Laporan Kegiatan
Karya Tulis Pengabdian Kepada Masyarakat
Yang Tidak Dipublikasikan

Statistika Deskriptif menggunakan R pada Google Colab untuk Ilmu Komputer

Oleh:
Irwan Prasetya Gunawan

Program Studi Informatika
Fakultas Teknik dan Ilmu Komputer
Universitas Bakrie
Jakarta
2021

HALAMAN PENGESAHAN

1. **Judul Karya Tulis** : Statistika Deskriptif menggunakan R pada Google Colab untuk Ilmu Komputer

2. Ketua Tim Pengusul
   a. Nama Lengkap : Irwan Prasetya Gunawan
   b. NIDN : 0301107306
   c. Pangkat/Golongan : III/C
   d. Jabatan Fungsional : Lektor
   e. Telp/Email : 021-5261448/irwan.gunawan@bakrie.ac.id

3. Anggota Tim Pengusul Kegiatan
   a. Dosen : –
      –
   b. Praktisi : –

4. Peserta
   a. Mahasiswa : –
   b. Alumni : –

5. Biaya Kegiatan
   a. Universitas Bakrie : –
   b. Sumber lain : –

6. Tahun Pelaksanaan : Semester Genap TA 2020/2021

Mengetahui,
Kaprodi Informatika

Prof. Dr. Hoga Saragih
NIDN. 0315087601

Bandung, 1 September 2021
Ketua Tim Pengusul

Irwan Prasetya Gunawan, Ph.D
NIDN. 0301107306

Mengetahui,
Kepala LPkM Universitas Bakrie

Ardiansyah, Ph.D
NIDN. 0318107501

Abstrak

Tulisan ini merupakan paparan pengantar bagi penggunaan bahasa R untuk analisis statistika deskriptif dengan menggunakan platform Google Colab. Ilmu statistika modern tidak bisa lepas dari aplikasi komputer sebagai alat bantu esensial dalam pengolahan data. Fokus tulisan kali ini adalah statistika deskriptif yang sering kali digunakan sebagai langkah awal bagi analisis lebih lanjut terhadap data. Dalam era big data seperti saat ini, maka tidak hanya volume data yang jumlahnya sangat signifikan, tetapi juga metode untuk analisisnya. Metode komputasi untuk analisis data akan menjadi elemen yang sangat penting dalam ilmu statistika modern. Bahasa R sebagai bahasa pemrograman yang dikembangkan untuk statistik sangat cocok untuk kebutuhan komputasi ini. Pada umumnya, penggunaan bahasa R dilakukan dalam lingkungan perangkat lunak aplikasi khusus, yang beberapa di antaranya membutuhkan proses instalasi yang cukup kompleks. Namun, saat ini kita bisa memanfaatkan platform komputasi berbasis *cloud* Google Colab untuk pemrograman bahasa R ini. Keuntungan dari pendekatan yang dipaparkan dalam tulisan ini adalah caranya yang sangat praktis karena pengguna tidak perlu melakukan proses unduh aplikasi serta instalasinya. Selain itu, fungsi-fungsi bawaan standar yang sudah disediakan untuk kernel bahasa R di Google Colab sudah lebih dari cukup untuk kebutuhan berbagai macam analisis statistik, khususnya statistika deskriptif sebagaimana yang ditunjukkan dalam tulisan ini.

Statistika Deskriptif menggunakan R pada Google Colab untuk Ilmu Komputer

Daftar Isi

1 Pendahuluan

1.1 Latar Belakang

Perkembangan ilmu komputer saat ini dapat dirasakan hampir di semua bidang lain, termasuk di antaranya pada ilmu statistik. Kontribusi ilmu komputer pada ilmu statistik bisa dilihat misalnya pada implementasi komputasi untuk metode statistik. Pada saat ini, penggunaan komputasi untuk bisa mendapatkan wawasan serta pengetahuan yang cukup mendalam dari berbagai macam data membawa kita pada ranah ilmu pengolahan data yang memanfaatkan metode statistik. Seiring dengan bertambahnya jumlah serta volume data yang bisa kita dapatkan, maka pengolahan dengan menggunakan teknologi informasi menjadi satu keharusan. Jika dalam beberapa tahun ke belakang istilah *big data* selalu dikonotasikan dengan ukuran data yang terlalu besar untuk disimpan, diolah, dan dianalisis dengan perangkat lunak basis data konvensional, maka *big data* ini juga tidak lepas dari bagaimana kita bisa melakukan lebih banyak analisis terhadap data yang ada. Jelas bagaimana peranan ilmu statistik akan sangat penting di sini. Dan jelas pula bahwa seiring dengan volume data yang semakin besar, metode komputasi untuk ilmu statistik akan sangat dibutuhkan.

Perangkat lunak untuk pengolahan data yang sudah tersedia saat ini secara komersial antara lain SPSS, Minitab, S-Plus, dan SAS. Sementara itu, perangkat lunak yang bersifat open source untuk kebutuhan yang sama bisa kita dapatkan pada R. R sebagai bahasa pemrograman dikembangkan dari bahasa S yang merupakan bahasa pemrograman statistik yang didesain dan dikembangkan pertama kali di pertengahan dekade 1970-an di Bell Laboratories. Pada saat itu, komputasi statistik masih banyak menggunakan bahasa Fortran; namun, bahasa S menawarkan pendekatan yang lebih interaktif daripada Fortran. Menjelang akhir dekade 80-an, sudah banyak perubahan yang dibuat terhadap bahasa S ini sehingga bahasa S yang baru sering disebut sebagai 'New S'. Bahasa 'New S' inilah yang sangat mirip dengan bahasa yang digunakan pada R saat ini.

Bahasa R itu sendiri dikembangkan pertama kali oleh Ross Ihaka dan Robert Gentleman dari Universitas Auckland pada pertengahan dekade 90-an. Bahasa S dibangun dari bahasa C, Fortran, dan bahasa S itu sendiri dan tersedia dengan lisensi GNU General Public License. Oleh karena itu juga, bahasa R bisa dianggap sebagai salah satu GNU package yang tersedia secara bebas (*open source*).

Keuntungan dari sifat *open source* bahasa R adalah banyaknya kontribusi yang bisa didapatkan dari pengguna bahasa ini untuk bisa dimanfaatkan secara umum. Ini terbukti dari banyaknya pustaka (*library*) yang bisa digunakan pada R yang jumlahnya mencapai ribuan pustaka yang tersedia saat ini secara online.

Secara konvensional, untuk bisa menggunakan R, maka kita membutuhkan program R terlebih dahulu yang mesti kita install di platform komputer yang kita gunakan. Program R ini bisa kita unduh dari https://cran.r-project.org/. Kemudian, untuk mempermudah penggunaannya, maka kita bisa menggunakan beberapa alternatif antar muka seperti R-Console, R-Commander, atau RGui editor lainnya. Bahasa R ini juga bisa kita install melalui Anaconda bersama dengan bahasa python, dan bahkan penggunaannya bisa dilakukan secara integrasi dengan bahasa python ini. Alternatif terakhir inilah yang akan menjadi titik tolak kita dalam menggunakan bahasa R kali ini.

