Analisis Cluster dengan R: Panduan Lengkap untuk Pemula

Deden Istiawan

2024-12-31

Contents

Kata Pengantar

Rekayasa Perangkat Lunak Fakultas Sains dan Teknologi Institut Teknologi Statistika dan Bisnis Muhammadiyah Semarang, Indonesia deden.istiawan@itesa.ac.id. Last updated on Tuesday, December 31, 2024.

Analisis *cluster* merupakan teknik yang sangat penting dalam bidang data science, karena kemampuannya untuk mengelompokkan data berdasarkan kesamaan karakteristik. Dengan mengelompokkan objek atau data, analisis cluster membantu mengidentifikasi pola yang mungkin tidak terlihat secara langsung dalam data mentah. Hal ini sangat berguna di berbagai bidang, seperti pemasaran, biologi, dan pengenalan pola, di mana pemahaman yang lebih dalam tentang struktur data dapat memberikan wawasan yang berharga (Kaufman and Rousseeuw, 1990a).

Salah satu manfaat utama dari analisis cluster adalah segmentasi pasar. Dalam dunia bisnis, perusahaan dapat menggunakan teknik ini untuk mengelompokkan pelanggan berdasarkan perilaku atau karakteristik tertentu. Dengan memahami segmen-segmen ini, perusahaan dapat merancang strategi pemasaran yang lebih efektif dan personalisasi produk untuk memenuhi kebutuhan spesifik dari setiap kelompok. Selain itu, analisis cluster juga memungkinkan pengelompokan data yang besar menjadi lebih mudah dipahami dan diinterpretasikan, sehingga memfasilitasi pengambilan keputusan yang lebih baik (Han et al., 2012).

Dalam konteks machine learning, analisis cluster sering digunakan sebagai metode unsupervised learning. Ini berarti bahwa algoritma clustering dapat menemukan struktur dalam data tanpa memerlukan label atau kategori sebelumnya. Hal ini sangat berguna ketika menghadapi dataset yang tidak terstruktur dan kompleks, di mana tujuan utamanya adalah untuk mengeksplorasi dan mendapatkan wawasan awal tentang data tersebut (Jain, 2010).

Secara keseluruhan, analisis cluster tidak hanya membantu dalam pengelompokan data tetapi juga berfungsi sebagai alat eksplorasi yang kuat dalam memahami hubungan dan pola dalam dataset. Dengan demikian, penerapan analisis cluster dapat meningkatkan kualitas analisis data dan mendukung pengambilan keputusan berbasis data yang lebih baik di berbagai sektor industri .

Acknowledgement

Saya ingin mengucapkan terima kasih kepada keluarga, teman, dan rekan kerja yang telah mendukung saya dalam menyelesaikan buku ini. Terima kasih atas cinta, dukungan, dan motivasi yang telah Anda berikan

Chapter 1

Pendahuluan

> "Why is that when one man builds a wall, the next needs to to know what's on the other side?"
>
> — Tyrion Lannister-Game of Thrones

Minat terhadap bahasa pemrograman statistik dan lingkungan perangkat lunak `R` (R Core Team, 2024) yang tersedia secara gratis terus berkembang pesat. Saat draf awal proyek ini disusun, lebih dari 11.000 add-on, termasuk metode terbaru, telah tersedia di Comprehensive R Archive Network (CRAN), jaringan server FTP global yang menyimpan versi kode `R` dan dokumentasi terbaru secara identik. Dalam banyak bidang penelitian statistik terapan, `R` menjadi dominan dibandingkan perangkat lunak statistik lainnya, termasuk yang bersifat komersial. Keunggulan `R` sebagai perangkat lunak sumber terbuka yang gratis, ditambah dengan komunitas pengguna yang besar dan aktif yang terus berkontribusi pada CRAN, menjadikannya pilihan menarik bagi peneliti di berbagai disiplin ilmu seperti statistik, ekonomi, dan teknik.

Salah satu keuntungan utama menggunakan `R` dalam analisis data adalah kemampuannya untuk mendokumentasikan setiap langkah analisis secara eksplisit sehingga mudah untuk diperbarui dan diperluas. Hal ini memungkinkan pengguna untuk menggunakan kembali kode untuk aplikasi serupa dengan data yang berbeda. Selain itu, program `R`sepenuhnya dapat direproduksi, yang membuat hasil analisis mudah dipahami dan divalidasi oleh pihak lain.

Selama beberapa tahun terakhir, `R` telah menjadi bagian integrasi dari kurikulum mata kuliah analisis data dan statistika yang kami ajarkan di berbagai universitas. Dalam konteks pembelajaran, mempelajari pemrograman `R` sering kali dianggap sebanding dengan mempelajari bahasa asing, di mana praktik yang berkelanjutan sangat penting untuk keberhasilan belajar. Namun, hanya menampilkan kode R mentah pada slide presentasi tidak cukup untuk mendorong mahasiswa berinteraksi langsung dengan materi pembelajaran. Oleh karena itu, `R` sangat penting untuk memberikan pengalaman belajar yang lebih mendalam dan interaktif.

Dalam hal referensi literatur, sudah ada beberapa buku luar biasa yang membahas penggunaan `R` dan penerapannya, seperti *An Introduction to Statistical Learning* (James et al., 2013). Namun, banyak sumber tersebut mungkin terlalu kompleks untuk mahasiswa pemula yang baru memulai belajar statistik atau memiliki sedikit pengalaman pemrograman. Untuk menjawab tantangan ini, kami mulai menyusun koleksi panduan yang dapat direproduksi untuk digunakan dalam kelas. Panduan ini memberikan arahan tentang bagaimana menerapkan berbagai metode analisis klaster yang diambil dari literatur statistik dengan menggunakan `R` sebagai alat utama. Proses ini sangat terbantu oleh paket-paket R seperti `knitr` dan `rmarkdown` yang memungkinkan pembuatan laporan dinamis dengan mengintegrasikan teks, kode `R`, dan hasil output dalam berbagai format, termasuk PDF dan HTML.

Selain itu, paket `bookdown` telah menjadi alat utama kami dalam proyek ini. Paket ini dibangun di atas `rmarkdown` dan memungkinkan kami untuk membuat dokumen yang menarik secara visual, seperti halaman web interaktif. Dengan dukungan perangkat ini, kami menyusun sebuah buku berjudul *Analisis Cluster dengan R: Panduan Lengkap untuk Pemula*. Buku ini dirancang sebagai skrip interaktif dalam gaya laporan penelitian yang dapat direproduksi, dengan tujuan menyediakan pengaturan pembelajaran elektronik yang platform-independen dengan mengintegrasikan pengetahuan teoretis dan keterampilan empiris dalam analisis data.

Proyek ini bukanlah buku teks analisis klaster yang komprehensif atau pengenalan umum tentang `R`. Fokus kami adalah memberikan panduan praktis tentang bagaimana metode analisis klaster dapat diterapkan menggunakan `R`, tanpa terlalu banyak membahas derivasi matematis atau bukti formal. Tujuan kami adalah memungkinkan pembaca tidak hanya mereplikasi hasil analisis kasus menggunakan `R` tetapi juga memperkuat kemampuan mereka untuk menerapkan keterampilan baru tersebut pada aplikasi empiris lainnya.

Setiap bab dalam buku ini mencakup latihan pemrograman `R` interaktif. Latihan ini dirancang sebagai pelengkap potongan kode yang menampilkan implementasi teknik-teknik yang telah dibahas. Latihan-latihan ini dibuat menggunakan DataCamp light widget yang terhubung dengan sesi `R` yang dikelola di server DataCamp. Dengan pendekatan ini, pembaca dapat langsung mencoba dan bereksperimen dengan latihan yang disediakan untuk memperdalam pemahaman mereka terhadap metode yang dipelajari. Dengan cara ini, kami berharap dapat memberikan pengalaman belajar yang menyeluruh dan menarik bagi pembaca yang ingin mendalami analisis klaster menggunakan `R`.

Widget ini terdiri dari dua tab. Tab `script.R` menyerupai file dengan format `.R`, format file yang umum digunakan untuk menyimpan kode `R`. Baris yang diawali dengan tanda # adalah komentar, yang tidak dianggap sebagai kode oleh `R`. Selain itu, tab script.R bekerja seperti lembar latihan di mana Anda dapat menuliskan solusi yang Anda buat. Jika Anda menekan tombol *Run*, kode akan dijalankan, pengujian kebenaran solusi dilakukan, dan Anda akan diberi tahu apakah pendekatan Anda benar. Jika tidak benar, Anda akan menerima umpan balik yang memberikan saran perbaikan atau petunjuk. Tab lainnya, `R Console`, adalah konsol `R` yang sepenuhnya berfungsi yang dapat digunakan untuk mencoba solusi sebelum mengirimkannya. Tentu saja, Anda dapat mengirimkan hampir semua kode `R` dan menggunakan konsol untuk bereksperimen dan mengeksplorasi. Cukup ketikkan perintah dan tekan tombol *Enter* pada keyboard Anda.

Pada konsol, Anda akan melihat simbol > di panel kanan (konsol). Simbol ini disebut "prompt" dan menunjukkan bahwa pengguna dapat memasukkan kode yang akan dijalankan. Untuk menghindari kebingungan, kami tidak akan menampilkan simbol ini di buku ini. Output yang dihasilkan oleh kode R dikomentari dengan #>.

Biasanya, kami menampilkan kode `R` bersama dengan output yang dihasilkan dalam potongan kode. Sebagai contoh, perhatikan baris kode berikut yang ditampilkan dalam potongan di bawah ini. Kode ini memberi tahu `R` untuk menghitung jumlah paket yang tersedia di `CRAN`. Potongan kode diikuti oleh output yang dihasilkan.

```
# Cek jumlah paket R yang tersedia di CRAN
nrow(available.packages(repos = "http://cran.us.r-project.org"))
#> [1] 21834
```

Setiap potongan kode dilengkapi dengan tombol di sisi kanan luar yang memungkinkan Anda menyalin kode ke clipboard Anda. Ini membuat pekerjaan dengan segmen kode yang lebih besar menjadi lebih nyaman, baik di versi R/*RStudio* Anda maupun di widget-widget yang disajikan di sepanjang buku ini. Pada widget di atas, Anda dapat mengklik tab `R Console` Console dan mengetikkan `nrow(available.packages(repos = "http://cran.us.r-project.org"))` (perintah dari potongan

kode di atas) dan menjalankannya dengan menekan tombol *Enter* pada keyboard Anda. [1]

Perhatikan bahwa beberapa baris dalam widget tersebut dikomentari, yang meminta Anda untuk menetapkan nilai numerik pada sebuah variabel dan kemudian mencetak isi variabel tersebut ke konsol. Anda dapat memasukkan pendekatan solusi Anda ke dalam `script.R` dan menekan tombol *Run* untuk mendapatkan umpan balik seperti yang dijelaskan di atas. Jika Anda tidak tahu bagaimana menyelesaikan latihan ini (jangan panik, itu mungkin alasan mengapa Anda membaca ini), klik *Hint* untuk mendapatkan saran. Jika Anda masih belum menemukan solusi, klik *Solution* untuk membuka tab lain, `Solution.R`, yang berisi kode solusi sampel. Seringkali, latihan dapat diselesaikan dengan berbagai cara, dan `Solution.R` akan menyajikan solusi yang kami anggap sebagai solusi yang mudah dipahami dan idiomatik.

1.1 Kolofon

Buku ini ditulis menggunakan bookdown di dalam RStudio. Seluruh kode sumbernya dapat diakses di GitHub.

```
#> - Session info ---------------------------------------------------------------
#>  setting  value
#>  version  R version 4.4.0 (2024-04-24 ucrt)
#>  os       Windows 11 x64 (build 22631)
#>  system   x86_64, mingw32
#>  ui       RTerm
#>  language (EN)
#>  collate  English_Indonesia.utf8
#>  ctype    English_Indonesia.utf8
#>  tz       Asia/Jakarta
#>  date     2024-12-31
#>  pandoc   3.2 @ C:/Program Files/RStudio/resources/app/bin/quarto/bin/tools/ (via rmarkdown)
#>
#> - Packages -------------------------------------------------------------------
#>  package     * version date (UTC) lib source
#>  bookdown      0.39    2024-04-15 [1] CRAN (R 4.4.0)
#>  cli           3.6.2   2023-12-11 [1] CRAN (R 4.4.0)
#>  digest        0.6.35  2024-03-11 [1] CRAN (R 4.4.0)
#>  evaluate      1.0.1   2024-10-10 [1] CRAN (R 4.4.2)
#>  fastmap       1.2.0   2024-05-15 [1] CRAN (R 4.4.0)
#>  htmltools     0.5.8.1 2024-04-04 [1] CRAN (R 4.4.0)
#>  knitr         1.47    2024-05-29 [1] CRAN (R 4.4.0)
#>  rlang         1.1.4   2024-06-04 [1] CRAN (R 4.4.0)
#>  rmarkdown     2.27    2024-05-17 [1] CRAN (R 4.4.0)
#>  rstudioapi    0.16.0  2024-03-24 [1] CRAN (R 4.4.0)
#>  sessioninfo   1.2.2   2021-12-06 [1] CRAN (R 4.4.2)
#>  tinytex       0.51    2024-05-06 [1] CRAN (R 4.4.0)
#>  xfun          0.49    2024-10-31 [1] CRAN (R 4.4.1)
#>  yaml          2.3.8   2023-12-11 [1] CRAN (R 4.4.0)
#>
#>  [1] C:/Users/Deden/AppData/Local/R/win-library/4.4
```

[1] The `R` session is initialized by clicking into the widget. This might take a few seconds. Just wait for the indicator next to the button *Run* to turn green.

```
#>  [2] C:/Program Files/R/R-4.4.0/library
#>
#> ------------------------------------------------------------------------------
```

1.2 Pengenalan Singkat R and RStudio

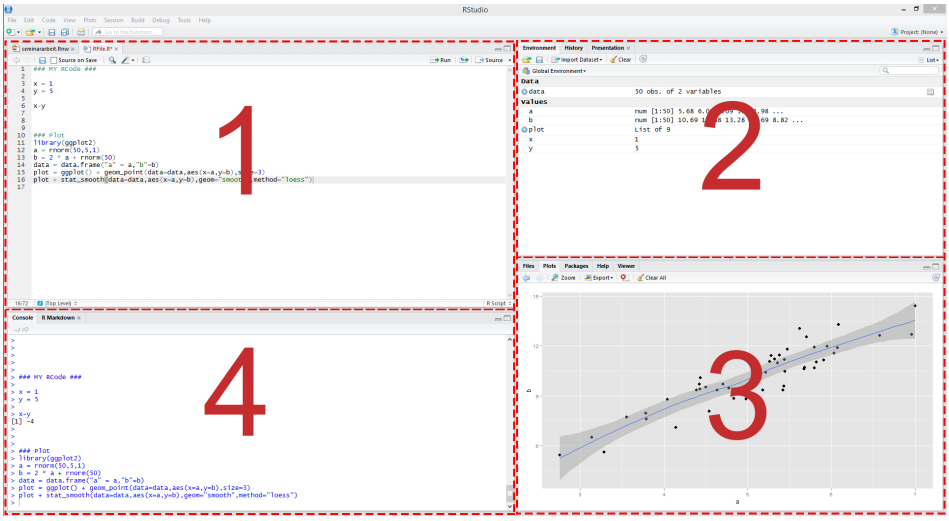

Figure 1.1: RStudio: empat panel

Dasar-Dasar R

Seperti disebutkan sebelumnya, buku ini bukan dimaksudkan sebagai pengantar R, melainkan panduan tentang cara memanfaatkan kemampuannya untuk aplikasi yang umum analisis *cluster* dengan R. Pembaca yang memiliki pengetahuan dasar tentang pemrograman R mungkin akan merasa nyaman untuk langsung memulai dari **Bab 2**. Namun, bagian ini ditujukan bagi mereka yang belum pernah bekerja dengan R atau *RStudio* sebelumnya. Jika Anda setidaknya sudah tahu cara membuat objek dan memanggil fungsi, Anda bisa melewati bagian ini. Namun, jika Anda ingin menyegarkan keterampilan Anda atau mendapatkan gambaran tentang cara bekerja dengan *RStudio*, lanjutkan membaca.

Pertama-tama, buka *RStudio* dan buat skrip R baru dengan memilih *File*, *New File*, *R Script*. Di panel editor, ketikkan

```
1 + 1
```

dan klik tombol berlabel *Run* di pojok kanan atas editor. Dengan melakukan ini, baris kode Anda akan dikirim ke konsol, dan hasil operasi tersebut akan ditampilkan tepat di bawahnya. Seperti yang Anda lihat, R berfungsi layaknya kalkulator. Anda dapat melakukan semua perhitungan aritmatika menggunakan operator yang sesuai (`+`, `-`, `*`, `/` or `^`). Jika Anda tidak yakin apa fungsi operator terakhir, coba gunakan dan periksa hasilnya.

Vektor

Dalam R, Anda dapat menggunakan variabel atau objek. Untuk membuat objek, gunakan operator penugasan `<-`. Misalnya, untuk mendefinisikan variabel bernama `x` dengan nilai `10`, ketik `x <- 10`

dan tekan tombol *Run*. Variabel tersebut akan muncul di panel environment di kanan atas. Namun, konsol tidak akan menunjukkan output karena kode ini tidak menghasilkan tampilan apa pun. Untuk melihat nilai `x`, cukup ketik `x` di konsol dan tekan *Enter*, lalu `R` akan menampilkan nilainya di konsol.

`x` merupakan skalar, yaitu vektor dengan panjang 1. Untuk membuat vektor yang lebih panjang, Anda bisa menggunakan fungsi `c()` (*c* berarti "gabungkan" atau "kombinasikan"). Misalnya, untuk membuat vektor `y` yang berisi angka 1 hingga 5 dan menampilkannya, ikuti langkah berikut.

```
y <- c(1, 2, 3, 4, 5)
y
#> [1] 1 2 3 4 5
```

Anda juga dapat membuat vektor yang berisi huruf atau kata. Perlu diingat bahwa karakter harus diberi tanda kutip, jika tidak, mereka akan diartikan sebagai nama objek.

```
hello <- c("Hello", "World")
```

Di sini, kita telah membuat vektor yang terdiri dari dua elemen, yaitu kata `Hello` dan `World`.

Jangan lupa untuk menyimpan skrip Anda! Caranya, pilih *File* dan kemudian *Save*.

Funsi

Anda telah melihat bahwa fungsi `c()` digunakan untuk menggabungkan objek. Secara umum, setiap pemanggilan fungsi memiliki format yang serupa: nama fungsi diikuti oleh tanda kurung bulat. Biasanya, tanda kurung tersebut berisi argumen.

Berikut dua contoh sederhana.

```
# membuat vektor `z`
z <- seq(from = 1, to = 5, by = 1)

# menghitung rata-rata dari entri dalam `z`
mean(z)
#> [1] 3
```

Pada baris pertama, kita menggunakan fungsi `seq()` untuk menghasilkan vektor yang sama persis seperti yang dibuat pada bagian sebelumnya, dan menamakannya `seq()`. Fungsi ini menerima argumen `from`,`to` dan `by`, yang sudah cukup jelas. Fungsi `mean()` menghitung nilai rata-rata dari argumennya, yaitu `x`. Karena vektor `z` diberikan sebagai argumen `x`, hasilnya adalah `3`!

Jika Anda tidak yakin tentang argumen apa yang dibutuhkan oleh suatu fungsi, Anda dapat memeriksa dokumentasinya. Misalnya, jika kita ragu mengenai cara kerja argumen di `seq()`, ketik `?seq` di konsol. Setelah menekan *Enter*, halaman dokumentasi fungsi tersebut akan muncul di panel kanan bawah *RStudio*. Di bagian *Arguments*, Anda akan menemukan informasi yang dibutuhkan. Di bagian bawah hampir setiap halaman dokumentasi, terdapat contoh penggunaan fungsi tersebut. Ini sangat bermanfaat bagi pemula, dan kami sarankan untuk melihatnya.

Tentu saja, semua perintah yang telah dijelaskan di atas juga dapat digunakan pada widget interaktif di seluruh buku ini. Anda dapat mencobanya di bawah ini.

This interactive application is only available in the HTML version.

Chapter 2

Algoritma K-Means

Algoritma K-Means merupakan salah satu metode yang paling populer dalam analisis *cluster*, yang digunakan untuk mengelompokkan data ke dalam beberapa kelompok berdasarkan kesamaan fitur (Jain, 2010) . Metode ini bekerja dengan cara membagi data ke dalam K cluster, di mana K adalah jumlah *cluster* yang ditentukan sebelumnya oleh pengguna (MacQueen, 1967). Proses pengelompokan ini dilakukan dengan meminimalkan jarak antara data dalam satu *cluster* dan pusat *cluster* (centroid) yang dihasilkan. Langkah pertama dalam algoritma K-Means adalah inisialisasi centroid untuk setiap *cluster*, yang dapat dilakukan secara acak atau menggunakan metode tertentu seperti K-Means++ (Arthur and Vassilvitskii, 2007). Setelah centroid diinisialisasi, algoritma kemudian mengelompokkan setiap data ke dalam *cluster* terdekat berdasarkan jarak Euclidean(Hastie et al., 2009). Proses ini diulang hingga tidak ada perubahan signifikan dalam posisi *centroid* atau tidak ada perubahan dalam pengelompokan data.

Salah satu keunggulan dari algoritma K-Means adalah kesederhanaannya, yang membuatnya mudah dipahami dan diimplementasikan, bahkan oleh pemula (Han et al., 2011). Namun, algoritma ini juga memiliki beberapa kelemahan, seperti ketergantungan pada pemilihan jumlah *cluster* K yang tepat dan sensitivitas terhadap *outlier*. Oleh karena itu, pemilihan K yang optimal sering kali menjadi tantangan tersendiri dalam penerapan algoritma ini (Elbow, 1975). Dalam praktiknya, algorima K-Means banyak digunakan dalam berbagai bidang, termasuk pemasaran, pengenalan pola, dan analisis citra (Xu and Wunsch, 2005). Misalnya, dalam pemasaran, K-Means dapat digunakan untuk segmentasi pelanggan berdasarkan perilaku pembelian mereka, sehingga perusahaan dapat menargetkan strategi pemasaran yang lebih efektif (Kumar and Reinartz, 2016). Dengan demikian, pemahaman yang baik tentang algoritma K-Means sangat penting bagi para peneliti dan praktisi yang ingin menerapkan analisis *cluster* dalam pekerjaan mereka.

