BAB II
LANDASAN TEORI

2.1 *Text Mining*

Text Mining merupakan penerapan konsep dari teknik *data mining* untuk mencari pola dalam teks, bertujuan untuk mencari informasi yang bermanfaat dengan tujuan tertentu. Berdasarkan ketidakteraturan struktur data teks, maka proses *text mining* memerlukan beberapa tahap awal yang pada intinya mempersiapkan agar teks dapat diubah menjadi lebih terstruktur (Budi Susanto, Teknik Informatika UKDW Yogyakarta).

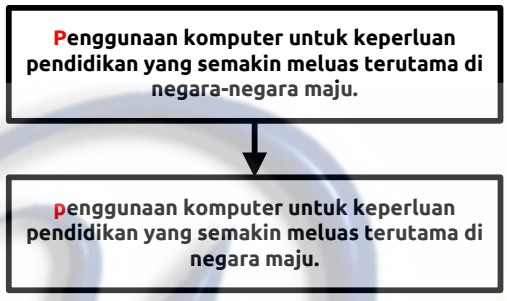
Proses *text mining* dibagi menjadi 3 tahap utama, yaitu proses awal terhadap teks (*text preprocessing*), transformasi teks (*text transformation*), dan penemuan pola (*pattern discovery*) (Even, Yahir Zohar, 2002).

2.1.1 *Text Preprocessing*

Pada *text mining*, struktur data yang baik dapat mempermudah proses komputerisasi secara otomatis. Maka dari itu, diperlukan beberapa tahapan untuk pengubahan dari informasi yang strukturnya sembarang menjadi lebih terstruktur sesuai dengan kebutuhan. Tahapan awal dari *text mining* adalah *text preprocessing* yang bertujuan untuk mempersiapkan teks menjadi data yang terstruktur dan dapat diproses pada tahapan berikutnya. Secara umum tahapan-tahapan dari *text preprocessing* adalah sebagai berikut:

1. *Case Folding*

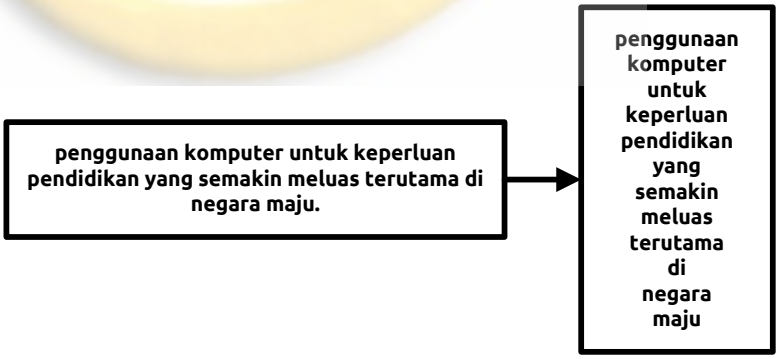
   Tahap *case folding* adalah mengubah seluruh huruf dari ‘a’ sampai dengan ‘z’ dalam dokumen menjadi huruf kecil. Tidak semua dokumen konsisten dengan penggunaan huruf kapital. Maka dari itu *case folding* mengkonversi keseluruhan teks dalam dokumen menjadi huruf kecil.



Gambar 2.1 Proses *Case Folding*

2. *Tokenizing*

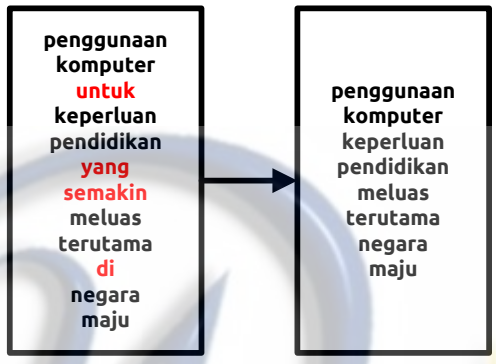
   Tahap pemotongan *string* input berdasarkan tiap kata yang menyusunnya. Dalam *tokenizing* juga dapat membuang beberapa karakter yang dianggap sebagai tanda baca.