1.2 Menggunakan R pada Google Colab

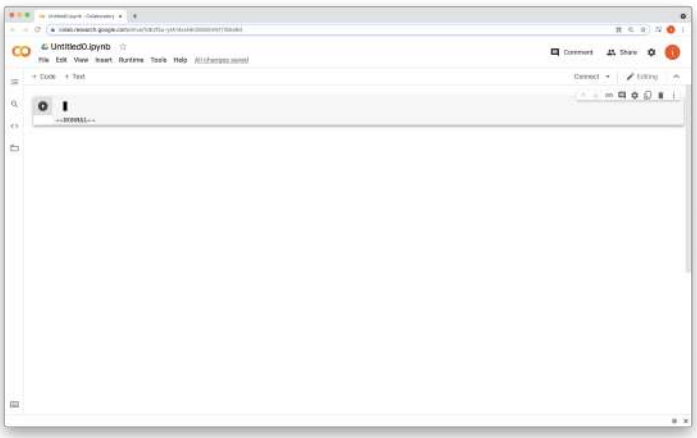
Saat ini, kita bisa menggunakan R secara mudah tanpa harus melakukan proses instalasi pada komputer yang kita gunakan. Metode alternatif ini memanfaatkan platform Google Colab yang tersedia secara online dan bisa digunakan tanpa biaya sama sekali. Google Colab (kepanjangan dari "Colaboratory") adalah produk dari Google Research yang dibangun berdasarkan Jupyter untuk menulis dan mengeksekusi kode program yang ditulis dalam bahasa python. Jupyter adalah aplikasi berbasis web yang sifatnya open source tempat kita bisa membuat 'notebook', yaitu dokumen yang berisi kode program python yang terintegrasi dengan komponen teks seperti paragraf, persamaan matematika, gambar, tautan web, dan sebagainya. Python itu sendiri merupakan bahasa pemrograman umum yang bisa digunakan untuk berbagai macam tujuan seperti pengembangan web, kontrol, automasi, serta *data science*.

Untuk bisa menggunakan R pada Google Colab maka kita hanya membutuhkan koneksi Internet dan browser. Browser versi terakhir yang sudah diuji untuk bisa digunakan pada Google Colab antara lain Chrome, Firefox, dan Safari, walaupun secara umum hampir semua browser yang terhubung dengan Internet tentunya akan bisa digunakan untuk Google Colab ini.

Tautan yang mesti kita akses untuk bisa menggunakan kernel bahasa R pada Google Colab adalah

`http://colab.to/r`

Di sini kita dibawa ke halaman kosong dengan judul halaman (judul notebook) standar `Untitled0.ipynb`. Judul ini bisa kita ubah dengan nama apapun dengan cara mengeklik nama judul halaman tersebut, dan mengedit nama halaman/notebook sesuai dengan keinginan kita.

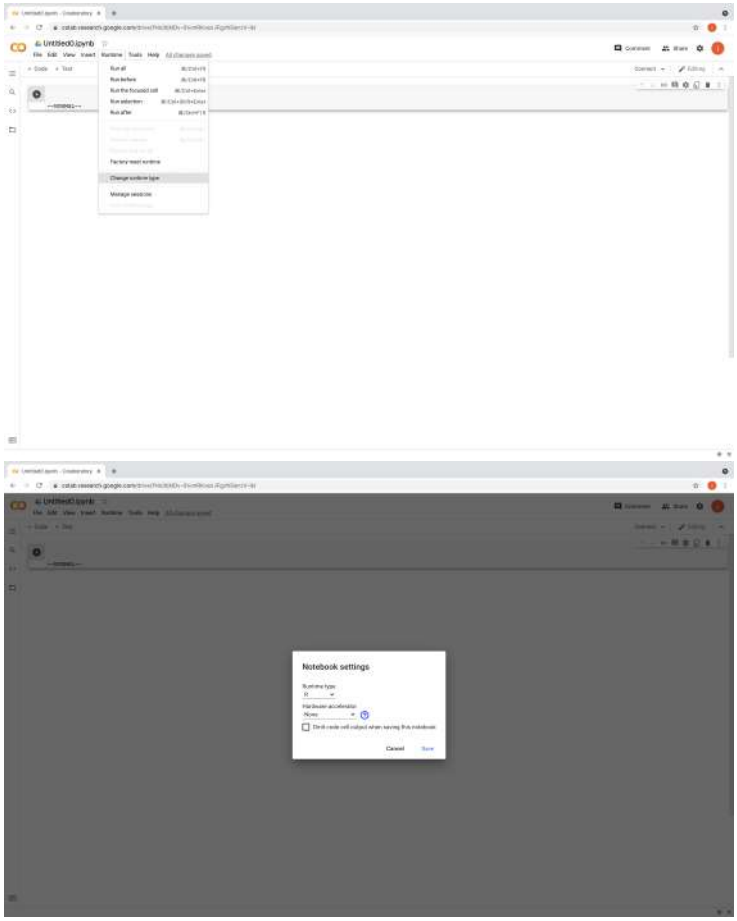


Di halaman kosong ini juga sudah disediakan sebuah sel kosong yang bisa kita isi langsung dengan kode program dalam bahasa R. Sel kode program pada Google Colab ditandai dengan *button* segitiga yang apabila kita klik akan mengeksekusi kode program yang kita tulis.

Selain kode program, kita juga bisa menuliskan komponen teks, paragraf, tabel, persamaan matematika, dan sebagainya pada sel `Text`. Untuk menambahkan sel `Text` pada notebook ini, kita bisa klik *button* + `Text` yang ada di kiri atas. Setelah itu, kita *double click* sel teks ini untuk menambahkan teks yang kita inginkan.

Dengan mengakses tautan untuk membuka Google Colab di atas, maka kita secara langsung sudah mendapatkan kernel R pada notebook yang kita tulis. Ini bisa kita konfirmasi dengan meng-

ecek menu `Runtime >> Change runtime type`; apabila kernel R sudah siap untuk kita gunakan, maka akan terlihat bahwa setting notebook yang kita pilih menggunakan runtime tipe R (seperti yang kita inginkan).



