodes apa yang memiliki centrality tinggi di masing-masing kelompok yang ada.

3.7 Latihan 5

Berdasarkan penjelasan pada [bagian Memahami Centrality dalam Network](#), jawablah pertanyaan-pertanyaan berikut:

1. Apa fungsi dari centrality secara umum?
2. Jelaskan masing-masing centrality yang ada dan dijelaskan dalam modul!
3. Jelaskan fungsi modularity dan sebutkan indikator-indikator yang bisa dilihat dari Gambar 9!
4. Centrality apa yang cocok untuk mengetahui nodes yang menjadi penghubung didalam sebuah jejaring?
5. Centrality apa yang cocok untuk mengetahui nodes yang menjadi influencer didalam sebuah jejaring?

4 Pre-processing untuk Network Analysis (3 Jam)

Setelah mengetahui fungsi dari R dan Rstudio pada bagian pertama, modul ini selanjutnya juga menjelaskan beberapa konsep centrality dan parameter yang sering digunakan pada saat melakukan Social Network Analysis. Bagian ini dan selanjutnya akan lebih banyak membahas teknis.

Pre-processing sendiri merupakan salah satu tahapan yang akan menentukan hasil analisis yang dilakukan, terlepas dari metode yang akan digunakan oleh analisi. Dalam melakukan pre-processing ini, kita akan melibatkan bukan hanya keterampilan menulis skrip atau kode, tapi juga melibatkan pengetahuan umum atau teoretis bidang ilmu tertentu.

Misalnya, pada saat melakukan analisis konten, kita membutuhkan pengetahuan linguistik selain juga pemahaman tentang Natural Language Processing. Karena pada tahap ini kita akan menentukan apakah term/kata stopwords akan dihilangkan, apakah normalisasi atau bahkan translasi perlu dilakukan.

Umumnya, dalam pelaksanaan analisis data justru proses data pre-processing inilah yang akan memakan waktu yang lebih lama dibanding proses lainnya. Hal ini dikarenakan pada tahap ini, kita akan mengerjakan percobaan yang ketika hasilnya belum sesuai dengan syarat untuk bisa dilakukan analisis perlu diulang kembali hingga mencapai kualitas yang ditentukan atau sudah sesuai dengan kebutuhan analisis.

Kebutuhan analisis di sini akan banyak ditentukan oleh tujuan dan metode analisis yang

dugunakan. Untuk analisis konten, data dianggap sudah siap di analisis, misalnya setelah teks yang ada didalamnya telah bisa dinormalkan 80 persen. Artinya, ketika dicocokkan dengan kamus, hanya ada dua puluh persen term saja dari keseluruhan yang tidak ada dalam kamus. Selain itu, misalnya kita menggunakan data yang bersumber dari media sosial, data teks dianggap siap dianalisis jika tidak ada term yang diawali tanda @ atau # dan seterusnya. Hal-hal seperti itulah yang perlu kita tentukan terlebih dahulu sebelum melakukan *pre-processing*.

4.1 Impor dan Ekspor Data

Sebelum bisa melakukan pre-processing, kita harus terlebih dahulu mengimpor atau memasukkan data yang akan diolah ke dalam Rstudio. Jika kita telah mengikuti langkah pada bagian [Menulis Skrip](#), maka kita bisa mengarahkan *mouse* pada data yang dituju dan pilih **impor datasets** hingga tampil layar seperti pada Gambar 10.

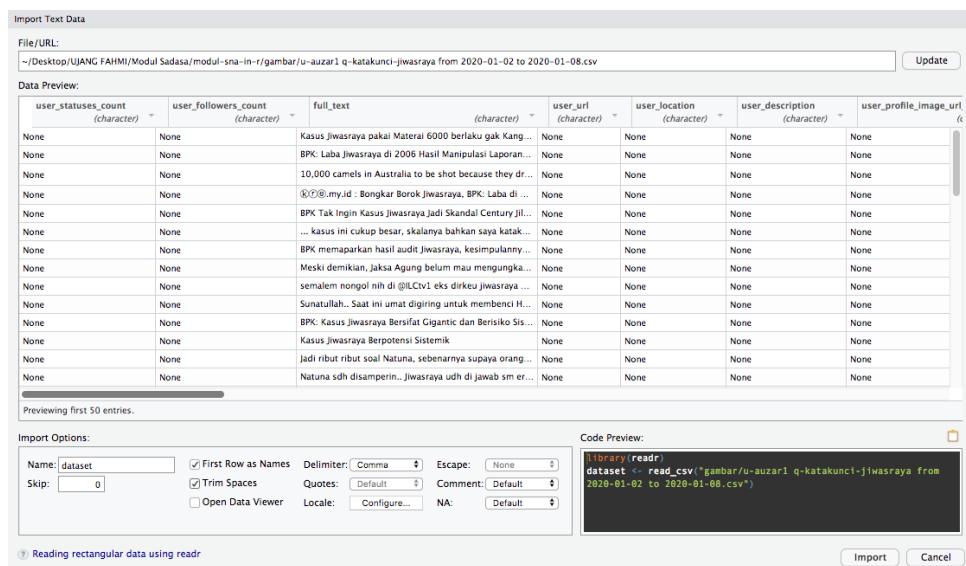


Figure 10: Ilustrasi Cara Mengimpor Data

Setelah visual seperti yang terlihat pada Gambar 10 muncul, kita bisa menekan tombol impor. Di dalam contoh gambar tersebut, kita perhatikan arah pojok kanan bawah, di mana di sana ada library, dan ada dataset. Artinya, untuk mengimpor data dengan format .csv, kita perlu menggunakan fungsi `read_csv()` dari package `readr`. Data yang diimpor selanjutnya disebut atau diberi nama `dataset`, dengan demikian pada bagian

environment akan muncul dataset.

Masih pada layar seperti pada Gambar 10, kita juga bisa memberi nama yang berbeda dengan mengisi atau mengganti isian pada kolom `name`, isian `Skip` digunakan untuk melewati kolom berdasarkan urutan nomor kolom dari kiri ke kanan. Parameter delimiter disini digunakan untuk menentukan pembatas antar kolom, dalam konteks ini karena csv pada umumnya pembatasnya koma, maka isian dibiarkan `comma` jika kita menggunakan pembatas lain maka hal tersebut perlu disesuaikan.

Selain menggunakan cara di atas, kita juga bisa membuat skrip langsung di file dengan ekstensi `.R` dengan menggunakan fungsi kode seperti berikut.

```
nama_objek <- fungsi_untuk_membaca_file("nama_folder/nama_file.csv")
```

Bagian `nama_objek` merupakan nama yang harus diawali dengan huruf (tidak bisa dengan angka) dan tidak boleh memiliki spasi. Kemudian, bagian fungsi digunakan untuk menempatkan nama fungsinya yang diikuti dengan tanda `()` yang perlu diisi dengan argumen yang sesuai dengan fungsi. Dalam konteks ini, kita akan membiasakan diri dengan fungsi yang terdapat dalam library `tidyverse`. Lebih tepatnya fungsi `read_csv()` dari package `readr` seperti dapat dilihat pada contoh skrip berikut.

```
data1 <- read_csv("data/sample_data.csv")
```

Dari skrip diatas, kita membuat sebuah objek `data1` dari file `.csv` yang ada dalam folder `data`. Argument yang dibutuhkan di sini adalah nama folder dan nama datanya yang dipisahkan dengan garis miring `/`. Sama seperti pada saat kita menggunakan cara pertama, di dalam skrip diatas kita juga bisa menyertakan argumen lain seperti jenis separator, tipe data di dalam kolom, dan lain sebagainya seperti yang terdapat dalam dokumentasi yang bisa diakses dengan menggunakan tanda tanya `(?)` seperti contoh berikut:

```
?read.csv()
```

Untuk mengekspor data kita bisa menggunakan fungsi dengan awalan `write_`. Se-

baliknya untuk mengimpornya kita bisa menggunakan fungsi `read_`. Argumen yang dibutuhkan kurang lebih sama dengan skema pada saat kita mencoba untuk mengimpor data. Bedanya di sini kita membutuhkan objek data frame yang ada pada bagian environment. Misalnya, di environment kita sudah ada ada dengan nama `data1` maka skripnya bisa dituliskan sebagai berikut:

```
write_csv(x = "nama objek", path = "nama directory/nama file.csv")
```

4.2 Latihan 6

Berdasarkan penjelasan pada [bagian Impor data](#), lengkapilah skrip berikut:

1. Kita memiliki data dengan format csv dengan nama `Twitter Raw.csv` di dalam sebuah folder dengan nama `raw data`, Data tersebut perlu diimpor terlebih untuk di analisis. Di sini kita akan menggunakan fungsi `read_csv()` dari package `readr` yang terdapat dalam library `tidyverse`. Oleh karena itu, kita membuat skrip berikut.

```
library(---)
raw_data1 <- read_csv(file = "----/----",
                      col_names = ---,
                      trim_ws = ---)
```

2. Setelah melakukan pre-processing, kita siap untuk menyimpan hasilnya dalam format csv. Di sini kita akan menggunakan salah satu fungsi dari `readr` untuk menyimpan file dari environment `rstudio` kedalam sub directory bernama `clean data` dalam directory proyek yang telah kita buat di awal. Lengkapilah skrip berikut.

```
library(---)
---()(x = ---, path = "----/----")
```

4.3 Memilih kolom

Terdapat dua cara untuk memilih kolom yang akan dianalisis. Pertama sebelum dimasukkan kedalam Rstdvio dan Kedua setelah dimasukkan ke Rstudio. Jika datanya tidak terlalu besar, secara teknis akan lebih mudah jika diimpot terlebih dahulu, tapi, jika datanya terlalu besar untuk ukuran memory maka akan lebih baik kita sudah memilih terlebih dahulu kolom mana saja yang perlu diimpot.

Untuk memilih kolom dari data yang ada di environment kita bisa menggunakan fungsi `select()` dari packages `dplyr`. Sebagai contoh, setelah diimpot, maka untuk memilih kolom bisa dilakukan dengan skrip berikut:

```
library(dplyr)
library(readr)

dataset <- read_csv("data/Twitter Raw.csv",
  trim_ws = FALSE)

# Memilih kolom berdasar nomor urut kolom dari kiri ke kanan
kol_terpilih1 <- dataset %>%
  dplyr::select(c(31, 12))

# Memilih kolom berdasar nomor urut kolom dari kiri ke kanan
kol_terpilih2 <- dataset %>%
  dplyr::select(c("user_screen_name", "entities_user_mentions"))
```

Kita pastikan terlebih dahulu bahwa di environment kita terdapat objek dengan nama `dataset`, yaitu data yang diambil dari Twitter. Dari data tersebut kemudian kita memilih kolom berdasarkan nomor kolom, yaitu kolom 31 dan kolom 12. Selain berdasarkan nomor atau indek kolom, kita juga bisa memilih kolom berdasarkan nama kolomnya. Skrip diatas, menggunakan sintak `c()` karena memilih lebih dari satu, artinya kita membutuhkan sintak untuk menampung nama atau indeks atau namanya.

Selain setelah diimpot, dari awal kita juga bisa memilih kolom tertentu untuk diimpot.

Hal ini bisa dilakukan masih dengan menggunakan skema yang sama seperti impor data csv yang telah dijelaskan sebelumnya. Bedanya, untuk hanya mengambil kolom tertentu kita perlu menggunakan parameter tambahan seperti contoh skrip berikut.

```
library(readr)

Twitter_Raw <- read_csv("data/Twitter_Raw.csv",
  col_types = cols(
    is_quote_status = col_skip(),
    user_profile_image_url_https = col_skip(),
    user_url = col_skip()
  ),
  trim_ws = FALSE)
```

Dengan menggunakan skrip diatas, secara konsep kita mengimpor data dari folder data yang bernama Twitter_Raw.csv kecuali kolom dengan nama: (1) `is_quote_status`; (2) `user_profile_image_url_https`; (3) `user_url`. Ketiga kolom tersebut berada dalam sintak `col_types = cols("nama kolom" = col_skip())`.

4.4 Latihan 7

Berdasarkan penjelasan pada bagian Memilih kolom, jawablah pertanyaan-pertanyaan berikut:

1. Lengkapilah skrip untuk memilih kolom nomor 1 sampai 5 berikut:

```
kolo_1_5 <- dataset %>%
  ---(1:5)
```

2. Lengkapilah skrip untuk memilih kolom berdasarkan nama dari data twitter yang telah diimpor berikut, dan buat sebagai objek R dengan nama `user`.

```

library(---)
library(---)

--- <- dataset %>%
  select("----")

```

4.5 Memabagi kolom

Membagi atau memisahkan kolom merupakan salah satu kegiatan yang umumnya perlu dilakukan pada saat melakukan pre-processing. Misalnya, kita punya satu kolom yang yang di dalamnya terdapat beberapa nilai. Contohnya, dalam konteks twitter adalah kolom username yang dimention dalam sebuah twit. Pada saat kita akan membuat sebuah network maka, setiap username perlu dijadikan satu baris. Berikut adalah contohnya.

```

library(readr)
library(dplyr)

# Data
dataset <- read_csv("data/Twitter Raw.csv",
                     trim_ws = FALSE)

# pilih kolom
dataset <- dataset %>%
  filter(!is.na(entities_user_mentions)) %>%
  select(c(user_screen_name, entities_user_mentions))

# memisahkan isi kolom entities_user_mentions
library(tidyr)
dataset <- dataset %>%
  separate(entities_user_mentions, into = c("kol1", "kol2"), sep = " ")

```

Dalam skema sederhana, cara membagi kolom bisa menggunakan skrip diatas, yaitu dengan fungsi `separate()` dari `tidyR` (Wickham & Henry, 2019). Konsepnya, fungsi

separate di atas membagi kolom `entities_user_mentions` menjadi dua kolom yaitu `kol1` dan `kol2` dengan tanda (`sep = " "`) atau spasi. Jadi, setiap ada spasi dalam kolom tersebut akan dibuat atau dimasukkan kedalam kolom 1 dan 2 yang ditunjukkan. Namun, untuk tujuan membuat network hal ini terkadang tidak bisa digunakan. Khususnya ketika dalam sebuah kolom kita tidak tahu harus membaginya menjadi berapa kolom.