Gambar 2.2 Proses *Tokenizing*

3. *Filtering*

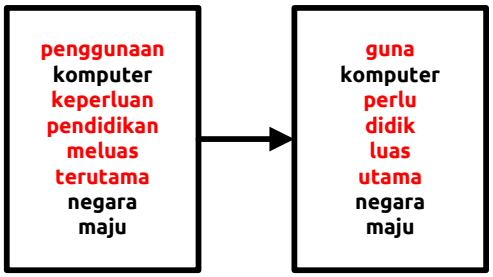
Tahap *filtering* adalah tahap mengambil kata-kata penting dari hasil *tokenizing* menggunakan algoritma *stopword removal* (membuang kata yang kurang penting). *Stopword* adalah kata-kata yang tidak deskriptif yang dapat dibuang dalam pendekatan *bag-of-words*.



Gambar 2.3 Proses *Filtering*

4. *Stemming*

Tahap *stemming* adalah tahap mencari *root* kata dari tiap kata hasil filtering. Pada tahap ini dilakukan proses pengembalian berbagai bentukan kata ke dalam suatu representasi yang sama.



Gambar 2.4 Proses *Stemming*

2.2 TF (*Term Frequency*)

Term frequency merupakan salah satu metode untuk menghitung bobot tiap *term* dalam teks. Dalam metode ini, tiap *term* diasumsikan memiliki nilai kepentingan yang sebanding dengan jumlah kemunculan *term* tersebut pada teks (Mark Hall & Lloyd Smith, 1999). Bobot sebuah *term* t pada sebuah teks dirumuskan dalam persamaan berikut :

$$W(d,t) = TF(d,t) \quad (1)$$

Dimana TF(d,t) adalah *term frequency* dari *term* t di teks d. *Term frequency* dapat memperbaiki nilai *recall* pada *information retrieval*, tetapi tidak selalu memperbaiki nilai *precision*. Hal ini disebabkan *term* yang *frequent* cenderung muncul di banyak teks, sehingga *term-term* tersebut memiliki kekuatan diskriminatif/keunikan yang kecil. Untuk memperbaiki permasalahan ini, *term* dengan nilai frekuensi yang tinggi sebaiknya dibuang dari *set term*. Menemukan *threshold* yang optimal merupakan fokus dari metode ini.

2.3 DF (*Document Frequency*)

Document frequency adalah jumlah dokumen yang mengandung suatu *term* tertentu. Tiap *term* akan dihitung nilai *document frequency*-nya (DF). Lalu *term* tersebut diseleksi berdasarkan jumlah nilai DF. Jika nilai DF berada di bawah *threshold* yang telah ditentukan, maka *term* akan dibuang. Asumsi awalnya adalah bahwa *term* yang lebih jarang muncul tidak memiliki pengaruh besar dalam proses pengelompokkan dokumen. Pembuangan *term* yang jarang ini dapat mengurangi dimensi fitur yang besar pada *text mining*.

Perbaikan dalam pengelompokkan dokumen ini juga dapat terjadi jika *term* yang dibuang tersebut juga merupakan *noise term*. *Document frequency* merupakan metode *feature selection* yang paling sederhana dengan waktu komputasi yang rendah (Yiming Yang & Jan O. Pedersen, 1997). Ilustrasi dari metode *document frequency* ini adalah sebagai berikut. Jika terdapat data

berjumlah 5000 dokumen, dan jumlah dokumen yang mengandung *term* "teknologi" adalah 150 dokumen. Maka nilai DF(teknologi) adalah 150.

2.4 IDF (*Invers Document Frequency*)

Invers document frequency (IDF) yaitu pengurangan dominasi *term* yang sering muncul di berbagai dokumen. Hal ini diperlukan karena *term* yang banyak muncul di berbagai dokumen, dapat dianggap sebagai *term* umum (*common term*) sehingga tidak penting nilainya. Sebaliknya faktor kejarangmunculan kata (*term scarity*) dalam koleksi dokumen harus diperhatikan dalam pemberian bobot. Menurut Mandala Witten (Witten, 1999) 'Kata yang muncul pada sedikit dokumen harus dipandang sebagai kata yang lebih penting (*uncommon term*) daripada kata yang muncul dari banyak dokumen'. Pembobotan akan memperhitungkan faktor dari kebalikan frekuensi dokumen yang mengandung suatu kata (*invers document frequency*). Hal ini merupakan usulan dari George Zipf. Zipf mengamati bahwa frekuensi dari sesuatu cenderung kebalikan secara proposional dengan urutannya.