1.3 Dataset R

Dengan mengaktifkan kernel R pada Google Colab seperti yang dijelaskan sebelumnya, maka kita secara otomatis sudah mendapatkan akses ke sekumpulan pustaka data (dataset) standar yang bisa kita gunakan. Salah satu satu contoh dataset ini adalah dataset `iris` sebagai berikut:

[1]:
```
str(iris)
```

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1
1 ...
```

Hasil eksekusi perintahnya akan memberikan kita keterangan mengenai salah satu satu dataset yang bisa kita gunakan pada R di Google Colab ini. Selain dataset ini, berbagai macam dataset

yang tersedia secara default ketika kita mengaktifkan R di Google Colab bisa kita lihat dengan menggunakan perintah sebagai berikut:

[2]:
```
data()
```

Contoh isi data untuk dataset `AirPassengers` bisa kita lihat dengan perintah-perintah berikut ini:

[3]:
```
data(AirPassengers)
AirPassengers
```

A Time Series: 12 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1949	112	118	132	129	121	135	148	148	136	119	104	118
1950	115	126	141	135	125	149	170	170	158	133	114	140
1951	145	150	178	163	172	178	199	199	184	162	146	166
1952	171	180	193	181	183	218	230	242	209	191	172	194
1953	196	196	236	235	229	243	264	272	237	211	180	201
1954	204	188	235	227	234	264	302	293	259	229	203	229
1955	242	233	267	269	270	315	364	347	312	274	237	278
1956	284	277	317	313	318	374	413	405	355	306	271	306
1957	315	301	356	348	355	422	465	467	404	347	305	336
1958	340	318	362	348	363	435	491	505	404	359	310	337
1959	360	342	406	396	420	472	548	559	463	407	362	405
1960	417	391	419	461	472	535	622	606	508	461	390	432

1.4 Library/Pustaka pada R

Penggunaan R sangat terbantu dengan berbagai macam library atau pustaka yang bisa kita gunakan. Untuk menggunakan pustaka ini, maka kita gunakan perintah seperti `install.package()` yang disusul dengan perintah `library()`, dengan nama pustaka (atau package/paket) sebagai argumen dari perintah-perintah tersebut.

Contoh, jika kita ingin menggunakan variabel acak diskrit, maka kita bisa menggunakan paket `discreteRV` yang kita aktifkan dengan rangkaian perintah berikut:

[4]:
```
install.packages("discreteRV")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

[5]:
```
library(discreteRV)
```

```
Attaching package: ‘discreteRV’


The following object is masked from ‘package:base’:

    %in%
```

Dari output/keluaran kode program yang diberikan di atas, kita bisa melihat bahwa ketika kita memasang paket `discreteRV`, kernel R secara otomatis akan memasang paket yang dibutuhkan oleh paket `discreteRV` ini, yaitu paket `plyr`.

Dari sini, kita bisa menggunakan fungsi yang tersedia pada paket `discreteRV` untuk membuat sebuah variabel acak sebagai berikut:

[6]:
```
(X <- RV(0:1, probs = c(.2, 0.8)))
```

```
Random variable with 2 outcomes

Outcomes   0   1
Probs    1/5 4/5
```

Yang baru saja kita lakukan adalah membuat sebuah variabel acak dengan dua keluaran (outcomes) yang masing-masing memiliki probabilitas kemunculan $\frac{1}{5}$ dan $\frac{4}{5}$.

Dari beberapa metode pengecekan singkat di atas, kita sudah memastikan bahwa notebook pada Google Colab yang kita buat sudah siap untuk kita edit dengan menggunakan R.

2 Dasar-dasar R

Pada bagian ini akan diberikan sedikit penjelasan mengenai dasar-dasar penulisan program dalam bahasa R secara sederhana.

2.1 Pemrograman Interaktif

Secara umum, bahasa R digunakan secara interaktif dengan hasil perhitungan yang bisa kita lihat secara langsung. Cara pengunaannya sangat mirip dengan menggunakan konsol interaktif Matlab/Octave. Namun, dengan adanya fasilitas notebook, maka kode program interaktif ini bisa kita gabung dengan catatan/dokumentasi non-kode program, seperti halnya pada MathCad.

Sebagai contoh, kita bisa menggunakan R pada Google Colab seperti halnya kita menggunakan kalkulator, dengan cara mengetikkan ekspresi matematika yang ingin kita hitung secara langsung seperti berikut:

[7]:
```
2 + 5 # penjumlahan
```

7

[8]:
```
# Contoh perkalian
2 * 5
```

10

[9]:
```
# Contoh pemangkatan
2**5
```

32

Sebagai catatan, artikel yang ditulis ini juga menggunakan notebook Google Colab. Namun, untuk keperluan publikasi, maka notebook ini kemudian diekspor sebagai file PDF setelah sebelumnya diedit layout-nya terlebih dahulu dengan menggunakan LaTeX.

2.2 Objek

Bahasa R itu sendiri merupakan bahasa pemrograman yang bersifat OOP (*object oriented programming*), sehingga setiap variabel, fungsi, keluaran, dan sebagainya akan disimpan dalam bentuk objek yang memiliki atribut/sifat tertentu. Kita bisa memanipulasi objek-objek ini dengan menggunakan operator ataupun fungsi untuk mendapatkan objek lain dengan karakteristik tertentu.

Kita bisa mengeset variabel tertentu dengan menggunakan <- atau tanda sama dengan = sebagai berikut:

$$y = 2^5$$

[10]:
```
y <- 2**5
```

[11]:
```
y
```

32

$$m = 2^3$$

[12]:
```
(m = 2**3)
```

8

$$n = \log_2 2^3$$

[13]:
```
(n = log2(2**3))
```

3

Di salah satu contoh di atas, kita menggunakan fungsi `log2`, yaitu fungsi logaritma basis 2.

Kita juga bisa mendefinisikan vektor sebagai berikut:

[14]:
```
(v <- c(5, 7, 3, 8, 9))
```

1. 5 2. 7 3. 3 4. 8 5. 9

[15]:
```
v + 1
```

1. 6 2. 8 3. 4 4. 9 5. 10

Terlihat bahwa operator penjumlahan pada vektor akan diterapkan pada semua elemen vektor pada contoh di atas.

Ringkasan (summary) pada data yang kita simpan dalam vektor bisa kita dapatkan sebagai berikut:

[16]:
```
summary(v)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    3.0     5.0     7.0     6.4     8.0     9.0 
```

Dari fungsi `summary` ini kita bisa mendapatkan informasi yang cukup lengkap mulai dari nilai minimum, maksimum, median, rata-rata (mean), serta nilai quartil pertama dan ketiga.

Nilai rata-rata serta median juga bisa kita peroleh secara langsung sebagai berikut:

[17]:
```
mean(v)
```

6.4

[18]:
```
median(v)
```

7

Sebagai tambahan, varians dari data yang kita simpan dalam bentuk vektor kita dapatkan dengan perintah `var` berikut:

[19]:
```
var(v)
```

5.8

2.3 Dataframe

Selain data dalam bentuk vektor, kita juga bisa mengumpulkan atau menyusun data dalam bentuk `dataframe` seperti halnya kita menyusun data dalam bentuk tabel.