4.6 Latihan 8

Berdasarkan penjelasan pada bagian Membagi kolom, jawablah pertanyaan-pertanyaan berikut:

1. Lengkapilah skrip untuk membagi kolom tagar menjadi tiga kolom berikut:

```
tagar <- dataset %>%
  select(entities_hashtags) %>%
  filter(!is.na(---)) %>%
  separate(---, into = c("----", "----", "----"),
           sep = ----)
```

2. Bagilah kolom `created_at` dari data `Twitter_Raw.csv` menjadi kolom tahun, bulan, dan tanggal.

4.7 Tokenisasi

Tokenisasi adalah proses membagi teks yang dapat berupa kalimat, paragraph, baris dan lain sebagainya menjadi bagian-bagian tertentu. Contoh, kalimat “aku sangat suka membaca” jika ditokenisasi dengan menggunakan n-grams 1 menjadi empat token. Tokenasi dapat dilakukan dalam beberapa bentuk, misalnya n-gram maka 1 term = 1 token, bigram berarti 2 term = 1 token dan seterusnya. Kalimat aku sangat suka membaca jika dibuat token dengan aturan bigram maka hanya menjadi 3 token, yaitu aku sangat, sangat suka, dan suka membaca. Dalam beberapa kasus, bigram dan seterusnya bisa mengurangi jumlah token.

Untuk melakukannya, kita bisa menggunakan fungsi `unnest_tokens()` dari tidytext (Silge & Robinson, 2016). Berikut adalah contoh penggunaannya.

```
library(readr)
library(tidytext)

# impor data
dataset <- read_csv("data/Twitter Raw.csv", trim_ws = FALSE)

# pilih kolom
dataset <- dataset %>%
  dplyr::filter(!is.na(entities_user_mentions)) %>%
  dplyr::select(c(user_screen_name, entities_user_mentions))

# tokenisasi
datenet <- dataset %>%
  unnest_tokens(mentioned, entities_user_mentions,
                token = "ngrams", n = 1, to_lower = FALSE)
```

Secara kronologis skrip diatas melakukan: (1) Mengimpor data; (2) memfilter/memilih baris data yang di dalam kolom `entities_user_mentions` tidak sama dengan NA; (3) memilih kolom `user_screen_name` dan `entities_user_mentions`; (4) membuat objek R dengan nama `datenet` yang berisi hasil dari tokenisasi kolom `entities_user_mentions` dengan nama kolom `mentioned`. Dalam contoh di atas, kita hanya menggunakan `n = 1`. Artinya, kita hanya membagi kalimat berdasarkan per term. Term di sini secara otomatis dipisahkan dengan spasi.

4.8 Latihan 9

Berdasarkan penjelasan pada bagian Tokenisasi, jawablah pertanyaan-pertanyaan berikut:

1. Lakukan tokenisasi kolom `full_text` dari data `Twitter Raw.csv`
2. Lengkapi skrip berikut untuk mengolah data `Twitter Raw.csv`

```

token_tagar <- dataset %>%
  ---:::select(entities_hashtags) %>%
  dplyr::filter(!is.na(---)) %>%
  ---:::unnest_tokens(tagar, ---, token = "----", n = 1) %>%
  count(tagar, sort = TRUE)

```

4.9 Nodes dan Edges

Nodes dan Edges merupakan dua hal utama yang harus ada dalam network. Sebagai ilustrasi, jika nodesnya merupakan nama manusia, maka edges-nya bisa berupa hubungan antar nama tersebut. Menggunakan contoh twit dibawah ini, kita bisa membuat conversation network dengan menggunakan username dan di mana nodesnya berupa username, dan edgesnya adalah mention.



Figure 11: Contoh twit untuk menentukan nodes dan edges network

Sehingga, dari twit yang terdapat pada gambar diatas, hubungannya bisa seperti:

@ForeignPolicy -> @Energy_Liz dan @ForeignPolicy -> @Neilbathiya. Mekanisme yang sama juga bisa digunakan untuk retweet network data dari Twitter. Hanya saja dalam retweet network edgesnya bukan posisi mention, melainkan retweet, yaitu dari yang diretweet ke yang meretweet.

Data untuk latihan dapat diambil dari: <https://drive.google.com/drive/folders/1cOnF2ireDSulStDKKeALyZUK8O87D042?usp=sharing>

Menggunakan data yang terdapat dalam tautan di atas, kita akan berlatih untuk menyiapakan data yang selanjutnya akan digunakan untuk melakukan analisis network baik menggunakan Gephi maupun secara langsung di RStudio. Langkahnya secara kronologis adalah sebagai berikut.

1. Mengimpor data
2. Memilih kolom yang akan dijadikan sumber analisis
3. Membuat data adjacency
4. Menyiapakan data untuk gephi
5. Ekspor data

Beberapa langkah di atas sudah dipersiapkan dalam bagian-bagian sebelumnya, yaitu mengimpor, memilih dan membagi kolom. Oleh karena itu, saat ini mari kita mempraktikkannya. Dikarenakan data kita dalam format csv, maka kita bisa menggunakan `read_csv()` dari `readr` (Wickham, Hester, & Francois, 2018), jika data kita dalam format lain, misalnya excel maka sintak untuk mengimportnya perlu disesuaikan kembali dengan menggunakan sintaks `read_excel()` dari package `readxl` (Wickham & Bryan, 2019) dan sintak lain sejenisnya yang berfungsi untuk mengimpor data.

Langkah 1: Mengimpor data

```
library(readr)
Twitter_Raw <- read_csv("data/Twitter_Raw.csv",
                         trim_ws = FALSE)
```

Menggunakan skrip di atas, kita telah mendapatkan objek baru berupa data dari Twitter yang terdiri dari 1000 baris dan 32 kolom. Anggap saja satu kolom sama dengan satu

variabel. Untuk melakukan analisis network, kita tidak membutuhkan semua variabel, melainkan hanya dua kolom saja, yaitu kolom yang berisi username. Username di sini dicirikan dengan diawali tanda @ terdapat dalam kolom full_text dan user_screen_name. Oleh karena itu, langkah selanjutnya adalah memilih dua kolom tersebut dengan menggunakan fungsi `select()` dari (Wickham et al., 2019) seperti dalam skrip berikut.

Langkah 2: Memilih kolom yang akan dijadikan sumber analisis

```
library(dplyr)

data_for_net <- Twitter_Raw %>%
  dplyr::select(user_screen_name, full_text)
```

Menggunakan skrip di atas, kita telah membuat sebuah objek data baru yang disebut `data_for_net` yang terdiri dari 1000 baris dan hanya dua kolom. Salah satu keuntungan melakukan pemilihan variabel seperti ini adalah kita membatasi kinerja memori yang digunakan untuk mengolah data. Dengan demikian, kita bisa melakukan pemrosesan yang jauh lebih cepat.

Namun, jika kita membuat data dan melihat isi dalam kolom `full_text`, sebenarnya di sana masih banyak data yang tidak relevan. Misalnya, masih banyak konten yang ditulis dengan menggunakan huruf non latin, dan lain sebagainya. Untuk itu, biasanya kita masih perlu untuk melakukan pre-processing terlebih dahulu sebelum sampai ketahap ini. Tahap pre-processing yang diperlukan diantaranya adalah menghilangkan twit-twit yang tidak relevan dengan ciri seperti di atas. Untuk saat ini, anggap saja datanya sudah bersih dan kita langsung melangkah pada tahap berikutnya yaitu membuat kolom `full_text` hanya berisi username dan menghapus text lain selain yang diawali tanda @.

Untuk memisahkan teks biasa dengan teks yang diawali tanda @ kita membutuhkan regular expression (regex), yaitu kumpulan karakter yang dapat digunakan untuk menjadi penanda sebuah jenis teks. Namun pada kesempatan ini, kita tidak akan terlalu banyak mempelajari regex. Untuk kasus username Twitter kita bisa menggunakan pengekstrakkan dari sintak yang sudah tersedia seperti berikut.

Langkah 3: Membuat data adjacency

```
library(stringr)
library(tidyr)

mentioned <- str_extract_all(string = data_for_net$full_text,
                             pattern = "@[[alnum:]_]*", simplify = TRUE)
mentioned <- data.frame(mentioned)
mentioned <- mentioned %>%
  unite("mention", sep = " ")
data_for_net$mentioned <- paste0(mentioned$mention)
```

Sampai ditahap ini, kita telah berhasil membuat sebuah kolom baru dengan nama `mention` di objek `data_for_net`. Konten yang ada di dalamnya merupakan username yang diekstrak dari kolom `full_text`. Dengan kata lain, username tersebut merupakan yang dimention dalam tubuh teks twit. Sekarang kita telah memiliki kolom baru pada objek `data_for_net`, yaitu kolom `mentioned`. Oleh karena itu, kita sudah bisa membuat adjacency list dari kolom `user_screen_name` dan `mentioned`. Namun seperti dapat dilihat pada kolom `mentioned` masih terdapat baris yang kosong dan berisi lebih dari satu username. Sementara untuk membuat network kita butuh satu ke satu. Artinya satu username ke satu username. Hal ini dapat dilakukan dengan langkah berikut.

```
library(tidytext)
library(stringr)

# tambah kolom total word per row
data_for_net$total <- str_count(data_for_net$mentioned, '\\S+')

# Membuat tokenisasi kolom mentioned
adj_objek <- data_for_net %>%
  filter(total > 0) %>%
```

```
dplyr::select(-c(full_text, total)) %>%
unnest_tokens(target, mentioned, token = "ngrams",
n = 1, to_lower = FALSE)
```

Skrip di atas digunakan untuk melakukan beberapa hal berikut: Pertama, menambah kolom `total` yang berisi jumlah kata pada kolom `mentioned`. Kolom ini selanjutnya digunakan untuk memilih baris yang memiliki konten lebih dari 0 atau minimal 1. Setelah bisa memfilter baris, untuk membuat dua kolom adjacency kita bisa menggunakan sintaks `unnest_tokens()` dari (Silge & Robinson, 2016). Tokenisasi di sini dilakukan dengan `n = 1`, dan membuat teks dibiarkan seperti apa adanya, dengan `to_lower = FALSE`.

Langkah 4: Menyiapakan data untuk gephi

Objek `adj_objek` sebenarnya sudah bisa langsung dimasukkan ke Gephi, namun di R kita bisa membuat pekerjaan di Gephi sedikit lebih mudah dengan membuat data yang lebih familiar untuk network. Walupun sebenarnya gephi bisa mengolah data csv, di sini kita akan membuat data sebagai objek graph dengan ekstensi `.gexf`.

```
library(igraph)
library(rgexf)

# create igraph objek
d_net <- simplify(graph_from_data_frame(d = adj_objek, directed = TRUE),
remove.loops = TRUE, remove.multiple = FALSE,
edge.attr.comb = igraph_opt("edge.attr.comb"))

# save langsung ----
write_graph(graph = d_net, file = "data/tes_net.graphml", format = "graphml")
```

Setelah membuat objek adjacency pekerjaan pra pemerosesan untuk network sebenarnya sudah hampir selesai, di mana dengan menggunakan skrip diatas, kita bisa langsung mengekspornya dan dimasukkan ke gephi sebagai sebuah objek graph. Objek `d_net`, di sini dibuat dengan menggunakan sintaks dari `graph_from_data_frame()` dari (Csardi

& Nepusz, 2006) dengan beberapa parameter yang ada didalamnya. Lalu, setelah objek d_net berhasil dibuat kita menyimpannya sebagai sebuah objek graph. Dalam konteks ini, kita membuatnya dalam format .gexf dengan menggunakan `write_graph()` dari (Yon, Lacoa, & Kunst, 2015).

Langkah 5: Ekspor data

Setelah berhasil membuat objek graph, langkah terakhir yang perlu dilakukan adalah menyimpannya di directory. Namun, untuk memudahkan, kita juga bisa membuat label nodes dan edgesnya. Di sini kita membuat data frame dari data d_net. Di mana nomor nodes kita namai kolom ID, dan Nama dinamai NAME. Di dalam sebuah objek graph, atau lebih tepatnya objek igraph di dalamnya terdapat nama nodes yang bisa diakses melalui atribut. Untuk mengakses nama, kita bisa menggunakan `V(d_net)$name`. Dimana V sebelum d_net merupakan indikator dari vertices, yaitu sebuah istilah lain untuk menunjukkan nodes.

```
# Create a dataframe nodes: 1st column - node ID, 2nd column -node name
nodes_df <- data.frame(ID = c(1:vcount(d_net)), NAME = V(d_net)$name)

# Create a dataframe edges: 1st column - source node ID,
# 2nd column -target node ID
edges_df <- as.data.frame(get.edges(d_net, c(1:ecount(d_net)))) 

# save dengan nama nodes
write.gexf(nodes = nodes_df, edges = edges_df,
           defaultedgetype = "directed",
           output = "data/sample_net.gexf")
```

Selanjutnya untuk mendapatkan data frame edgesnya kita bisa menggunakan `get.edges()` di mana hasilnya adalah dua kolom ID Nodes Sumber dan ID Nodes Target. Id-nya sendiri berasal dari ID nodes yang telah dibuat sebelumnya. Setelah di masukkan gephi, data di atas bisa menghasilkan visual network seperti [Gambar 12](#). Di mana pada gambar 12 tersebut kita sudah bisa melihat nodes dengan centrality serta kelompok berdasarkan