Invers Document Frequency dihitung dengan formula sebagai berikut :

$$idf_j = \log(\frac{D}{df_j}) \qquad (2)$$

Dimana :

D adalah jumlah semua dokumen koleksi

df_j adalah jumlah dokumen yang mengandung *term* t_j

IDF dapat memperbaiki nilai *precision*, karena IDF mengkhususkan untuk fokus pada sebuah *term* dalam keseluruhan dokumen. Menurut penelitian yang dilakukan G. Salton (G. Salton, 1989), kombinasi antara TF dan IDF untuk menghitung bobot *term* menunjukkan bahwa gabungan keduanya menghasilkan performansi yang lebih baik. Kombinasi bobot dari sebuah *term* t pada teks d didefinisikan sebagai berikut :

$$TFIDF(d,t) = TF(d,t) \cdot IDF(t) \quad (3)$$

Faktor TF dan IDF dapat berkontribusi untuk memperbaiki nilai *recall* dan *precision* (G. Salton, 1989)

2.5 *K-Means Clustering*

K-Means Clustering adalah algoritma *clustering* yang paling populer dan banyak digunakan. Algoritma ini disusun atas dasar ide yang sederhana. Pada awalnya ditentukan berapa *cluster* yang akan dibentuk. Sembarang objek atau elemen pertama dalam *cluster* dapat dipilih untuk dijadikan titik tengah (*centroid point*) *cluster*. Pada tahapan selanjutnya Algoritma *K-Means Clustering* akan melakukan pengulangan langkah-langkah berikut sampai terjadi kestabilan (tidak ada objek yang dapat dipindahkan) :

1. Menentukan koordinat titik tengah setiap *cluster*
2. Menentukan jarak setiap objek terhadap titik tengah
3. Mengelompokan objek-objek tersebut berdasarkan jarak minimumnya.

K-Means Clustering dapat digunakan untuk pembentukan *cluster* dari sebuah basis data yang atributnya berasal dari tipe yang berbeda-beda, dengan cara mengubah atribut-atribut tersebut ke dalam indeks *similarity* dan *dissimilarity* (Andayani, FMIPA UNY Yogyakarta, 2007). Pada penerapan metode *K-Means*, data yang bisa diolah dalam perhitungan adalah data numerik yang berbentuk angka (Ediyanto, N. M. Mara dan N. Satyahadewi., 2013). Sedangkan data yang selain angka juga bisa diterapkan tetapi terlebih dahulu harus dilakukan pengkodean untuk mempermudah perhitungan jarak/kesamaan karakteristik yang dimiliki dari setiap objek. Setiap objek dihitung dari kedekatan jaraknya berdasarkan karakter yang dimiliki dengan pusat *cluster* yang sudah ditentukan sebelumnya, jarak terkecil antara objek dengan masing-masing *cluster* merupakan anggota *cluster* yang terdekat. Setelah jumlah *cluster* ditentukan,

selanjutnya dipilih objek secara acak sesuai dengan jumlah *cluster* yang dibentuk sebagai pusat *cluster* awal untuk dihitung jarak kedekatannya terhadap semua objek yang ada.

Distance space digunakan untuk menghitung jarak antara data dan *centroid*. Adapun persamaan yang dapat digunakan salah satunya yaitu *Euclidean Distance Space*. *Euclidean Distance Space* sering digunakan dalam perhitungan jarak, hal ini dikarenakan hasil yang diperoleh merupakan jarak terpendek antara dua titik yang diperhitungkan. Adapun persamaannya adalah sebagai berikut (Saban Muliadinata, 2012):

$$d_{ij} = \sqrt{\sum_{k=1}^{p} \{x_{ik} - x_{jk}\}^2} \quad (4)$$

Keterangan :

d_{ij} = Jarak objek antara objek i dan j

P = Dimensi data

X_{ik} = Koordinat dari obyek i pada dimensi k

X_{jk} = Koordinat dari obyek j pada dimensi k

Rumus klasifikasi *K-Means* :