Secara keseluruhan, algoritma K-Means adalah alat yang kuat dan fleksibel untuk analisis *cluster*, meskipun penggunaannya memerlukan pemahaman yang mendalam tentang data dan konteks analisis (Bishop, 2006). Dalam buku ini, kami akan membahas secara rinci tentang cara menerapkan algoritma K-Means menggunakan `R`, serta memberikan panduan langkah demi langkah untuk membantu pemula memahami dan menguasai teknik ini. Dengan demikian, diharapkan pembaca dapat memanfaatkan algoritma K-Means dalam analisis data mereka secara efektif.

2.1 Tahapan Algoritma K-Means

1. Inisialisasi Centroid

Pilih jumlah cluster K yang diinginkan. Inisialisasi K centroid secara acak dari dataset. Centroid adalah titik yang mewakili pusat dari setiap cluster.

$$C_k = (x_{k1}, x_{k2}, \ldots, x_{kn}) \quad \text{untuk } k = 1, 2, \ldots, K$$

di mana C_k adalah centroid untuk cluster ke-k dan x_{ki} adalah nilai fitur ke-i dari centroid ke-k.

2. Penugasan Cluster

Untuk setiap titik data x_i, hitung jarak ke setiap centroid C_k dan tetapkan titik data tersebut ke cluster dengan centroid terdekat. Jarak yang umum digunakan adalah jarak Euclidean.

$$d(x_i, C_k) = \sqrt{\sum_{j=1}^{m} (x_{ij} - C_{kj})^2}$$

di mana $d(x_i, C_k)$ adalah jarak antara titik data x_i dan centroid C_k, m adalah jumlah fitur, dan x_{ij} adalah nilai fitur ke-j dari titik data ke-i.

Penugasan cluster dapat dinyatakan sebagai:

$$\text{Cluster}(x_i) = \arg\min_k d(x_i, C_k)$$

3. Perbaharui Centroid

Setelah semua titik data ditugaskan ke cluster, hitung ulang posisi centroid untuk setiap cluster dengan mengambil rata-rata dari semua titik data yang ditugaskan ke cluster tersebut.

$$C_k = \frac{1}{N_k} \sum_{x_i \in \text{Cluster}_k} x_i$$

di mana N_k adalah jumlah titik data dalam cluster ke-k dan Cluster_k adalah himpunan titik data yang ditugaskan ke cluster ke-k.

4. Ulangi Langkah 2 dan 3

Ulangi langkah penugasan cluster dan pembaruan centroid hingga tidak ada perubahan signifikan dalam posisi centroid atau tidak ada perubahan dalam pengelompokan data. Kriteria konvergensi dapat berupa:

$$\text{Jika } \|C_k^{(t)} - C_k^{(t-1)}\| < \epsilon \quad \text{atau tidak ada perubahan cluster}$$

di mana ϵ adalah ambang batas kecil yang ditentukan sebelumnya.

Table 2.1: Basis Data Terpadu Jawa Tengah

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
CILACAP	5.19	5.67	5.08	5.44	5.22	6.05	11.47	9.78	5.55	5.12
BANYUMAS	5.71	4.47	5.18	5.51	5.02	6.21	7.39	6.96	5.98	8.22
PURBALINGGA	3.30	2.19	3.80	3.13	3.73	3.34	8.71	7.41	3.21	4.65
BANJARNEGARA	2.73	2.34	3.76	2.80	2.57	2.99	3.31	5.45	4.21	6.05
KEBUMEN	4.17	2.55	3.26	4.16	3.15	4.15	4.30	9.29	4.61	4.34
PURWOREJO	1.87	2.12	1.48	3.05	1.78	1.83	5.00	4.90	3.12	2.09
WONOSOBO	2.13	1.95	3.00	1.78	1.62	2.06	0.45	2.32	3.57	0.84
MAGELANG	3.95	3.01	4.22	4.15	3.01	3.64	1.44	3.35	5.69	3.67
BOYOLALI	2.19	3.07	1.61	2.74	2.11	1.82	1.71	2.34	3.41	1.55
KLATEN	3.84	5.15	1.93	4.64	4.04	3.78	8.71	4.45	3.99	3.09

5. Hasil Akhir

Setelah konvergensi tercapai, hasil akhir adalah centroid dari setiap cluster dan pengelompokan titik data ke dalam cluster yang sesuai.

2.2 Eksperimen Algoritma K-Means

Deskripsi Data

Data yang digunakan dalam eksperimen k-means ini berasal dari Tim Percepatan Penanggulangan Kemiskinan (TNP2K). TNP2K merupakan lembaga yang bertugas untuk merumuskan dan melaksanakan kebijakan serta program penanggulangan kemiskinan di Indonesia. Data yang diperoleh mencakup berbagai variabel yang relevan untuk analisis kemiskinan, termasuk indikator sosial, ekonomi, dan demografis. Penggunaan data ini bertujuan untuk mengidentifikasi pola dan kelompok dalam populasi yang berpotensi mengalami kemiskinan, sehingga dapat membantu dalam merumuskan strategi yang lebih efektif dalam penanggulangan kemiskinan. Melalui metode k-means, kami berharap dapat menemukan segmentasi yang jelas dalam data, yang pada gilirannya dapat memberikan wawasan yang lebih mendalam mengenai karakteristik kelompok-kelompok yang rentan terhadap kemiskinan.

Data

Dengan menggunakan fungsi `read.csv()` dari R, kami dapat mengimpor data langsung dari URL tersebut ke dalam lingkungan kerja R. Berikut adalah kode yang digunakan untuk membaca data: Package `reader` menyiapkan fungsi `read_csv()` untuk import data dari file CSV. Pada kasus ini digunakan data Data 40% Kemiskinan di jawa Tengah.

```
library (readr)
urlfile = "https://bit.ly/3VO3kRE"
data<-read.csv(url(urlfile), row.names = "Kabupaten")
```

Memeriksa *Missing Value*

Sebelum melakukan analisis cluster menggunakan metode k-means, penting untuk memeriksa apakah terdapat nilai yang hilang (missing values) dalam dataset. Nilai yang hilang dapat mempengaruhi hasil analisis dan interpretasi data, sehingga perlu ditangani dengan tepat. Kami akan menggunakan fungsi `is.na()` dan `colsum()` untuk menghitung jumlah nilai yang hilang dalam setiap kolom dari dataset. Jika ditemukan nilai yang hilang, langkah selanjutnya adalah memutuskan bagaimana cara menanganinya, apakah dengan menghapus baris yang mengandung nilai hilang, mengganti nilai hilang dengan rata-rata, median, atau metode imputasi lainnya.

Berikut adalah kode untuk memeriksa nilai yang hilang dalam dataset:

```
colSums(is.na(data))
#>  X1  X2  X3  X4  X5  X6  X7  X8  X9 X10
#>   0   0   0   0   0   0   0   0   0   0
```

Setelah melakukan pemeriksaan terhadap dataset, tidak ada nilai yang hilang (*missing values*) dalam data yang kami gunakan. Hal ini penting karena keberadaan nilai yang hilang dapat mempengaruhi hasil analisis dan interpretasi data.

Visualisasi Matriks jarak

Setelah mempersiapkan data dan memastikan tidak ada nilai yang hilang, langkah selanjutnya adalah membuat visualisasi matriks jarak. Visualisasi ini penting untuk memahami seberapa dekat atau jauh objek-objek dalam dataset berdasarkan variabel yang ada. Dengan menggunakan matriks jarak, kita dapat melihat pola dan hubungan antar data yang akan membantu dalam analisis *cluster*. Kami akan menggunakan library `factoextra` untuk menghitung dan memvisualisasikan matriks jarak. Fungsi `get_dist()` akan digunakan untuk menghitung jarak antar objek, dan `fviz_dist()` dari `factoextra` akan digunakan untuk membuat visualisasi. Berikut adalah kode untuk membuat visualisasi matriks jarak:

Setelah membuat visualisasi matriks jarak, kita dapat melakukan analisis untuk memahami pola dan hubungan antar objek dalam dataset. Matriks jarak memberikan informasi tentang seberapa mirip atau berbeda objek-objek dalam data berdasarkan variabel yang digunakan.

Dalam visualisasi matriks jarak yang telah dibuat, kita dapat mengamati beberapa hal:

1. **Warna dan Jarak**: Warna yang lebih gelap menunjukkan jarak yang lebih dekat antara objek, sedangkan warna yang lebih terang menunjukkan jarak yang lebih jauh. Dengan demikian, objek-objek yang memiliki warna serupa cenderung memiliki karakteristik yang lebih mirip.
2. **Kelompok Objek**: Dari visualisasi, kita dapat mengidentifikasi kelompok objek yang mungkin memiliki kesamaan. Misalnya, jika terdapat beberapa objek yang berdekatan dan memiliki warna yang sama, ini menunjukkan bahwa mereka mungkin termasuk dalam cluster yang sama.
3. **Outlier**: Objek yang jauh dari kelompok lainnya dapat dianggap sebagai outlier. Outlier ini mungkin memiliki karakteristik yang unik atau berbeda dari mayoritas data, dan perlu dianalisis lebih lanjut untuk memahami penyebab perbedaannya.

Selanjutnya, kita dapat melanjutkan dengan analisis cluster menggunakan algoritma k-means untuk mengelompokkan objek-objek dalam dataset berdasarkan jarak yang telah dihitung. Dengan demikian, kita dapat lebih memahami pola dalam data dan membuat keputusan yang lebih baik berdasarkan hasil analisis.

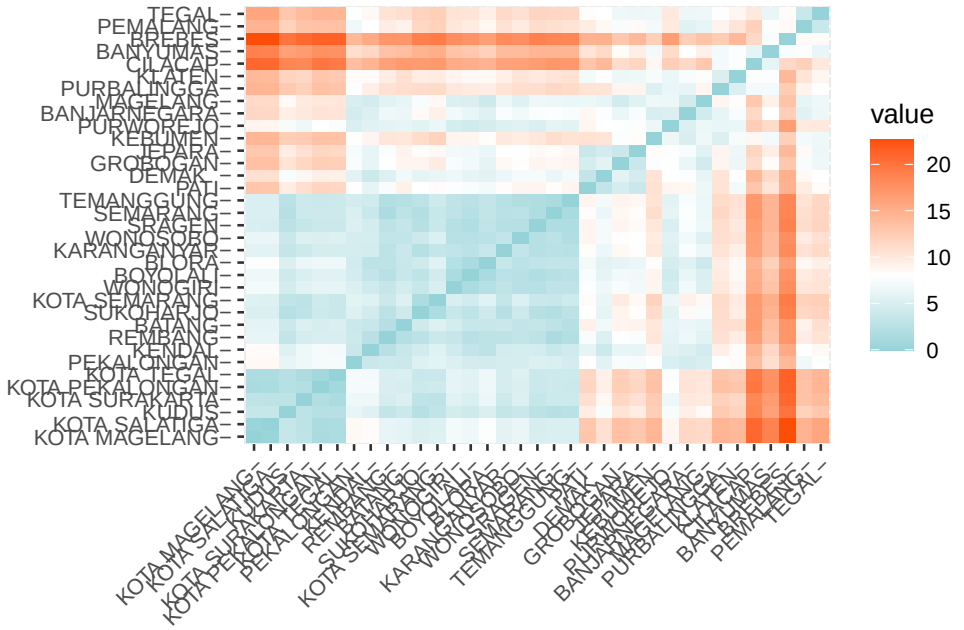

Figure 2.1: Matrik Jarak

Estimasi Jumlah *Cluster* Optimal

Dalam metode k-means banyaknya klaster ditentukan sendiri oleh pengguna. Maka dari itu perlu dicari jumlah klaster yang optimum yang dapat mengelompokkan objek dengan baik (Perlu diketahui bahwa metode ini relatif subjektif). Salah satu metode yang digunakan adalah Elbow Plot. Elbow Plot merupakan plot antara banyak klaster dengan total within-cluster variation (total dari simpangan per kluster). Banyak klaster yang dipilih adalah bagian "siku" atau titik dimana terdapat penurunan yang tajam sebelum titik tersebut dan disusul penurunan yang tidak tajam setelah titik tersebut. Hal ini karena penambahan jumlah klaster tidak membawa pengaruh banyak atas variasi yang ada di dalam klaster tersebut.

Metode Elbow

Metode Elbow merupakan suatu metode yang digunakan untuk menghasilkan informasi dalam menentukan jumlah cluster terbaik dengan cara melihat persentase hasil perbandingan antara jumlah cluster yang akan membentuk siku pada suatu titik. Metode ini memberikan ide/gagasan dengan cara memilih nilai cluster dan kemudian menambah nilai cluster tersebut untuk dijadikan model data dalam penentuan cluster terbaik. Dan selain itu persentase perhitungan yang dihasilkan menjadi pembanding antara jumlah cluster yang ditambah. Hasil persentase yang berbeda dari setiap nilai cluster dapat ditunjukan dengan menggunakan grafik sebagai sumber informasinya. Jika nilai cluster pertama dengan nilai cluster kedua memberikan sudut dalam grafik atau nilainya mengalami penurunan paling besar maka nilai cluster tersebut yang terbaik.

Metode elbow menggunakan nilai total wss (whitin sum square) sebagai penentu optimalnya. Dari gambar di atas terlihat garis mengalami patahan yang membentuk elbow atau siku pada saat = 2. Maka dengan menggunakan metode ini diperoleh optimal pada saat berada di = 2.

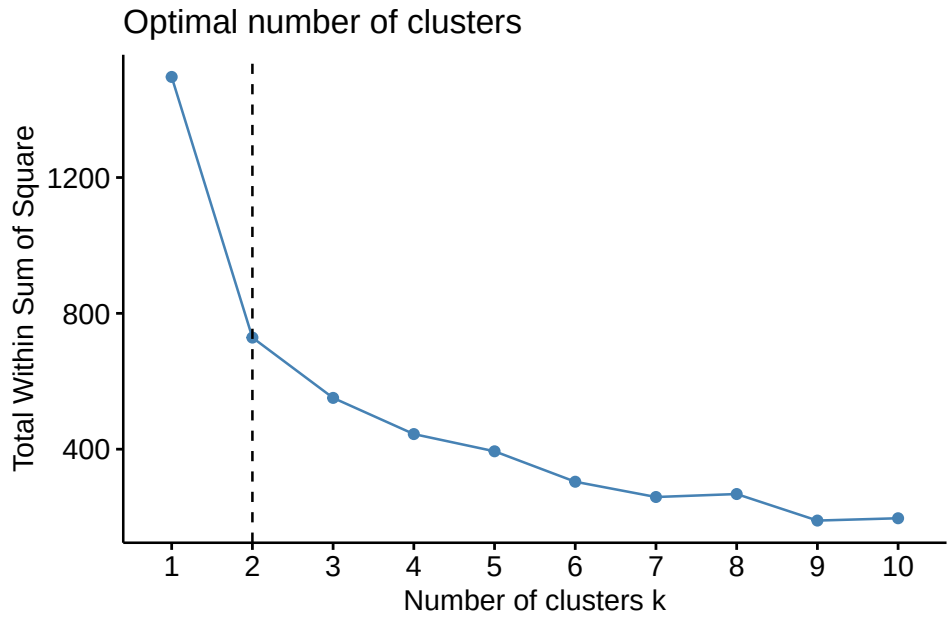

Figure 2.2: Plot Jumlah Cluster Metode Elbow

Metode Silhouette

Silhouette Coefficient digunakan untuk melihat kualitas dan kekuatan cluster, seberapa baik suatu objek ditempatkan dalam suatu cluster. Metode ini merupakan gabungan dari metode cohesion dan separation.

Pendekatan rata-rata nilai metode silhouette untuk menduga kualitas dari klaster yang terbentuk. Semakin tinggi nilai rata-ratanya maka akan semakin baik. Berdasarkan grafik pada gambar di atas banyak klaster optimal yang terbentuk pada = 2.

Membuat Plot *Cluster*

Pada eksperimen ini, algoritma K-Means diterapkan pada dataset dengan jumlah cluster yang bervariasi dari 2 hingga 5 untuk menganalisis bagaimana sebaran data berubah dengan jumlah cluster yang berbeda. Fungsi `kmeans()` digunakan untuk melakukan clustering, dengan parameter `centers` yang menentukan jumlah cluster yang diinginkan. Pada setiap percobaan, parameter `nstart = 25` digunakan untuk menjalankan algoritma K-Means sebanyak 25 kali dengan inisialisasi pusat cluster yang berbeda, untuk meningkatkan kemungkinan mendapatkan hasil yang lebih baik dan menghindari solusi lokal yang tidak optimal. Hasil clustering dengan 2, 3, 4, dan 5 cluster ini memberikan gambaran mengenai pemisahan data dalam ruang fitur dan memungkinkan kita untuk mengevaluasi sebaran data pada setiap cluster serta menentukan jumlah cluster yang paling sesuai untuk analisis lebih lanjut.

```
#gunakan beberapa nilai K
k2 <- kmeans(data, centers = 2, nstart = 25)
k3 <- kmeans(data, centers = 3, nstart = 25)
k4 <- kmeans(data, centers = 4, nstart = 25)
k5 <- kmeans(data, centers = 5, nstart = 25)
```

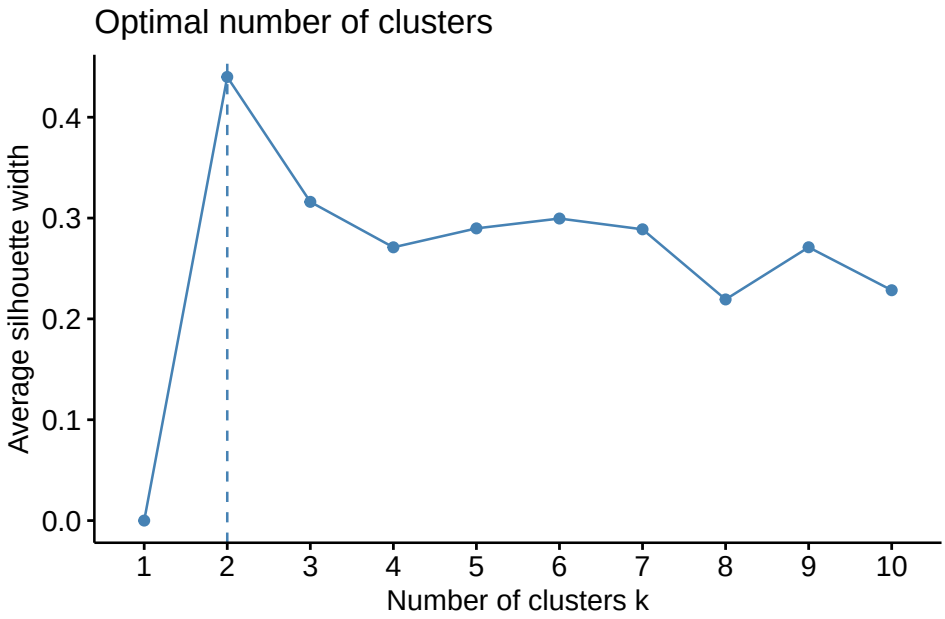

Figure 2.3: Plot Jumlah Cluster Metode silhouette

Kode ini digunakan untuk membandingkan hasil clustering dengan jumlah cluster yang berbeda, mulai dari 2 hingga 5, dengan memvisualisasikan setiap hasil clustering menggunakan fungsi `fviz_cluster()` dari paket `factoextra`. Setiap plot menggambarkan hasil clustering untuk jumlah cluster yang berbeda, di mana parameter `geom = "point"` memastikan bahwa titik data diplot sebagai titik pada grafik. Judul setiap plot, seperti `"k = 2"`, `"k = 3"`, `"k = 4"`, dan `"k = 5"`, menunjukkan jumlah cluster yang digunakan dalam setiap eksperimen clustering. Dengan menghasilkan empat plot ini, kita dapat membandingkan sebaran data dalam setiap cluster dan menganalisis bagaimana pemisahan data berubah dengan jumlah cluster yang berbeda. Visualisasi ini sangat berguna untuk membantu menentukan jumlah cluster yang optimal berdasarkan struktur data yang terlihat dalam plot.

Kode ini digunakan untuk menampilkan empat plot hasil clustering dalam satu tampilan teratur dengan menggunakan fungsi `grid.arrange()` dari paket `gridExtra`. Setiap plot yang mewakili jumlah cluster yang berbeda (2, 3, 4, dan 5) disusun dalam dua baris, masing-masing berisi dua plot, sehingga memudahkan perbandingan antar hasil clustering.