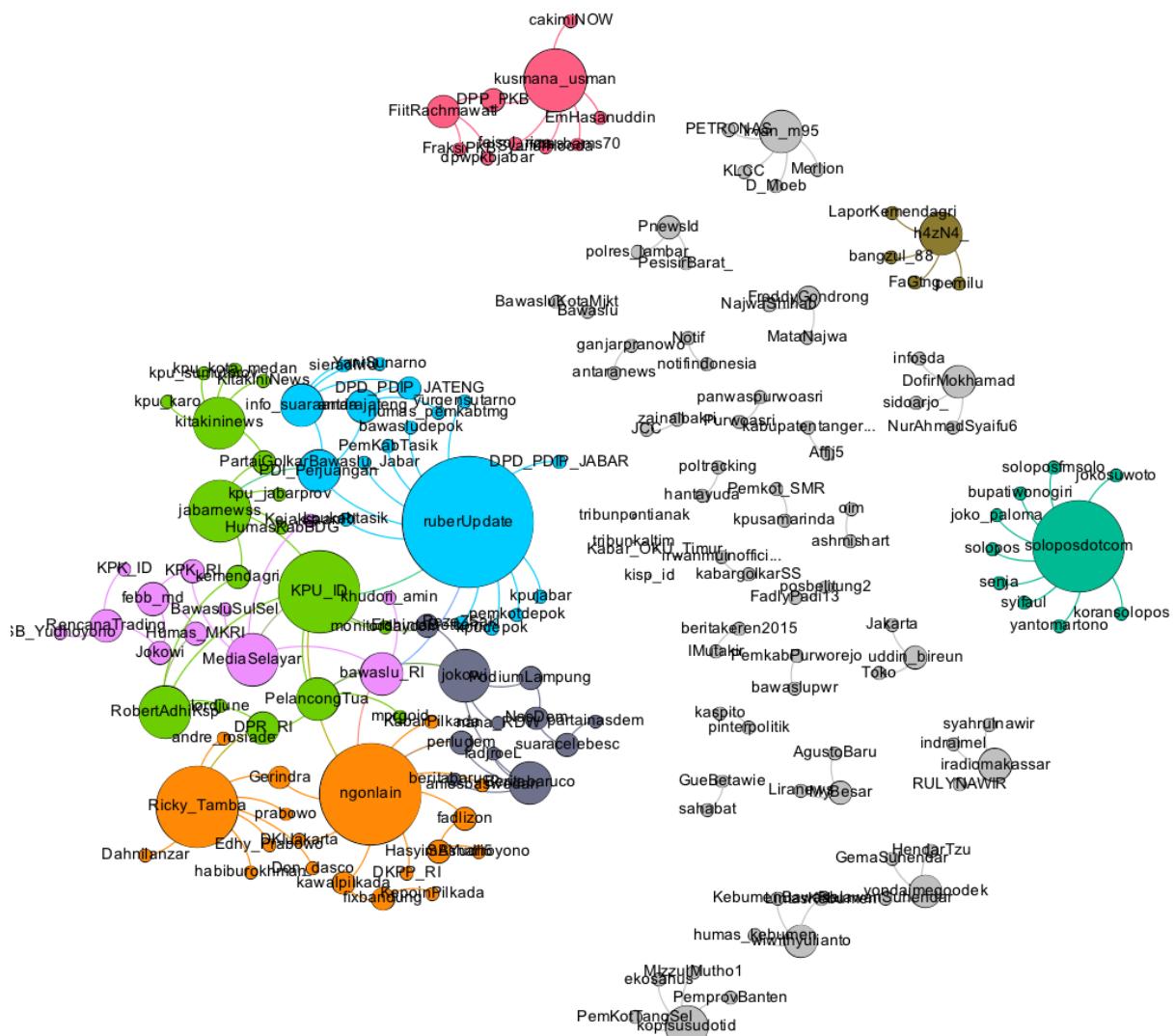


Figure 12: Hasil visualisasi network di Gephi

modularity.

4.10 Latihan 10

Berdasarkan penjelasan pada bagian **Nodes dan Edges**, jawablah pertanyaan-pertanyaan berikut:

1. Apa yang dimaksud dengan nodes dan edges?
2. Bagaimana cara menentukan nodes dan edges untuk membuat sebuah network?

3. Jelaskan langkah membuat adjacency list secara detail!
4. Lengkapi skrip berikut untuk membuat adjacency list dari sebuah data Twitter dengan skema *undirected network*

```
# menambahkan tanda @ dengan paste
data_for_net$user_screen_name <- ---("@", data_for_net$user_screen_name)

# tambah kolom total word per row
data_for_net$total <- str_count(data_for_net$mentioned, '\\S+')

# Tokenisasi
hasil <- data_for_net %>%
  dplyr::---(total > 5) %>%
  dplyr::---select(user_screen_name, mentioned) %>%
  unite(col = username, sep = " ") %>%
  tidytext::---(kata, username, token = "ngrams", n = 2, to_lower = FALSE) %>%
  separate(col = kata, into = c("----", "----"), sep = " ") %>%
  count(sumber, target, sort = TRUE)
```

5 Menggunakan Igraph dan Gephi (3 Jam)

Igraph, Ggraph dan Gephi merupakan paket dan perangkat lunak yang bisa memenuhi kebutuhan analisis network. Dua paket yang disebutkan diawal merupakan paket yang bisa digunakan untuk membuat dan mengukur centrality serta parameter network lainnya dan membuat visualisasinya di R. Sedangkan Gephi merupakan perangkat lunak yang tidak terkait dengan R dan khusus dibuat untuk melakukan analisis network. Bagian ini akan menjelaskan beberapa fitur utama dari ketiga alat yang akan digunakan.

5.1 Package igraph

Untuk menginstall paket ini bisa menggunakan `install.packages()`. Paket ini, bisa digunakan untuk membuat beberapa hal, baik untuk menyiapkan data untuk kemudian diolah di Gephi, maupun melakukan analisis centrality langsung di R. Beberapa fitur utamanya adalah sebagai berikut:

1. Membuat objek graph atau igraph dari data frame

Untuk melakukan analisis network di perangkat lunak yang lain, sebaiknya kita sudah memiliki objek graph. Di R, umumnya objek tersebut berjenis list, di mana di dalamnya terdapat nodes/vertices, edges, dan centrality atau parameter lain yang akan digunakan. Parameter lain tersebut biasanya disebut atribut yang melekat baik pada setiap nodes maupun hubungannya/edges. Salah satu fitur yang akan sangat membantu adalah fitur untuk membuat graph dari data frame atau sebuah *data flat*, yang memiliki minimal dua kolom, yaitu nodes sumber dan nodes target. Contohnya adalah sebagai berikut:

```
# load data
library(tidyverse)
sampled_data <- read_csv("data/sample_adjacency.csv",
                         trim_ws = FALSE)

sampled_data <- sampled_data %>%
  count(user_screen_name, target, sort = TRUE) %>%
  filter(n >= 3)

# membuat objek igraph
library(igraph)
graph_sample <- graph_from_data_frame(d = sampled_data, directed = TRUE)

# mengimpor
write_graph(graph = graph_sample, file = "data/sample1.graphml",
            format = "graphml")
```

Skrip diatas digunakan untuk membuat objek `graph_sample` dari sebuah data frame yang terdiri dua kolom yang merepresentasikan nodes sumber dan target. Sintak pertama yang digunakan adalah `graph_from_data_frame()`, di mana `d` berupa data frame, dan `directed = TRUE/FALSE` sesuai dengan konsep network yang akan dibuat. Selanjutnya, setelah berhasil membuat objek tersebut kita bisa mengolahnya lebih lanjut atau mengekspornya dan diolah di gephi atau perangkat lain. Untuk mengekspor menggunakan package `igraph`, kita bisa menggunakan sintak `write_graph()`, dengan format yang bisa dipilih yaitu: “edgelist”, “pajek”, “ncol”, “lgl”, “graphml”, “dimacs”, “gml”, “dot”, “leda”. Untuk contoh skrip diatas, kita menggunakan format `graphml`.

2. Menghitung centrality

Centrality, seperti telah dijelaskan sebelumnya merupakan salah satu parameter utama yang sering digunakan saat melakukan network analisis. Package `igraph` (Csardi & Nepusz, 2006) memiliki beberapa fungsionalitas yang bisa dimanfaatkan untuk menghitung berbagai centrality seperti skrip berikut.

```
# mendapatkan nama nodes
name <- data_frame(nodes= V(graph = graph_sample)$name)

# degree centrality
degree.cent <- centr_degree(graph_sample, mode = "all")
degree.cent <- data_frame(degree = degree.cent$res)

# closeness centrality
closeness.cent <- closeness(graph_sample, mode="all")
closeness.cent <- data_frame(closeness = closeness.cent)

# betweenness
betweenness <- betweenness(graph = graph_sample)
betweenness <- data_frame(betweenness = betweenness)

# Eigenvector
```

```
eigenvector <- eigen_centrality(graph = graph_sample, directed = TRUE)
eigenvector <- data_frame(eigen = eigenvector[["vector"]])
```

Skrip di atas digunakan untuk menghitung degree, closeness, dan centrality yang lain. Seperti dapat dilihat, sintak yang digunakan juga mirip dengan centrality-nya, yaitu `centr_degree()`, `closeness()`, `betweenness()`, dan `eigen_centrality()`. Parameter utama yang dibutuhkan adalah objek `graph`. Dalam konteks ini kita isi dengan `graph_sample` yaitu sebuah objek igraph yang telah kita buat sebelumnya. Selain itu, juga ada mode dan tipe network yang juga diperlukan untuk mendapatkan centrality. Di sini kita perlu menyesuaikan, apakah network yang kita buat directed atau undirected. Untuk lebih lengkapnya terkait dengan fungsi yang dimiliki `igraph` kita bisa melihat dokumentasi pada tautan berikut: <https://igraph.org/r/doc/>

2. Menghitung struktur komunitas

Untuk mengidentifikasi keberadaan komunitas, di `igraph` kita memiliki beberapa pilihan metode. Fungsi `cluster_walktrap()` adalah salah satu yang juga paling simple dan cepat dibandingkan metode yang lain. Fungsi tersebut mencoba untuk mencari komunitas dalam sebuah jejaring melalui jarak yang acak dari satu nodes ke nodes yang lain. Intusinya adalah, jarak yang acak terdekat cenderung sama dalam sebuah komunitas. Eksekusi kodenya adalah sebagai berikut.

```
wc <- cluster_walktrap(graph = graph_sample)
modularity(wc)
membership(wc)
```

Hasil dari skrip di atas selanjutnya dapat di lihat pada gambar 13, di mana dalam gambar yang dihasilkan dari data terdapat sembilan komunitas. Masing-masing komunitas memiliki warna sendiri. Hasil analisis ini, selanjutnya bisa menjadi parameter untuk menggali lebih jauh terkait dengan komunitas yang ada. Karena dalam konteks ini, hasil hanya bisa mengelompokkan username saja. Sementara untuk mengetahui siapa atau kelompok apa yang ada, salah satu analisis lanjutan adalah dengan mengkaji konten dari tiap kelompoknya.

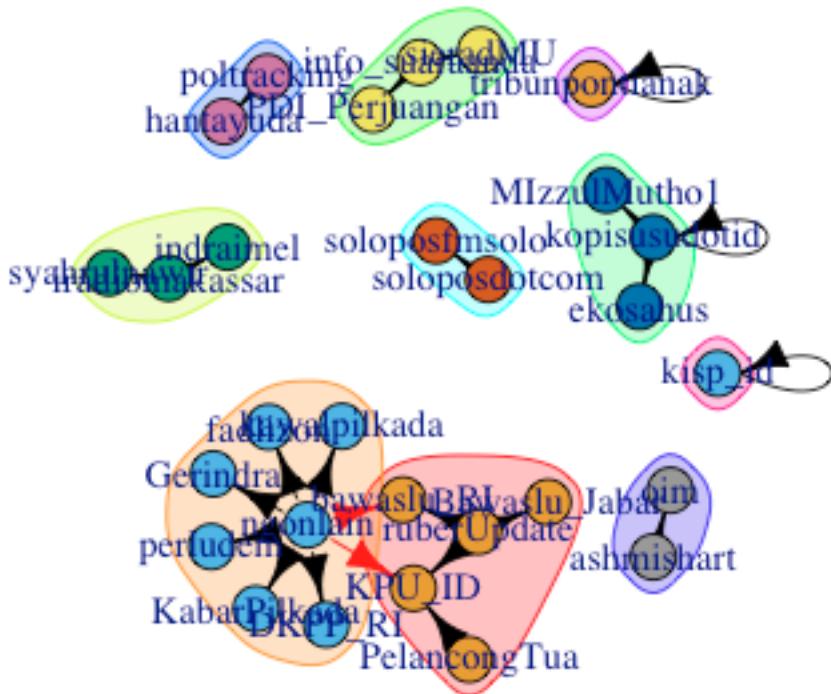


Figure 13: Komunitas dalam jeiring

Jika dilihat secara sekilas, keberadaan akun dalam satu komunitas juga cukup logis. Misalnya, soloposcom berada satu kelompok dengan soloposfm. Dengan kata lain, dalam realitanya pun kedua akun tersebut seharusnya memang berdekatan karena dimiliki oleh grup solopos.

5.2 Aplikasi Gephi

Gephi merupakan aplikasi yang khusus dibuat untuk melakukan analisis network secara visual. Gephi bisa didapat dari <https://gephi.org/> dan digunakan secara gratis (*free*). Gephi bisa digunakan untuk berbagai analisis network yang maksimal bisa menangani 1 juta nodes dan edges. Selain itu, gephi juga bisa mengolah berbagai jenis file graph seperti graphml, gexf, net dan lain-lain.

Pada bagian ini kita akan mencoba untuk lebih mengenal hal apa saja yang bisa dilakukan di Gephi, dan memahami cara untuk menggunakaninya. Untuk bisa mengenal dan

menggunakan gephi dapat mengikuti langkah-langkah berikut.

1. Mendownload dan menginstall gephi

Hal pertama yang perlu dilakukan adalah mengunduh dan menginstallnya di komputer atau laptop masing-masing. Installer gephi bisa didapat dari tautan berikut: <https://gephi.org/>. Setelah menginstall, kita akan mendapatkan tampilan seperti Gambar 14 berikut pada saat membuka aplikasi gephi.