$$\sqrt{(x_i - x_{avg})^2 + (y_i - y_{avg})^2 + (z_i - z_{avg})^2} \quad (5)$$

2.6 *K-Nearest Neighbors*

K-Nearest Neighbors (KNN) adalah sebuah metode klasifikasi terhadap sekumpulan data berdasarkan pembelajaran data yang sudah terklasifikasikan sebelumnya. KNN termasuk dalam golongan *supervised learning*, dimana hasil *query instance* yang baru diklasifikasikan berdasarkan mayoritas kedekatan jarak dari kategori yang ada dalam KNN. Nantinya kelas yang baru dari suatu data akan dipilih berdasarkan grup kelas yang paling dekat jarak vektornya. Tujuan dari

algoritma ini adalah mengklasifikasikan obyek baru berdasarkan atribut dan *training sample*. *Classifier* tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. Diberikan titik *query*, akan ditemukan sejumlah k obyek atau (titik *training*) yang paling dekat dengan titik *query*. Klasifikasi menggunakan *voting* terbanyak diantara klasifikasi dari k obyek. Algoritma *K-Nearest Neighbors* menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari *query instance* yang baru.

Metode *K-Nearest Neighbors* sangatlah sederhana, bekerja berdasarkan jarak terpendek dari *query instance* ke *training sample* untuk menentukan KNN-nya. *Training sample* diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi *training sample*. Sebuah titik pada ruang ini ditandai jika kelas c merupakan klasifikasi yang paling banyak ditemui pada k buah tetangga terdekat dari titik tersebut. Dekat atau jauhnya tetangga biasanya dihitung berdasarkan *Euclidean Distance*.

Jarak *Euclidean* paling sering digunakan dalam menghitung jarak karena sangat cocok untuk menggunakan jarak terdekat (lurus) antara dua data. Jarak *euclidean* berfungsi menguji ukuran yang bisa digunakan sebagai interpretasi kedekatan jarak antara dua obyek. yang direpresentasikan sebagai berikut:

$$D(a,b) = \sqrt{\sum_{k=1}^{d} (a_k - b_k)^2} \qquad (6)$$

Dimana D(a,b) adalah jarak skalar dari dua buah vektor a dan b dari matrik berukuran D dimensi. Pada fase *training*, algoritma ini hanya melakukan penyimpanan vektor-vektor fitur dan klasifikasi data *training sample*. Pada fase klasifikasi, fitur-fitur yang sama dihitung untuk data *testing* (yang klasifikasinya

tidak diketahui). Jarak dari vektor yang baru ini terhadap seluruh vektor *training* dihitung, dan sejumlah k yang paling dekat diambil. Titik yang baru klasifikasinya diprediksikan masuk pada klasifikasi terbanyak dari titik-titik tersebut.

Menurut Eko Prasetyo (Eko Prasetyo, 2012) dalam buku Konsep dan Aplikasi *Data Mining*, salah satu masalah yang dihadapi KNN adalah pemilihan nilai K yang tepat. Misalnya, diambil K bernilai 13, kelas 0 dimiliki oleh 7 tetangga yang jauh, sedangkan kelas 1 dimiliki oleh 6 tetangga yang lebih dekat. Hal ini mengakibatkan data uji tersebut akan terdistorsi sehingga ikut tergabung dengan kelas 0. Hal ini karena setiap tetangga tersebut mempunyai bobot yang sama terhadap data uji, sedangkan K yang terlalu kecil bisa menyebabkan algoritma terlalu sensitif terhadap *noise*. Nilai k yang bagus dapat dipilih berdasarkan optimisasi parameter, misalkan dengan *cross validation*. Kasus dimana klasifikasi diprediksikan berdasarkan *training* data yang paling dekat (dengan kata lain, k = 1) disebut algoritma *nearest neighbors*.

Ketepatan algoritma KNN sangat dipengaruhi oleh ada atau tidaknya fitur-fitur yang tidak relevan atau jika bobot fitur tersebut tidak setara dengan relevansinya terhadap klasifikasi. Riset terhadap algoritma ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur agar performa klasifikasi menjadi lebih baik (Aryo, 2012). Berikut ini adalah urutan proses pada algoritma *K-Nearest Neighbors* :

1. Menentukan parameter K (jumlah tetangga paling dekat).
2. Menghitung kuadrat jarak *euclidean* masing-masing obyek terhadap data sampel yang diberikan.
3. Mengurutkan objek-objek tersebut ke dalam kelompok yang mempunyai jarak *euclidean* terkecil.
4. Mengumpulkan kategori Y (klasifikasi *nearest neighbors*).
5. Dengan menggunakan kategori mayoritas, maka dapat diprediksikan nilai *query instance* yang telah dihitung.