Eksperimen K-Means Clustering

Berdasarkan pendekatan metode elbow dan metode Silhouette, jumlah cluster optimal yang diperoleh adalah K=2. Metode elbow digunakan untuk menentukan jumlah cluster yang meminimalkan total within-cluster sum of squares (WSS), sementara metode Silhouette mengukur seberapa baik titik data dikelompokkan dalam cluster yang benar. Hasil dari kedua metode ini menunjukkan bahwa dua cluster adalah jumlah yang paling optimal untuk dataset ini. Oleh karena itu, eksperimen selanjutnya dilakukan dengan menggunakan K=2 untuk melakukan clustering, dengan tujuan untuk memverifikasi apakah pembagian data ke dalam dua cluster menghasilkan pemisahan yang jelas dan bermakna berdasarkan karakteristik data yang ada.

```
#jalankan k-means dengan k = 2
set.seed(123)
km.res <- kmeans(data, 2, nstart = 25)
```

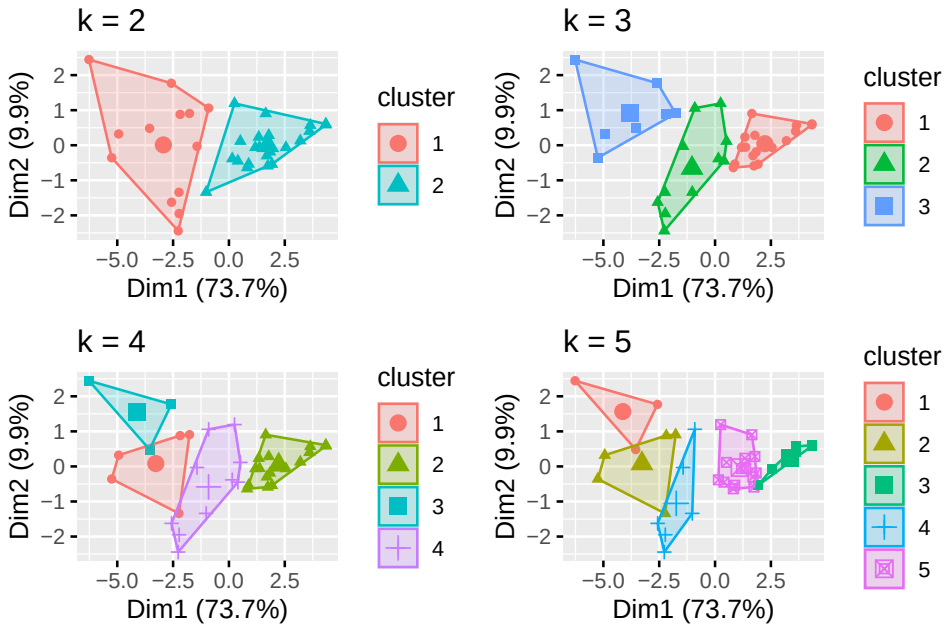

Figure 2.4: Plot Jumlah Cluster

```
# cetak hasil
print(km.res)
#> K-means clustering with 2 clusters of sizes 22, 13
#>
#> Cluster means:
#>         X1       X2       X3       X4       X5       X6       X7       X8
#> 1 1.918182 2.017273 1.675000 1.949091 1.890455 1.728182 1.557273 1.286818
#> 2 4.446923 4.276923 4.854615 4.394615 4.494615 4.769231 5.056154 5.515385
#>         X9      X10
#> 1 1.952727 1.455455
#> 2 4.389231 5.230000
#>
#> Clustering vector:
#>         CILACAP        BANYUMAS     PURBALINGGA     BANJARNEGARA         KEBUMEN
#>               2               2               2               2               2
#>       PURWOREJO        WONOSOBO        MAGELANG        BOYOLALI         KLATEN
#>               1               1               2               1               2
#>       SUKOHARJO        WONOGIRI     KARANGANYAR         SRAGEN        GROBOGAN
#>               1               1               1               1               2
#>           BLORA         REMBANG            PATI           KUDUS         JEPARA
#>               1               1               2               1               2
#>         DEMAK           SEMARANG      TEMANGGUNG         KENDAL         BATANG
#>               1               1               1               1               1
#>      PEKALONGAN        PEMALANG           TEGAL         BREBES   KOTA MAGELANG
#>               1               2               2               2               1
#>  KOTA SURAKARTA   KOTA SALATIGA   KOTA SEMARANG KOTA PEKALONGAN      KOTA TEGAL
#>               1               1               1               1               1
#>
```

```
#> Within cluster sum of squares by cluster:
#> [1] 262.6536 466.0163
#>  (between_SS / total_SS =  51.3 %)
#>
#> Available components:
#>
#> [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
#> [6] "betweenss"    "size"         "iter"         "ifault"
```

Untuk melihat hasil akhir clustering K-Means pada setiap kabupaten dalam dataset. Dengan menggunakan `km.res$cluster`, kita dapat memperoleh informasi mengenai anggota setiap cluster yang terbentuk setelah algoritma K-Means dijalankan. Hasil ini akan menunjukkan cluster mana yang menjadi bagian dari setiap kabupaten berdasarkan hasil eksperimen dengan K=2, yang sebelumnya ditentukan sebagai jumlah cluster optimal. Setiap nilai dalam `km.res$cluster` mengindikasikan nomor cluster (misalnya, 1 atau 2) yang sesuai dengan kabupaten tertentu, yang memungkinkan kita untuk menganalisis distribusi dan pemisahan kabupaten berdasarkan karakteristik yang relevan dalam dataset.

```
# Jumlah anggota cluster
km.res$cluster
#>         CILACAP        BANYUMAS     PURBALINGGA    BANJARNEGARA         KEBUMEN
#>               2               2               2               2               2
#>       PURWOREJO        WONOSOBO        MAGELANG        BOYOLALI         KLATEN
#>               1               1               2               1               2
#>       SUKOHARJO        WONOGIRI     KARANGANYAR          SRAGEN        GROBOGAN
#>               1               1               1               1               2
#>           BLORA         REMBANG            PATI           KUDUS         JEPARA
#>               1               1               2               1               2
#>         DEMAK          SEMARANG      TEMANGGUNG          KENDAL          BATANG
#>               1               1               1               1               1
#>      PEKALONGAN        PEMALANG           TEGAL          BREBES   KOTA MAGELANG
#>               1               2               2               2               1
#>  KOTA SURAKARTA   KOTA SALATIGA   KOTA SEMARANG KOTA PEKALONGAN      KOTA TEGAL
#>               1               1               1               1               1
```

Kode ini digunakan untuk melihat ukuran masing-masing cluster setelah algoritma K-Means dijalankan. Dengan menggunakan `km.res$size`, kita dapat memperoleh jumlah anggota atau titik data yang termasuk dalam setiap cluster. Hasil dari `km.res$size` akan memberikan informasi mengenai berapa banyak kabupaten yang tergolong dalam setiap cluster yang terbentuk, baik itu cluster 1 atau cluster 2, berdasarkan hasil eksperimen dengan K=2. Informasi ini sangat berguna untuk memahami distribusi data dan sebaran anggota di setiap cluster yang dihasilkan oleh proses clustering.

```
# ukuran cluster
km.res$size
#> [1] 22 13
```

Visualisasi Hasil *clustering*

Kode ini digunakan untuk memvisualisasikan hasil clustering K-Means pada dataset. Pertama, `km.res$centers` digunakan untuk menampilkan pusat cluster yang ditemukan oleh algoritma

K-Means, yang mewakili rata-rata dari titik-titik data yang tergolong dalam setiap cluster. Pusat cluster ini memberikan gambaran tentang karakteristik masing-masing cluster. Selanjutnya, fungsi `fviz_cluster(km.res, data = data)` digunakan untuk memvisualisasikan hasil clustering dalam bentuk plot. Titik-titik data diplot dengan warna yang berbeda untuk masing-masing cluster, memudahkan kita untuk melihat pemisahan antar cluster serta sebaran data dalam ruang fitur. Visualisasi ini sangat berguna untuk memahami seberapa baik pemisahan antara cluster dan memberikan wawasan mengenai struktur data setelah dilakukan clustering.

```
#>         X1       X2       X3       X4       X5       X6       X7       X8
#> 1 1.918182 2.017273 1.675000 1.949091 1.890455 1.728182 1.557273 1.286818
#> 2 4.446923 4.276923 4.854615 4.394615 4.494615 4.769231 5.056154 5.515385
#>         X9      X10
#> 1 1.952727 1.455455
#> 2 4.389231 5.230000
```

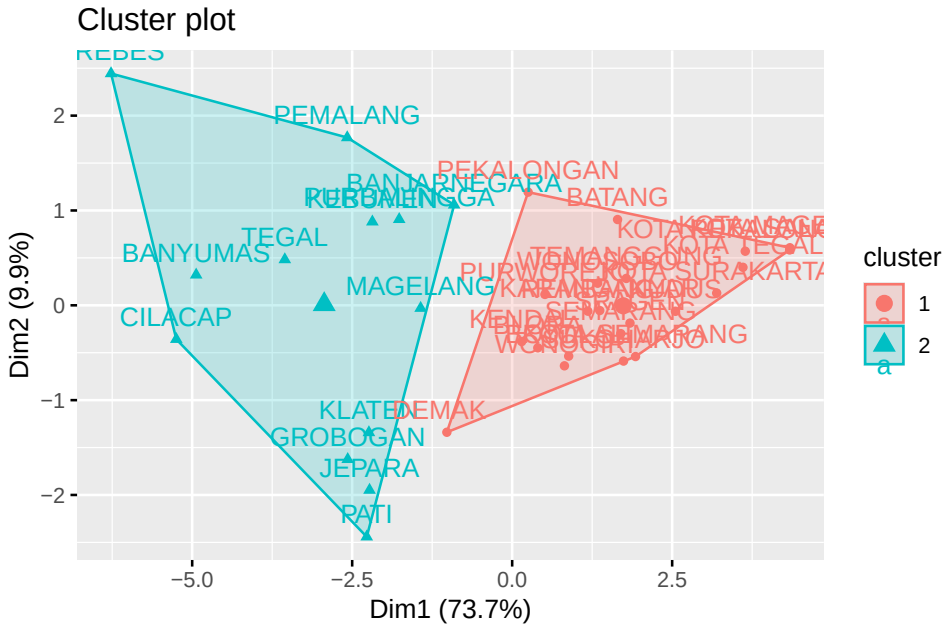

Figure 2.5: Plot Hasil Cluster

Kode ini digunakan untuk memvisualisasikan hasil clustering K-Means dengan tampilan yang lebih informatif dan menarik. Fungsi `fviz_cluster()` digunakan untuk menggambarkan hasil clustering, di mana setiap cluster diberi warna yang berbeda—oranye untuk cluster pertama dan biru kehijauan untuk cluster kedua. Penambahan ellips dengan tipe Euclidean memperjelas batasan area setiap cluster, menunjukkan sebaran titik data di dalamnya. Selain itu, fitur star plot diaktifkan untuk menggambarkan titik pusat setiap cluster, dengan garis yang menghubungkannya ke titik data dalam cluster tersebut, memberikan gambaran visual yang jelas tentang posisi pusat dan distribusi data. Untuk meningkatkan keterbacaan, fitur `repel` digunakan agar label titik data tidak tumpang tindih, dan tema minimal dari `ggplot2` (`theme_minimal()`) diterapkan untuk menghasilkan tampilan plot yang lebih bersih dan sederhana. Dengan pengaturan ini, visualisasi hasil clustering menjadi lebih mudah dipahami dan lebih menarik secara visual.

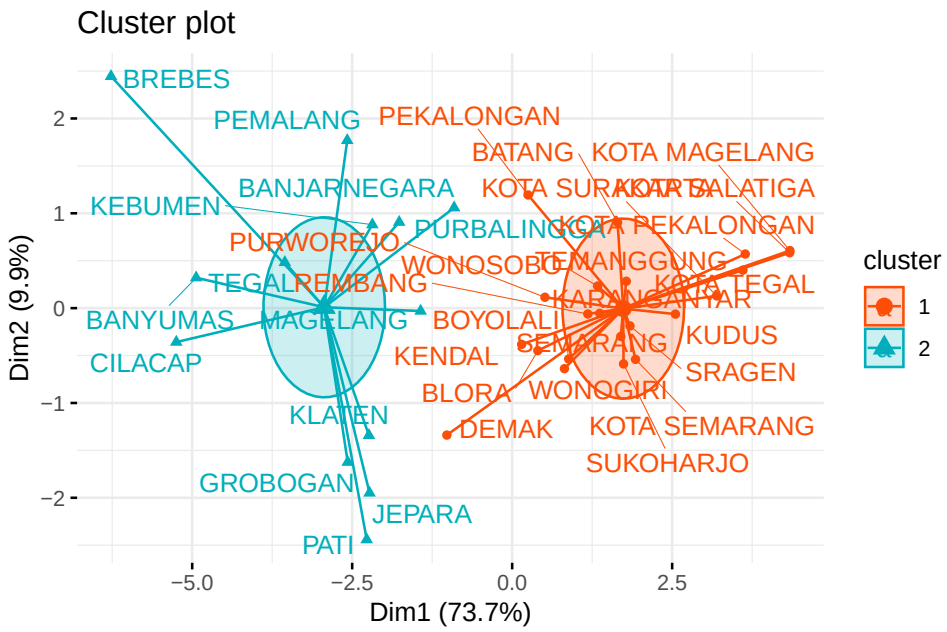

Figure 2.6: Plot Hasil Cluster

Chapter 3

Algoritma K-Medoids

K-Medoids adalah salah satu metode pengelompokan yang termasuk dalam kategori analisis cluster non-hierarki. Metode ini merupakan varian dari K-Means, yang dirancang untuk mengatasi kelemahan K-Means dalam menghadapi data yang mengandung outlier. Dalam K-Medoids, objek yang mewakili setiap cluster disebut medoid, yang merupakan titik pusat dari cluster tersebut. Berbeda dengan K-Means yang menggunakan rata-rata (centroid) sebagai pusat cluster, K-Medoids lebih robust terhadap outlier karena menggunakan objek yang sebenarnya dari dataset sebagai medoid (Jiawei and Kamber, 2006).

Proses kerja K-Medoids dimulai dengan menentukan jumlah cluster (k) yang diinginkan. Setelah itu, algoritma akan memilih secara acak medoid awal dari dataset. Selanjutnya, setiap objek dalam dataset akan dialokasikan ke cluster terdekat berdasarkan jarak ke medoid. Jarak yang umum digunakan dalam K-Medoids adalah jarak Euclidean. Proses ini diulang hingga tidak ada perubahan dalam pemilihan medoid, yang menunjukkan bahwa algoritma telah konvergen (Setyawati, 2017).

Salah satu keunggulan K-Medoids adalah kemampuannya untuk memberikan hasil yang lebih stabil dan representatif ketika data mengandung outlier. Hal ini menjadikan K-Medoids pilihan yang lebih baik dibandingkan K-Means dalam situasi di mana data tidak terdistribusi secara normal atau terdapat nilai ekstrim yang dapat mempengaruhi hasil clustering (Vercellis, 2009). Dengan demikian, K-Medoids sangat cocok untuk analisis data yang kompleks dan beragam.

Dalam implementasinya, K-Medoids dapat dilakukan dengan menggunakan berbagai perangkat lunak statistik, termasuk R. R menyediakan paket-paket yang memudahkan pengguna untuk melakukan analisis clustering, termasuk K-Medoids. Dengan menggunakan fungsi-fungsi yang tersedia, pengguna dapat dengan mudah melakukan standarisasi data, menentukan jumlah cluster optimal, dan memvisualisasikan hasil clustering (Santoso, 2012).

Secara keseluruhan, K-Medoids merupakan metode yang efektif untuk analisis cluster, terutama dalam konteks data yang mengandung outlier. Dengan pemahaman yang baik tentang cara kerja dan implementasinya, pengguna dapat memanfaatkan K-Medoids untuk mendapatkan wawasan yang lebih dalam dari data yang dianalisis.

3.1 Tahapan Algoritma K-Means

1. Inisialisasi Centroid

Tentukan jumlah cluster K yang diinginkan. Pilih secara acak K titik data dari dataset sebagai medoid awal untuk setiap cluster.

2. Penugasan Anggota Cluster

Setiap titik data x_i dalam dataset dialokasikan ke cluster yang memiliki medoid terdekat. Penugasan dilakukan berdasarkan jarak antara titik data dan medoid menggunakan rumus jarak (misalnya, jarak Euclidean).

$$\text{Jarak}(x_i, m_k) = \sqrt{\sum_{j=1}^{n}(x_{ij} - m_{kj})^2}$$

Di mana: x_i adalah titik data ke-i, m_k adalah medoid untuk cluster k, x_{ij} adalah nilai fitur ke-j dari titik data x_i, m_{kj} adalah nilai fitur ke-j dari medoid m_k, n adalah jumlah fitur.

Titik data x_i akan dimasukkan ke dalam cluster C_k yang memiliki medoid dengan jarak terkecil:

$$C_k = \{x_i \mid \text{Jarak}(x_i, m_k) \leq \text{Jarak}(x_i, m_j) \,\text{untuk semua}\, j \neq k\}$$

3. Pembaruan Medoid

Setelah anggota cluster ditugaskan, tentukan medoid baru untuk setiap cluster. Medoid baru adalah titik data yang meminimalkan jumlah jarak ke semua titik dalam cluster tersebut.

$$m_k = \arg\min_{x_j \in C_k} \sum_{x_i \in C_k} \text{Jarak}(x_i, x_j)$$

Di mana: m_k adalah medoid baru untuk cluster C_k, x_j adalah kandidat medoid di dalam cluster C_k, $\text{Jarak}(x_i, x_j)$ adalah jarak antara titik data x_i dan kandidat medoid x_j.

Medoid baru adalah titik data x_j yang meminimalkan jumlah total jarak ke titik-titik lainnya dalam cluster C_k.

4. Iterasi

Langkah 2 dan 3 diulang hingga tidak ada perubahan pada medoid atau hingga perubahan medoid sangat kecil, menandakan konvergensi.

5. Hasil Akhir

Setelah konvergensi tercapai, algoritma berhenti, dan hasil akhir adalah pembagian dataset ke dalam K cluster dengan medoid yang mewakili setiap cluster. Titik data yang tergabung dalam cluster C_k lebih dekat ke medoid m_k dibandingkan medoid lainnya.

3.2 Eksperimen Algoritma K-Medoids

Data

Dengan menggunakan fungsi `read.csv()` dari R, kami dapat mengimpor data langsung dari URL tersebut ke dalam lingkungan kerja R. Berikut adalah kode yang digunakan untuk membaca data: Package `reader` menyiapkan fungsi `read_csv()` untuk import data dari file CSV. Pada kasus ini digunakan data Data 40% Kemiskinan di jawa Tengah.

Table 3.1: Basis Data Terpadu Jawa Tengah

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
CILACAP	5.19	5.67	5.08	5.44	5.22	6.05	11.47	9.78	5.55	5.12
BANYUMAS	5.71	4.47	5.18	5.51	5.02	6.21	7.39	6.96	5.98	8.22
PURBALINGGA	3.30	2.19	3.80	3.13	3.73	3.34	8.71	7.41	3.21	4.65
BANJARNEGARA	2.73	2.34	3.76	2.80	2.57	2.99	3.31	5.45	4.21	6.05
KEBUMEN	4.17	2.55	3.26	4.16	3.15	4.15	4.30	9.29	4.61	4.34
PURWOREJO	1.87	2.12	1.48	3.05	1.78	1.83	5.00	4.90	3.12	2.09
WONOSOBO	2.13	1.95	3.00	1.78	1.62	2.06	0.45	2.32	3.57	0.84
MAGELANG	3.95	3.01	4.22	4.15	3.01	3.64	1.44	3.35	5.69	3.67
BOYOLALI	2.19	3.07	1.61	2.74	2.11	1.82	1.71	2.34	3.41	1.55
KLATEN	3.84	5.15	1.93	4.64	4.04	3.78	8.71	4.45	3.99	3.09

```
library (readr)
urlfile = "https://bit.ly/3VO3kRE"
data<-read.csv(url(urlfile), row.names = "Kabupaten")
```

R packages dan fungsi yang dibutuhkan

Untuk melakukan analisis cluster dengan metode k-medoids di R, diperlukan beberapa paket dan fungsi utama. Paket yang wajib digunakan adalah **`cluster`**, yang menyediakan fungsi `pam()` (Partitioning Around Medoids) sebagai implementasi utama algoritma k-medoids. Selain itu, paket **`factoextra`** berguna untuk visualisasi hasil clustering dengan fungsi `fviz_cluster()`. Untuk mengevaluasi kualitas clustering, paket **`fpc`** dan **`clusterCrit`** dapat digunakan sebagai opsi tambahan dengan fungsi seperti `cluster.stats()` untuk evaluasi statistik dan `intCriteria()` untuk menghitung indeks validasi clustering. Dalam proses analisis, fungsi `silhouette()` dapat digunakan untuk mengevaluasi kualitas clustering dengan Silhouette Width, sementara `dist()` diperlukan untuk membuat matriks jarak yang digunakan oleh algoritma. Sebagai langkah awal, data biasanya diskalakan menggunakan fungsi seperti `scale()` sebelum diterapkan algoritma k-medoids untuk memastikan hasil yang optimal. Kombinasi paket dan fungsi ini memungkinkan analisis cluster yang efektif, mulai dari pembentukan cluster hingga evaluasi hasil.

```
library(cluster)
library(factoextra)
```

Estimasi Jumlah *Cluster* Optimal

Untuk menentukan jumlah cluster yang optimal, metode average silhouette dapat digunakan. Konsepnya adalah menjalankan algoritma PAM dengan berbagai jumlah cluster k. Kemudian, rata-rata silhouette dari masing-masing cluster dihitung dan diplot berdasarkan jumlah cluster. Average silhouette digunakan untuk menilai kualitas clustering, di mana nilai yang tinggi menunjukkan clustering yang baik. Jumlah cluster k yang optimal adalah yang menghasilkan rata-rata silhouette tertinggi dalam rentang nilai k yang dipertimbangkan.

Fungsi `fviz_nbclust()` dari paket `factoextra` menyediakan cara praktis untuk memperkirakan jumlah cluster yang optimal. Dengan memanfaatkan metode average silhouette, fungsi ini dapat mem-

bantu menentukan jumlah cluster yang menghasilkan hasil clustering terbaik. Contoh implementasinya adalah sebagai berikut:

```
fviz_nbclust(data, pam, method = "silhouette") +
  theme_classic()
```

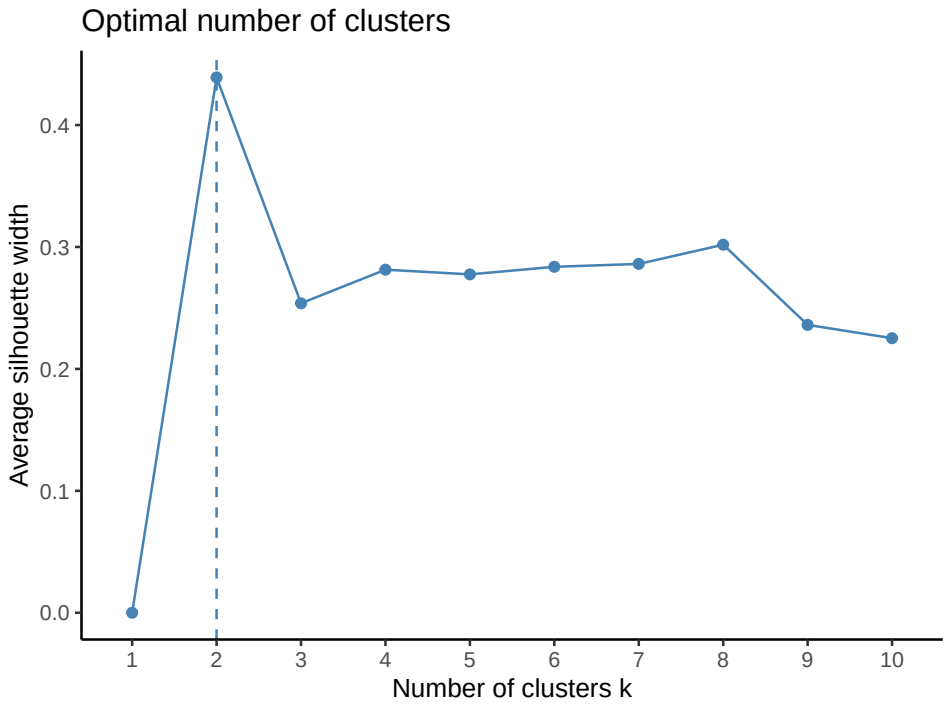

Figure 3.1: Plot Jumlah Cluster Metode silhouette

Hasilnya adalah grafik yang menampilkan nilai rata-rata silhouette untuk setiap jumlah cluster. Jumlah cluster optimal ditentukan oleh nilai silhouette tertinggi, yang menunjukkan pembagian cluster terbaik dalam data.Berdasarkan plot, jumlah cluster yang disarankan adalah 2. Pada langkah berikutnya, kita akan mengelompokkan observasi ke dalam 2 cluster.