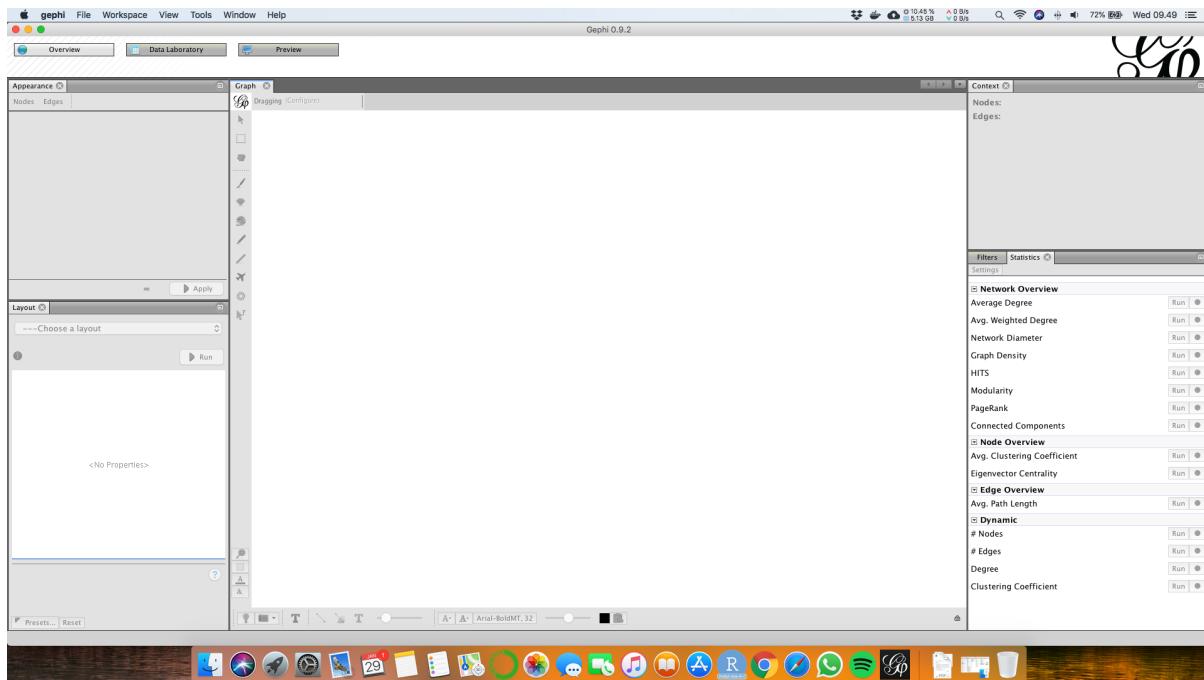


Figure 14: Tampilan awal gephi

Gambar 14 merupakan tampilan awal, dimana kita belum memasukkan data apapun kedalam Gephi. Oleh karena itu, di sana kita masih belum melihat nodes dan edges atau network apapun. Untuk bisa menginstall dengan sukses, sebaiknya perangkat kita sudah terinstall **Java JRE** terlebih dahulu yang bisa didapat dari tautan berikut: <https://www.java.com/en/download/>. Di mana Gephi compatible dengan Java versi 7 atau 8.

2. Memiliki file graph atau file yang bisa terima/baca oleh gephi

Setelah menginstall Gephi, langkah selanjutnya kita perlu memiliki data yang bisa diterima oleh gephi. Berdasarkan keterangan yang dimuat di laman resminya, gephi bisa menerima file-file diantaranya sebagai berikut.

- GEXF: adalah Graph Exchange XML Format, yaitu sebuah bahasa untuk menggambarkan struktur network yang kompleks, asosiasi dan kedinamisannya. Contoh file gexf bisa didapat pada tautan berikut: <https://gephi.org/gexf/format/datasets.html>
- GDF: Adalah format file yang digunakan oleh GUESS. File ini lebih mirip dengan CSV. Contohnya adalah sebagai berikut.
- GML: adalah Graph Modeling Language adalah format file yang bisa mensupport data network dengan sintak yang mudah. Format ini juga digunakan oleh: Graphlet, Pajek, yEd, LEDA and NetworkX. Contoh-contoh file GML bisa didapatkan dan dilihat di Gephi dari tautan berikut: <http://www-personal.umich.edu/~mejn/netdata/>
- GraphML: Sama seperti format file sebelumnya, graphml juga mendukung untuk menyimpan struktur jaringan dalam sebuah file dalam format struktur XML.
- CSV: Gephi juga mendukung untuk membaca file dalam format CSV yang merepresentasikan hubungan nodes-nodes network. Misalnya dari A ke B dan A ke C direpresentasikan dalam CSV sebagai: A, B dan A, C.
- Spreadsheet: Selain menggunakan format file yang sudah disebutkan di atas, kita juga bisa mengimpor data excel dan mengubahnya menjadi visual network di Gephi. Sama seperti csv, file excel harus merepresentasikan hubungan, dan juga bisa disertai dengan atribut lain termasuk time interval yang merepresentasikan kedinamisan terbentuknya sebuah network secara keseluruhan.

3. Impor data ke Gephi

Untuk bisa berlatih mengimpor data ke Gephi, sebaiknya file yang terdapat dalam poin nomor 2 di atas telah di unduh. Jika sudah, maka kita bisa melanjutkan dengan dua paradigma. Pertama, mengimpor file dalam format graph dan file dalam format tabel konvesional (CSV dan atau Spreadsheet).

Impor file csv

Untuk mengimpor file csv: file > import spreadsheet > pilih file yang akan diimpor > open > Import as: Adjacency List/Nodes Tabel, etc. > Next > Time Representation > Finish > Pilih directed atau Undirected Network.

Hal penting yang perlu diperhatikan di sini adalah jenis konten yang ada dalam spreadsheet kita. Apakah ia sebagai adjacency list, tabel nodes, atau edges. Dalam konteks

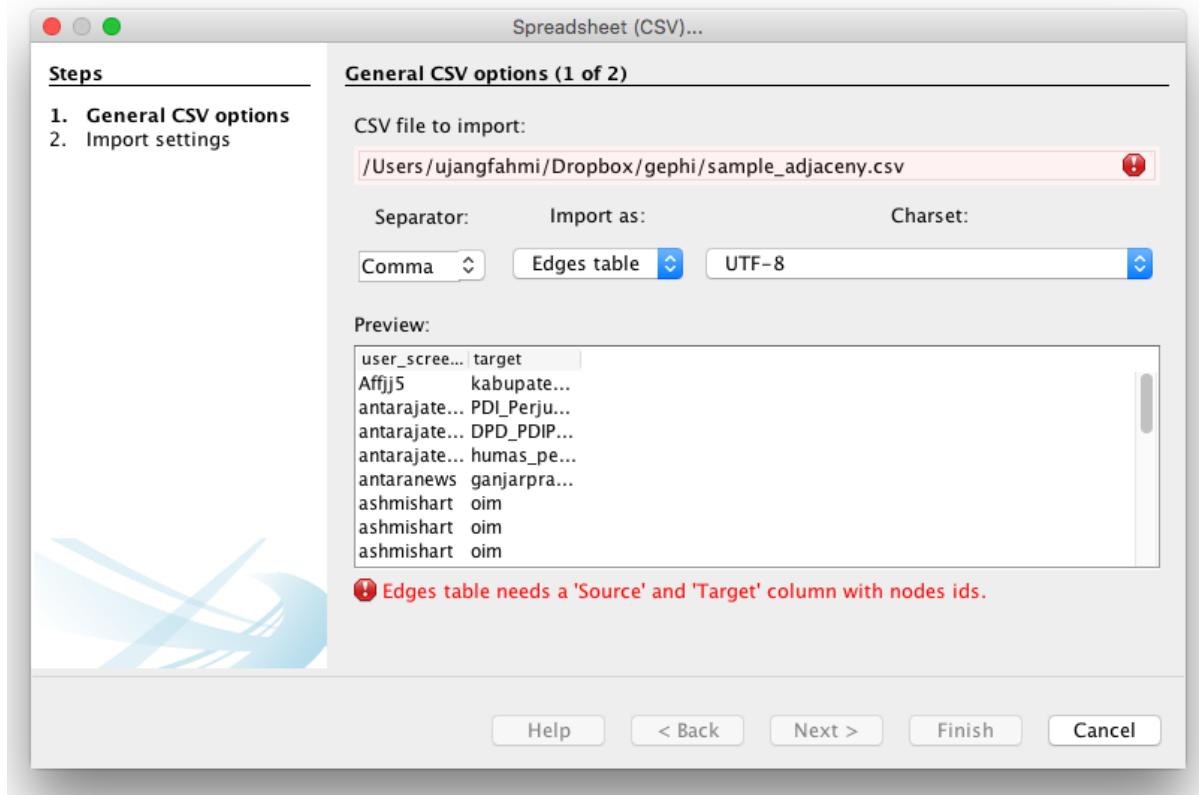


Figure 15: Tampilan saat impor spreadsheet

data yang sudah kita buat, di mana dalam data tersebut terdiri dari dua kolom yang merepresentasikan nodes sumber dan target, kita bisa memilih import as adjacency list.

Pada saat tipe data yang kita impor belum sesuai, maka kita akan mendapatkan keterangan seperti pada Gambar 15 di atas. Pada saat hal seperti itu terjadi, maka kita perlu menyesuaikan tipe datanya hingga tidak ada lagi peringatan seperti di atas.

Impor file graph

Import file graph kurang lebih memiliki langkah yang sama, namun di sini kita bisa langsung membuka file tersebut dengan cara:

File > Open > Pilih directed/undirected

Sama seperti langkah sebelumnya, setelah menekan tombol **Open** kita harus memilih tipe network yang akan dibuat (Lihat Gambar 16), apakah ia merupakan directed network atau undirected. Namun terkadang, jika perangkat laptop atau komputer kita memiliki kemampuan terbatas, kita perlu melakukan filter. Seperti contoh **BTR_directed.gexf**, file

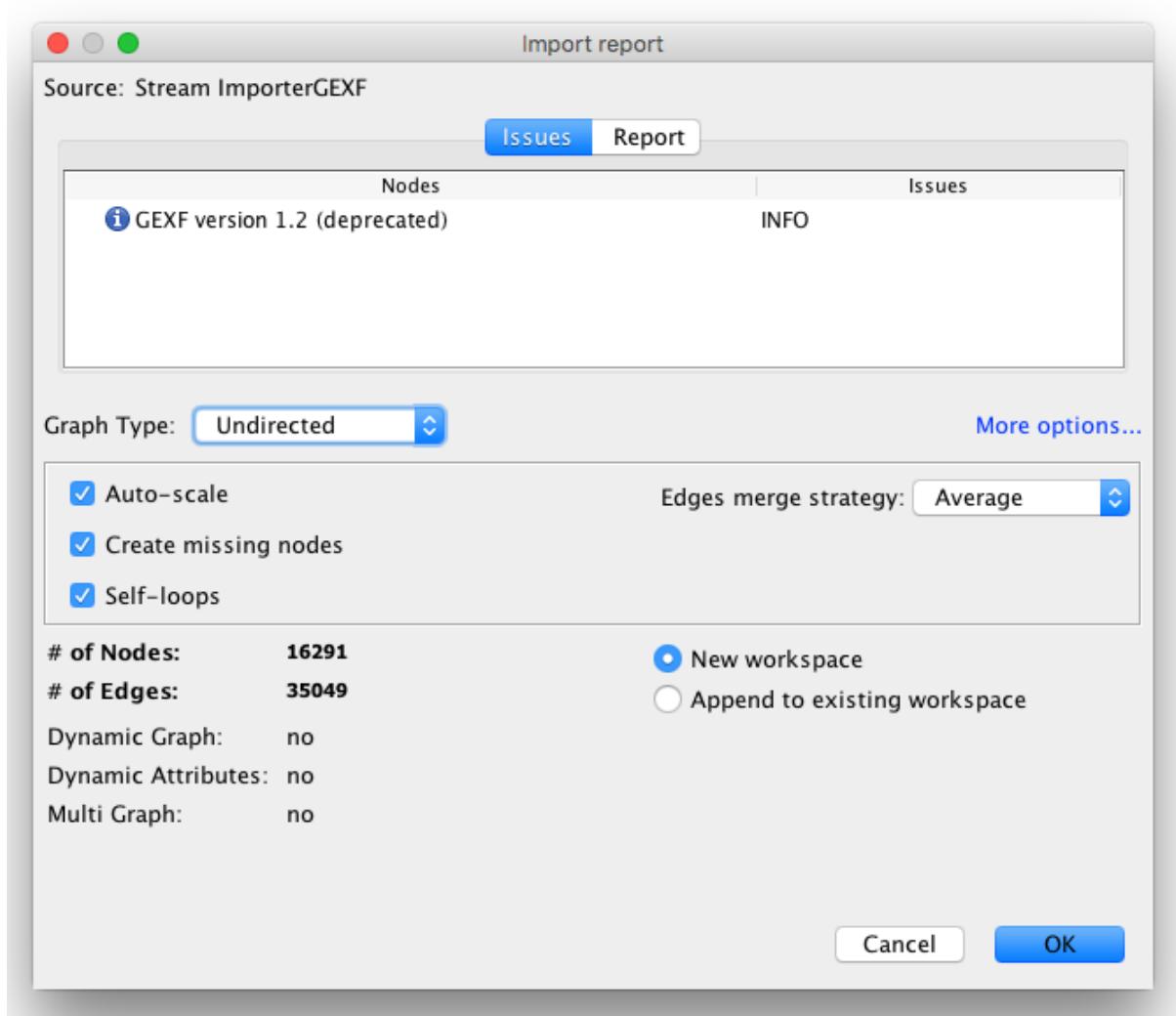


Figure 16: Tampilan saat menentukan jenis network

tersebut memiliki 16 ribu dan 30 ribu lebih nodes dan edges, sehingga cukup berat untuk diproses secara langsung.

Oleh karena itu, kita terkadang perlu memfilter nodes dan edges terlebih dahulu. Gephi memiliki beberapa sarana untuk melakukan *filtering*. Namun dalam melakukan filtering, kita perlu mempertimbangkan representasi dari network yang akan dihasilkan.

4. Membuat Filter

Filter berfungsi untuk membuat network hanya berisi nodes atau edges yang sesuai dengan kondisi filter. Gambar 17 Menunjukkan filter yang ada di posisi sebelah kanan bawah. Filter bisa dilakukan dengan beberapa kondisi, misalnya kita hanya akan menganalisis nodes yang memiliki degree tertentu.