[20]:
```
Students = data.frame(
  Name = c("Berkah", "Guson", "Hoga", "Iwan", "Yusuf"),              # Kolom nama
  Program = c("Networking", "Web", "CDMA", "Mobile", "Database"),    # Kolom␣
 ↪program
  EDOM = c(3.77, 3.65, 3.70, 3.55, 3.75)                             # Kolom EDOM
)
```

Untuk menampilkan isi dataframe ini, kita hanya perlu menuliskan nama dataframe-nya saja seperti berikut:

[21]:
```
Students
```

A data.frame: 5 × 3

Name <chr>	Program <chr>	EDOM <dbl>
Berkah	Networking	3.77
Guson	Web	3.65
Hoga	CDMA	3.70
Iwan	Mobile	3.55
Yusuf	Database	3.75

Atau, jika hanya sebagian datanya yang akan ditampilkan, kita bisa gunakan fungsi `head` atau `tail`:

[22]:
```
tail(Students)
```

A data.frame: 5 × 3

	Name <chr>	Program <chr>	EDOM <dbl>
1	Berkah	Networking	3.77
2	Guson	Web	3.65
3	Hoga	CDMA	3.70
4	Iwan	Mobile	3.55
5	Yusuf	Database	3.75

Untuk dataframe yang kita gunakan, karena jumlah barisnya hanya sedikit, maka tampilan `head`, `tail`, dan tampilan data keseluruhan tidak berbeda. Perbedaan akan sangat terasa jika kita menggunakan data yang ukurannya cukup besar (misalnya seperti contoh di atas pada dataset `iris`):

[23]:
```
dim(iris); nrow(iris); ncol(iris)      # mencari dimensi dataset iris, jumlah␣
↪baris, dan jumlah kolom
```

1. 150 2. 5

150

5

[24]:
```
head(iris); tail(iris)
```

A data.frame: 6 × 5

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

A data.frame: 6 × 5

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
145	6.7	3.3	5.7	2.5	virginica
146	6.7	3.0	5.2	2.3	virginica
147	6.3	2.5	5.0	1.9	virginica
148	6.5	3.0	5.2	2.0	virginica
149	6.2	3.4	5.4	2.3	virginica
150	5.9	3.0	5.1	1.8	virginica

Untuk dataframe ini kita juga bisa menerapkan fungsi `summary` untuk melihat berbagai macam ringkasan statistik yang ada pada dataframe tersebut.

[25]:
```
summary(Students)
```

```
     Name              Program              EDOM
```

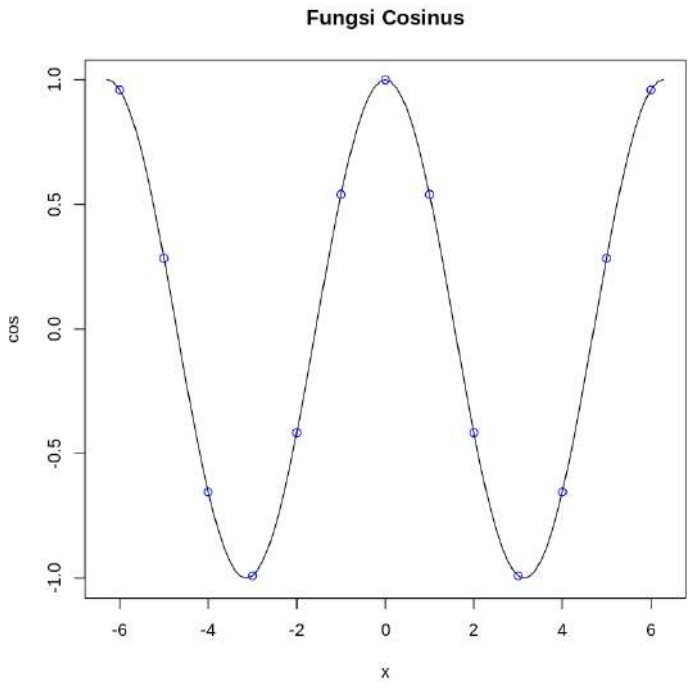
```
Length:5           Length:5           Min.   :3.550
Class :character   Class :character   1st Qu.:3.650
Mode  :character   Mode  :character   Median :3.700
                                      Mean   :3.684
                                      3rd Qu.:3.750
                                      Max.   :3.770
```

Kita bisa menggunakan dataframe ini sebagai salah satu metode pengisian data yang akan kita analisis atau olah dengan R pada Google Colab.

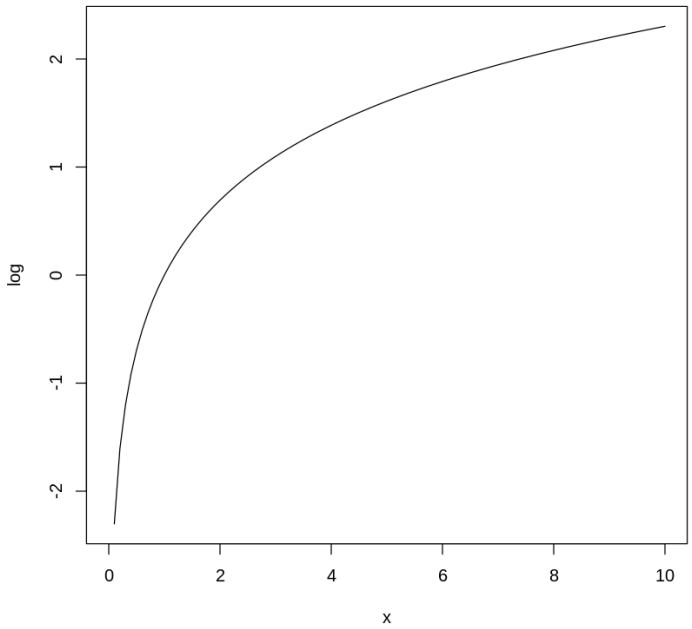
2.4 Grafik

Salah satu fasilitas lain pada R yang banyak digunakan adalah fasilitas tampilan grafik. Sebagai contoh, berikut ini adalah tampilan grafik sebuah fungsi sinusoidal:

[26]:
```
x <- -6:6
plot(cos, -2*pi, 2*pi, main = "Fungsi Cosinus")
points(x, cos(x), col="blue")
```



[27]:
```
plot(log, 0, 10)
```



3 Statistika Deskriptif

3.1 Metode Bergambar dan Tabular dalam Statistika Deskriptif

Bagian ini menjelaskan teknik untuk *memvisualisasikan* distribusi data dengan variabel tunggal. Visualisasi adalah langkah pertama yang penting dalam analisis statistik, karena metode visualisasi kadang kala bisa mengungkapkan *pola* yang sulit dijelaskan jika hanya dengan menggunakan angka. Selanjutnya, metode visualisasi ini *dapat memberikan wawasan* mengenai *prosedur statistik* yang sesuai yang akan dilakukan untuk analisis datanya.