Eksperimen K-Medoids Clustering

Berdasarkan plot hasil analisis dengan metode average silhouette, jumlah cluster optimal yang disarankan adalah 2. Oleh karena itu, dalam analisis cluster menggunakan algoritma k-medoids, kita akan menggunakan jumlah cluster $k = 2$. Langkah ini diharapkan dapat menghasilkan pembagian cluster yang optimal dan berkualitas tinggi sesuai dengan data yang dianalisis. Berikut adalah kode untuk menjalankan algoritma k-medoids dengan $k = 2$ menggunakan fungsi pam() dan mencetak hasilnya:

```
## Menjalankan k-medoids clustering dengan k = 2
pam.res <- pam(data, 2)
```

Kode ini akan menampilkan detail hasil clustering, termasuk informasi tentang medoid, anggota cluster, dan ukuran setiap cluster.

```
# Mencetak hasil clustering
print(pam.res)
#> Medoids:
#>          ID   X1   X2   X3   X4   X5   X6   X7   X8   X9  X10
#> TEGAL    28 4.78 3.91 6.70 4.84 6.35 5.82 3.00 5.99 2.98 5.98
#> SEMARANG 22 1.76 2.19 1.61 2.33 1.88 1.61 0.92 0.49 2.67 1.56
#> Clustering vector:
#>         CILACAP        BANYUMAS     PURBALINGGA    BANJARNEGARA         KEBUMEN
#>               1               1               1               1               1
#>       PURWOREJO        WONOSOBO        MAGELANG        BOYOLALI         KLATEN
#>               2               2               2               2               1
#>       SUKOHARJO        WONOGIRI     KARANGANYAR         SRAGEN        GROBOGAN
#>               2               2               2               2               1
#>          BLORA         REMBANG            PATI           KUDUS         JEPARA
#>               2               2               2               2               1
#>         DEMAK          SEMARANG      TEMANGGUNG         KENDAL         BATANG
#>               2               2               2               2               2
#>      PEKALONGAN        PEMALANG           TEGAL         BREBES   KOTA MAGELANG
#>               2               1               1               1               2
#>  KOTA SURAKARTA   KOTA SALATIGA   KOTA SEMARANG KOTA PEKALONGAN      KOTA TEGAL
#>               2               2               2               2               2
#> Objective function:
#>    build     swap
#> 4.702218 4.605625
#>
#> Available components:
#>  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
#>  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

Hasil analisis k-medoids dengan $k = 2$ menunjukkan bahwa data berhasil dikelompokkan ke dalam dua cluster. Medoid yang terpilih sebagai representasi masing-masing cluster adalah TEGAL untuk Cluster 1 dan SEMARANG untuk Cluster 2, dengan nilai atribut masing-masing ditampilkan dalam tabel. Observasi lainnya dikelompokkan berdasarkan kedekatan mereka dengan medoid, misalnya CILACAP, BANYUMAS, dan PURBALINGGA masuk ke Cluster 1, sementara PURWOREJO, WONOSOBO, dan MAGELANG masuk ke Cluster 2. Secara keseluruhan, clustering ini memberikan pembagian yang optimal dengan dua cluster yang masing-masing diwakili oleh medoid terpilih.

Untuk menambahkan klasifikasi titik (cluster) ke data asli, Anda dapat menggunakan kode berikut. Kode ini akan menggabungkan hasil klasifikasi (cluster) dari analisis k-medoids ke dalam `data` dan menampilkan 3 baris pertama dari data yang telah diperbarui:

```
# Menambahkan klasifikasi cluster ke data asli
hasil_cluster <- cbind(data, cluster = pam.res$cluster)

# Menampilkan 3 baris pertama
head(hasil_cluster, n = 3)
#>               X1   X2   X3   X4   X5   X6    X7   X8   X9  X10 cluster
#> CILACAP     5.19 5.67 5.08 5.44 5.22 6.05 11.47 9.78 5.55 5.12       1
#> BANYUMAS    5.71 4.47 5.18 5.51 5.02 6.21  7.39 6.96 5.98 8.22       1
#> PURBALINGGA 3.30 2.19 3.80 3.13 3.73 3.34  8.71 7.41 3.21 4.65       1
```

Hasil dari kode ini akan menampilkan tiga baris pertama dari `data`, dengan kolom tambahan yang

menunjukkan cluster tempat setiap observasi dikelompokkan. Kolom baru cluster berisi angka 1 atau 2 yang mewakili dua cluster yang terbentuk.

Visualisasi K-Medoids Clustering

Untuk memvisualisasikan hasil pengelompokan dengan k-medoids, kita akan menggunakan fungsi **`fviz_cluster()`** dari paket **`factoextra`**. Fungsi ini akan menghasilkan plot sebar titik data yang diwarnai sesuai dengan nomor cluster. Dalam kode ini, palet warna yang digunakan adalah `#00AFBB` untuk cluster pertama dan `#FC4E07` untuk cluster kedua. Elips konsentrasi ditambahkan dengan parameter ellipse.type = "t", yang menunjukkan distribusi data di setiap cluster. Selain itu, untuk menghindari tumpang tindih label, parameter repel = TRUE digunakan, meskipun ini dapat sedikit memperlambat proses visualisasi. Terakhir, tema plot yang klasik diterapkan dengan ggtheme = theme_classic(), memberikan tampilan yang bersih dan sederhana pada grafik.

```
fviz_cluster(pam.res,
  palette = c("#00AFBB", "#FC4E07"), # Palet warna
  ellipse.type = "t", # Elips konsentrasi
  repel = TRUE, # Menghindari label bertumpuk (proses ini dapat sedikit lebih lambat)
  ggtheme = theme_classic() # Tema plot klasik
)
```

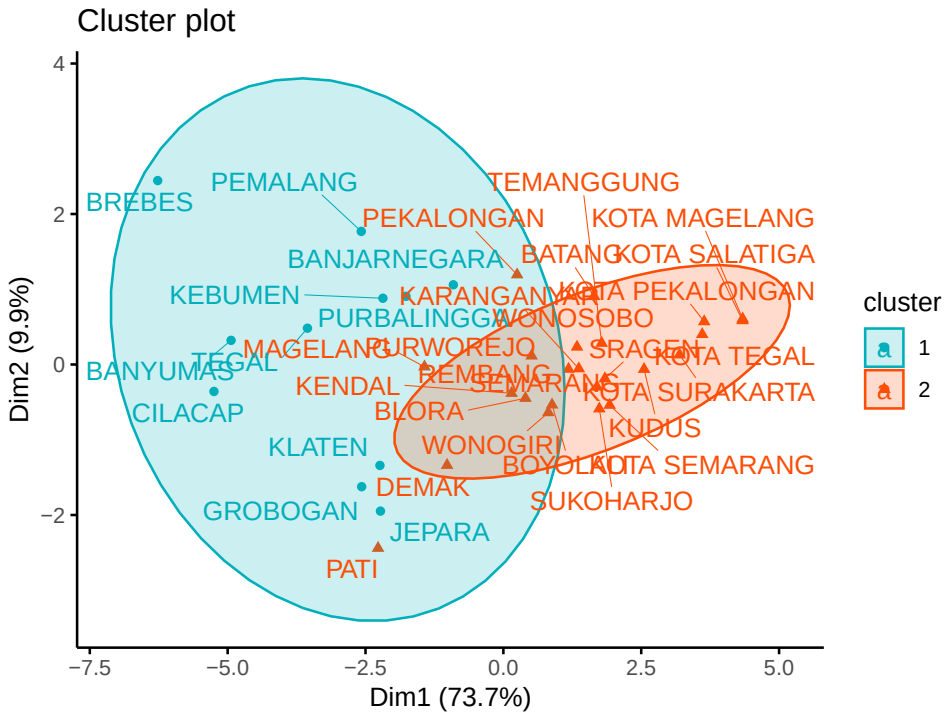

Figure 3.2: Visualisasi Cluster K-Medoids

Chapter 4

CLARA - Clustering Large Applications

Analisis clustering adalah teknik statistik yang digunakan untuk mengelompokkan objek-objek ke dalam kelompok atau kluster berdasarkan kesamaan karakteristik. Tujuan utama dari clustering adalah untuk memaksimalkan kesamaan dalam satu kluster dan meminimalkan kesamaan antar kluster. Dalam konteks data besar, metode clustering tradisional sering kali tidak efisien, sehingga diperlukan algoritma yang lebih canggih seperti CLARA (Clustering Large Applications). CLARA dirancang untuk menangani dataset besar dengan cara yang lebih efisien dibandingkan dengan algoritma clustering lainnya seperti K-Means atau K-Medoids (Kaufman and Rousseeuw, 1990b).

CLARA mengadopsi pendekatan sampling untuk mengatasi masalah komputasi yang muncul saat bekerja dengan dataset besar. Algoritma ini pertama-tama mengambil sampel dari dataset dan kemudian menerapkan algoritma K-Medoids pada sampel tersebut untuk menemukan medoid. Setelah itu, CLARA menghitung jarak antara setiap objek dalam dataset asli dengan medoid yang ditemukan, dan mengelompokkan objek berdasarkan kedekatannya dengan medoid tersebut. Proses ini diulang beberapa kali untuk meningkatkan akurasi hasil clustering (Kaufman and Rousseeuw, 1990b).

Salah satu keunggulan utama dari CLARA adalah kemampuannya untuk mengurangi waktu komputasi yang diperlukan untuk clustering dataset besar. Dengan menggunakan teknik sampling, CLARA dapat memberikan hasil yang representatif tanpa harus memproses seluruh dataset secara langsung. Hal ini sangat berguna dalam aplikasi dunia nyata di mana data sering kali sangat besar dan kompleks, seperti dalam analisis data pelanggan, pengolahan citra, dan bioinformatika (Halkidi et al., 2001).

CLARA telah diterapkan dalam berbagai bidang, termasuk pemasaran, analisis sosial, dan ilmu kesehatan. Misalnya, dalam pemasaran, CLARA dapat digunakan untuk mengelompokkan pelanggan berdasarkan perilaku pembelian mereka, sehingga perusahaan dapat menyesuaikan strategi pemasaran mereka dengan lebih efektif. Di bidang kesehatan, CLARA dapat membantu dalam pengelompokan pasien berdasarkan gejala atau respons terhadap pengobatan, yang dapat meningkatkan perawatan pasien

Dengan meningkatnya volume data yang dihasilkan setiap hari, metode clustering yang efisien seperti CLARA menjadi semakin penting. Algoritma ini tidak hanya menawarkan solusi untuk masalah komputasi yang dihadapi oleh metode clustering tradisional, tetapi juga memberikan hasil yang dapat diandalkan dalam konteks data besar. Di masa depan, pengembangan lebih lanjut dari algoritma ini dan integrasinya dengan teknik pembelajaran mesin lainnya dapat membuka jalan bagi analisis data yang lebih mendalam dan akurat (Halkidi et al., 2001).

4.1 Tahapan Algoritma CLARA

Algoritma CLARA (*Clustering Large Applications*) adalah metode clustering berbasis partisi yang efisien untuk dataset besar. Berikut adalah tahapan-tahapannya:

1. Inisialisasi

Tentukan jumlah cluster k. Tentukan jumlah sampel (*subsets*) yang akan diambil (s).

2. Pengambilan Sampel Subset Data

CLARA mengambil s subset data secara acak, di mana ukuran subset data adalah m. Nilai m biasanya cukup besar untuk mencakup representasi seluruh dataset.

3. Penerapan Algoritma PAM pada Subset Data

Untuk setiap subset, algoritma PAM (*Partitioning Around Medoids*), Pilih k medoid awal secara acak. Tetapkan setiap data x_i ke medoid terdekat berdasarkan fungsi jarak $d(x_i, m_j)$, di mana:

$$d(x_i, m_j) = \sqrt{\sum_{p=1}^{P} (x_{ip} - m_{jp})^2}$$

dengan P adalah jumlah atribut. Hitung *total cost* untuk setiap medoid:

$$\text{Cost} = \sum_{i=1}^{n} \min_{j=1}^{k} d(x_i, m_j)$$

Tukar medoid dengan non-medoid secara iteratif untuk mengurangi *total cost* hingga konvergen.

4. Pemilihan Medoid Terbaik

Evaluasi semua medoid yang diperoleh dari s subset menggunakan seluruh dataset. Medoid terbaik adalah yang memiliki nilai *total cost* terkecil.

5. Penugasan Data ke Cluster

Tetapkan seluruh data ke medoid terdekat yang dipilih pada langkah sebelumnya.

6. Evaluasi Kualitas Cluster

Gunakan metrik evaluasi seperti *silhouette coefficient*:

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

di mana: $a(i)$ adalah rata-rata jarak data i ke data lain dalam cluster yang sama. $b(i)$ adalah rata-rata jarak data i ke data dalam cluster terdekat.

4.2 Eksperimen Algoritma CLARA

Dalam eksperimen ini, kita akan menerapkan algoritma CLARA pada dataset yang terdiri dari dua variabel acak. Kita akan membangkitkan 500 sampel dari dua distribusi normal yang berbeda.

1. Memuat library yang diperlukan

Library pertama adalah `cluster`, yang digunakan untuk melakukan analisis clustering dengan berbagai algoritma, seperti PAM (*Partitioning Around Medoids*) dan CLARA (*Clustering Large Applications*). Dalam kasus ini, CLARA digunakan untuk mengelompokkan data besar secara efisien. Library kedua adalah `ggplot2`, yang digunakan untuk membuat visualisasi data secara estetis dan informatif. Dengan pendekatan *layered grammar of graphics*, `ggplot2` memungkinkan pembuatan grafik kompleks, seperti visualisasi hasil clustering berdasarkan dua variabel dalam dataset.

```
library(cluster)
library(ggplot2)
library(factoextra)
```

2. Membangkitkan data acak

Data acak dibangkitkan menggunakan fungsi `rnorm()` dalam bahasa pemrograman `R`. Fungsi `rnorm()` digunakan untuk menghasilkan angka acak yang terdistribusi normal, dengan parameter pertama menunjukkan jumlah data yang ingin dibangkitkan, parameter kedua adalah rata-rata (mean), dan parameter ketiga adalah simpangan baku (standard deviation). Pada bagian pertama, dua set data acak dengan jumlah 200 titik data untuk setiap variabel `x` dan `y` dihasilkan, yang masing-masing memiliki distribusi normal dengan rata-rata 0 dan simpangan baku 8. Data ini kemudian digabungkan menggunakan fungsi `rbind()`. Pada bagian kedua, dua set data tambahan dengan jumlah 300 titik data masing-masing untuk variabel `x` dan `y` dihasilkan, dengan rata-rata 50 dan simpangan baku 8. Akhirnya, nama kolom untuk data acak tersebut ditentukan dengan fungsi `colnames()`, yaitu `x` untuk variabel pertama dan `y` untuk variabel kedua. Hasilnya adalah sebuah matriks data dua kolom yang berisi dua grup data acak dengan distribusi normal berbeda.

```
set.seed(1234)
data <- rbind(
  cbind(rnorm(200, 0, 8), rnorm(200, 0, 8)),
  cbind(rnorm(300, 50, 8), rnorm(300, 50, 8)))
colnames(data) <- c("x", "y")
```

3. Visualisasi data acak

Visualisasi data acak di atas dilakukan menggunakan paket `ggplot2` di R. Fungsi `ggplot()` digunakan untuk memulai pembuatan plot, dengan parameter `data` berisi data yang ingin divisualisasikan, dan `aes()` untuk menentukan hubungan antara variabel-variabel yang akan diplot, yaitu `x` dan `y`. Selanjutnya, fungsi `geom_point()` digunakan untuk membuat grafik titik (scatter plot), dengan argumen `alpha = 0.5` untuk memberikan tingkat transparansi pada titik-titik, sehingga memungkinkan visualisasi yang lebih jelas ketika titik-titik saling tumpang tindih. Judul plot diatur menggunakan `labs()` dengan parameter `title`, `x`, dan `y` untuk memberikan label pada plot dan sumbu X serta Y. Terakhir, `theme_minimal()` diterapkan untuk memberikan tampilan plot yang bersih dan sederhana. Hasil dari

kode ini adalah visualisasi data acak dalam bentuk scatter plot dengan dua variabel, menunjukkan dua kelompok data dengan distribusi normal yang berbeda.

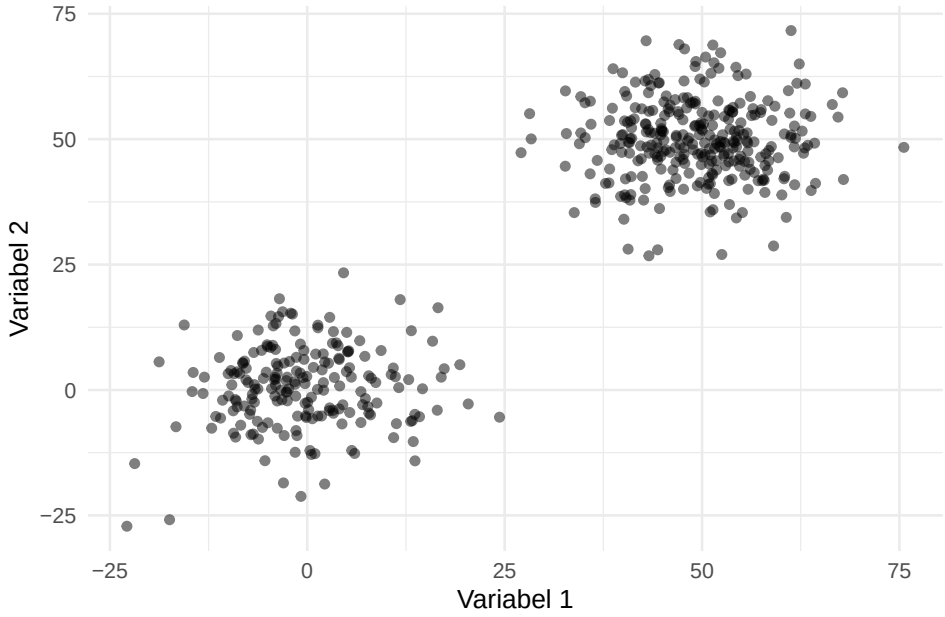

Figure 4.1: Visualisasi Data

4. Mencari Jumlah CLluster Optimal

Visualisasi jumlah klaster yang optimal dilakukan menggunakan metode silhouette dengan paket `cluster` dan `factoextra` di R. Fungsi `fviz_nbclust()` digunakan untuk menilai jumlah klaster terbaik berdasarkan metode yang dipilih, dalam hal ini menggunakan algoritma `clara` (Clustering Large Applications). Metode silhouette mengukur seberapa baik setiap objek dipisahkan dari klaster lain, dengan nilai yang lebih tinggi menunjukkan pemisahan yang lebih baik. Parameter `method = "silhouette"` mengindikasikan bahwa visualisasi menggunakan metode tersebut. Hasilnya akan memperlihatkan grafik yang menggambarkan nilai rata-rata silhouette untuk berbagai jumlah klaster. Terakhir, fungsi `theme_classic()` diterapkan untuk memberikan tampilan plot yang bersih dan sederhana. Grafik ini dapat membantu dalam menentukan jumlah klaster yang paling sesuai dengan data berdasarkan evaluasi kualitas pemisahan antar klaster.

```
fviz_nbclust(data, clara, method = "silhouette")+
theme_classic()
```

5. Menjalankan CLARA

Algoritma CLARA (Clustering Large Applications) digunakan untuk melakukan klasterisasi data dengan jumlah klaster yang ditentukan sebanyak 2, yang ditetapkan melalui parameter `k = 2`. Fungsi `clara()` dari paket `cluster` di R dirancang untuk melakukan klasterisasi pada dataset besar dengan cara mengacak sampel data untuk mempercepat proses klasterisasi. Parameter `samples = 5` menunjukkan bahwa lima sampel data acak akan diambil untuk setiap iterasi klasterisasi. Hasil dari klasterisasi ini disimpan dalam objek `clara_result`, yang berisi informasi tentang klaster yang terbentuk, termasuk elemen-elemen data yang termasuk dalam setiap klaster dan pusat klaster.

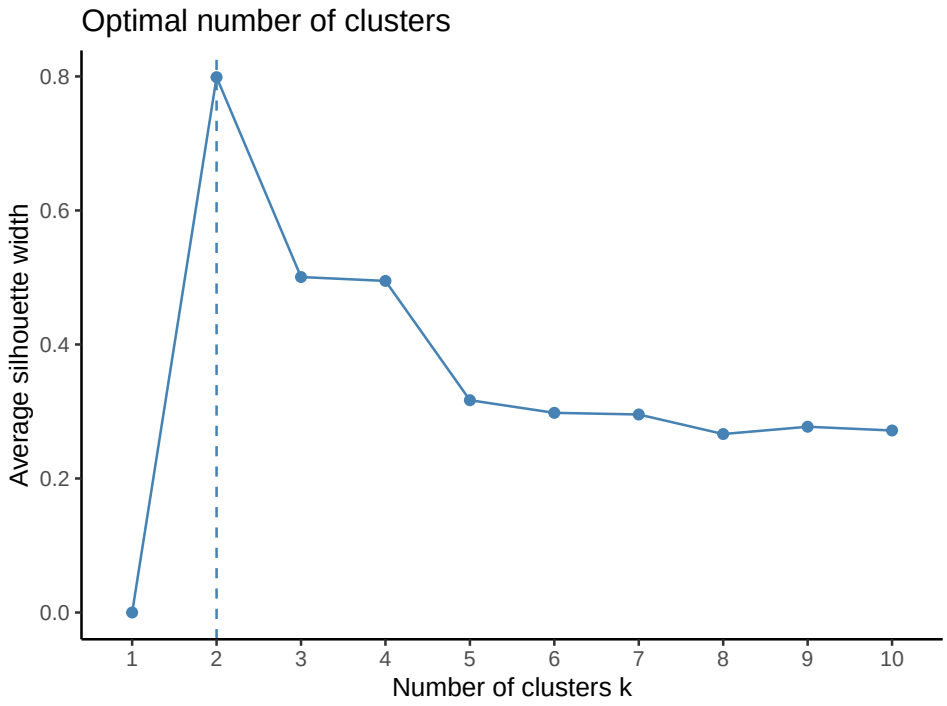

Figure 4.2: Jumlah Cluster Optimal

```
clara_result <- clara(data, k = 2, samples = 5)
```

6. Evaluasi hasil

`print(clara_result)` digunakan untuk menampilkan hasil dari proses klasterisasi yang dilakukan dengan algoritma CLARA. Setelah data dikelompokkan menjadi dua klaster menggunakan parameter `k = 2`, hasil ini memberikan informasi tentang bagaimana data dibagi. Secara spesifik, hasil tersebut akan menunjukkan anggota klaster, yaitu data mana yang tergabung dalam setiap klaster, serta pusat klaster, yang menggambarkan titik tengah dari masing-masing klaster. Selain itu, hasil ini juga dapat mencakup ukuran atau indeks kualitas klaster, yang digunakan untuk mengevaluasi sejauh mana klaster-klaster tersebut terpisah dengan baik satu sama lain. Dengan demikian, fungsi `print(clara_result)` memberikan gambaran yang jelas tentang hasil klasterisasi dan seberapa baik data dapat dikelompokkan dalam dua klaster.