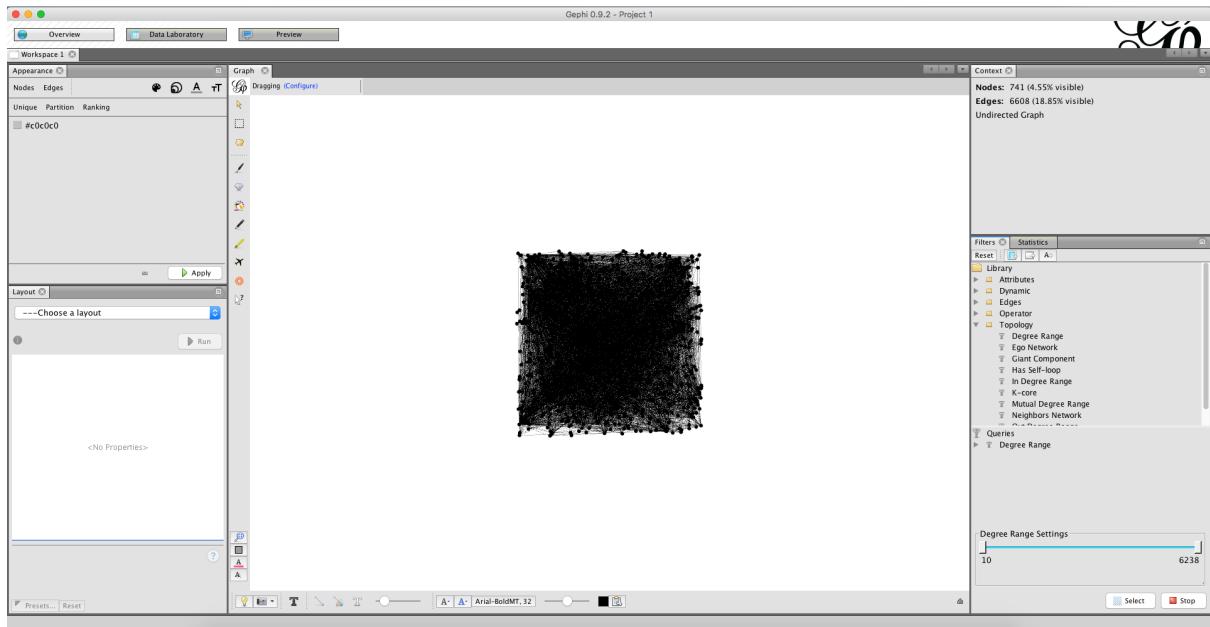


Figure 17: Tampilan saat import file graph

Untuk menggunakan filter, kita perlu memilih jenis filternya. Misalnya, yang paling umum dan sering digunakan adalah Topology, yang di dalamnya ada berbagai macam parameter. Kita hanya perlu menarik filter yang akan digunakan dan menentukan angkanya, lalu tekan enter. Dalam Gambar 17, kita menggunakan degree range dengan angka 10. Artinya, network hanya akan menampilkan nodes yang memiliki degree minimal 10 atau lebih. Selain itu, pada Gambar juga menunjukkan terdapat indikator stop. Selama tanda tersebut tidak ditekan, maka network kita hanya akan menunjukkan sesuai dengan filter yang ditentukan.

5. Plugins gephi

Plugins Gephi bisa didapatkan pada menu **tools** (bagian atas). Di sana kita bisa memilih plugins yang sesuai dengan kebutuhan. Plugins ini berfungsi untuk membantu kita dalam menggunakan gephi. Misalnya untuk mengimpor data, membuat visualisasi, dan membuat output yang diinginkan. Tentunya, untuk mendapatkan plugins yang sesuai, kita melihat plugins yang tersedia juga. Lebih penting lagi, kita juga perlu membaca dokumentasinya.

Secara umum, jika kita sudah berhasil menginstall gephi maka sebagian besar plugins juga bisa diinstall. Hanya untuk beberapa plugins tertentu saja yang membutuhkan dependensi baru. Misalnya, java versi tertentu untuk bisa menginstall plugins untuk

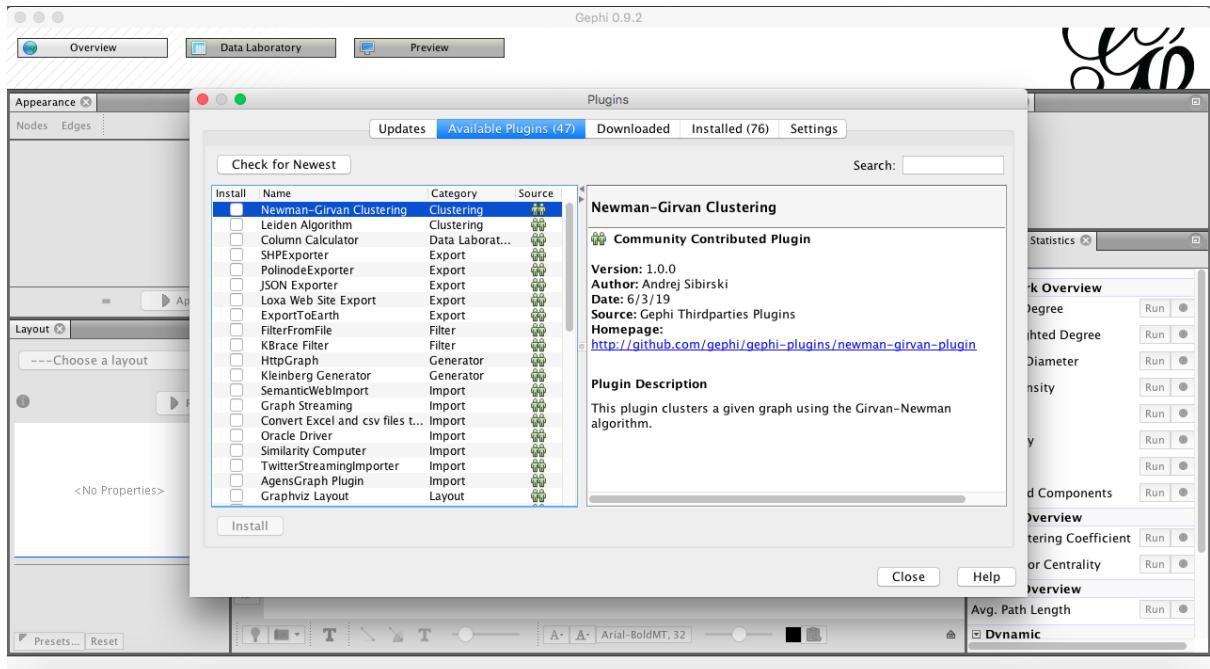


Figure 18: Plugins yang tersedia untuk gephi

mengekspor hasil visualisasi dalam bentuk html yang lebih interaktif atau bisa diklik.

5.3 Latihan 11

Berdasarkan penjelasan pada bagian Mengenal Igraph dan Gephi, jawablah pertanyaan-pertanyaan berikut:

1. Sebutkan data apa saja yang bisa diterima oleh gephi?
2. Lengkapi skrip untuk mengekspor data dari R dengan nama NET dalam format graphml dengan nama NET_SAMPLE.

```
--> (graph = -->, file = "data/---.---", format = "---")
```

3. Lengkapi skrip untuk menghitung betweenness centrality dari objek graph bernama NET berikut:

```
--> <- (graph = -->)
```

6 Visualisasi Network (4 Jam)

Setalah melakukan persiapan data network dan perhitungan atribut masing-masing nodes dan edges kita sebenarnya sudah selesai melakukan analisis network. Namun untuk memastikan hasil analisis bisa dilihat dan dipahami oleh orang lain, kita perlu memvisualisasikannya. Visualisasi dapat dilakukan dengan menggunakan berbagai alat yang tersedia. Pada kesempatan kali ini, kita akan berlatih untuk membuat visualisasi network menggunakan R dan Gephi.

6.1 Visualisasi network menggunakan R

Sebagai informasi, di R kita bisa membuat visualisasi network baik yang statis maupun yang dinamis dan interaktif. Visualisasi statis biasanya diproduksi dengan output file berupa .png atau .jpeg dan sejenisnya. Sementara visualisasi interaktif umumnya berupa file .html.

Pada kesempatan ini, kita terlebih dahulu akan membuat visualisasi dalam format yang statis dengan memanfaatkan fungsi dasar `plot()` di R dan fungsi dari `ggraph` (Pedersen, 2019). Namun sama seperti pada saat melakukan persiapan membuat file untuk graph, di sini kita juga sudah harus memahami elemen dan atribut yang ada di dalam sebuah network.

Untuk mengingatkan, dalam sebuah network terdapat dua elemen utama, yaitu nodes/vertices dan edges. Selain itu, dalam sebuah network masing-masing nodes dan edges juga bisa memiliki atribut masing-masing, misalnya centrality dan modularity.

6.1.1 Visualisasi dengan `plot()`

`plot()` merupakan fungsi dasar dari R untuk membuat visualisasi atau tampilan hasil olahan data. Untuk menggunakan fungsi ini, kita perlu menyiapkan datanya terlebih dahulu. Adapun data yang dimaksud adalah data nodes dan atributnya. Jika hal tersebut sudah dilakukan, pada bagian sebelumnya maka di sini kita hanya akan mempelajari terkait dengan elemen apa saja yang bisa diplotkan menggunakan fungsi `plot()` ini.

1. Plot untuk melihat hasil awal

Sebagaimana fungsi dasar dan umum lainnya, fungsi plot ini cocok untuk digunakan pada tahap awal analisis. Dengan kata lain fungsi ini akan sangat berguna pada saat kita melakukan pre-processing untuk membuat data final untuk analisis network.

```
library(igraph)
library(tidyverse)

# 1. import data

sampled_data <- read_csv("data/sample_adjaceny.csv",
                         trim_ws = FALSE)

sampled_data <- sampled_data %>%
  dplyr::count(user_screen_name, target, sort = TRUE) %>%
  filter(n >= 2)

# 2. Membuat objek graph

net <- graph_from_data_frame(d = sampled_data, directed = FALSE)

# 3. Membuat visualisasi

plot(net)
```



Figure 19: Hasil visualisasi network dasar

Contoh skrip di atas hanya membutuhkan tiga langkah untuk menghasilkan sebuah visualisasi network, yaitu impor data, membuat objek graph, dan meletakkannya dalam fungsi `plot()`. Namun masalahnya, hasil visualisasinya belum jelas untuk diinterpretasikan. Tapi, setidaknya hasil plot telah mengonfirmasi bahwa kita telah melakukan hal yang benar dalam membuat objek graph dari adjacency list. Jika sebelumnya kita juga telah membuat list tersebut, maka hal ini juga menjadi konfirmasinya.

2. Melihat atribut network

Hal selanjutnya yang juga akan bermanfaat saat melakukan eksplorasi network adalah pada saat melakukan plotting atribut. Di mana dengan menggunakan fungsi `plot()` kita bisa dengan segera membuat visualisasi seperti pada gambar 20 berikut.

```
ec <- eigen_centrality(graph = net, directed = FALSE, weights = NA)$vector
plot(net, vertex.size=ec*10, layout=layout_with_fr)
```



Figure 20: Hasil visualisasi atribut eigencentrality

Sama seperti skrip sebelumnya, untuk membuat visualisasi seperti pada Gambar 20, kita hanya membutuhkan dua langkah, yaitu: Menambahkan atribut nodes berupa eigencentrality, dan kemudian meletakkannya dalam fungsi `plot`. Di dalam skrip diatas, parameter yang digunakan untuk membuat besar nodes yang berbeda adalah `vertex.size`, yaitu = `ec` yang merupakan eigencentrality.

Namun sekali lagi, walaupun di sini kita sudah melihat adanya nodes yang lebih besar

dibanding yang lain, kita masih melihat network yang tumpang tindih, dan tidak bisa untuk melihat secara detil. Untuk itu, dalam membuat visualisasi, kita juga perlu memperhatikan layout yang dipilih.

3. Layout network

Terdapat banyak layout yang bisa dimanfaatkan untuk membuat visualisasi objek graph dengan menggunakan `igraph`. Walaupun tidak ada aturan khusus mengenai layout apa yang baik untuk sebuah network. Secara umum analis perlu melakukan eksplorasi layout yang cocok. Hal ini sebenarnya juga berlaku baik dalam R maupun nanti pada saat menggunakan Gephi.

```
coords <- layout_(net, as_star(), normalize())
plot(net, vertex.size=ec*10, layout = coords)

plot(net, vertex.size=ec*10, layout = layout_with_dh)

wc <- cluster_walktrap(graph = net)
plot(wc, net, vertex.size=ec*30, layout=coords)
```

Lima baris skrip diatas digunakan untuk: (1) membuat objek layout graph, di mana parameter yang digunakan adalah objek `graph = net`, nama/fungsi `layout = as_star()`, dan di sini kita juga menambahkan normalisasi untuk atribut masing-masing nodes. Sama dengan langkah sebelumnya, layout yang sudah dibuat hanya perlu dimasukkan kedalam parameter fungsi `plot()`.

Selain dengan membuatnya terlebih dahulu, layout juga bisa langsung dimasukkan kedalam fungsi `plot()`. Untuk mengetahui layout apa saja yang tersedia kita bisa menelusurinya melalui tautan berikut: https://igraph.org/r/doc/layout_.html. Sementara, dua bari terakhir digunakan untuk: (1) membuat kluster nodes, lalu memvisualisasikannya dalam fungsi `plot` dengan menambahkan objek `wc` atau `walk cluster`.