Beberapa istilah penting dalam proses visualisasi data ini antara lain:

- **Distribusi**: menjelaskan
  - nilai yang dimiliki oleh variabel
  - seberapa sering nilai ini muncul
- **Ukuran sampel**: jumlah pengamatan dalam kumpulan data.
  - Untuk menyatakan nilai dari variabel kumpulan data, kita sering menggunakan notasi $x_1, x_2, ..., x_n$, dengan x_i adalah pengamatan ke-i dari kumpulan data
  - Kemudian, secara umum, data *tidak* dianggap terurut, kecuali dinyatakan lain.

3.1.1 Plot Batang-dan-Daun (Stem-and-Leaf)

Visualisasi data pertama adalah plot **stem-and-leaf** (batang-dan-daun). Plot ini dibuat menggunakan langkah-langkah berikut:

1. Pilih jumlah *digit depan* untuk menjadi nilai **stem**. Digit yang tersisa adalah nilai **leaf**.
2. Gambarkan *garis vertikal* dan buat daftar nilai batang di sebelah kiri garis ini, secara berurutan.
3. *Catat* daun dari setiap pengamatan pada baris yang sesuai dengan nilai batangnya. (Komputer sering mengurutkan nilai daun, tetapi jika dilakukan dengan tangan, ini tidak perlu.)
4. Sebagai informasi tambahan, tunjukkan *satuan* batang dan daun. (Misalnya, batang mulai di tempat puluhan, dan daun mulai di tempat satuan.)

Perintah/fungsi dasar pada R untuk penggambaran diagram batang-dan-daun: `stem`

Contoh 1: Stem-and-Leaf Plot

[28]:
```
tinggi_badan <- c(5.55, 5.30, 5.63, 5.30, 5.13,
                  5.05, 5.38, 5.96, 5.21, 5.38)
```

Tabulasi/penggambaran secara manual:

Stem	Leaf
55	5
53	0088
56	3
51	3
50	5
59	6
52	1

Stem	Leaf
50	5
51	3
52	1
53	0088
54	
55	5
56	3
57	
58	
59	6

[29]:
```
stem(tinggi_badan, scale = 2)
```

```
The decimal point is 1 digit(s) to the left of the |

50 | 5
51 | 3
```

```
52 | 1
53 | 0088
54 |
55 | 5
56 | 3
57 |
58 |
59 | 6
```

[30]:
```
stem(tinggi_badan)
```

```
The decimal point is 1 digit(s) to the left of the |

50 | 53
52 | 10088
54 | 5
56 | 3
58 | 6
```

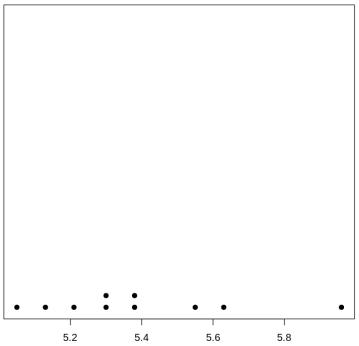
3.1.2 Dotplot

Grafik **dotplot** mewakili setiap titik data sebagai titik di sepanjang garis bilangan nyata, dengan menempatkan titik pada garis sesuai dengan nilainya. Jika dua titik hampir tumpang tindih, maka penggambarannya akan ditumpuk.

Perintah dasar untuk membuat diagram **dotplot**: `stripchart`

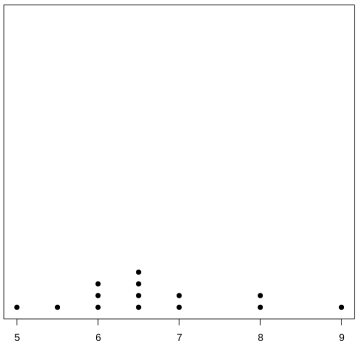
Contoh 2: Dotplot Kita gambarkan data `tinggi_badan` menggunakan dotplot, sebagai perbandingan dengan diagram batang-dan-daun di bagian sebelum ini.

[31]:
```
stripchart(tinggi_badan, method = "stack", pch = 19, offset = 0.9, at = 0, ylim␣
↪= c(0,1))
```



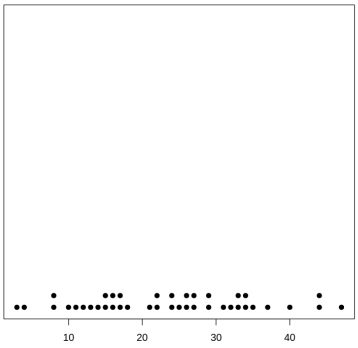
Sebagai contoh lain, kita buat data baru dalam bentuk vektor (dengan nama variabel `waktu_tidur`), kemudian kita gambarkan diagram dotplot-nya.

[32]:
```
waktu_tidur <- c(5, 5.5, 6, 6, 6, 6.5, 6.5, 6.5, 6.5, 7, 7, 8, 8, 9)
stripchart(waktu_tidur, method = "stack", pch = 19, offset = 0.9, at = 0, ylim =␣
↪c(0,1))
```



Contoh lain dengan variabel `waktu_recovery` yang kita gambarkan diagram dotplot dan diagram batang-dan-daunnya sebagai perbandingan.

[33]:
```
waktu_recovery <- c(3, 4, 11, 15, 16, 17, 22, 44, 37, 16, 14, 24, 25, 15, 26,␣
↪27, 33, 29, 35, 44, 13, 21, 22, 10, 12, 8, 40, 32, 26, 27, 31, 34, 29, 17, 8,␣
↪24, 18, 47, 33, 34)
stripchart(waktu_recovery, method = "stack", pch = 19, offset = 0.9, at = 0,␣
↪ylim = c(0,1))
```



[34]:
```
stem(waktu_recovery)
```

```
The decimal point is 1 digit(s) to the right of the |

0 | 34
0 | 88
1 | 01234
1 | 5566778
2 | 12244
2 | 5667799
3 | 123344
3 | 57
4 | 044
4 | 7
```

3.1.3 Table

Penggambaran data dalam bentuk tabel bisa kita lakukan langsung dari data yang kita input dalam bentuk vektor.

Contoh 3: Table Data berikut adalah vektor berisi nilai score yang didapat oleh sebuah tim sepak bola.