```
print(clara_result)
#> Call:     clara(x = data, k = 2, samples = 5)
#> Medoids:
#>                x         y
#> [1,] -0.5495492  2.458514
#> [2,] 47.5964047 50.892735
#> Objective function:   9.9971
#> Clustering vector:    int [1:500] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ...
#> Cluster sizes:            200 300
#> Best sample:
#>  [1]   6  45  51  67  75  85  90  94  97 110 111 160 168 170 176 181 201 219 249
```

```
#> [20] 260 264 275 296 304 317 319 337 361 362 369 370 374 379 397 398 411 420 422
#> [39] 424 436 448 458 465 489
#>
#> Available components:
#>  [1] "sample"     "medoids"    "i.med"      "clustering" "objective"
#>  [6] "clusinfo"   "diss"       "call"       "silinfo"    "data"
```

7. Visualisasi Hasil Cluster

Visualisasi silhouette digunakan untuk mengevaluasi kualitas klaster yang dihasilkan oleh algoritma CLARA. Fungsi `silhouette()` menghitung nilai silhouette untuk setiap titik data berdasarkan klaster yang telah ditetapkan (`clara_result$clustering`) dan jarak antar data (`dist(data)`). Nilai silhouette mengukur sejauh mana setiap titik data berada dalam klaster yang benar, dengan nilai mendekati +1 menunjukkan bahwa titik data tersebut sangat cocok dengan klasternya, sedangkan nilai mendekati -1 menunjukkan bahwa titik tersebut lebih cocok dengan klaster lain.

Selanjutnya, fungsi `plot(sil, border = NA)` digunakan untuk menghasilkan plot visualisasi dari hasil silhouette. Parameter `border = NA` menghilangkan batas pada setiap segmen plot untuk memberikan tampilan yang lebih bersih. Hasil visualisasi ini memberikan gambaran tentang kualitas pemisahan antar klaster, dengan tinggi nilai silhouette menunjukkan klasterisasi yang baik.

```
sil <- silhouette(clara_result$clustering, dist(data))
plot(sil, border = NA)
```

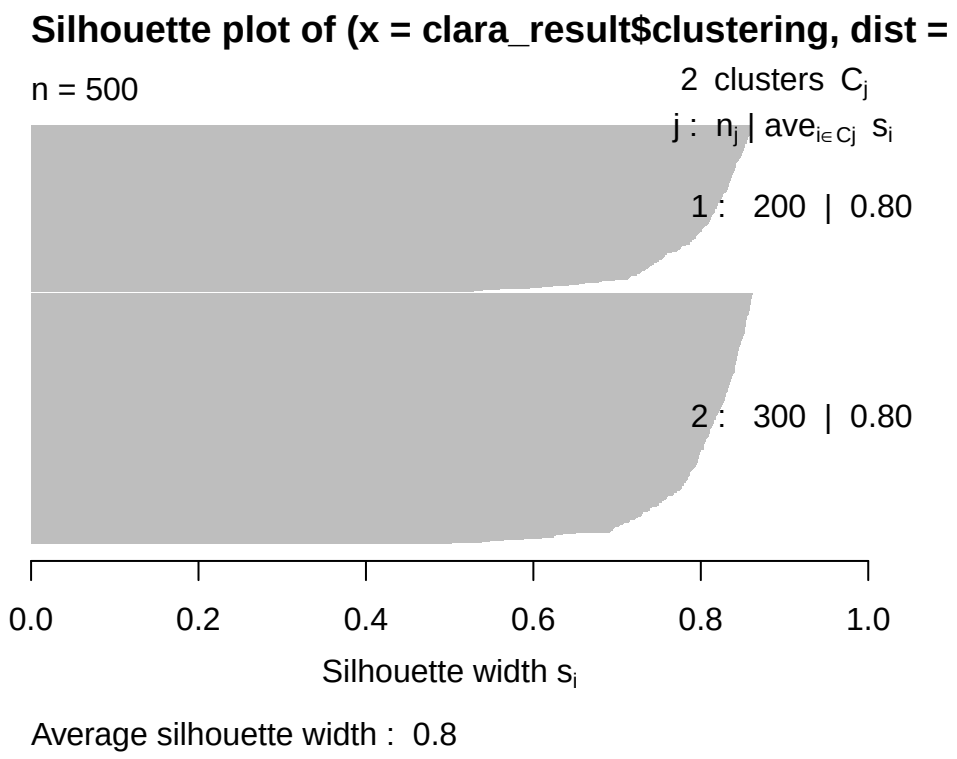

Figure 4.3: Visualisasi silhouette

Kode ini menggunakan fungsi `fviz_cluster()` untuk memvisualisasikan hasil klasterisasi yang dilakukan dengan algoritma CLARA. Visualisasi ini menampilkan titik-titik data yang dikelompokkan berdasarkan klaster yang telah ditentukan, dengan dua klaster yang masing-masing diberi warna berbeda menggunakan palet warna `#00AFBB` dan `#FC4E07`. Elips konsentrasi (`ellipse.type = "t"`) ditambahkan di sekitar setiap klaster, yang menunjukkan area distribusi data dalam klaster tersebut, memberikan gambaran tentang seberapa padat atau tersebarnya data. Dengan parameter `geom = "point"`, visualisasi ini menampilkan titik-titik data, dan ukuran titik diatur menggunakan `pointsize = 1`. Tema klasik (`theme_classic()`) diterapkan untuk memberikan tampilan plot yang bersih dan sederhana. Hasil dari kode ini adalah visualisasi yang jelas tentang pembagian data ke dalam dua klaster dengan warna yang berbeda, serta gambaran tentang penyebaran data di dalam setiap klaster.

```
fviz_cluster(clara_result,
palette = c("#00AFBB", "#FC4E07"), # color palette
ellipse.type = "t", # Concentration ellipse
geom = "point", pointsize = 1,
ggtheme = theme_classic()
)
```

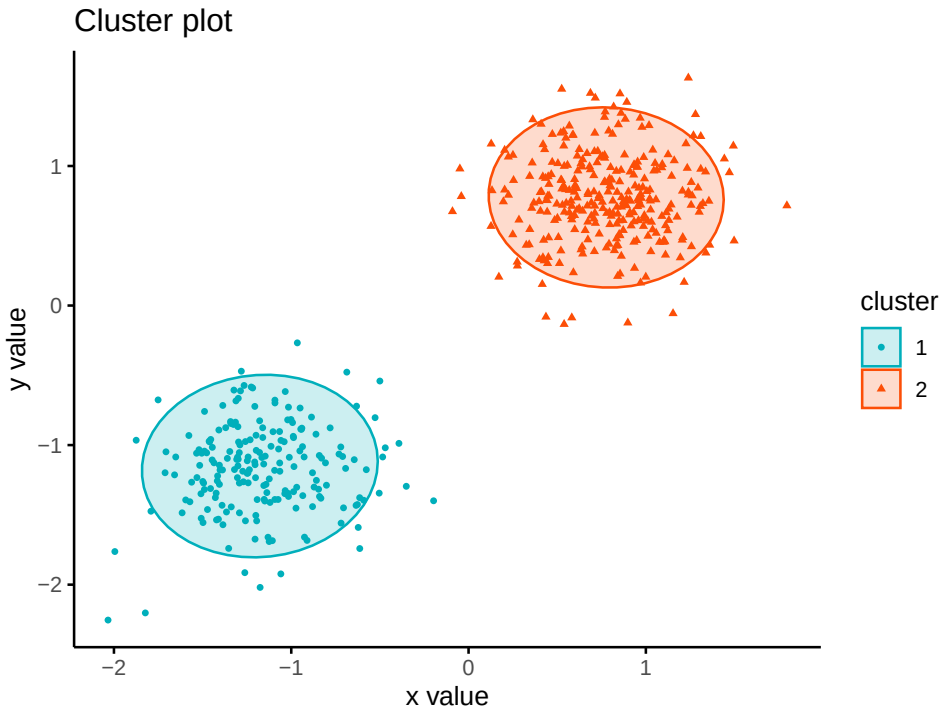

Figure 4.4: Visualisasi Hasil Cluster

Chapter 5

Algoritma K-Modes

Algoritma K-Modes merupakan salah satu metode dalam analisis clustering yang dirancang khusus untuk mengelompokkan data kategorik. Berbeda dengan algoritma K-Means yang hanya dapat digunakan untuk data numerik, K-Modes mengatasi keterbatasan ini dengan menggunakan modus sebagai pusat cluster. Hal ini memungkinkan K-Modes untuk menangani data yang memiliki atribut kategorik, seperti jenis kelamin, status perkawinan, atau kategori produk, yang sering dijumpai dalam berbagai aplikasi analisis data (Handayani, 2020).

Proses kerja algoritma K-Modes mirip dengan K-Means, di mana algoritma ini juga memerlukan penentuan jumlah cluster (K) sebelum proses clustering dimulai. K-Modes melakukan inisialisasi centroid dengan memilih modus dari setiap atribut dalam cluster. Selanjutnya, algoritma ini menghitung jarak antara data dan centroid menggunakan metode yang disebut "dissimilarity measure," yang mengukur seberapa berbeda dua data berdasarkan atribut kategorik (Buulolo, 2020).

Salah satu keunggulan K-Modes adalah kemampuannya untuk mengatasi data yang mengandung noise atau outlier. Dalam konteks data kategorik, outlier dapat mempengaruhi hasil clustering secara signifikan. K-Modes menggunakan pendekatan yang lebih robust dengan memfokuskan pada modus, sehingga hasil clustering menjadi lebih stabil dan representatif (Hardandy et al., 2017).

Implementasi K-Modes dalam R dapat dilakukan dengan menggunakan paket seperti klaR dan cluster, yang menyediakan fungsi-fungsi yang diperlukan untuk melakukan clustering. Dengan menggunakan dataset yang sesuai, pengguna dapat dengan mudah menerapkan algoritma ini untuk mendapatkan insight yang berharga dari data kategorik yang dimiliki (Arum, 2024).

Dalam bab ini, kita akan membahas lebih dalam mengenai langkah-langkah implementasi algoritma K-Modes, termasuk pemilihan jumlah cluster yang optimal, serta analisis hasil clustering yang diperoleh. Dengan pemahaman yang baik tentang algoritma ini, diharapkan pembaca dapat menerapkan K-Modes dalam berbagai konteks analisis data yang relevan.

5.1 Tahapan Algoritma K-Modes

1. Inisialisasi

Tentukan jumlah cluster k. Pilih k modus dari dataset sebagai centroid awal. Modus adalah nilai yang paling sering muncul dalam setiap atribut kategorik.

2. Pengelompokan

Untuk setiap data point, hitung jarak dissimilarity ke setiap centroid. Dalam K-Modes, jarak dissimilarity dihitung menggunakan rumus berikut:

$$d(x_i, c_j) = \sum_{k=1}^{n} \delta(x_{ik}, c_{jk})$$

di mana: $d(x_i, c_j)$ adalah jarak dissimilarity antara data point x_i dan centroid c_j. $\delta(x_{ik}, c_{jk})$ adalah fungsi yang mengembalikan 0 jika $x_{ik} = c_{jk}$ (nilai sama) dan 1 jika $x_{ik} \neq c_{jk}$ (nilai berbeda). n adalah jumlah atribut.

3. Penugasan Cluster

Setiap data point x_i ditugaskan ke cluster dengan centroid terdekat (dengan jarak dissimilarity terkecil).

4. Update Centroid

Setelah semua data point ditugaskan ke cluster, hitung ulang centroid untuk setiap cluster dengan mengambil modus dari setiap atribut dalam cluster tersebut. Modus dapat dihitung dengan:

$$c_j = \text{modus}(X_j)$$

di mana X_j adalah himpunan data point yang ditugaskan ke cluster j.

5. Kondisi Berhenti

Ulangi langkah 2 hingga 4 sampai tidak ada perubahan dalam penugasan cluster atau sampai jumlah iterasi maksimum tercapai.

Penjelasan Tambahan

- **Centroid**: Dalam konteks K-Modes, centroid adalah modus dari atribut-atribut dalam cluster. Ini berbeda dengan K-Means, di mana centroid adalah rata-rata dari atribut numerik.
- **Iterasi**: Proses ini diulang hingga tidak ada perubahan dalam penugasan cluster, yang menunjukkan bahwa algoritma telah konvergen.

5.2 Eksperimen Algoritma K-Modes

1. Memuat library yang diperlukan

Algoritma k-modes adalah salah satu metode clustering yang dirancang khusus untuk menangani data kategorikal. Di R, algoritma ini dapat diimplementasikan dengan menggunakan library Klar, yang menyediakan fungsi bawaan untuk k-modes. Selain itu, library MASS sering digunakan untuk mendukung manipulasi data atau keperluan lainnya dalam analisis clustering.

Table 5.1: Data Daerah Aliran Sungai

PRODUKTIVITAS	KEMIRINGAN LERENG	TINGKAT EROSI	MANAJEMEN LAHAN
Sangat Tinggi	Datar	Berat	Baik
Sangat Tinggi	Datar	Sangat Berat	Baik
Sangat Rendah	Datar	Sedang	Buruk
Sangat Rendah	Datar	Sedang	Buruk
Sangat Tinggi	Datar	Sedang	Baik
Sangat Tinggi	Datar	Sedang	Baik
Sangat Tinggi	Datar	Sedang	Baik
Sangat Rendah	Datar	Sedang	Buruk
Sangat Tinggi	Datar	Sedang	Baik
Sangat Tinggi	Datar	Sedang	Baik

```
library(klaR)
library(MASS)
```

2. Persiapan Dataset

Package **readr** menyiapkan fungsi `read_csv()` untuk import data dari file CSV. Pada kasus ini digunakan data Daerah Aliran Sungai (DAS).

```
library (readr)
urlfile = "https://raw.githubusercontent.com/dedenistiawan/Dataset/main/Dataset%20DAS.csv"
data<-read_csv(url(urlfile))
```

3. Memeriksa Missing value

Untuk memeriksa keberadaan **missing value** dalam dataset, kita dapat menggunakan fungsi `is.na()` yang mengidentifikasi elemen dengan nilai yang hilang (NA). Dalam kasus ini, perintah `colSums(is.na(data))` digunakan untuk menghitung jumlah nilai yang hilang pada setiap kolom dalam dataset. Fungsi `is.na(data)` menghasilkan matriks logika yang menunjukkan posisi nilai yang hilang, di mana nilai `TRUE` menandakan keberadaan missing value. Selanjutnya, `colSums()` menjumlahkan nilai `TRUE` tersebut untuk setiap kolom, sehingga memberikan informasi tentang jumlah nilai yang hilang per kolom. Dengan hasil ini, pengguna dapat dengan mudah mengidentifikasi kolom yang memerlukan penanganan lebih lanjut, seperti imputasi atau penghapusan data.

```
colSums(is.na(data))
#>     PRODUKTIVITAS KEMIRINGAN LERENG     TINGKAT EROSI   MANAJEMEN LAHAN
#>                 0                 0                 0                 0
```

3. Menjalankan K-Modes

Proses diawali dengan menjalankan fungsi `kmodes()`, di mana dataset dibagi ke dalam 3 cluster dengan parameter iterasi maksimum sebesar 10 dan tanpa pembobotan (parameter `weighted = FALSE`). Hasil

clustering disimpan dalam variabel **cluster.results**, yang mencakup informasi tentang mode dari setiap cluster dan label cluster untuk setiap baris data. Selanjutnya, hasil cluster tersebut ditambahkan sebagai kolom baru ke dataset asli menggunakan fungsi **cbind()**, menghasilkan data frame baru yang mencakup informasi label cluster untuk setiap data. Dengan demikian, dataset hasil clustering ini dapat digunakan untuk analisis lebih lanjut, seperti mengidentifikasi pola atau karakteristik di dalam setiap cluster.

```
kmodes_result <- kmodes(data , 3, iter.max = 10, weighted = FALSE )
cluster.output <- cbind(data ,kmodes_result$cluster)
```

4. Menyimpan Hasil Cluster

Fungsi **write.csv()** digunakan untuk mengekspor data frame **cluster.output**, yang berisi dataset asli beserta label cluster hasil algoritma k-modes, ke file bernama *"kmodes clusters.csv"*. Parameter **row.names = TRUE** memastikan bahwa nama baris dari data frame akan disertakan dalam file CSV yang dihasilkan. File ini disimpan di direktori kerja R saat ini dan dapat diakses menggunakan perangkat lunak spreadsheet seperti Microsoft Excel atau diimpor kembali ke R untuk analisis lebih lanjut.

```
write.csv(cluster.output, file = "kmodes clusters.csv", row.names = TRUE)
```

Chapter 6

Metode Cluster Hirarki

Hierarchical clustering adalah salah satu metode analisis klaster yang digunakan untuk mengelompokkan data berdasarkan kemiripan atau jarak antar objek. Berbeda dengan metode klaster lainnya seperti K-Means, hierarchical clustering tidak memerlukan jumlah klaster yang telah ditentukan sebelumnya. Prosesnya dimulai dengan setiap objek dianggap sebagai klaster tersendiri, yang kemudian digabungkan secara bertahap hingga membentuk satu klaster besar atau, sebaliknya, dengan memisahkan satu klaster besar menjadi klaster-klaster kecil. Teknik ini memberikan fleksibilitas dalam mengeksplorasi struktur data tanpa asumsi awal yang ketat (Everitt et al., 2011) .

Hierarchical clustering terdiri dari dua pendekatan utama, yaitu agglomerative dan divisive. Pendekatan agglomerative, yang lebih umum digunakan, memulai dengan setiap objek sebagai klaster individu dan secara iteratif menggabungkan klaster yang paling mirip. Sebaliknya, divisive memulai dengan satu klaster besar yang mencakup semua objek, kemudian secara bertahap membagi klaster menjadi klaster-klaster yang lebih kecil. Kedua pendekatan ini menggunakan matriks jarak untuk mengukur kedekatan antara objek atau klaster, dengan berbagai metrik seperti jarak Euclidean atau korelasi Pearson (Hartigan and Wong, 1979).

Salah satu keuntungan utama hierarchical clustering adalah kemampuannya untuk menghasilkan dendrogram, yaitu representasi visual yang menunjukkan hubungan hierarkis antara klaster. Dendrogram memungkinkan pengguna untuk memutuskan jumlah klaster yang optimal dengan memotong pohon pada level tertentu. Selain itu, hierarchical clustering juga bermanfaat untuk mengeksplorasi data yang tidak memiliki struktur klaster yang jelas atau memiliki pola yang kompleks (Sokal and Michener, 1958).

Namun, hierarchical clustering memiliki beberapa keterbatasan. Metode ini cenderung kurang efisien untuk dataset besar karena kompleksitas komputasinya yang tinggi. Selain itu, keputusan penggabungan atau pemisahan pada tahap awal bersifat permanen, sehingga kesalahan awal dapat memengaruhi hasil akhir. Oleh karena itu, pemilihan metrik jarak dan metode penggabungan menjadi faktor krusial dalam menghasilkan klaster yang relevan dan bermakna (Jain, 2010).

6.1 Tahapan Agglomerative Clustering

Agglomerative Clustering adalah salah satu metode dalam hierarchical clustering yang menggunakan pendekatan bottom-up. Metode ini mengelompokkan data dengan cara menggabungkan cluster yang paling dekat satu sama lain hingga semua data tergabung dalam satu cluster atau hingga jumlah cluster yang diinginkan tercapai.

1. Inisialisasi

Setiap objek (data point) dianggap sebagai satu cluster terpisah. Jika ada n objek, maka akan ada n cluster pada awalnya.

2. Menghitung Jarak

Hitung jarak antara semua pasangan cluster. Jarak ini dapat dihitung menggunakan berbagai metrik, seperti Euclidean, Manhattan, atau Cosine. Untuk dua titik $A(x_1, y_1)$ dan $B(x_2, y_2)$, jarak Euclidean dapat dihitung dengan rumus:

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

3. Menggabungkan Cluster

Temukan dua cluster terdekat (dengan jarak terkecil) dan gabungkan mereka menjadi satu cluster baru. Proses ini mengurangi jumlah cluster sebanyak satu.

4. Memperbarui Jarak

Setelah penggabungan, perbarui matriks jarak untuk mencerminkan jarak antara cluster baru dan cluster lainnya. Ada beberapa metode untuk memperbarui jarak, termasuk:

- **Single Linkage**: Jarak minimum antara anggota dua cluster.

$$d(A, B) = \min\{d(a, b) \mid a \in A, b \in B\}$$

- **Complete Linkage**: Jarak maksimum antara anggota dua cluster.

$$d(A, B) = \max\{d(a, b) \mid a \in A, b \in B\}$$

- **Average Linkage**: Rata-rata jarak antara semua pasangan anggota dari dua cluster.

$$d(A, B) = \frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$$

- **Ward's Method**: Menghitung jarak berdasarkan peningkatan varians yang dihasilkan dari penggabungan dua cluster.

$$d(A, B) = \sqrt{\frac{n_A n_B}{n_A + n_B} \cdot d^2(A, B)}$$

  di mana n_A dan n_B adalah jumlah anggota dalam cluster A dan B.

5. Ulangi Proses

Ulangi langkah 3 dan 4 hingga semua objek tergabung dalam satu cluster atau hingga jumlah cluster yang diinginkan tercapai.

Table 6.1: Basis Data Terpadu Jawa Tengah

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
CILACAP	5.19	5.67	5.08	5.44	5.22	6.05	11.47	9.78	5.55	5.12
BANYUMAS	5.71	4.47	5.18	5.51	5.02	6.21	7.39	6.96	5.98	8.22
PURBALINGGA	3.30	2.19	3.80	3.13	3.73	3.34	8.71	7.41	3.21	4.65
BANJARNEGARA	2.73	2.34	3.76	2.80	2.57	2.99	3.31	5.45	4.21	6.05
KEBUMEN	4.17	2.55	3.26	4.16	3.15	4.15	4.30	9.29	4.61	4.34
PURWOREJO	1.87	2.12	1.48	3.05	1.78	1.83	5.00	4.90	3.12	2.09
WONOSOBO	2.13	1.95	3.00	1.78	1.62	2.06	0.45	2.32	3.57	0.84
MAGELANG	3.95	3.01	4.22	4.15	3.01	3.64	1.44	3.35	5.69	3.67
BOYOLALI	2.19	3.07	1.61	2.74	2.11	1.82	1.71	2.34	3.41	1.55
KLATEN	3.84	5.15	1.93	4.64	4.04	3.78	8.71	4.45	3.99	3.09

6. Membuat Dendrogram

Setelah semua penggabungan selesai, buat dendrogram untuk memvisualisasikan proses penggabungan cluster. Dendrogram menunjukkan hubungan antar cluster dan dapat digunakan untuk menentukan jumlah cluster yang optimal dengan memotong dendrogram pada ketinggian tertentu.

Kesimpulan

Agglomerative Clustering adalah metode yang efektif untuk mengelompokkan data berdasarkan kesamaan. Dengan mengikuti tahapan di atas, Anda dapat menerapkan teknik ini untuk berbagai aplikasi analisis data. Pastikan untuk memilih metrik jarak dan metode penggabungan yang sesuai dengan karakteristik data Anda untuk mendapatkan hasil yang optimal.

6.2 Eksperimen Agglomerative Clustering