6.1.2 Visualisasi dengan ggraph

ggraph (Pedersen, 2019) adalah persamaan ggplot2 yang dikhkususkan untuk membuat visualiasi network. Jika sebelumnya kita telah mempelajari cara menggunakan ggplot, maka kita akan menemukan tanda +, geom, dan aesthetic yang juga akan digunakan dalam fungsi-fungsi di ggraph. Berikut adalah skrip dasar untuk membuat visualisasi dengan ggraph menggunakan objek graph `net` yang telah dibuat sebelumnya.

```
library(ggraph)
```

```
net %>%
  ggraph(layout = "kk") +
  geom_node_point() +
  geom_edge_link() +
  theme_graph()
```

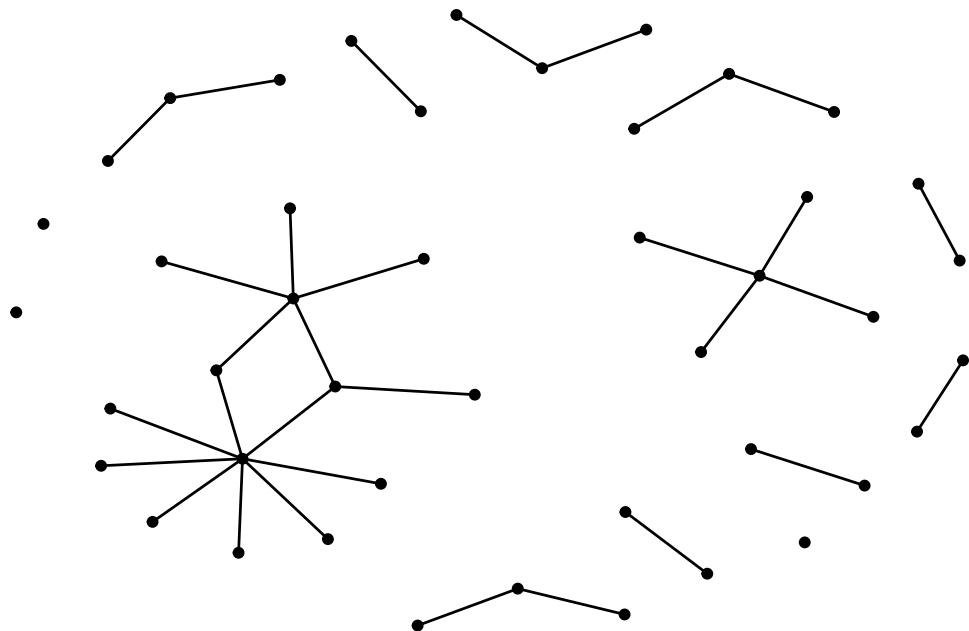


Figure 21: Hasil visualisasi ggraph dasar

Skrip diatas melakukan: (1) mengambil objek `net`; (2) menyatakan plot network menggunakan fungsi `ggraph()`; (3) plotting nodes menggunakan fungsi `geom_node_point()` dan edges menggunakan `geom_edge_link()` sehingga menghasilkan plot seperti Gam-

bar 21. Secara umum terdapat tiga hal yang perlu diperhatikan dalam membuat plot menggunakan ggraph, yaitu: Layout, Nodes, dan Edges.

6.1.2.1 Layout

Layout yang tersedia di ggraph untuk objek yang dibuat menggunakan igraph adalah: auto, igraph, dendrogram, manual, linear, treemap, circlepack, partition, dan hive. Layout merupakan argument yang ditujukan untuk objek `ggraph`. Di dalam skrip sebelumnya, kita menggunakan layout `auto` yang merupakan fungsi dasar untuk memilihkan layout untuk graph yang akan divisualkan.

Selain itu, terdapat layout lain yang juga bisa dimanfaatkan. Untuk memilih layout sendiri kita perlu menyesuaikan dengan jenis network yang akan divisualkan. Misalnya, layout `dendrogram` dan `partition` hanya bisa digunakan untuk *directed network*.

6.1.2.2 Nodes

Jika layout berfungsi untuk menentukan di mana sebuah nodes terplot, maka sekarang kita akan mempelajari bagaimana sebuah nodes di visualkan. Dalam konteks ini, kita akan menggunakan objek `ec` yang telah dibuat sebelumnya. Di mana value `ec` akan menentukan besar atau size sebuah nodes. Berikut adalah contohnya.

```
wc <- cluster_walktrap(graph = net)
col <- wc$membership

net %>%
  ggraph(layout = "kk") +
  geom_edge_link(edge_colour = "grey") +
  geom_node_point(aes(size = ec,
                       color = as.factor(col)),
                  show.legend = FALSE) +
  geom_node_text(aes(label = name), repel=TRUE) +
  scale_fill_manual(wc)
```

theme_graph()

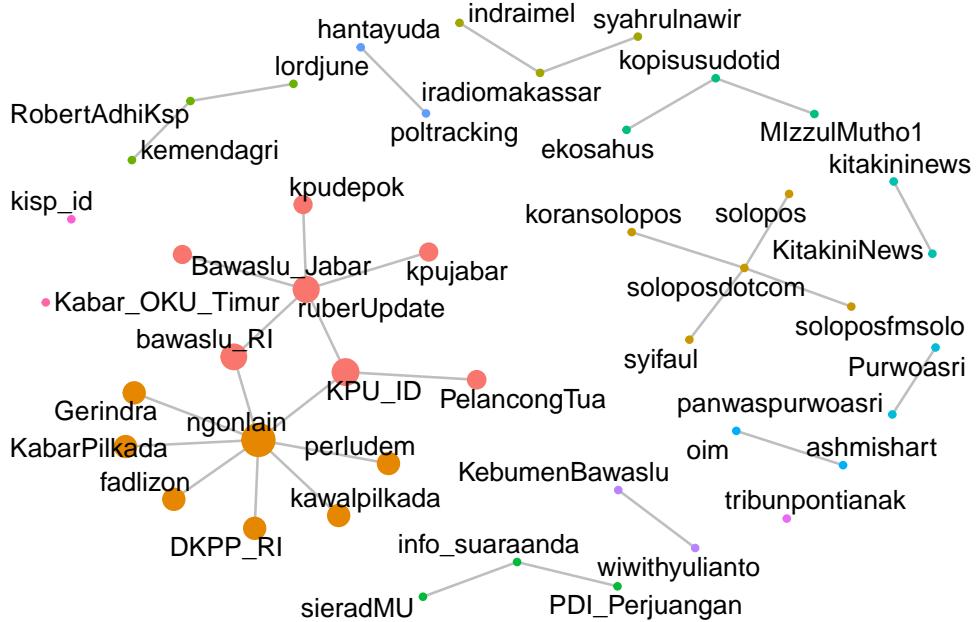


Figure 22: Hasil visualisasi dengan besar nodes yang berbeda

Skrip di atas digunakan untuk membuat plot dengan nodes yang disesuaikan dengan value pada objek `ec`. Selain itu, kita juga sudah memasukkan elemen warna yang berbeda untuk tiap nodes berdasarkan keanggotannya pada sebuah kelompok hasil dari modularity. Dengan menggunakan plot seperti ini, kita bisa mengetahui bukan hanya kelompok, tapi juga nodes penting pada tiap kelompok.

6.1.2.3 Edges

Sebuah edges merepresentasikan hubungan antara dua entitas, dalam konteks ini disebut nodes atau vertices. Untuk membuat edge pada graph dari objek igraph kita hanya perlu menambahkan fungsi `geom_edge_link()`. Walaupun demikian, sama seperti fungsi `geom_` pada ggplot, fungsi ini juga memungkinkan untuk diisi dengan parameter lain termasuk aesthetic-nya sendiri.

6.2 Latihan 12

Berdasarkan penjelasan pada bagian Visualisasi Network, jawablah pertanyaan-pertanyaan berikut:

1. Sebutkan elemen apa saja yang harus dipersiapkan untuk membuat visualisasi network di R?
2. Lengkapi skrip berikut untuk membuat visualisasi dasar menggunakan fungsi `plot()`

```
layouts <- layout_(net, as_star(), normalize())
community <- cluster_walktrap(graph = net)
plot(---, net, vertex.size=ec*10, layout=---)
```

3. Lengkapi skrip berikut untuk membuat visualisasi dasar menggunakan fungsi dari `ggraph`

```
community <- cluster_walktrap(graph = net)
col <- ---$membership

net %>%
  ggraph(layout = "kk") +
  geom_edge_link(edge_colour = "grey") +
  geom_node_---(aes(size = ec,
                    color = as.factor(---)),
                show.legend = FALSE) +
  geom_node_text(aes(label = ---), repel=TRUE) +
  scale_fill_manual(---) +
  theme_graph()
```

6.3 Visualisasi network menggunakan Gephi

Gephi merupakan salah satu alat yang cukup lengkap untuk melakukan network analysis dengan kemampuan membaca berbagai jenis file, perhitungan matriks dan visualisasi yang beragam. Pada kesempatan kali ini, kita akan membuat visualisasi sekaligus analisis dari file network yang telah kita buat sebelumnya, yaitu `sample_net.gexf`. Jika belum, file tersebut dapat diunduh melalui tautan: <https://drive.google.com/drive/folders/1cOnF2ireDSulStDKKeALyZUK8O87D042?usp=sharing>.

1. Impor Data

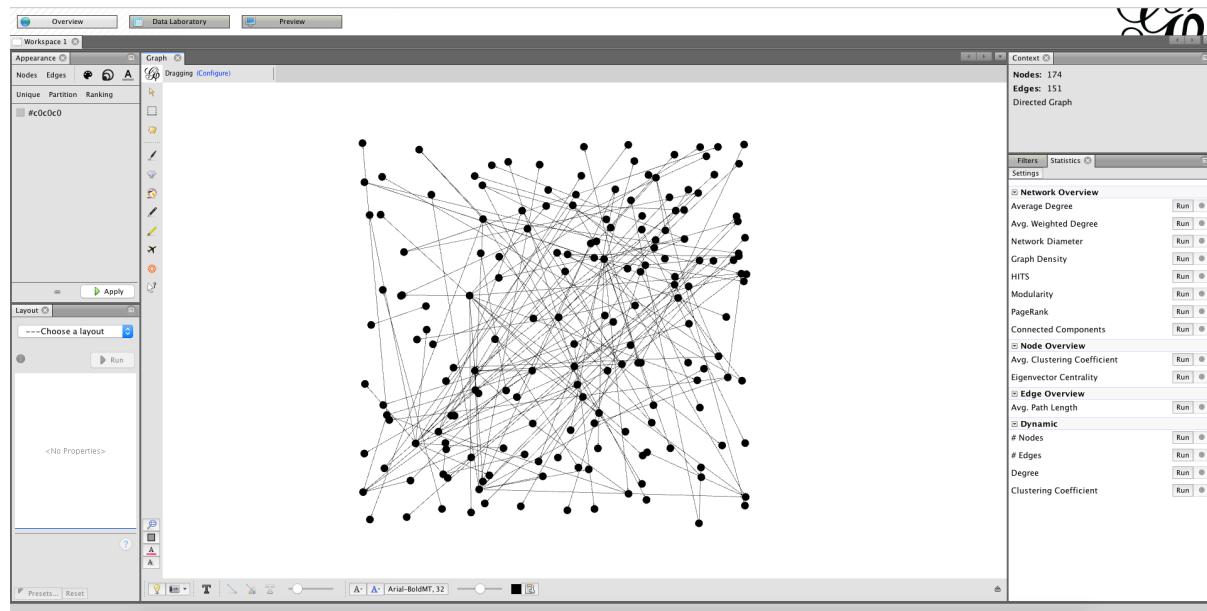


Figure 23: Tampilan awal setelah impor data

Setelah mengimpor data, maka kita akan mendapatkan tampilan seperti yang terdapat pada Gambar 23. Pada gambar tersebut, kita sudah melihat network yang nodes dan edgesnya masih berwarna hitam semua.

2. Mengekstrak Matriks

Setelah melihat layar menampilkan seperti Gambar 23, selanjutnya kita bisa melakukan filter network atau langsung mengekstrak matriks networknya. Karena untuk melakukan filter telah disampaikan, saat ini kita akan langsung membuat tampilan hitam Gambar 23, menjadi lebih berwarna dan menarik dengan menambahkan centrality dan modularity.

Di mana dalam layar yang sama, seharusnya kita melihat parameter seperti statistic di sebelah kanan.

Untuk mengekstrak modularity atau mendeteksi komunitas, kita hanya perlu menekan tombol modularity pada menu statistik sebelah kanan, dan kemudian membiarkan prosesnya berjalan hingga muncul layar baru seperti Gambar 24.

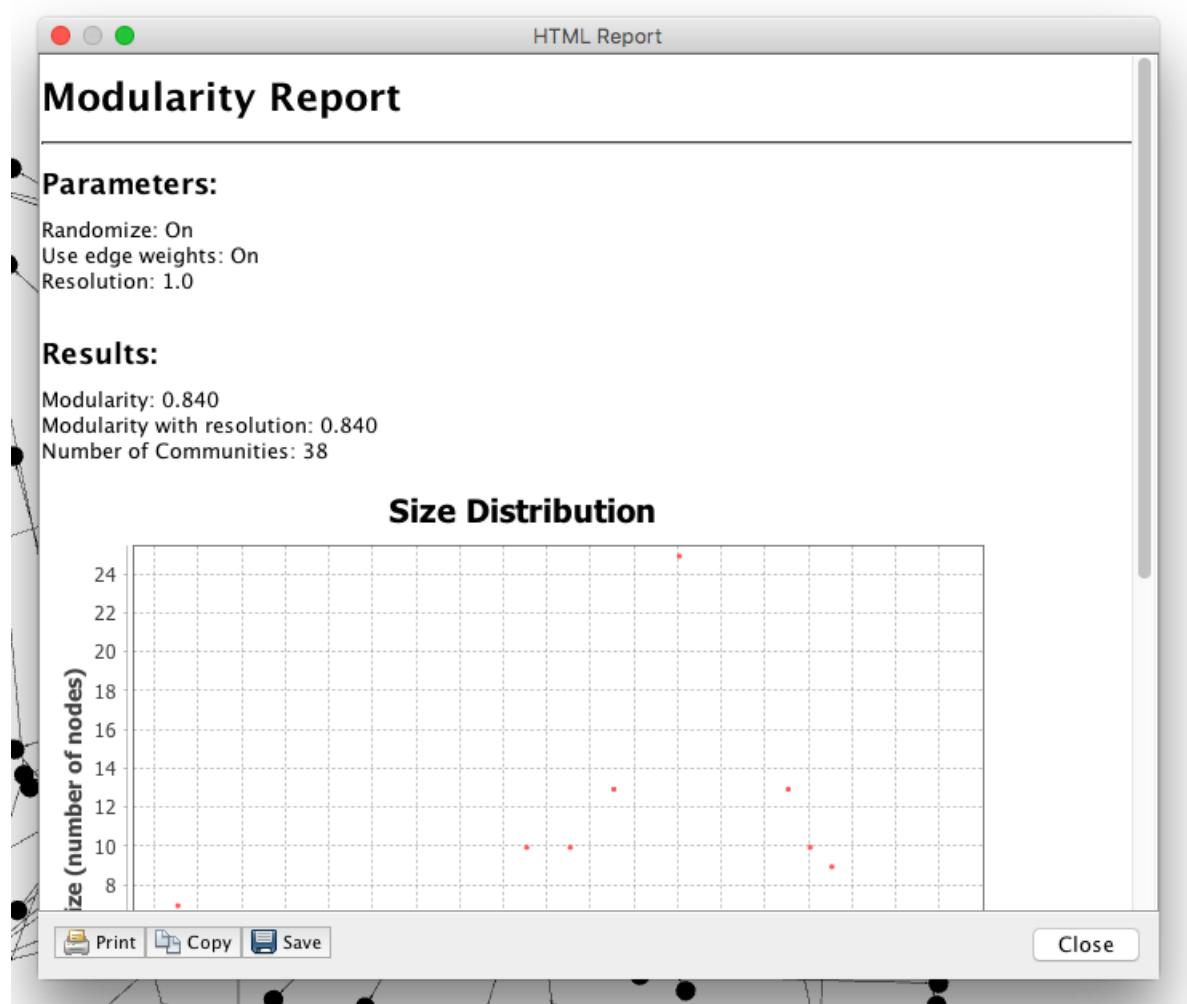


Figure 24: Hasil dan referensi modularity yang digunakan Gephi

Namun, sebelum sampai ke Gambar 24, kita perlu menentukan metode komputasi yang diinginkan. Jika sudah dibaca dan dimengerti kita bisa menekan tombol OK dan kemudian muncul Gambar 24. Pada tahap ini, kita telah membuat perhitungan modularity, di mana metodenya juga dijelaskan pada tampilan seperti Gambar 24. Jika kita hendak menggunakan hasilnya, maka kita bisa menggunakan referensi yang digunakan gephil dalam bibliografi tulisan kita.