KNN merupakan teknik klasifikasi sederhana, tetapi mempunyai hasil kerja yang cukup bagus. Meskipun begitu, KNN juga mempunyai kelebihan dan kekurangan. Beberapa karakteristik KNN adalah sebagai berikut (Eko Prasetyo, 2012).

1. KNN merupakan algoritma yang menggunakan seluruh data latih untuk melakukan proses klasifikasi (*complete storage*). Hal ini mengakibatkan proses prediksi yang sangat lama untuk data dalam jumlah yang sangat besar. Pendekatan lain adalah dengan menggunakan *mean* data dari setiap kelas, kemudian menghitung jarak terdekat data uji ke *mean* data setiap kelas tersebut. Hal ini memberi keuntungan kerja yang lebih cepat, tetapi hasilnya kurang memuaskan karena model hanya membentuk *hyperplane* tepat di tengah-tengah di antara 2 kelas yang memisahkan 2 kelas (untuk kasus 2 kelas). Semakin banyak data latih, semakin halus *hyperplane* yang dibuat. Ada relasi pertukaran (*trade- off relation*) antara jumlah data latih pada biaya komputasi dengan kualitas batas keputusan (*decision boundary*) yang dihasilkan.
2. Algoritma KNN tidak membedakan setiap fitur dengan suatu bobot seperti pada *Artificial Neural Network (ANN)* yang berusaha menekan fitur yang tidak mempunyai kontribusi terhadap klasifikasi menjadi 0 pada bagian bobot. KNN tidak memiliki bobot untuk masing- masing fitur.
3. Karena KNN masuk kategori *lazy learning* yang menyimpan sebagian atau semua data dan hampir tidak ada proses pelatihan, KNN sangat cepat dalam proses pelatihan (karena memang tidak ada), tetapi sangat lambat dalam proses prediksi.
4. Hal yang rumit adalah menentukan nilai K yang paling sesuai.
5. Karena KNN pada prinsipnya memilih tetangga terdekat, parameter jarak juga penting untuk dipertimbangkan sesuai dengan kasus datanya. *Euclidean* sangat cocok untuk menggunakan jarak terdekat

(lurus) antara dua data, tetapi *Manhattan* sangat teguh (*robust*) untuk mendeteksi *outlier* dalam data.

2.7 Plagiarisme

Plagiarisme berasal dari dua kata Latin, yaitu *"plagiarius"* yang berarti penculik, dan *"plagiare"* yang berarti mencuri. Menurut Kamus Besar Bahasa Indonesia, plagiat adalah perbuatan mengambil karangan atau pendapat orang lain dan menjadikannya seolah-olah karangan atau pendapat itu sebagai miliknya sendiri.

Menurut Peraturan Menteri Pendidikan Nasional nomor 17 tahun 2010, plagiat adalah tindakan secara sengaja ataupun tidak sengaja untuk memperoleh atau mencoba memperoleh kredit atau nilai untuk suatu karya ilmiah dengan mengutip sebagian atau seluruh karya dan/atau karya ilmiah pihak lain yang diakui sebagai karya ilmiahnya tanpa menyatakan sumber secara tepat dan memadai. Sementara, plagiator adalah orang perseorangan atau sekelompok orang pelaku plagiat, masing-masing bertindak untuk sendiri, untuk kelompok atau untuk dan atas nama suatu badan.

2.8 Pemrograman R

2.8.1 Sejarah dan Pengembangan R

R adalah suatu kesatuan *software* yang terintegrasi dengan beberapa fasilitas untuk manipulasi, perhitungan dan penampilan grafik yang handal. R berbasis pada bahasa pemrograman S, yang dikembangkan oleh AT&T Bell Laboratories (sekarang *Lucent Technologies*) pada akhir tahun 1970. R merupakan versi gratis bahasa S dari software (berbayar) yang sejenis yakni S-PLUS yang banyak digunakan para peneliti dan akademisi dalam melakukan kegiatan ilmiahnya.

Pada awalnya, versi pertama R dibuat oleh Ross Ihaka dan Robert Gentleman dari University of Auckland, namun selanjutnya R dikembangkan oleh tim yang disebut tim inti. Tim Inti (*core team*) terdiri dari ahli statistik, ahli komputer & pemrograman, geografi, ekonomi dari institusi yang berbeda-beda dari seluruh dunia yang mencoba membangun sebuah sistem (*software)* yang handal namun dengan biaya yang murah.