```
library (readr)
urlfile = "https://bit.ly/3VO3kRE"
data<-read.csv(url(urlfile), row.names = "Kabupaten")
```

```
knitr::kable(
  head(data, 10), caption = 'Basis Data Terpadu Jawa Tengah',
  booktabs = TRUE)
```

Standarisasi Data

Sebelum melakukan analisis clustering, penting untuk menstandarisasi data agar setiap fitur memiliki skala yang sama. Standarisasi membantu menghindari bias yang mungkin muncul akibat perbedaan skala antar fitur, yang dapat mempengaruhi hasil analisis, terutama dalam pengukuran jarak.

Dalam R, kita dapat menggunakan fungsi `scale()` untuk menstandarisasi dataset. Fungsi ini mengubah setiap fitur dalam dataset sehingga memiliki rata-rata 0 dan deviasi standar 1. Proses ini

Table 6.2: Data Hasil Standarisasi

	X1	X2	X3	X4	X5	X6	X7	
CILACAP	1.4450108	1.6357957	1.0041319	1.6983202	1.3694832	1.7177332	3.2150170	2.4
BANYUMAS	1.7671470	0.9380866	1.0492817	1.7443527	1.2535376	1.8038274	1.6920805	1.4
PURBALINGGA	0.2741697	-0.3875608	0.4262142	0.1792452	0.5056885	0.2595125	2.1847953	1.6
BANJARNEGARA	-0.0789411	-0.3003472	0.4081543	-0.0377655	-0.1667960	0.0711815	0.1691441	0.9
KEBUMEN	0.8131283	-0.1782481	0.1824052	0.8565817	0.1694462	0.6953645	0.5386801	2.2
PURWOREJO	-0.6117047	-0.4282605	-0.6212614	0.1266366	-0.6247812	-0.5530016	0.7999682	0.7
WONOSOBO	-0.4506367	-0.5271026	0.0650157	-0.7085259	-0.7175376	-0.4292411	-0.8984045	-0.1
MAGELANG	0.6768400	0.0892071	0.6158435	0.8500056	0.0882843	0.4209392	-0.5288685	0.1
BOYOLALI	-0.4134671	0.1240926	-0.5625667	-0.0772220	-0.4334709	-0.5583825	-0.4280859	-0.1
KLATEN	0.6086958	1.3334551	-0.4180873	1.1722336	0.6854041	0.4962716	2.1847953	0.5

dilakukan dengan cara mengurangi rata-rata dari setiap nilai dan membaginya dengan deviasi standar fitur tersebut.

```
# Standarisasi Data
df <- scale(data)
```

Ukuran Similarity dan Dissimilarity

Dalam analisis clustering, ukuran similarity (kesamaan) dan dissimilarity (ketidaksamaan) sangat penting untuk menentukan seberapa dekat atau jauh objek satu sama lain dalam ruang fitur. Ukuran dissimilarity sering digunakan untuk mengukur jarak antara data point, yang kemudian digunakan dalam algoritma clustering untuk mengelompokkan data berdasarkan kedekatan mereka.

Salah satu langkah awal dalam proses clustering adalah menghitung matriks dissimilarity, yang memberikan informasi tentang jarak antara setiap pasangan objek dalam dataset. Dalam konteks ini, kita menggunakan ukuran jarak Euclidean, yang merupakan salah satu metode paling umum untuk mengukur jarak antara dua titik dalam ruang multidimensi.

```
res.dist <- dist(df, method = "euclidean")
```

Pada kode di atas, kita menggunakan fungsi `dist()` dari R untuk menghitung matriks dissimilarity. Parameter `df` merujuk pada data yang telah distandarisasi, yang berarti bahwa setiap fitur dalam dataset telah dinormalisasi untuk memiliki rata-rata 0 dan deviasi standar 1. Proses standarisasi ini penting untuk memastikan bahwa semua fitur berkontribusi secara seimbang terhadap perhitungan jarak, terutama ketika fitur memiliki skala yang berbeda.

Setelah menghitung matriks dissimilarity menggunakan fungsi `dist()`, kita sering kali ingin melihat representasi matriks tersebut dalam format yang lebih mudah dibaca. Dalam R, kita dapat menggunakan fungsi `as.matrix()` untuk mengonversi objek dissimilarity menjadi matriks biasa. Ini memungkinkan kita untuk melihat nilai-nilai jarak antara objek-objek dalam dataset.

Pada kode di atas, `res.dist` adalah objek dissimilarity yang telah dihitung sebelumnya. Dengan menggunakan `as.matrix(res.dist)`, kita mengonversi objek tersebut menjadi matriks. Kemudian, kita menggunakan indeks `[1:5, 1:5]` untuk menampilkan lima baris dan lima kolom pertama dari matriks dissimilarity.

```
as.matrix(res.dist)[1:5, 1:5]
#>               CILACAP BANYUMAS PURBALINGGA BANJARNEGARA  KEBUMEN
#> CILACAP      0.000000 2.327193    3.828424     5.188508 3.891360
#> BANYUMAS     2.327193 0.000000    3.809719     4.232529 3.310710
#> PURBALINGGA  3.828424 3.809719    0.000000     2.418211 2.235801
#> BANJARNEGARA 5.188508 4.232529    2.418211     0.000000 2.159694
#> KEBUMEN      3.891360 3.310710    2.235801     2.159694 0.000000
```

Melakukan Pengelompokan Hierarkis

Setelah menghitung matriks dissimilarity, langkah selanjutnya dalam analisis clustering adalah menerapkan algoritma pengelompokan hierarkis. Dalam R, kita dapat menggunakan fungsi `hclust()` untuk melakukan ini. Salah satu metode yang umum digunakan dalam pengelompokan hierarkis adalah metode Ward, yang bertujuan untuk meminimalkan varians dalam setiap cluster yang terbentuk.

```
res.hc <- hclust(d =res.dist, method = "ward.D2")
```

Pada kode di atas, `res.dist` adalah matriks dissimilarity yang telah dihitung sebelumnya. Parameter `method = "ward.D2"` menunjukkan bahwa kita menggunakan metode Ward untuk menggabungkan cluster. Metode Ward bekerja dengan cara menggabungkan dua cluster yang menghasilkan peningkatan terkecil dalam total varians dalam cluster yang baru terbentuk. Ini membantu dalam menghasilkan cluster yang lebih homogen.

Hasil dari fungsi `hclust()` adalah objek yang berisi informasi tentang struktur pengelompokan hierarkis. Objek ini dapat digunakan untuk membuat dendrogram, yang merupakan representasi visual dari proses penggabungan cluster. Dendrogram ini membantu dalam menentukan jumlah cluster yang optimal dengan memotong dendrogram pada ketinggian tertentu.

Visualisasi dengan Dendrogram

Setelah melakukan pengelompokan hierarkis menggunakan metode Ward, langkah selanjutnya adalah memvisualisasikan hasil clustering untuk memahami struktur dan hubungan antar cluster. Salah satu cara yang efektif untuk melakukan ini adalah dengan menggunakan dendrogram, yang menunjukkan bagaimana objek-objek dalam dataset digabungkan menjadi cluster.

Library `factoextra` menyediakan fungsi yang memudahkan visualisasi hasil analisis multivariat, termasuk dendrogram untuk pengelompokan hierarkis. Berikut adalah kode untuk memvisualisasikan dendrogram dari hasil pengelompokan hierarkis yang telah dilakukan:

```
library("factoextra")
library(ggplot2)
fviz_dend(res.hc, cex = 0.5)
```

Pada kode di atas, kita pertama-tama memuat library `factoextra` dan `ggplot2`. Fungsi `fviz_dend()` digunakan untuk membuat dendrogram dari objek hasil pengelompokan hierarkis `res.hc`. Parameter `cex = 0.5` digunakan untuk mengatur ukuran label pada dendrogram, sehingga label lebih mudah dibaca.

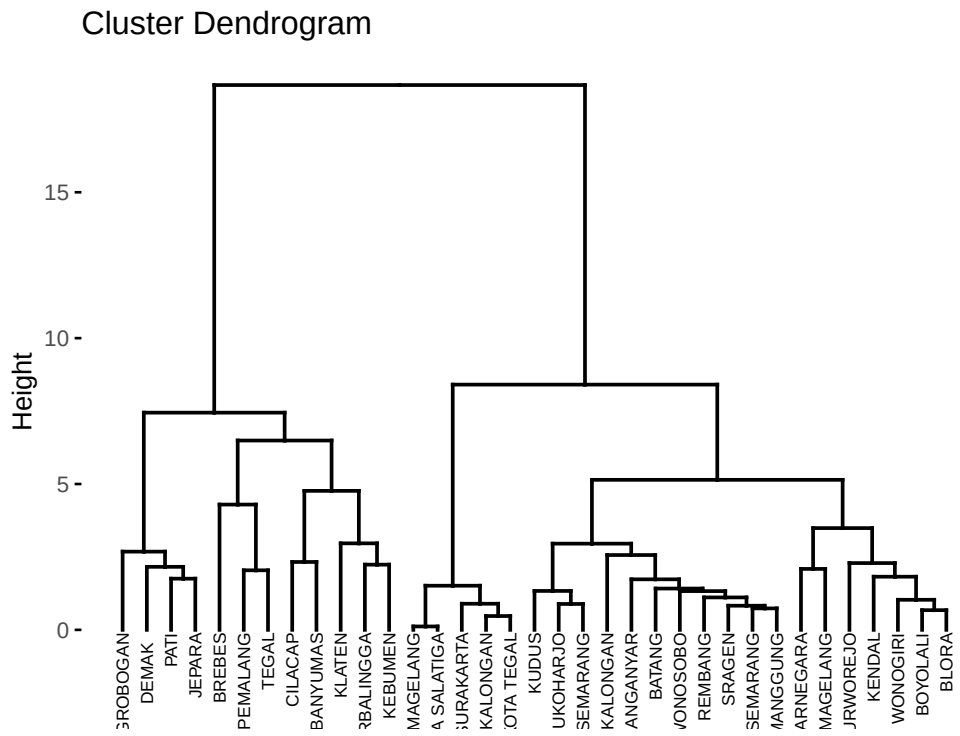

Figure 6.1: Visualisasi Dendogram

Visualisasi dengan Dendrogram

Setelah memvisualisasikan dendrogram dari hasil pengelompokan hierarkis, langkah selanjutnya adalah menentukan jumlah cluster yang diinginkan dan memotong dendrogram untuk membentuk kelompok-kelompok tersebut. Dalam contoh ini, kita akan memotong dendrogram menjadi dua kelompok.

Fungsi `cutree()` digunakan untuk memotong dendrogram dan mengelompokkan objek-objek ke dalam jumlah cluster yang ditentukan. Berikut adalah kode untuk memotong dendrogram menjadi dua kelompok:

```
grp <- cutree(res.hc, k =2)
head(grp, n =2)
#>  CILACAP BANYUMAS
#>        1        1
```

Pada kode di atas, `res.hc` adalah objek hasil pengelompokan hierarkis yang telah kita buat sebelumnya. Parameter `k = 2` menunjukkan bahwa kita ingin memotong dendrogram menjadi dua kelompok. Hasil dari fungsi `cutree()` disimpan dalam variabel `grp`, yang berisi penugasan kelompok untuk setiap objek dalam dataset.

Jumlah Anggota dalam Setiap Cluster

Setelah memotong dendrogram dan mengelompokkan objek-objek ke dalam cluster, langkah selanjutnya adalah menghitung jumlah anggota dalam setiap cluster. Dalam contoh ini, kita akan menggunakan fungsi table() untuk menghitung jumlah anggota dalam setiap cluster.

Fungsi `table()` digunakan untuk menghitung frekuensi atau jumlah anggota dalam setiap kategori atau cluster. Berikut adalah kode untuk menghitung jumlah anggota dalam setiap cluster:

```
table(grp)
#> grp
#>  1  2
#> 12 23
```

Visualisasi Dendrogram

Setelah memotong dendrogram untuk membentuk kelompok, kita dapat memvisualisasikan hasil pengelompokan dengan menandai setiap kelompok menggunakan warna yang berbeda. Ini membantu dalam memahami struktur cluster dan memudahkan interpretasi hasil.

```
fviz_dend(res.hc, k =2,
          cex = 0.5,
          k_colors = c("#E7B800", "#FC4E07"),
          color_labels_by_k = TRUE,
          rect = TRUE
          )
```

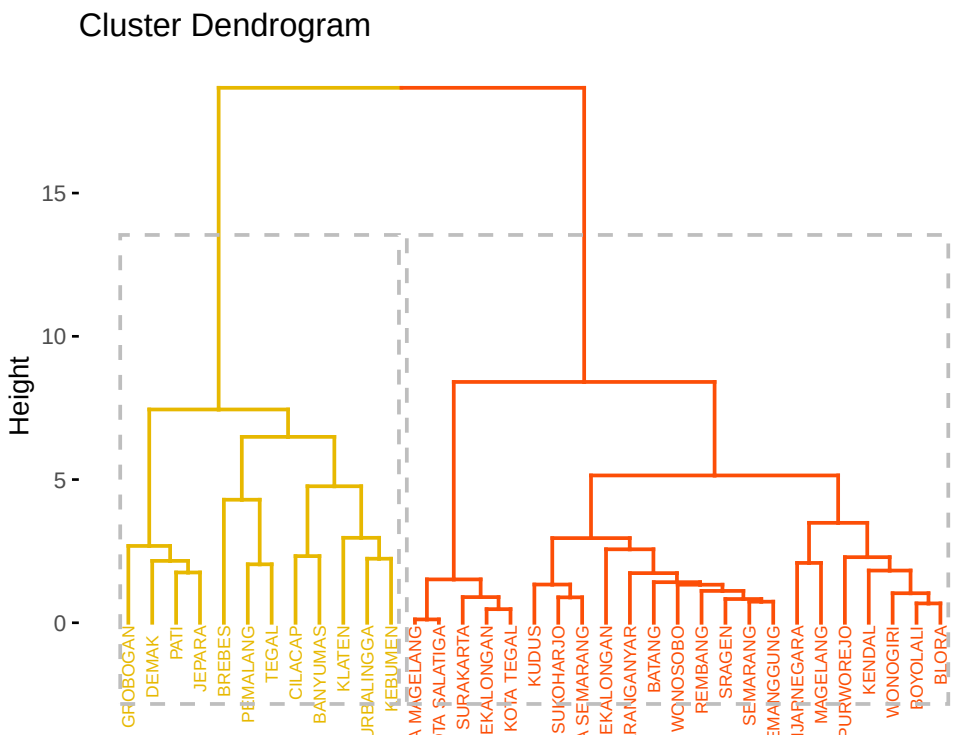

Figure 6.2: Visualisasi Dendogram

Pada kode di atas, kita menggunakan fungsi `fviz_dend()` untuk membuat dendrogram dari objek hasil pengelompokan hierarkis res.hc. Parameter yang digunakan adalah sebagai berikut: `k = 2` Menunjukkan bahwa kita ingin memotong dendrogram menjadi dua kelompok. `cex = 0.5`

Mengatur ukuran label pada dendrogram agar lebih mudah dibaca. `k_colors = c("#E7B800", "#FC4E07")` Menentukan warna yang akan digunakan untuk masing-masing kelompok. Dalam hal ini, kelompok pertama akan berwarna kuning (#E7B800) dan kelompok kedua berwarna oranye (#FC4E07).`color_labels_by_k = TRUE` Mengatur agar label pada dendrogram diwarnai sesuai dengan kelompok yang ditentukan. `rect = TRUE` Menambahkan kotak di sekitar kelompok untuk menyoroti batasan antar cluster.

Visualisasi Hasil Clustering

Setelah melakukan pengelompokan data dan mendapatkan penugasan kelompok untuk setiap objek, langkah selanjutnya adalah memvisualisasikan hasil clustering. Visualisasi ini membantu kita memahami distribusi objek dalam setiap cluster dan bagaimana cluster tersebut terpisah satu sama lain.

Pada kode di bwah, kita menggunakan fungsi `fviz_cluster()` untuk membuat visualisasi clustering. Parameter yang digunakan adalah sebagai berikut:

- **`list(data = df, cluster = grp):`** Menyediakan data yang telah distandarisasi (df) dan penugasan kelompok (grp) yang dihasilkan dari pemotongan dendrogram.
- **`palette = c("#E7B800", "#FC4E07")`**: Menentukan warna yang akan digunakan untuk masing-masing cluster. Dalam hal ini, cluster pertama akan berwarna kuning (#E7B800) dan cluster kedua berwarna oranye (#FC4E07).
- **`ellipse.type = "convex"`**: Menambahkan elips konsentrasi di sekitar setiap cluster, yang memberikan gambaran visual tentang sebaran objek dalam cluster.
- **`repel = TRUE`**: Menghindari tumpang tindih label pada plot, meskipun ini dapat memperlambat proses rendering.
- **`show.clust.cent = FALSE`**: Menentukan untuk tidak menampilkan pusat cluster pada visualisasi.
- **`ggtheme = theme_minimal()`**: Menggunakan tema minimal untuk plot, yang memberikan tampilan yang bersih dan profesional.

```
fviz_cluster(list(data = df, cluster = grp),
palette = c("#E7B800", "#FC4E07"),
ellipse.type = "convex",
repel = TRUE,
show.clust.cent = FALSE, ggtheme = theme_minimal())
```

6.3 Perbandingan Dendogram

Dalam analisis klaster hierarkis, dendrogram adalah representasi grafis yang menunjukkan bagaimana objek dikelompokkan secara hierarkis berdasarkan kemiripan atau jarak mereka. Namun, hasil klastering dapat sangat dipengaruhi oleh metode yang digunakan untuk menghitung jarak antar objek maupun cara menggabungkan klaster. Oleh karena itu, membandingkan dendrogram yang dihasilkan oleh berbagai metode menjadi langkah penting untuk mengevaluasi konsistensi dan validitas struktur klaster.

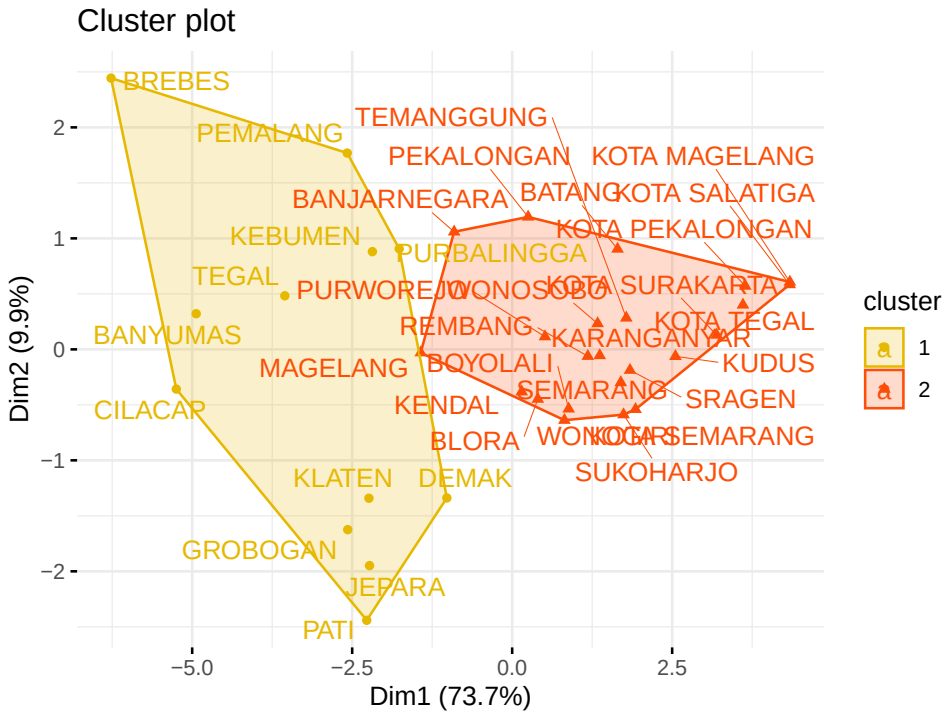

Figure 6.3: Visualisasi Hasil CLustering

Membandingkan dua dendrogram melibatkan analisis kesamaan struktur yang dihasilkan oleh metode klastering yang berbeda. Hal ini dapat membantu mengidentifikasi pola yang stabil dalam data serta mendeteksi perbedaan yang muncul akibat perubahan algoritma, metrik jarak, atau parameter lainnya. Dengan memahami perbedaan ini, peneliti dapat memilih metode klastering yang paling sesuai dengan tujuan analisis dan karakteristik dataset.

Pendekatan untuk membandingkan dendrogram mencakup visualisasi seperti tanglegram, yang menunjukkan kesamaan dan perbedaan hubungan antar klaster, serta metrik kuantitatif seperti korelasi kophenetik. Analisis semacam ini memberikan wawasan yang mendalam tentang bagaimana data dikelompokkan, sekaligus memastikan bahwa hasil klastering relevan dan dapat diandalkan.