Selanjutnya, hasil modularity yang didapat bisa kita masukan dalam visualisasi kita dengan menggunakan menu bergambar warna, lalu memilih partition, dan modularity sebagai parameternya. Setelah selesai maka kita akan mendapatkan network seperti Gambar 25 berikut.

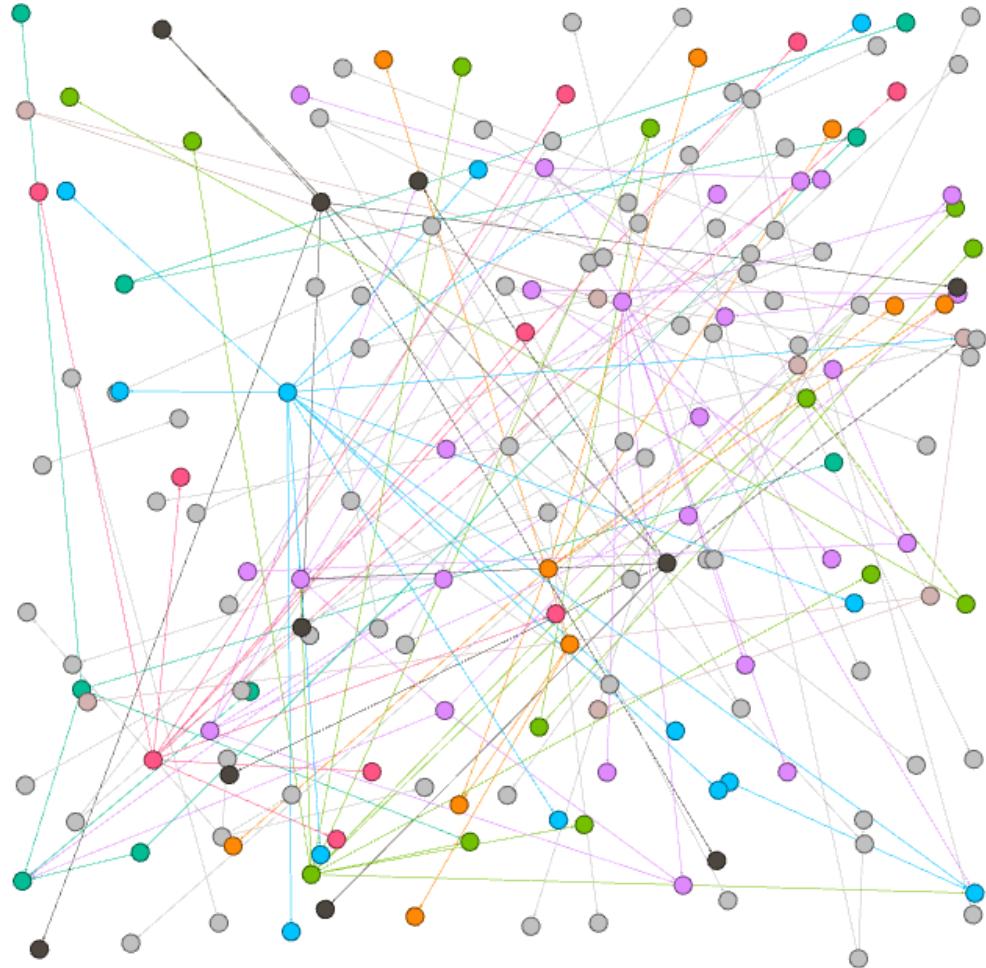


Figure 25: Hasil penghitungan modularitas secara visual

Gambar 25 telah menunjukkan kemungkinan adanya komunitas. Di mana masing-masing warna merepresentasikan satu kelompok. Selanjutnya kita bisa menambahkan ukuran nodes yang merepresentasikan salah satu centrality. Untuk melakukannya, kita bisa menggunakan tombol **Network Diameter** untuk mendapatkan nilai betweenness, closeness dan centrality yang lain. Sementara untuk mendapatkan nilai eigencentrality, kita bisa menggunakan tombol **Eigenvector Centrality**. Setelah hasilnya didapatkan, selanjutnya kita bisa menerapkannya dalam visualisasi sehingga di dapat hasil seperti Gambar 26.

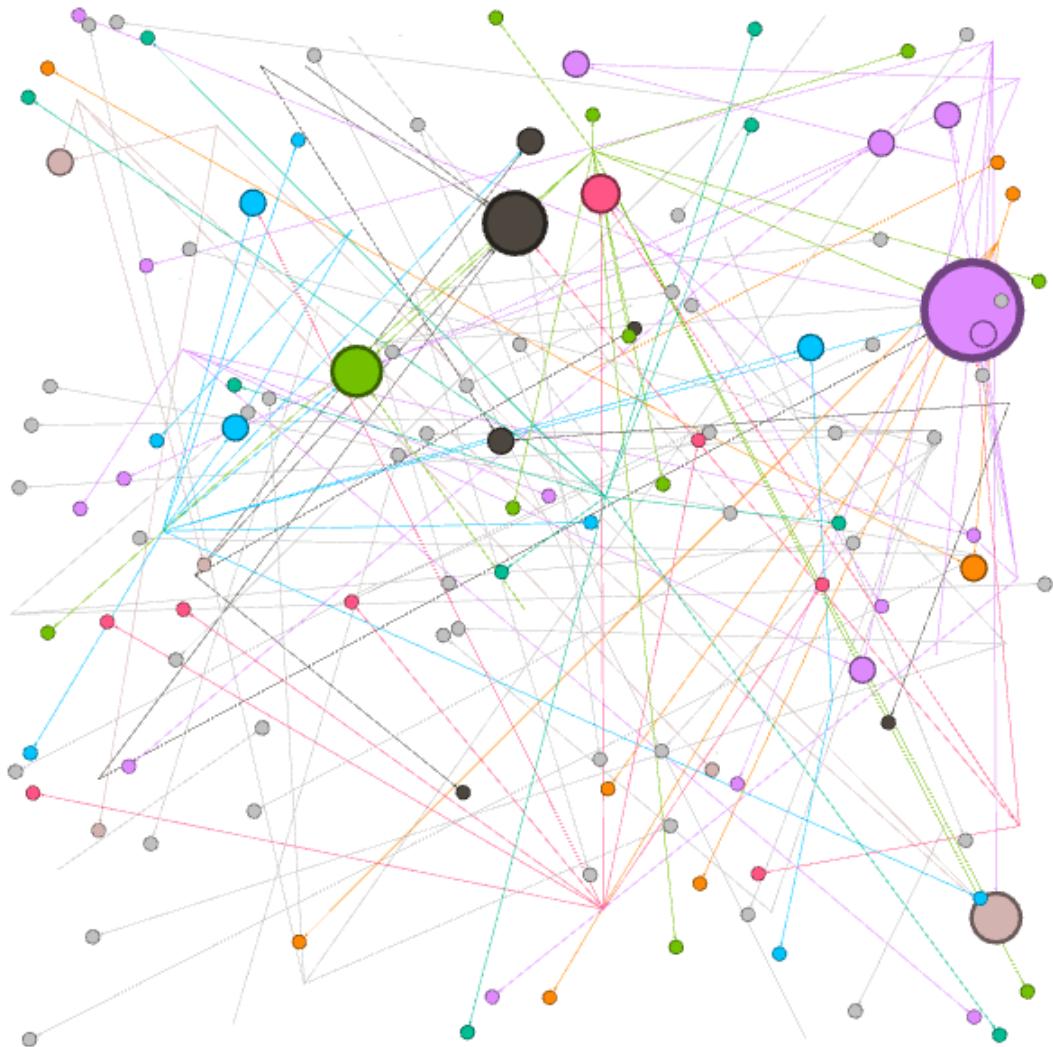


Figure 26: Hasil penghitungan centrality secara visual

Dalam Gambar 26, kita bukan hanya telah menunjukkan hasil modularity dengan indikator yang dapat dilihat berupa warna, tapi juga besaran nodes juga sudah berbeda sesuai dengan centrality yang dipilih untuk menentukannya. Untuk mendapatkan visual seperti Gambar 26, kita bisa menggunakan menu **Size** disamping menu **color** yang kita gunakan sebelumnya untuk mengubah warna nodes berdasarkan kelompok.

3. Nama Nodes

Visualisasi seperti gambar 26 mungkin telah memberikan beberapa ide interpretasi. Untuk mendukungnya, kita bisa tempatkan nama untuk masing-masing nodes tersebut sehingga tampak seperti gambar 27 berikut.

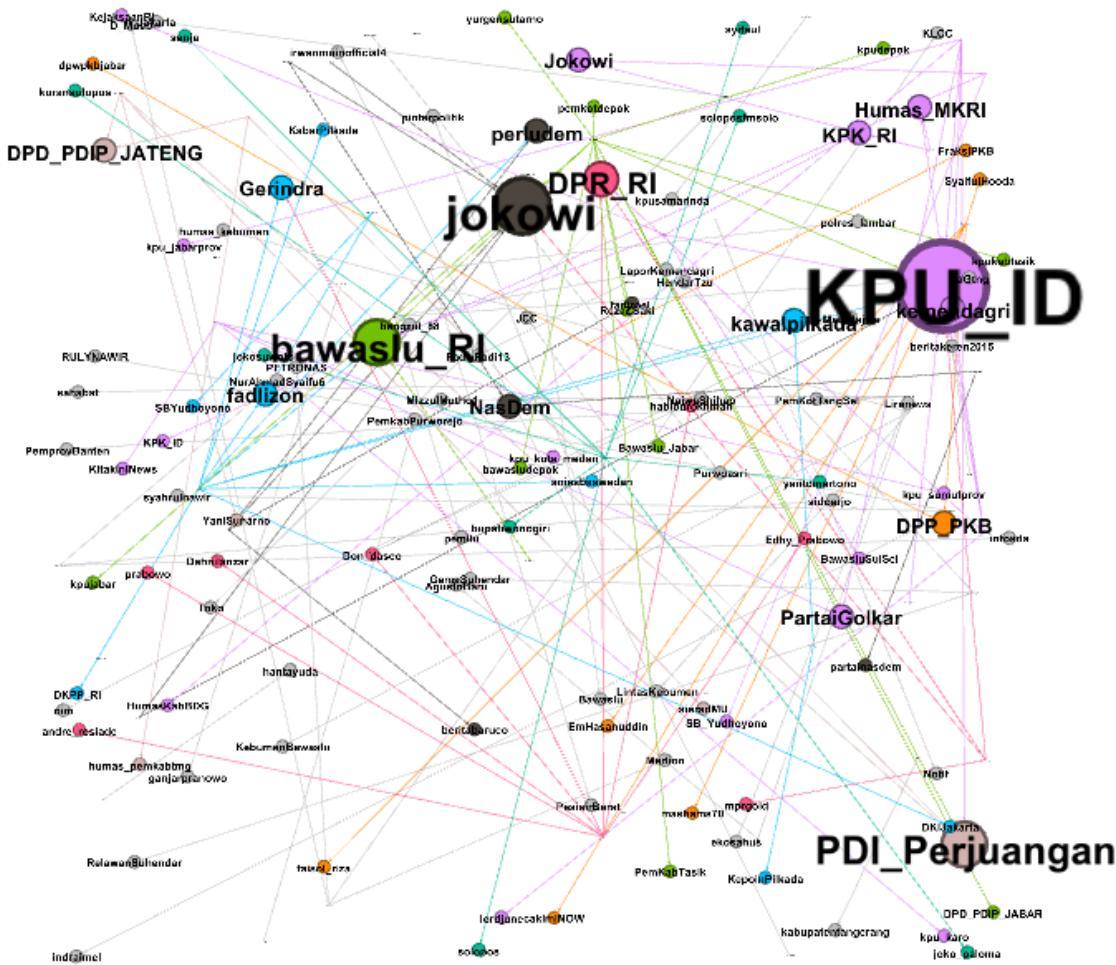


Figure 27: Memunculkan nama nodes dalam visual network

Pada layar gephi di bawah tampilan seperti gambar 27, terdapat tombol T, atau text yang berfungsi untuk menampilkan text nama nodes. Untuk pengaturan yang lebih lanjut kita bisa menekan menu dropdown yang ada di pojok kanan, di bawah tampilan network. Di sana kita bisa mengubah tampilan teks yang muncul hanya pada saat dipilih dengan kursor, besar teks yang menyesuaikan besar nodes, dan lain sebagainya.