Menurut kutipan dari penghargaan *Association for Computing Machinery Software*, John Chamber menyatakan bahwa bahasa pemrograman S telah "merubah orang dalam memanipulasi, visualisasi dan menganalisis data untuk selamanya". R dibuat searah dengan ide yang ada pada bahasa pemrograman S dan program statistik lainnya. (Mitra Riset.2002)

2.8.2 R dan Program Statistik Lainnya

Seperti dijelaskan sebelumnya, R merupakan "kerabat" dekat dari S-PLUS dimana secara fungsi dan sintaks/tata bahasa yang sama-sama menggunakan bahasa S, namun tidak identik. R dapat berinteraksi dengan program statistik, manipulasi, perhitungan dan penampilan grafik lainnya, seperti SPSS yang cukup populer, Microsoft Excel dengan menyediakan fasilitas *import* dan *export* data. Selain *software* di atas, R dapat melakukan *import file* dari *software* lainnya seperti, Minitab, SAS, Stat, Systat dan EpInfo.

2.8.3 Kelebihan dan Fitur-Fitur R

R mempunyai karakteristik tersendiri, dimana selalu dimulai dengan *prompt* ">" pada *console*-nya. R mempunyai beberapa kelebihan dan fitur-fitur canggih dan berguna, diantaranya :

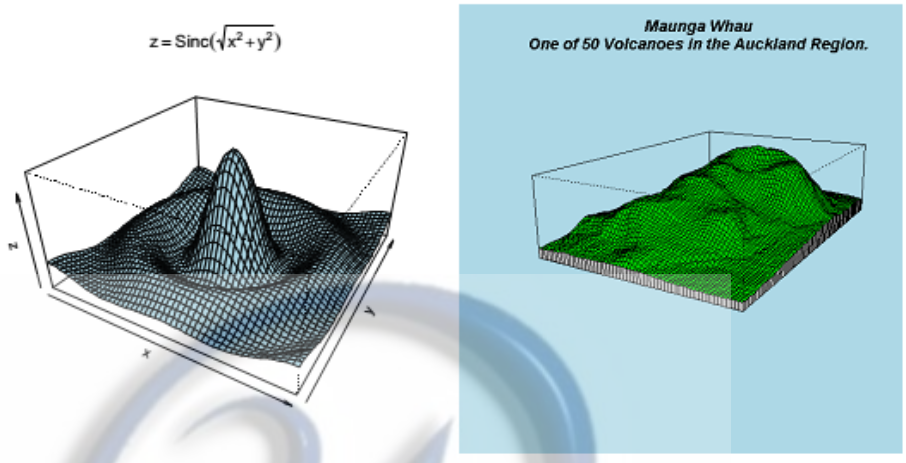
1. Efektif dalam pengelolaan data dan fasilitas penyimpanan. Ukuran *file* yang disimpan jauh lebih kecil dibandingkan dengan *software* yang lain.
2. Lengkap dalam operator perhitungan *array.*
3. Lengkap dan terdiri dari koleksi *tools* statistik yang terintegrasi untuk melakukan analisis data, dimulai dari statistik deskriptif, fungsi probabilitas, berbagai macam uji statistik, hingga *time series*.
4. Tampilan grafik yang menarik dan fleksibel ataupun *custimized*.
5. Dapat dikembangkan sesuai dengan keperluan dan kebutuhan data yang sifatnya terbuka, setiap orang dapat menambahkan fitur-fitur tambahan dalam bentuk sebuah *package* ke dalam *software* R.

Selain kelebihan dan kelengkapan fitur-fiturnya, hal terpenting lainnya yaitu, R bersifat *multiplatform*. R dapat dipasang dan digunakan baik pada sistem operasi Windows, UNIX/LINUX maupun pada Macintosh. Untuk dua sistem operasi yang disebutkan terakhir diperlukan sedikit penyesuaian. Selain itu R didukung oleh komunitas yang secara aktif saling berinteraksi satu sama lain melalui Internet dan didukung oleh *manual* atau R-help yang menyatu pada *software* R.