[35]:
```
score <- c(9, 6, 5, 5, 5, 6, 2, 8, 3, 4, 8, 1)
```

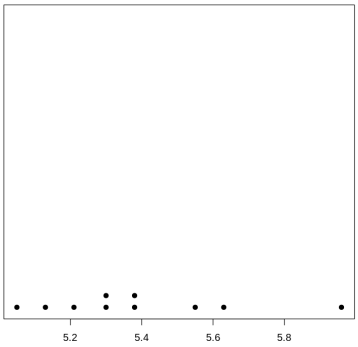
Distribusi frekuensi dari dataset ini adalah:

[36]:
```
table(score)
```

```
score
1 2 3 4 5 6 8 9
1 1 1 1 3 2 2 1
```

Example 4: Table Dengan menggunakan data `tinggi_badan` dalam Contoh 1, kita membuat distribusi frekuensi datanya sebagai berikut:

[37]:
```
stripchart(tinggi_badan, method = "stack", pch = 19, offset = 0.9, at = 0, ylim␣
↪= c(0,1))
```



[38]:
```
table(tinggi_badan)
```

```
tinggi_badan
5.05 5.13 5.21  5.3 5.38 5.55 5.63 5.96
   1    1    1    2    2    1    1    1
```

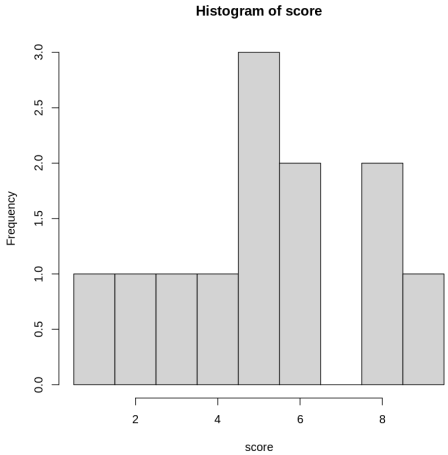
3.1.4 Histogram

Selain informasi distribusi frekuensi seperti yang kita lakukan sebelumnya dengan fungsi `table`, kita pun dapat menggunakan **histogram** untuk memvisualisasikan distribusi data kuantitatif ini.

Fungsi dasar yang digunakan: `hist`

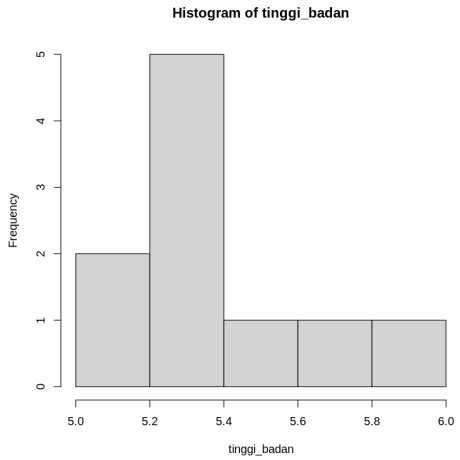
Contoh 5: Histogram Kita akan membuat histogram untuk dataset pada Contoh 3 (dataset score sepak bola).

[39]:
```
hist(score, breaks = min(score):max(score + 1) - 0.5)
```



Contoh 6: Histogram Histogram untuk dataset yang diberikan di Contoh 1 (dataset `tinggi_badan`) diberikan di bawah ini.

[40]:
```
hist(tinggi_badan)
```



Ketika kita melihat penggambaran distribusi data seperti pada histogram di atas, maka kita bisa mencari beberapa hal yang memberikan ciri khusus data tersebut:

- Apakah datanya **unimodal** (hanya memiliki satu "puncak")?
- Apakah datanya **bimodal** atau **multimodal** (banyak "puncak")?
- Apakah datanya **positif-skew**, **negatif-skew**, atau **simetris**?
- Apakah ada **pencilan**, yaitu titik yang jauh dari data lainnya?
- Seberapa tersebar datanya?

3.1.5 Plot Batang (Bar Plot)

Bar plot adalah metode untuk memvisualisasikan data kategorikal (atau yang sering disebut sebagai data **kualitatif**).

Fungsi dasar yang digunakan: `barplot`

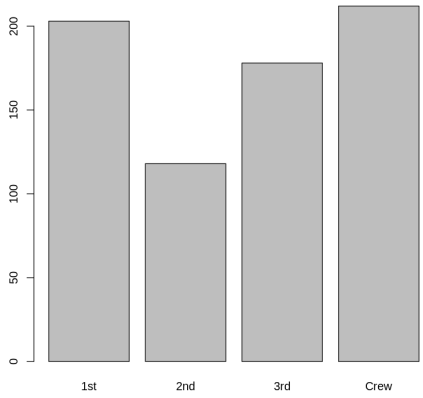
Contoh 7: Plot batang Di bawah ini adalah dataset yang menunjukkan frekuensi kelas penumpang kapal *Titanic* yang selamat dari tenggelamnya kapal tersebut.

[41]:
```
(t_kelas_penyintas <- apply(Titanic[, , , 2], 1, sum))
```

1st 203 **2nd** 118 **3rd** 178 **Crew** 212

Berikut ini adalah penggambaran plot batangnya:

[42]:
```
barplot(t_kelas_penyintas)
```



3.2 Ukuran Lokasi (Measures of Location)

Meskipun ringkasan visual data cukup bagus dan memberikan gambaran mengenai data yang kita miliki, ringkasan kuantitatif masih tetap dibutuhkan untuk dapat memberikan gambaran yang lebih lengkap mengenai kumpulan data ini. Di sini kita mulai dengan **measures of location**, yang memberi tahu kita di mana letak set data di sepanjang garis bilangan.

3.2.1 Rata-rata Sampel

Ukuran lokasi pertama dan paling umum untuk sampel adalah **rata-rata sampel**, yang ditentukan untuk kumpulan data $x_1, ..., x_n$ di bawah ini:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

Interpretasi fisik dari rata-rata sampel kira-kira sebagai berikut: jika kita menggambarkan data sebagai titik (dot) dan menganggap titik ini sebagai objek fisik, dengan bobot sebagai salah satu atribut dari titik ini dan garis bilangan sebagai sebuah jungkat-jungkit, maka nilai rata-rata sampel akan menjadi lokasi di mana jungkat-jungkit ini akan menjadi setimbang.

Contoh 8: Rata-rata sampel Di sini kita akan menghitung rata-rata jumlah poin yang dicetak oleh tim sepak bola yang diberikan di Contoh 3

[43]:
```
score
```

1. 9 2. 6 3. 5 4. 5 5. 5 6. 6 7. 2 8. 8 9. 3 10. 4 11. 8 12. 1

[44]:
```
mean(score)
```

5.16666666666667

3.2.2 Median/nilai tengah sampel

Misalkan $r_1, r_2, ..., r_n$ adalah kumpulan data *terurut* yang sesuai dengan kumpulan data $x_1, ..., x_n$, sehingga $r_1 \leq r_2 \leq ... \leq r_n$, maka **median sampel** adalah angka yang membagi set data ini menjadi dua. Notasi dari median diberikan di bawah ini sebagai:

$$\tilde{x}$$

Interpretasi fisik/geometris dari median cukup jelas; ketika kita mengatur data secara berurutan, kita bisa membagi datanya ke dalam dua bagian yang jumlahnya sama.

Contoh 9: Median sampel Nilai tengah atau median dari nilai score tim sepak bola yang sebelumnya diberikan di Contoh 3 adalah sebagai berikut:

[45]:
```
score
```

1. 9 2. 6 3. 5 4. 5 5. 5 6. 6 7. 2 8. 8 9. 3 10. 4 11. 8 12. 1

[46]:
```
sort(score)    # mengurutkan data
```

1. 1 2. 2 3. 3 4. 4 5. 5 6. 5 7. 5 8. 6 9. 6 10. 8 11. 8 12. 9

[47]:
```
median(score)
```

5

3.2.3 Kuartil

Misalkan $\alpha \in [0, 1]$, maka $\alpha \times 100$**th persentil** adalah angka sedemikian rupa sehingga kurang lebih $\alpha \times 100\%$ dari data dalam $r_1, ..., r_n$ terletak di sebelah kiri angka tersebut.