```
# Memuat pustaka dendextend
library(dendextend)

# Menghitung matriks jarak dengan metode Euclidean
res.dist <- dist(df, method = "euclidean")

# Melakukan hierarchical clustering dengan dua metode berbeda
hc1 <- hclust(res.dist, method = "average")  # Average linkage
hc2 <- hclust(res.dist, method = "ward.D2")  # Ward's method

# Mengubah hasil clustering menjadi objek dendrogram
dend1 <- as.dendrogram(hc1)
dend2 <- as.dendrogram(hc2)

# Membuat daftar untuk menyimpan dendrogram
```

```
dend_list <- dendlist(dend1, dend2)
```

Kode berikut digunakan untuk membuat tanglegram sederhana, yaitu representasi visual untuk membandingkan dua dendrogram:

```
tanglegram(dend1, dend2)
```

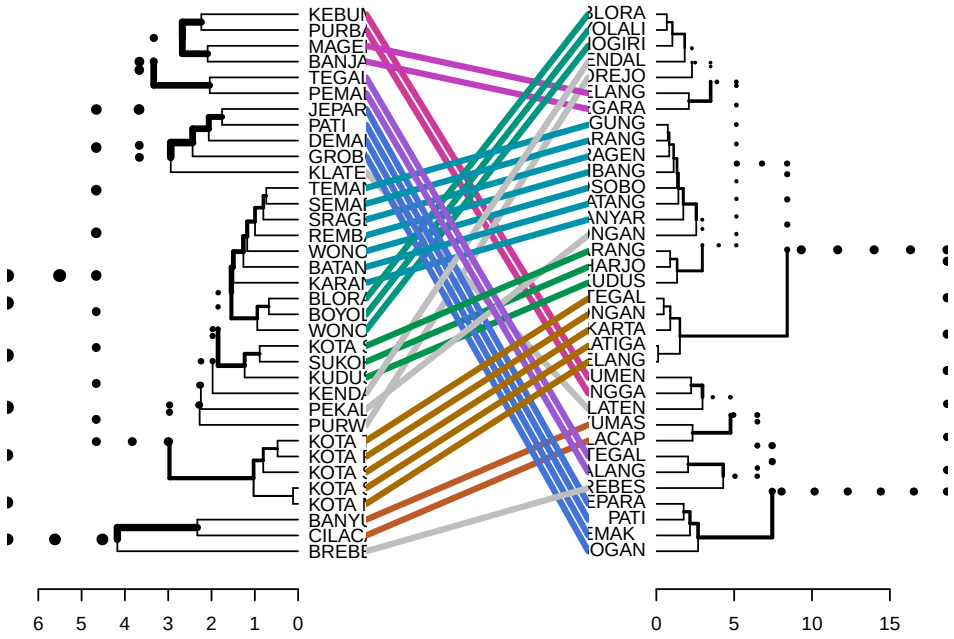

Figure 6.4: Perbandingan dua Dendogram

Kode berikut digunakan untuk membuat tanglegram dengan penyesuaian tertentu untuk menyoroti cabang yang sama (common branches) antara dua dendrogram:

```
tanglegram(dend1, dend2,
           highlight_distinct_edges = FALSE, # Nonaktifkan garis putus-putus
           common_subtrees_color_lines = FALSE, # Nonaktifkan pewarnaan garis
           common_subtrees_color_branches = TRUE, # Warnai cabang yang sama
           main = paste("Entanglement =", round(entanglement(dend_list), 2))
)
```

6.4 Tahapan Divisive Clustering

Divisive Clustering adalah metode pengelompokan hierarkis yang menggunakan pendekatan top-down. Metode ini dimulai dengan satu cluster besar yang mencakup semua objek dan secara bertahap membaginya menjadi sub-cluster. Dalam dokumen ini, kita akan menjelaskan tahapan-tahapan dalam Divisive Clustering beserta rumus yang relevan.

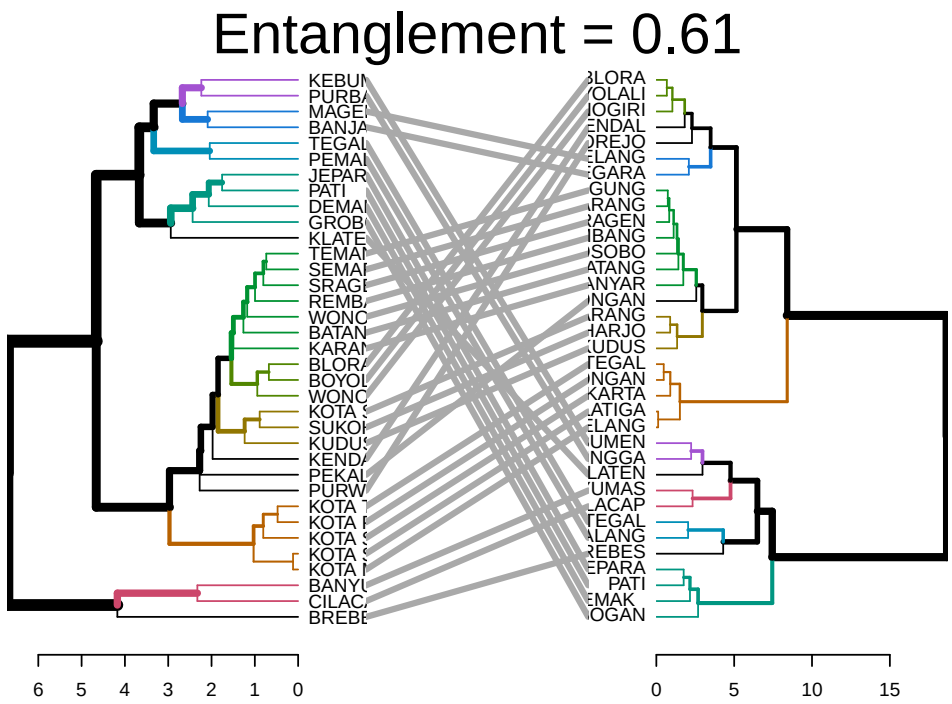

Figure 6.5: Perbandingan dua Dendogram

1. Inisialisasi

Mulai dengan satu cluster yang mencakup semua objek dalam dataset. Jika ada n objek, maka pada awalnya hanya ada satu cluster besar.

2. Menghitung Dissimilarity

Hitung matriks dissimilarity untuk semua objek dalam cluster. Dissimilarity dapat dihitung menggunakan berbagai metrik, seperti jarak Euclidean. Untuk dua titik $A(x_1, y_1)$ dan $B(x_2, y_2)$, jarak Euclidean dapat dihitung dengan rumus:

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

3. Memilih Cluster untuk Dibagi

Pilih cluster yang akan dibagi. Biasanya, cluster yang memiliki varians terbesar atau yang paling heterogen dipilih untuk dibagi.

4. Membagi Cluster

Bagi cluster yang dipilih menjadi dua sub-cluster. Ada beberapa metode untuk membagi cluster, termasuk: **K-Means**: Menggunakan algoritma K-Means untuk membagi cluster menjadi dua sub-cluster berdasarkan jarak. **PCA (Principal Component Analysis)**: Menggunakan PCA untuk mengidentifikasi arah varians terbesar dan membagi objek berdasarkan komponen utama.

5. Menghitung Dissimilarity untuk Sub-Cluster

Setelah membagi cluster, hitung kembali matriks dissimilarity untuk sub-cluster yang baru terbentuk.

6. Ulangi Proses

Ulangi langkah 3 hingga 5 hingga semua objek tergabung dalam cluster terpisah atau hingga jumlah cluster yang diinginkan tercapai.

7. Membuat Dendrogram

Setelah semua pembagian selesai, buat dendrogram untuk memvisualisasikan proses pembagian cluster. Dendrogram ini menunjukkan bagaimana objek-objek dikelompokkan dan dapat digunakan untuk menentukan jumlah cluster yang optimal.

6.5 Eksperimen Divisive Clustering

Berikut adalah contoh kode untuk melakukan Divisive Analysis Clustering (DIANA) menggunakan fungsi `diana()` dari paket `cluster` di R.

```
library(cluster)
res.diana <- diana(x = data,
                   stand = TRUE,
                   metric = "euclidean")
```

Visualisasi dendrogram menggunakan fungsi `fviz_dend` dari paket `factoextra`

```
fviz_dend(res.diana, k =2,
          cex = 0.5,
          k_colors = c("#E7B800", "#FC4E07"),
          color_labels_by_k = TRUE,
          rect = TRUE
          )
```

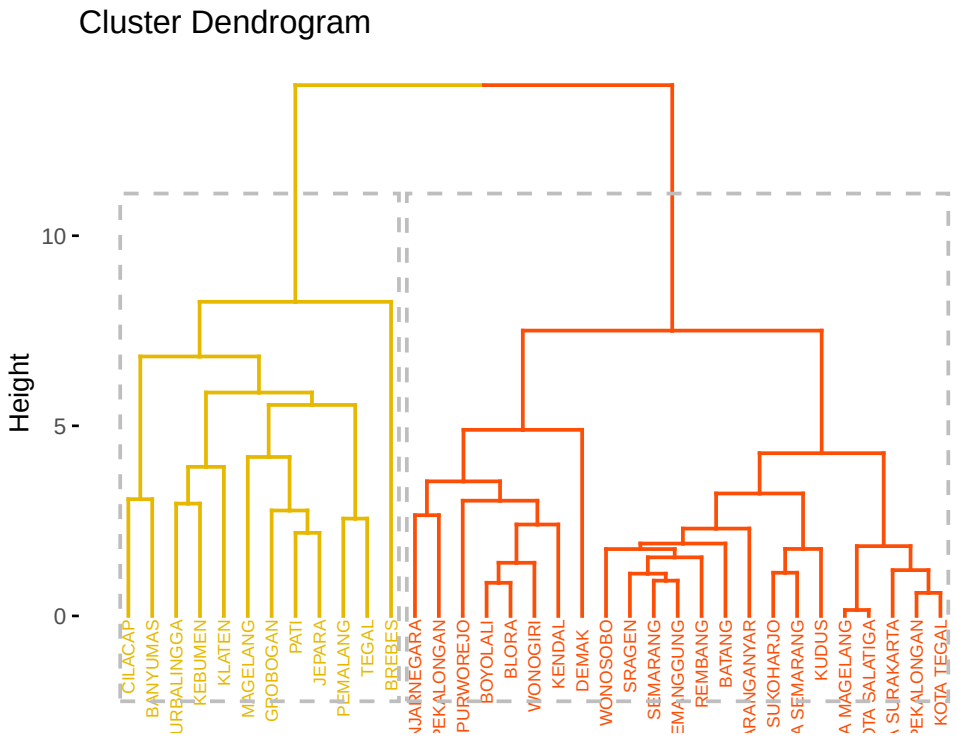

Figure 6.6: Visualisasi Dendogram

Chapter 7

Visualisasi Dendogram

Chapter 8

Algoritma Fuzzy C-Means

Algoritma fuzzy c-means (FCM) merupakan salah satu metode clustering yang banyak digunakan dalam analisis data dan pengolahan citra. Berbeda dengan algoritma clustering tradisional seperti k-means, yang mengelompokkan data ke dalam cluster yang jelas dan tegas, FCM memberikan fleksibilitas dengan memungkinkan setiap data untuk memiliki derajat keanggotaan pada lebih dari satu cluster. Hal ini menjadikan FCM sangat berguna dalam situasi di mana batasan antara cluster tidak dapat ditentukan dengan jelas (Bezdek, 1981).

FCM pertama kali diperkenalkan oleh Bezdek pada tahun 1981 dan sejak saat itu telah banyak diterapkan dalam berbagai bidang, termasuk pengolahan citra, pengenalan pola, dan analisis data multidimensi. Metode ini bekerja dengan meminimalkan fungsi objektif yang mengukur kesalahan antara data dan pusat cluster, dengan mempertimbangkan derajat keanggotaan setiap data terhadap cluster yang ada. Proses ini dilakukan secara iteratif hingga konvergensi tercapai, di mana perubahan pusat cluster dan derajat keanggotaan menjadi sangat kecil (Dunn, 1973).

Salah satu keunggulan FCM adalah kemampuannya untuk menangani data yang memiliki noise atau outlier. Dalam banyak aplikasi dunia nyata, data sering kali tidak bersih dan mengandung kesalahan pengukuran. Dengan menggunakan derajat keanggotaan, FCM dapat mengurangi pengaruh data yang tidak representatif terhadap hasil clustering, sehingga menghasilkan model yang lebih robust dan akurat (Pal and Bezdek, 1995).

Namun, meskipun FCM memiliki banyak kelebihan, algoritma ini juga memiliki beberapa kelemahan. Salah satunya adalah ketergantungan pada pemilihan jumlah cluster yang tepat, yang dapat mempengaruhi hasil akhir. Selain itu, FCM juga dapat menjadi sensitif terhadap inisialisasi pusat cluster, yang dapat menyebabkan hasil yang berbeda pada setiap iterasi (Huang, 1998). Oleh karena itu, penelitian lebih lanjut diperlukan untuk mengembangkan metode yang dapat mengatasi masalah ini.

Dalam konteks perkembangan teknologi dan kebutuhan analisis data yang semakin kompleks, FCM tetap menjadi salah satu metode yang relevan dan banyak digunakan. Penelitian dan pengembangan lebih lanjut dalam algoritma ini diharapkan dapat meningkatkan kinerjanya dan memperluas aplikasinya di berbagai bidang, termasuk kecerdasan buatan dan analisis big data. Dengan demikian, pemahaman yang mendalam tentang FCM dan aplikasinya sangat penting bagi para peneliti dan praktisi di bidang ini.

Klastering dengan algoritma Fuzzy C-Means didasarkan pada teori logika fuzzy yang diperkenalkan oleh Lotfi Zadeh pada tahun 1965 dengan nama himpunan fuzzy (fuzzy set). Fuzzy C-Means Clustering pertama kali diperkenalkan oleh Dun pada (1973) dan diperbaiki oleh Bezdek (Bezdek, 1981) . Dalam teori fuzzy, keangotaan sebuah data diberikan dengan suatu nilai derajat keanggotaan yang jangkauan nilainya 0 sampai 1. Semakin tinggi nilai derajat keanggotaannya maka semakin tinggi

nilai keanggotaan sebuah data dalam suatu kelompok dan semakin kecil nilai derajat keanggotaannya maka semakin rendah nilai keanggotaan sebuah data dalam suatu kelompok.

8.1 Tahapan Algoritma Fuzzy C-Means

Berikut adalah tahapan algoritma Fuzzy C-Means (FCM):

1. Inisialisasi Parameter

Tentukan jumlah cluster (c), parameter pembobot (m, biasanya $m > 1$), toleransi error (ϵ), dan maksimum iterasi.

2. Inisialisasi Matriks Keanggotaan

Buat matriks keanggotaan awal ($U^{(0)}$) secara acak. Matriks ini memiliki ukuran $N \times c$, di mana N adalah jumlah data, dan c adalah jumlah cluster. Pastikan bahwa $\sum_{j=1}^{c} u_{ij} = 1$ untuk setiap data i.

3. Perhitungan Pusat Cluster

Hitung pusat cluster (v_k) untuk setiap cluster k menggunakan rumus:

$$v_k = \frac{\sum_{i=1}^{N} u_{ik}^m x_i}{\sum_{i=1}^{N} u_{ik}^m}$$

di mana u_{ik} adalah nilai keanggotaan data i pada cluster k dan x_i adalah data ke-i.

4. Perbarui Matriks Keanggotaan

Hitung matriks keanggotaan baru ($U^{(t+1)}$) dengan rumus:

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{\|x_i - v_k\|}{\|x_i - v_j\|} \right)^{\frac{2}{m-1}}}$$

di mana $\|x_i - v_k\|$ adalah jarak antara data i dan pusat cluster k.

5. Evaluasi Konvergensi

Hitung perubahan matriks keanggotaan ($\|U^{(t+1)} - U^{(t)}\|$). Jika perubahan lebih kecil dari toleransi error (ϵ) atau iterasi maksimum tercapai, algoritma berhenti. Jika tidak, kembali ke langkah 3.

6. Hasil Akhir

Pusat cluster (v_k) dan matriks keanggotaan (u_{ik}) akhir menunjukkan distribusi data terhadap cluster. Data dapat diklasifikasikan ke cluster dengan keanggotaan tertinggi.

Table 8.1: Basis Data Terpadu Jawa Tengah

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
CILACAP	5.19	5.67	5.08	5.44	5.22	6.05	11.47	9.78	5.55	5.12
BANYUMAS	5.71	4.47	5.18	5.51	5.02	6.21	7.39	6.96	5.98	8.22
PURBALINGGA	3.30	2.19	3.80	3.13	3.73	3.34	8.71	7.41	3.21	4.65
BANJARNEGARA	2.73	2.34	3.76	2.80	2.57	2.99	3.31	5.45	4.21	6.05
KEBUMEN	4.17	2.55	3.26	4.16	3.15	4.15	4.30	9.29	4.61	4.34
PURWOREJO	1.87	2.12	1.48	3.05	1.78	1.83	5.00	4.90	3.12	2.09
WONOSOBO	2.13	1.95	3.00	1.78	1.62	2.06	0.45	2.32	3.57	0.84
MAGELANG	3.95	3.01	4.22	4.15	3.01	3.64	1.44	3.35	5.69	3.67
BOYOLALI	2.19	3.07	1.61	2.74	2.11	1.82	1.71	2.34	3.41	1.55
KLATEN	3.84	5.15	1.93	4.64	4.04	3.78	8.71	4.45	3.99	3.09

8.2 Eksperimeen Fuzzy C-Means

Install dan Load Packagaes

Untuk memulai eksperimen menggunakan algoritma Fuzzy C-Means (FCM) di `R`, beberapa library perlu diinstal dan dimuat untuk mendukung proses analisis, visualisasi, dan validasi hasil clustering. Library `ppclust` menyediakan berbagai metode clustering berbasis partisi, termasuk FCM, dan berfungsi untuk mengimplementasikan algoritma serta mengevaluasi hasil clustering. Library `factoextra` digunakan untuk memudahkan visualisasi hasil clustering, seperti menampilkan scatter plot dengan pembagian cluster yang jelas. Selain itu, library `fclust` mendukung berbagai fungsi terkait clustering berbasis fuzzy, termasuk penghitungan indeks validasi untuk menilai kualitas cluster. Terakhir, library `cluster` menawarkan berbagai alat untuk clustering dan validasi, termasuk perbandingan hasil clustering FCM dengan algoritma lain seperti K-Means atau PAM.

```
library(ppclust)
library(factoextra)
library(fclust)
library(cluster)
```

Data

Dataset diimpor ke dalam `R` menggunakan library `readr`, yang dirancang untuk membaca data dengan format seperti CSV secara efisien. Data diakses melalui URL `https://bit.ly/3VO3kRE` dan dimuat menggunakan fungsi `read.csv()`. Parameter `row.names = "Kabupaten"` digunakan untuk menjadikan kolom “Kabupaten” sebagai indeks baris, sehingga setiap baris data dapat diidentifikasi berdasarkan nama kabupaten. Setelah proses ini, dataset disimpan dalam variabel `data` sebagai sebuah *data frame*, yang siap digunakan untuk analisis lebih lanjut, seperti preprocessing, penerapan algoritma clustering, atau visualisasi hasil.

```
library (readr)
urlfile = "https://bit.ly/3VO3kRE"
data<-read.csv(url(urlfile), row.names = "Kabupaten")
```

Hasil Clustering

Hasil clustering diperoleh dengan menerapkan algoritma Fuzzy C-Means (FCM) menggunakan fungsi `fcm()` dari library `ppclust`. Dataset yang telah dipersiapkan dianalisis untuk menghasilkan tiga pusat cluster dengan parameter `centers=3`. Setelah proses clustering selesai, nilai derajat keanggotaan masing-masing data terhadap setiap cluster disimpan dalam matriks keanggotaan (u). Matriks ini kemudian dikonversi menjadi format *data frame* menggunakan fungsi `as.data.frame()` agar lebih mudah diakses dan dianalisis. Setiap baris dalam *data frame* merepresentasikan satu data observasi, sementara kolom-kolomnya menunjukkan derajat keanggotaan terhadap masing-masing cluster. Derajat keanggotaan ini mencerminkan probabilitas relatif atau tingkat afiliasi suatu data terhadap cluster tertentu.

```
library(ppclust)
res.fcm <- fcm(data, centers=3)
as.data.frame(res.fcm$u)
#>                   Cluster 1  Cluster 2  Cluster 3
#> CILACAP          0.19959671 0.09432155 0.70608174
#> BANYUMAS         0.09071562 0.03242738 0.87685700
#> PURBALINGGA      0.28641412 0.12503329 0.58855259
#> BANJARNEGARA     0.56797934 0.15770912 0.27431153
#> KEBUMEN          0.32490172 0.12183017 0.55326811
#> PURWOREJO        0.46825795 0.37665821 0.15508384
#> WONOSOBO         0.21208577 0.73997169 0.04794255
#> MAGELANG         0.75569090 0.12565978 0.11864932
#> BOYOLALI         0.29335751 0.65917928 0.04746322
#> KLATEN           0.44008300 0.15366683 0.40625017
#> SUKOHARJO        0.07528219 0.90659203 0.01812578
#> WONOGIRI         0.31366262 0.62649398 0.05984340
#> KARANGANYAR      0.13567704 0.83280963 0.03151333
#> SRAGEN           0.05643609 0.92987167 0.01369224
#> GROBOGAN         0.68584478 0.10906165 0.20509357
#> BLORA            0.51050938 0.42364979 0.06584083
#> REMBANG          0.18988443 0.77496097 0.03515460
#> PATI             0.63754615 0.18226626 0.18018760
#> KUDUS            0.04642043 0.93999864 0.01358093
#> JEPARA           0.69208387 0.13357477 0.17434136
#> DEMAK            0.65516864 0.22405973 0.12077163
#> SEMARANG         0.06180543 0.92443899 0.01375558
#> TEMANGGUNG       0.06302723 0.92181846 0.01515432
#> KENDAL           0.62112953 0.30783534 0.07103512
#> BATANG           0.20285008 0.74568211 0.05146780
#> PEKALONGAN       0.54367557 0.33879435 0.11753007
#> PEMALANG         0.39383397 0.13508254 0.47108349
#> TEGAL            0.30162583 0.08391671 0.61445746
#> BREBES           0.23941019 0.11958050 0.64100931
#> KOTA MAGELANG    0.12644712 0.82227544 0.05127744
#> KOTA SURAKARTA   0.08663885 0.88284897 0.03051218
#> KOTA SALATIGA    0.12550332 0.82356236 0.05093432
#> KOTA SEMARANG    0.14670134 0.81177689 0.04152177
#> KOTA PEKALONGAN 0.08784985 0.87988142 0.03226874
#> KOTA TEGAL       0.08711147 0.88100104 0.03188749
```

Visualisasi Matrik Keanggotaan

Untuk memvisualisasikan matriks keanggotaan yang dihasilkan oleh algoritma Fuzzy C-Means (FCM) menggunakan fungsi `corrplot()` dari library `corrplot`. Matriks keanggotaan yang terdapat dalam `res.fcm$u` menggambarkan derajat keanggotaan masing-masing data terhadap setiap cluster. Dengan parameter `is.corr = FALSE`, kita memberi tahu fungsi bahwa data yang akan divisualisasikan bukanlah matriks korelasi, melainkan matriks keanggotaan FCM. Visualisasi ini membantu untuk memahami bagaimana data tersebar di antara cluster dan seberapa kuat hubungan data dengan setiap cluster berdasarkan derajat keanggotaannya. Grafik ini dapat memperlihatkan pola atau struktur dalam data yang terkelompok dengan cara yang lebih mudah dipahami.