2.8.4 R. Riset dan Akademis

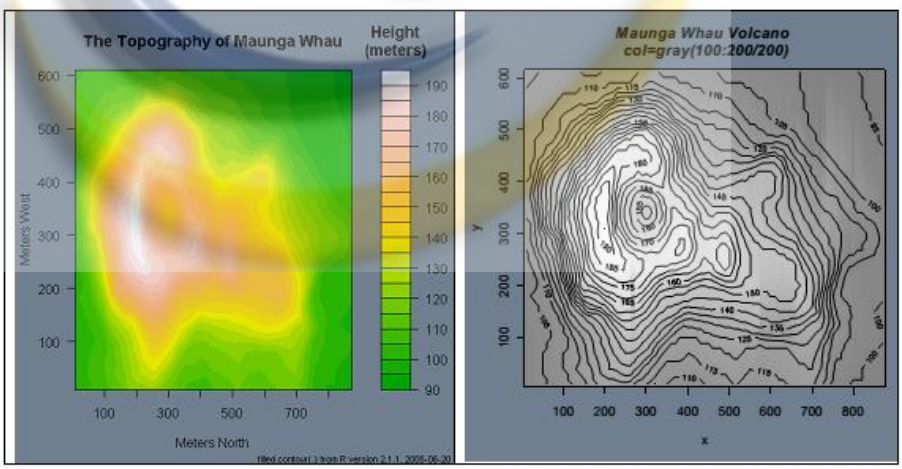
Software R sangat cocok untuk kegiatan riset, baik itu statistik, ekonomi, komputasi numerik dan pemrograman komputer. Karena didukung oleh banyak tenaga ahli dibidangnya, R layak dijadikan suatu perangkat lunak acuan oleh berbagai kalangan, terlebih di kalangan akademik (dosen dan mahasiswa). Selain itu R memiliki fitur yang lengkap dan handal serta faktor tanggung jawab moral dan legal/hukum bukan lagi menjadi kekhawatiran dalam penggunaannya, karena dapat diperoleh secara gratis. Berikut adalah beberapa contoh beserta gambar grafik (Miftah Andriansyah, 2005) yang terdapat di dalam R sebagai acuan implementasi pada :

1. Pemodelan matematis (seperti *software* MATLAB) dalam membentuk perspektif, cocok untuk jurusan teknik arsitek, sipil, mesin, dan ilmu komputer (pencitraan).



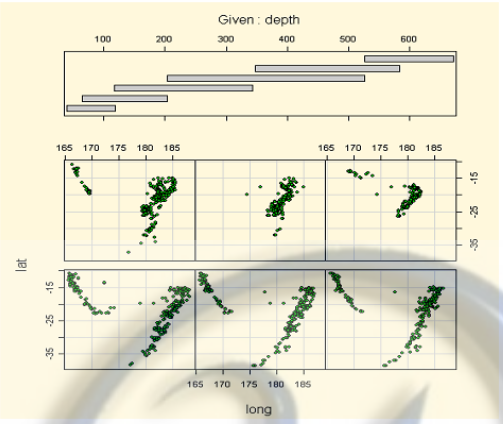
Gambar 2.5 Grafik Multidimensi

2. Pencitraan dan analisis kontur, cocok untuk jurusan geografi dan sejenisnya.

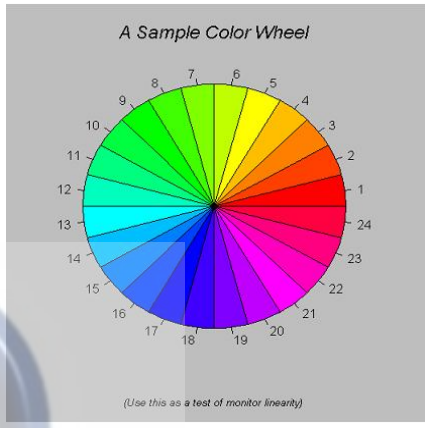


Gambar 2.6 Grafik Kontur dan Tofografi

3. Proses analisis data statistik, dengan tampilan grafik *plot* yang *customized* dan grafik fungsi idensitas yang dapat diparalelkan dengan *histogram*. Cocok untuk bidang statistika, ekonomi, dan lain-lain.



Gambar 2.7 Grafik *Scatter Plot*



Gambar 2.8 Grafik *Pie Chart*