- Contoh persentil yang paling umum adalah **kuartil**.
  - **Kuartil pertama** adalah persentil ke-25, dan **kuartil ketiga** adalah persentil ke-75.
  - **Kuartil kedua** adalah median (persentil ke-50).
  - Kuartil 0 dan 4 adalah minimum dan maksimum dari dataset.
  - Semua kuartil bersama-sama membentuk **ringkasan lima angka** dari kumpulan data

Contoh 10: Kuartil Kuartil pertama, kedua, dan ketiga untuk nilai score sepak bola diberikan sebagai berikut:

[48]:
```
quantile(score, c(0.25, 0.5, 0.75))
```

25\% 3.75 **50\%** 5 **75\%** 6.5

Bandingkan dengan summary data yang sama menggunakan fungsi `summary` sebagai berikut:

[49]:
```
summary(score)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   3.750   5.000   5.167   6.500   9.000
```

Contoh 11: Persentil Kita akan mencari persentil ke-10 dan ke-90 dari data tinggi badan pada Contoh 1.

[50]:
```
tinggi_badan
```

1. 5.55 2. 5.3 3. 5.63 4. 5.3 5. 5.13 6. 5.05 7. 5.38 8. 5.96 9. 5.21 10. 5.38

Sebagai ilustrasi, kita bisa urutkan terlebih dahulu datanya sebagai berikut:

[51]:
```
sort(tinggi_badan)
```

1. 5.05 2. 5.13 3. 5.21 4. 5.3 5. 5.3 6. 5.38 7. 5.38 8. 5.55 9. 5.63 10. 5.96

[52]:
```
quantile(tinggi_badan, c(0.1, 0.9))
```

10\% 5.122 **90\%** 5.663

Ada 10 data tinggi badan yang kita miliki, maka jelas bahwa persentil ke-10 dan ke-90 adalah data pertama (yaitu, sebelum data yang kedua) dan data ke sembilan.

3.2.4 Pencilan

Rata-rata sampel $\bar{x}$ bersifat **sensitif** terhadap pencilan; maksudnya, adanya pencilan dalam kumpulan data dapat memiliki efek mendalam pada rata-rata sampel. Di sisi lain, median sampel $\tilde{x}$ **tidak sensitif** terhadap keberadaan pencilan ini, karena pencilan hampir tidak pernah mengubah nilai median sampel.

Contoh 12: Pencilan Kita akan membandingkan nilai rata-rata sampel dan nilai tengah sampel dari nilai score tim sepak bola jika kita tambahkan sebuah nilai lain yang bersifat pencilan terhadap nilai score yang sudah ada. Pencilan ini disimulasikan dengan nilai score ke-12 yang bervariasi sebagai berikut:

[53]:
```
(pencilan <- c(score[12], score[12] + 3, score[12] * 2, max(score) * 2))
```

1. 1 2. 4 3. 2 4. 18

[54]:
```
score_tab <- sapply(pencilan, function(g) {
  dat <- c(score[1:11], g)
  return(c(g, median(dat), mean(dat)))
})
score_tab
score_tab <- t(score_tab) # transpos data agar lebih mudah untuk diaca
## Penamaan baris/kolom
rownames(score_tab) <- 1:nrow(score_tab)
colnames(score_tab) <- c("Nilai Pencilan", "Median", "Mean")
round(score_tab, digits = 2)
```

A matrix: 3 × 4 of type dbl	1.000000	4.000000	2.00	18.000000
	5.000000	5.000000	5.00	5.500000
	5.166667	5.416667	5.25	6.583333

A matrix: 4 × 3 of type dbl

	Nilai Pencilan	Median	Mean
1	1	5.0	5.17
2	4	5.0	5.42
3	2	5.0	5.25
4	18	5.5	6.58

3.2.5 Rata-rata dipangkas

Dari contoh yang diberikan sebelum ini, kita bisa melihat bahwa median lebih disukai untuk data yang **skew** (condong) sementara nilai rata-rata lebih disukai untuk data yang sifatnya simetris.

Kompromi antara sensitivitas nilai rata-rata terhadap pencilan dan nilai median yang tidak bergeming terhadap adanya pencilan dalam semua kumpulan data adalah **rata-rata yang dipangkas**, yang dilambangkan dengan $\bar{x}_{\text{tr}(100\alpha)}$. Rata-rata yang dipangkas adalah rata-rata data ketika $100\alpha\%$ data dihapus (atau diabaikan) dari setiap akhir kumpulan data.

Catatan: Ada kemungkinan bahwa kita tidak bisa menghapus secara tepat $100\alpha\%$ data yang ada; jika ini tidak bisa kita lakukan, maka kita bisa membuat perkiraan dengan melakukan interpolasi.

Contoh 13: Rata-rata terpangkas Kita akan mencari nilai rata-rata terpangkas $\bar{x}_{\text{tr}(10)}$ untuk data tinggi badan pada Contoh 1.

[55]:
```
sort(tinggi_badan)
```

1. 5.05 2. 5.13 3. 5.21 4. 5.3 5. 5.3 6. 5.38 7. 5.38 8. 5.55 9. 5.63 10. 5.96

[56]:
```
quantile(tinggi_badan, c(0.1, 0.9))
```

10\% 5.122 **90\%** 5.663

[57]:
```
mean(tinggi_badan, trim = 0.1)
```

5.36

Nilai rata-rata terpangkas dari data `tinggi_badan` bisa kita dapatkan dengan menambahkan argumen `trim` pada fungsi `mean` yang sebelumnya sudah pernah kita pakai.

Pemangkasan ini pun bisa kita lakukan secara manual dengan terlebih dahulu mengurutkan datanya lalu mengabaikan bagian penghujung dari kumpulan datanya (dalam hal ini, 10% atau 0.1 bagian dari ujung data yang kita miliki). Hal ini ditunjukkan sebagai berikut:

[58]:
```
sort(tinggi_badan)[2:9]
```

1. 5.13 2. 5.21 3. 5.3 4. 5.3 5. 5.38 6. 5.38 7. 5.55 8. 5.63

[59]:
```
mean(sort(tinggi_badan)[2:9])
```

5.36

Terlihat bahwa setelah kita mengabaikan data-data pada ujung awal dan akhir (yaitu data pertama dan data kesepuluh), maka perhitungan nilai rata-ratanya sama dengan perhitungan sebelumnya (menggunakan argumen tambahan `trim`).