```
# Visualize using corrplot
library(corrplot)
corrplot(res.fcm$u, is.corr = FALSE)
```

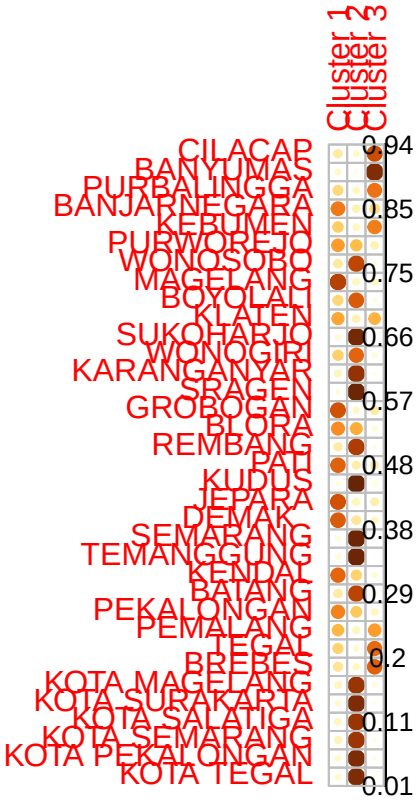

Figure 8.1: Matrik Keagotaan

Kode ini digunakan untuk mengakses dan menampilkan pusat cluster pertama (v_0) yang dihasilkan oleh algoritma Fuzzy C-Means (FCM) yang telah diterapkan pada dataset. Pada objek `res.fcm`, atribut `v0` menyimpan informasi tentang pusat cluster pada iterasi pertama. Nilai ini menunjukkan posisi awal dari setiap cluster sebelum algoritma mulai melakukan iterasi lebih lanjut untuk memperbarui posisi pusat cluster berdasarkan derajat keanggotaan data. Dengan menampilkan `res.fcm$v0`, kita dapat melihat posisi awal cluster dalam ruang fitur data dan mengevaluasi bagaimana posisi tersebut berfungsi dalam pemisahan data pada iterasi selanjutnya.

```
res.fcm$v0
#>              X1   X2   X3   X4   X5   X6   X7   X8   X9  X10
#> Cluster 1 3.79 1.95 1.22 2.05 1.81 1.72 1.98 1.36 2.20 0.93
#> Cluster 2 0.23 0.25 0.12 0.25 0.19 0.20 0.34 0.18 0.16 0.03
#> Cluster 3 4.17 2.55 3.26 4.16 3.15 4.15 4.30 9.29 4.61 4.34
```

Kode ini digunakan untuk mengakses dan menampilkan pusat cluster yang dihasilkan setelah algoritma Fuzzy C-Means (FCM) selesai dijalankan. Pada objek `res.fcm`, atribut `v` menyimpan informasi tentang posisi pusat cluster yang telah diperbarui setelah iterasi terakhir. Pusat cluster ini merupakan hasil akhir dari proses clustering dan menunjukkan lokasi sentroid (pusat) dari setiap cluster berdasarkan data dan derajat keanggotaan yang telah dihitung. Dengan menampilkan `res.fcm$v`, kita dapat melihat posisi akhir setiap cluster dalam ruang fitur data, yang membantu untuk memahami bagaimana data telah dikelompokkan ke dalam cluster yang berbeda setelah proses clustering selesai.

```
res.fcm$v
#>                 X1       X2       X3       X4       X5       X6       X7
#> Cluster 1 3.418238 4.025821 3.497306 3.657048 3.843888 3.477061 3.074419
#> Cluster 2 1.724771 1.743303 1.432408 1.706002 1.626688 1.504833 1.335312
#> Cluster 3 5.001721 4.041093 5.531054 4.669414 4.638562 5.500335 6.458007
#>                 X8       X9      X10
#> Cluster 1 3.057133 3.802309 3.529256
#> Cluster 2 1.021470 1.697120 1.056498
#> Cluster 3 7.009924 4.458629 6.474407
```

Ringkasan Hasil Clustering FCM

Untuk menampilkan ringkasan dari hasil algoritma Fuzzy C-Means (FCM) yang telah dijalankan, menggunakan fungsi `summary()` pada objek `res.fcm`. Fungsi ini memberikan gambaran umum tentang hasil clustering, termasuk informasi mengenai pusat cluster, derajat keanggotaan, dan konvergensi algoritma. Biasanya, hasil yang ditampilkan mencakup jumlah cluster yang terbentuk, posisi pusat cluster akhir, serta statistik terkait keanggotaan data terhadap cluster. Dengan menggunakan `summary(res.fcm)`, kita dapat memperoleh insight terkait seberapa baik algoritma telah mengelompokkan data dan apakah proses clustering telah konvergen, serta informasi tambahan yang dapat membantu evaluasi kualitas clustering.

```
summary(res.fcm)
#> Summary for 'res.fcm'
#>
#> Number of data objects:  35
#>
#> Number of clusters:  3
#>
#> Crisp clustering vector:
#>  [1] 3 3 3 1 3 1 2 1 2 1 2 2 2 2 1 1 2 1 2 1 1 2 2 1 2 1 3 3 3 2 2 2 2 2 2
#>
#> Initial cluster prototypes:
#>              X1   X2   X3   X4   X5   X6   X7   X8   X9  X10
#> Cluster 1 3.79 1.95 1.22 2.05 1.81 1.72 1.98 1.36 2.20 0.93
#> Cluster 2 0.23 0.25 0.12 0.25 0.19 0.20 0.34 0.18 0.16 0.03
```

```
#> Cluster 3 4.17 2.55 3.26 4.16 3.15 4.15 4.30 9.29 4.61 4.34
#>
#> Final cluster prototypes:
#>                 X1       X2       X3       X4       X5       X6       X7
#> Cluster 1 3.418238 4.025821 3.497306 3.657048 3.843888 3.477061 3.074419
#> Cluster 2 1.724771 1.743303 1.432408 1.706002 1.626688 1.504833 1.335312
#> Cluster 3 5.001721 4.041093 5.531054 4.669414 4.638562 5.500335 6.458007
#>                 X8       X9      X10
#> Cluster 1 3.057133 3.802309 3.529256
#> Cluster 2 1.021470 1.697120 1.056498
#> Cluster 3 7.009924 4.458629 6.474407
#>
#> Distance between the final cluster prototypes
#>           Cluster 1 Cluster 2
#> Cluster 2  42.66853
#> Cluster 3  48.57171 165.71763
#>
#> Difference between the initial and final cluster prototypes
#>                   X1       X2       X3        X4       X5       X6       X7
#> Cluster 1 -0.3717618 2.075821 2.277306 1.6070484 2.033888 1.757061 1.094419
#> Cluster 2  1.4947706 1.493303 1.312408 1.4560023 1.436688 1.304833 0.995312
#> Cluster 3  0.8317214 1.491093 2.271054 0.5094145 1.488562 1.350335 2.158007
#>                   X8         X9      X10
#> Cluster 1  1.6971326  1.6023093 2.599256
#> Cluster 2  0.8414704  1.5371205 1.026498
#> Cluster 3 -2.2800760 -0.1513711 2.134407
#>
#> Root Mean Squared Deviations (RMSD): 5.060824
#> Mean Absolute Deviation (MAD): 148.9348
#>
#> Membership degrees matrix (top and bottom 5 rows):
#>                Cluster 1  Cluster 2 Cluster 3
#> CILACAP       0.19959671 0.09432155 0.7060817
#> BANYUMAS      0.09071562 0.03242738 0.8768570
#> PURBALINGGA   0.28641412 0.12503329 0.5885526
#> BANJARNEGARA 0.56797934 0.15770912 0.2743115
#> KEBUMEN       0.32490172 0.12183017 0.5532681
#> ...
#>                   Cluster 1 Cluster 2  Cluster 3
#> KOTA SURAKARTA  0.08663885 0.8828490 0.03051218
#> KOTA SALATIGA   0.12550332 0.8235624 0.05093432
#> KOTA SEMARANG   0.14670134 0.8117769 0.04152177
#> KOTA PEKALONGAN 0.08784985 0.8798814 0.03226873
#> KOTA TEGAL      0.08711147 0.8810010 0.03188749
#>
#> Descriptive statistics for the membership degrees by clusters
#>           Size       Min        Q1      Mean    Median        Q3       Max
#> Cluster 1   11 0.4400830 0.5270925 0.5979972 0.6211295 0.6705067 0.7556909
#> Cluster 2   17 0.6264940 0.7749610 0.8295979 0.8328096 0.9065920 0.9399986
#> Cluster 3    7 0.4710835 0.5709104 0.6359014 0.6144575 0.6735455 0.8768570
#>
```

```
#> Dunn's Fuzziness Coefficients:
#> dunn_coeff normalized
#>  0.5999684  0.3999525
#>
#> Within cluster sum of squares by cluster:
#>       1       2       3
#> 200.0818 130.5953 220.3251
#> (between_SS / total_SS =  61.98%)
#>
#> Available components:
#>  [1] "u"          "v"          "v0"         "d"          "x"
#>  [6] "cluster"    "csize"      "sumsqrs"    "k"          "m"
#> [11] "iter"       "best.start" "func.val"   "comp.time"  "inpargs"
#> [16] "algorithm"  "call"
```

Menjalankan algoritma Fuzzy C-Means (FCM) pada dataset `data` dengan jumlah cluster yang ditentukan sebanyak 3 (`centers=3`) dan parameter `nstart=5`. Parameter `nstart=5` menunjukkan bahwa algoritma FCM akan dijalankan sebanyak 5 kali dengan inisialisasi pusat cluster yang berbeda pada setiap iterasi. Tujuan dari parameter ini adalah untuk meningkatkan kemungkinan menemukan solusi yang lebih baik dengan memulai algoritma dari berbagai kondisi awal, sehingga mengurangi kemungkinan terjebak pada solusi lokal yang tidak optimal. Setelah proses ini, hasil clustering akan disimpan dalam objek `res.fcm`, yang dapat digunakan untuk mengevaluasi pusat cluster, derajat keanggotaan, dan validitas clustering.

```
res.fcm <- fcm(data, centers=3, nstart=5)
```

Untuk menampilkan nilai objektif (fungsi tujuan) dari hasil terbaik yang ditemukan selama eksperimen Fuzzy C-Means (FCM). Nilai ini dapat diakses melalui `res.fcm$func.val`, yang memberikan informasi tentang nilai minimum dari fungsi tujuan (objective function) pada iterasi terakhir setelah algoritma selesai dijalankan.

Fungsi tujuan dalam FCM biasanya mengukur sejauh mana data sesuai dengan pusat cluster yang telah dihitung, dengan mempertimbangkan derajat keanggotaan data terhadap setiap cluster. Semakin rendah nilai fungsi tujuan, semakin baik hasil clustering yang dihasilkan, karena itu menunjukkan pemisahan yang lebih baik antara cluster dan data. Dengan menampilkan `res.fcm$func.val`, kita dapat mengevaluasi kualitas solusi terbaik yang ditemukan, yang sering digunakan sebagai indikator untuk memvalidasi hasil clustering yang diperoleh dengan multiple starts.

```
res.fcm$func.val
#> [1] 360.931 360.931 360.931 360.931 360.931
```

Untuk menampilkan jumlah iterasi yang dibutuhkan oleh algoritma Fuzzy C-Means (FCM) untuk mencapai konvergensi. Atribut `res.fcm$iter` menyimpan informasi tentang jumlah iterasi yang dilakukan selama proses clustering hingga algoritma mencapai kondisi konvergen, yaitu saat perubahan posisi pusat cluster atau matriks keanggotaan menjadi sangat kecil atau mencapai batas iterasi maksimum.

Menampilkan `res.fcm$iter` memberikan wawasan tentang seberapa cepat atau lambat algoritma FCM mencapai konvergensi dalam eksperimen yang dijalankan. Jumlah iterasi yang rendah menunjukkan bahwa algoritma dengan cepat menemukan solusi yang stabil, sementara jumlah iterasi yang tinggi dapat menunjukkan bahwa proses clustering membutuhkan waktu lebih lama untuk mencapai hasil yang konvergen.

```
res.fcm$iter
#> [1] 84 76 79 83 82
```

Untuk menampilkan hasil terbaik dari beberapa percobaan yang dilakukan selama eksekusi algoritma Fuzzy C-Means (FCM) dengan multiple starts. Atribut `res.fcm$best.start` menunjukkan indeks dari percobaan (start) yang menghasilkan solusi terbaik berdasarkan nilai fungsi tujuan (objective function) terendah.

Selama penggunaan `nstart`, algoritma FCM dijalankan beberapa kali dengan inisialisasi pusat cluster yang berbeda. `res.fcm$best.start` memberikan informasi tentang percobaan yang memberikan hasil terbaik, yang berarti percobaan tersebut memiliki nilai fungsi tujuan yang lebih kecil dibandingkan dengan percobaan lainnya. Ini membantu untuk mengetahui solusi yang paling optimal yang ditemukan oleh algoritma setelah menjalankan beberapa percobaan.

```
res.fcm$best.start
#> [1] 1
```

Visualisasi Hasil Cluster

Untuk memvisualisasikan hasil clustering yang diperoleh dari algoritma Fuzzy C-Means (FCM) menggunakan fungsi `fviz_cluster()` dari paket `factoextra`. Pertama, hasil clustering FCM yang disimpan dalam objek `res.fcm` diubah menjadi format yang kompatibel dengan visualisasi menggunakan fungsi `ppclust2()`. Hasil clustering kemudian diproses untuk diperlakukan seperti hasil dari algoritma K-Means. Selanjutnya, fungsi `fviz_cluster()` digunakan untuk memplot hasil clustering, dengan parameter tambahan seperti `ellipse.type = "convex"` untuk menambahkan elips konveks yang menggambarkan area setiap cluster, dan `palette = "jco"` untuk memberikan warna berbeda pada setiap cluster. Selain itu, `repel = TRUE` memastikan label titik data tidak saling tumpang tindih, sehingga grafik menjadi lebih mudah dibaca. Visualisasi ini memberikan gambaran yang jelas tentang pemisahan data ke dalam cluster yang berbeda, memudahkan analisis hasil clustering dan memahami struktur data.

```
res.fcm2 <- ppclust2(res.fcm, "kmeans")
factoextra::fviz_cluster(res.fcm2, data = data,
  ellipse.type = "convex",
  palette = "jco",
  repel = TRUE)
```

Visualisasi Hasil Cluster dengan clusplot

Kode ini digunakan untuk memvisualisasikan hasil clustering Fuzzy C-Means (FCM) menggunakan fungsi `clusplot()` dari paket `cluster`. Pertama, hasil clustering yang disimpan dalam objek `res.fcm` diubah menjadi format yang kompatibel dengan algoritma fuzzy `fanny` menggunakan fungsi `ppclust2()`. Selanjutnya, fungsi `clusplot()` digunakan untuk memplot hasil clustering dalam ruang dua dimensi. Data yang digunakan untuk clustering distandarisasi terlebih dahulu dengan `scale()` untuk memastikan setiap fitur memiliki skala yang seragam. Hasil clustering yang diperoleh kemudian diplot, dengan setiap cluster diberi warna yang berbeda, dan label ditampilkan pada titik data sesuai dengan cluster mereka. Plot ini juga menyertakan garis pemisah antara cluster, sehingga memudahkan pemahaman tentang bagaimana data terkelompok dan sejauh mana pemisahan antara

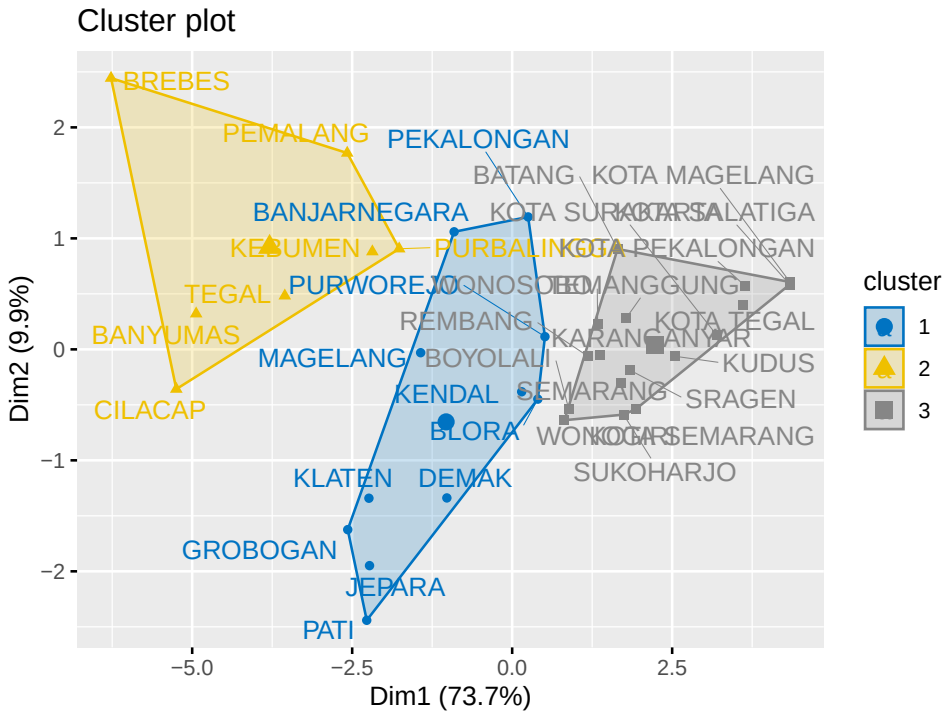

Figure 8.2: Hasil Clustering FCM

cluster tersebut. Visualisasi ini memberikan gambaran yang jelas tentang struktur data, yang sangat berguna dalam analisis hasil clustering FCM.

```
res.fcm3 <- ppclust2(res.fcm, "fanny")

cluster::clusplot(scale(data), res.fcm3$cluster,
  main = "Cluster plot of Iris data set",
  color=TRUE, labels = 2, lines = 2, cex=1)
```

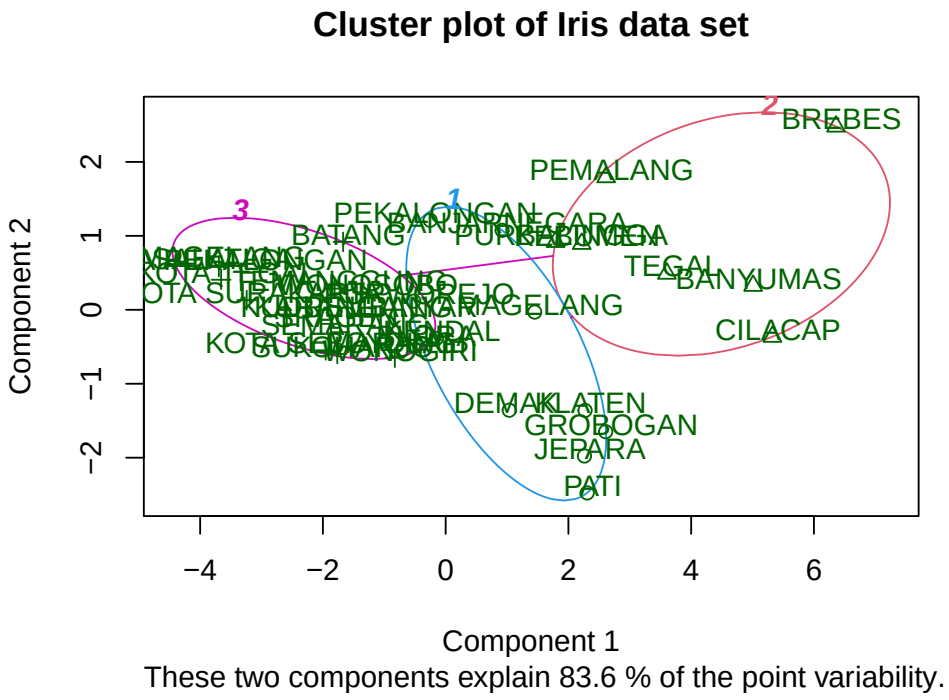

Figure 8.3: Visualisasi Hasil FCM dengan Clusplot

Referensi

Bibliography

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035.

Arum, W. K. (2024). Clustering dengan k-means dan k-modes algorithm pada dataset cost of living dan flight price prediction. *Medium*.

Bezdek, J. C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Buulolo, A. (2020). Clustering dengan k-means dan k-modes. *Medium*.

Dunn, J. C. (1973). A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57.

Elbow, W. (1975). Outline for a new approach to unsupervised clustering. In *Proceedings of the 5th Annual Conference on Statistical Computing*, pages 1–5.

Everitt, B. S., Landau, S., Leese, M., and Stahl, D. (2011). *Cluster Analysis*. Wiley.

Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2001). Cluster validity methods: A comparative study. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 31(6):576–585.

Han, J., Kamber, M., and Pei, J. (2011). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

Han, J., Kamber, M., and Pei, J. (2012). Data mining concepts and techniques, third edition.

Handayani, V. (2020). Analisis clustering menggunakan algoritma k-modes. Open Library Telkom University.

Hardandy, H., Wulan Sari, Y., and Lestari, V. (2017). *Ukuran Jarak Baru (New Dissimilarity) Dalam Algoritma Clustering K-Modes*. Universitas Gadjah Mada.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets. *Data Mining and Knowledge Discovery*, 2(3):283–304.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666.

Jiawei, H. and Kamber, M. (2006). *Data Mining: Concept and Techniques*. Morgan Kaufmann, 2 edition.

Kaufman, L. and Rousseeuw, P. (1990a). *Finding Groups in Data: an introduction to cluster analysis*. Wiley.

Kaufman, L. and Rousseeuw, P. J. (1990b). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley.

Kumar, V. and Reinartz, W. (2016). Creating enduring customer value. *Journal of Marketing*, 80(6):36–68.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297.

Pal, N. R. and Bezdek, J. C. (1995). On cluster validity for the fuzzy c-means model. *IEEE Transactions on Fuzzy Systems*, 3(3):370–379.

R Core Team (2024). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Santoso, S. (2012). *Aplikasi SPSS pada Statistik Parametrik*. PT. Elex Media Komputindo, Jakarta.

Setyawati, A. (2017). Analisis k-medoids clustering untuk pengelompokan data. *Jurnal Statistika*.

Sokal, R. R. and Michener, C. D. (1958). A statistical method for evaluating systematic relationships. *The University of Kansas Science Bulletin*, 38(21):1409–1438.

Vercellis, C. (2009). *Sistem Informasi*. Lokomedia, Yogyakarta.

Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678.