4. Layout

Tahap selanjutnya adalah memilih layout. Gephi, memiliki beberapa pilihan layout yang secara umum memiliki fungsi yang sama. Walaupun demikian, beberapa layout memiliki beberapa fungsi yang berbeda. Misalnya, untuk menunjukkan kedekatan antar akun, kita bisa menggunakan force atlas, di mana algoritma yang digunakan akan

memaksimalkan perhitungan jarak antar nodes. Sementara untuk tujuan yang berbeda, misalnya untuk menunjukkan akun central dari keseluruhan nodes kita bisa menggunakan layout Fruncterman Reingold yang dihasilkan oleh penggunaan algoritma yang berbeda.

Layout ada pada bagian kiri tengah, di mana kita bisa memilih sesuai dengan tujuan seperti yang telah disebutkan di atas. Jika kita belum sempat mempelajari fungsi dari masing-masing layout maka hal terpenting pada saat melihat network adalah apakah visualnya sudah memberikan kita informasi yang dibutuhkan atau belum, karena hal tersebutlah esensi dari visualisasi data termasuk network. Berikut adalah contoh hasilnya.

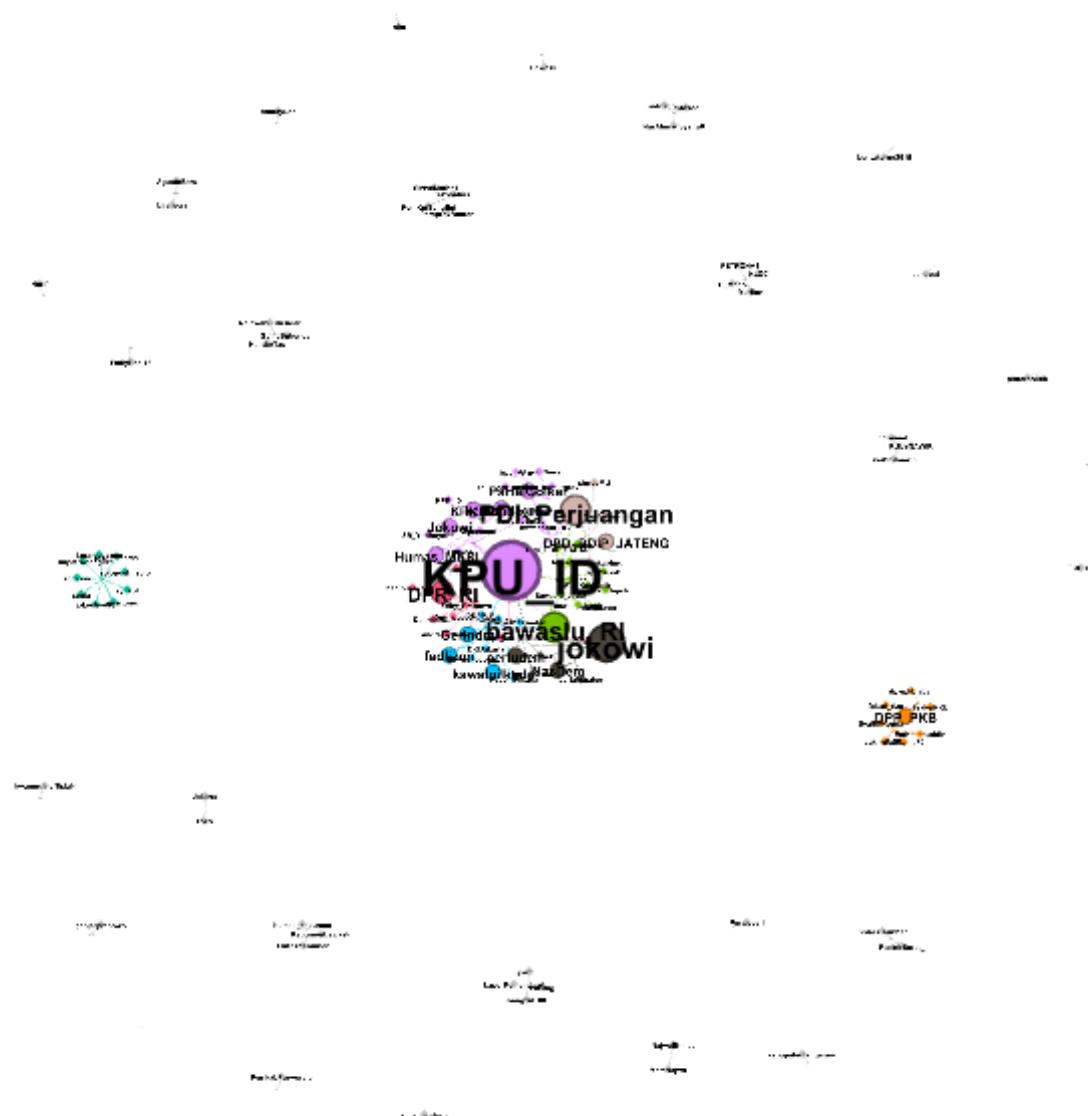


Figure 28: Hasil layout force atlas2

Gambar 28 merupakan hasil visualisasi dengan layout force atlas2. Melalui gambar tersebut kita bisa mendapatkan informasi bahwa secara umum mungkin terdapat beberapa kelompok username, yang masing-masing kelompoknya juga gabungan dari beberapa kelompok kecil lagi. Misalnya, nodes yang ada ditengah merupakan username yang terkait langsung dengan pemerintahan atau negara, seperti username jokowi, kpu, dan bawaslu.

5. Data

Hasil kalkulasi atau komputasi yang dilakukan di gephi bisa ditemukan pada menu Data Laboratory yang terletak di bagian atas. Di sana terdapat dua data, yaitu data nodes dan edges yang masing-masingnya bisa dieksport, disimpan dan diimpor kembali kedalam R, untuk tujuan visualisasi yang berbeda. Misalnya, untuk membuat plot diagram balok yang menunjukkan jumlah anggota masing-masing kelompok dan lain sebagainya.

```
library(tidyverse)

edges <- read_csv("data/edges.csv", trim_ws = FALSE)
nodes <- read_csv("data/nodes.csv", trim_ws = FALSE)

nodes %>%
  count(modularity_class, sort = TRUE) %>%
  mutate(kelompok = paste0("Kelompok ke ", modularity_class)) %>%
  head(n = 10) %>%
  ggplot(aes(x = reorder(kelompok, n), y = n)) +
  geom_col() +
  coord_flip() +
  labs(x = "Kelompok", y = "Jumlah")
```

Gambar 29 merupakan visualisasi jumlah kelompok yang datanya diambil setelah melakukan pengolahan di Gephi. Di sini kita bisa mengetahui dengan jelas, bahwa kelompok 32 memiliki anggota terbanyak. Analisis lebih lanjut bisa dilakukan berdasarkan data ini, di mana masing-masing nodes atau dalam konteks ini username bisa dijadikan sebagai basis analisis konten per kelompok modularity. Dengan demikian, kita memiliki

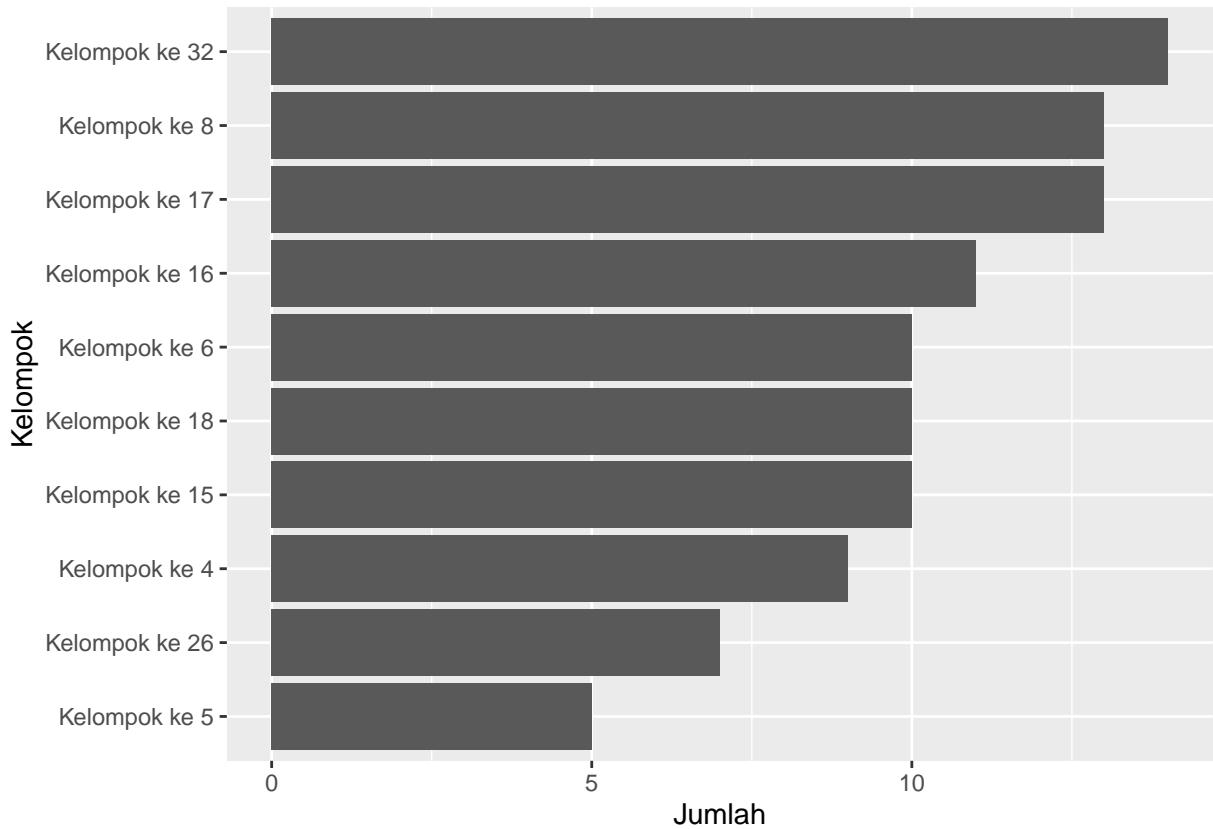


Figure 29: 10 Kelompok dengan jumlah anggota terbanyak

probabilitas lebih tinggi untuk memastikan apakah kelompok tersebut sesuai dengan yang ada di dunia nyata.

6. Menyimpan hasil visualisasi

Setelah selesai membuat visualisasi dan melakukan eksplorasi, hal terakhir yang perlu dilakukan adalah menyimpannya hasil visualisasi dalam bentuk png atau format lainnya. Untuk bisa mendapatkannya, kita bisa menggunakan menu **Preview**. Di dalam menu tersebut juga terdapat beberapa pengaturan yang bisa dilakukan, seperti menampilkan nama nodes, mengubah warna edges, background dan lain sejenisnya. Setelah pengaturan selesai dilakukan, selanjutnya kita bisa menyimpannya dengan menggunakan **Export** yang ada dibagian pojok kiri bawah. Pada saat mengekspor, besar file bisa ditentukan setelah menekan tombol tersebut.

Selain untuk menyimpan hasil visualisasi, gephi juga bisa untuk menyimpan graph objek, yaitu dengan menggunakan menu **file > export > graph file** dilanjutkan dengan memilih format yang diinginkan. Sementara untuk menyimpan hasil visualisasi dalam

format html, sejauh ini kita bisa memanfaatkan plugin yang tersedia, yaitu **SigmaExporter**. Untuk keterangan lebih lanjut terkait dengan plugin tersebut bisa merujuk pada tautan ini <https://github.com/oxfordinternetinstitute/gephi-plugins/tree/sigmaexporter-plugin> dan <http://blogs.ox.ac.uk/vis/>.

6.4 Latihan 13

Berdasarkan penjelasan pada bagian Visualisasi network menggunakan Gephi, jawablah pertanyaan-pertanyaan berikut:

1. Buatlah Social Network Analisis menggunakan gephi, baik dengan menggunakan data yang dibuat sendiri, maupun dengan mengambil data yang disediakan.
2. Buatlah interpretasi dari hasil analisis dan presentasikan di depan kelas selama 5 menit.

7 Penutup dan Referensi

Catatan Akhir

Referensi ini menunjukkan beberapa bahan yang bisa menjadi rujukan peserta pelatihan untuk dapat mengembangkan pengetahuan dan keterampilan melakukan **network analysis** menggunakan R dan Gephi. Dua alat ini dianggap bisa saling melengkapi dan memudahkan kita dalam melakukan analisis network. Secara umum, R akan banyak digunakan untuk melakukan pre-processing. Sementara Gephi sangat nyaman digunakan untuk mengolah dan mengekstrak data berupa graph objek.

Visualisasi network di R sendiri sebenarnya sangat kaya, berikut adalah beberapa **packages** untuk visualisasi network interaktif yang bisa dipelajari lebih lanjut.

1. NetworkD3: <https://github.com/christophergandrud/networkD3>
2. visNetwork: <https://github.com/datastorm-open/visNetwork>
3. sigmajs: <https://github.com/JohnCoene/sigmajs>

Referensi

- Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695. Retrieved from <http://igraph.org>
- Pedersen, T. L. (2019). *Ggraph: An implementation of grammar of graphics for graphs and networks*. Retrieved from <https://CRAN.R-project.org/package=ggraph>
- Silge, J., & Robinson, D. (2016). Tidytext: Text mining and analysis using tidy data principles in r. *JOSS*, 1(3). doi:[10.21105/joss.00037](https://doi.org/10.21105/joss.00037)
- Wickham, H., & Bryan, J. (2019). *Readxl: Read excel files*. Retrieved from <https://CRAN.R-project.org/package=readxl>
- Wickham, H., François, R., Henry, L., & Müller, K. (2019). *Dplyr: A grammar of data manipulation*. Retrieved from <https://CRAN.R-project.org/package=dplyr>
- Wickham, H., & Henry, L. (2019). *Tidyr: Tidy messy data*. Retrieved from <https://CRAN.R-project.org/package=tidyr>
- Wickham, H., Hester, J., & Francois, R. (2018). *Readr: Read rectangular text data*. Retrieved from <https://CRAN.R-project.org/package=readr>
- Yon, G. V., Lacoa, J. F., & Kunst, J. B. (2015). *Rgexf: Build, import and export gexf graph files*. Retrieved from <https://CRAN.R-project.org/package=rgexf>