3.3 Ukuran Variabilitas

Perhatikan tiga kumpulan data berikut:

1	2	3
4	2	1
5	5	3
6	6	6
7	7	9
8	10	11

Kita bisa membuat plot titik untuk setiap kumpulan data ini, lalu hitung rata-rata dan median dari kumpulan datanya masing-masing.

Sekarang kita anggap bahwa setiap kumpulan data ini mewakili waktu tunggu (dalam menit) untuk jalur busway yang akan membawa kita pulang ke rumah setiap malam setelah bekerja. Pertanyaannya adalah: kumpulan data mana yang lebih kita sukai? Mengapa?

Contoh di atas menggambarkan bahwa ukuran pusat (seperti nilai rata-rata dan nilai tengah) masih belum cukup untuk menggambarkan kumpulan data secara lengkap. Kita juga menginginkan adanya **ukuran variabilitas**, yang menjelaskan bagaimana kumpulan data ini "tersebar".

Ukuran penyebaran ini bisa kita dasarkan pada **deviasi**. Deviasi dari sebuah titik data i adalah $x_i - \bar{x}$. Kemudian nilai deviasi ini mesti kita hitung untuk semua titik data yang kita miliki, lalu kita akumulasikan untuk menggambarkan sebaran datanya, misalnya $\sum_{i=1}^{n}(x_i - \bar{x})$.

3.3.1 Varians sampel dan simpangan baku sampel

Hasil di atas menunjukkan bahwa kita harus mengukur variabilitas dengan sesuatu yang lain. Ukuran paling umum untuk variabilitas adalah **variasi sampel** dan **standar deviasi sampel**.

Standar deviasi sampel secara kasar dapat diartikan sebagai deviasi "tipikal" dari titik data dari rata-rata.

Rata-rata sampel sensitif terhadap outlier. Simpangan baku sampel *lebih* sensitif terhadap outlier daripada rata-rata sampel.

Idealnya kita harus menggunakan perangkat lunak atau kalkulator untuk menghitung varians sampel, tetapi perhitungan secara praktis diberikan oleh persamaan berikut:

$$S_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2 = \sum_{i=1}^{n} x_i^2 - n\bar{x}^2$$

3.3.2 Varians populasi dan simpangan baku populasi

Ada padanan populasi untuk kedua besaran ini, yaitu **varians populasi**, σ^2, dan **deviasi standar populasi**, $\sigma = \sqrt{\sigma^2}$.

Contoh 14 Kita akan menghitung varians sampel dan deviasi standar sampel dari skor pertandingan sepak bola yang diberikan di Contoh 1.

[60]:
```
score
```

1. 9 2. 6 3. 5 4. 5 5. 5 6. 6 7. 2 8. 8 9. 3 10. 4 11. 8 12. 1

[61]:
```
length(score)
```

12

[62]:
```
summary(score)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   3.750   5.000   5.167   6.500   9.000 
```

[63]:
```
# varians sampel
var(score)
```

5.96969696969697

[64]:
```
# standar deviasi sampel
sd(score)
```

2.44329633276379

3.3.3 Sebaran keempat (rentang antar-kuartil)

Sebaran keempat (juga dikenal sebagai **rentang antar-kuartil (IQR)**) adalah kuartil ketiga dikurangi kuartil pertama; kita gunakan notasi f_s untuk rentang antar-kuartil ini. Ini adalah ukuran lain dari dispersi atau sebaran data.

Fungsi dasar yang digunakan: `IQR`

Contoh 15 Berikut adalah sebaran keempat untuk skor pertandingan sepak bola yang diberikan pada Contoh 1; pertama, kita hitung dengan cara langsung (menggunakan fungsi `IQR`), lalu kita bandingkan dengan perhitungan tidak langsung menggunakan selisih antara kuartil ketiga dan pertama.

[65]:
```
IQR(score)
```

2.75

[66]:
```
quantile(score, 0.75) - quantile(score, 0.25)
```

75\%: 2.75

3.3.4 Boxplot

Boxplot adalah perangkat statistik yang memvisualisasikan sebaran kumpulan data. Dengan boxplot, kita bisa melihat bagaimana titik-titik data tersebar satu sama lain, misalnya seberapa jauh titik-titik ekstrim seperti nilai maksimum dan minimum dengan nilai tengah, kuartil pertama, kuartil ketiga, atau nilai rata-rata, dan seberapa jauh pencilan (jika ada). Dengan boxplot juga kita bisa membandingkan sebaran kumpulan data yang satu dengan kumpulan data lainnya (komparatif).

Pada R, titik pencilan akan digambarkan dengan titik, secara default.

Contoh 16 Dataset berikut berisi pertumbuhan gigi marmut yang diberi vitamin C melalui jus jeruk pada tiga tingkat dosis yang berbeda.

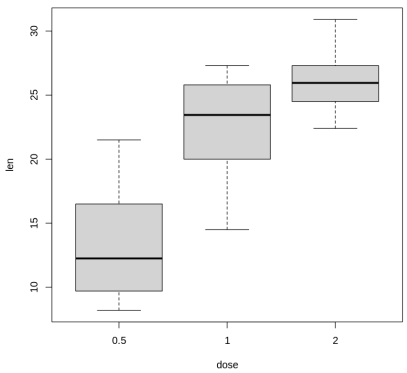
[67]:
```
suppressPackageStartupMessages(library(dplyr))  # Provides %>% operator

OJ <- ToothGrowth %>% filter(supp == "OJ") %>% select(len, dose) %>% unstack %>%
  lapply(sort) %>% as.data.frame
names(OJ) <- c(0.5, 1, 2)
OJ
```

A data.frame: 10 × 3

0.5 <dbl>	1 <dbl>	2 <dbl>
8.2	14.5	22.4
9.4	19.7	23.0
9.7	20.0	24.5
9.7	21.2	24.8
10.0	23.3	25.5
14.5	23.6	26.4
15.2	25.2	26.4
16.5	25.8	27.3
17.6	26.4	29.4
21.5	27.3	30.9

[68]:
```
boxplot(len ~ dose, data = ToothGrowth %>% filter(supp == "OJ"))
```



4 Kesimpulan

Analisis visual data berdasarkan statistika deskriptif dengan menggunakan R pada platform Google Colab sangat praktis untuk dilakukan karena kita tidak perlu melakukan proses unduh serta instalasi programnya terlebih dahulu. Fungsi-fungsi bawaan standar pada kernel R di platform Google Colab sudah tersedia dan sudah lebih dari cukup untuk melakukan analisis statistik deskriptif. Selain itu, dengan penggunaan fasilitas `notebook` pada Google Colab ini maka integrasi antara penulisan kode program serta dokumentasi tekstual sangat mudah untuk dilakukan.

5 Sumber

- https://cran.r-project.org/
- https://rseek.org/
- http://colab.to/r
- https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/00Index.html
- https://en.wikipedia.org/wiki/R_(programming_language)
- https://en.wikipedia.org/wiki/S_(programming_language)
- https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/index.html