MATERI 11

ANALISIS DATA

Beberapa fungsi untuk perhitungan statistik: rentang data, maksimum/minimum, rata-rata, deviasi, jumlah kumulatif, dan sebagainya. Di MATLAB fungsi fungsi statistik semacam ini telah ada dan bisa digunakan secara fleksibel.Misal **x** dan **y** adalah vektor (baris ataupun kolom), dan **A** dan **B** sebagai matriks m×n.

MAKSIMUM DAN MINIMUM

Nilai maksimum dan minimum diperoleh dengan **command** berikut ini:

max(x)	menghitung nilai maksimum dari elemen vektor **x**. Jika **x** bernilai kompleks maka dihitung **max(abs(x))**
max(A)	menghitung nilai maksimum dari setiap kolom di matriks **A**; hasilnya berupa vektor 1×n
max(max(A))	menghitung nilai maksimum dari elemen matriks **A**
max(A,B)	menghitung matriks berukuran sama dengan **A** dan **B** dengan elemen berisi nilai terbesar di antara elemen **A** dan **B** pada posisi yang sama
min(...)	sama dengan sintaks **max(...)** di atas, tetapi untuk mencari minimum

JUMLAH DAN PRODUK

Beberapa jenis operasi penjumlahan bisa dilakukan dengan *command* **sum** dan **cumsum**.

sum(x)	menjumlahkan nilai elemen vektor **x**
sum(A)	menjumlahkan nilai elemen dari setiap kolom di matriks **A**; hasilnya berupa vektor 1×n
sum(sum(A))	menjumlahkan nilai semua elemen matriks **A**
cumsum(x)	menghitung vektor berukuran sama dengan **x** berisi jumlah kumulatif elemen **x**; yaitu elemen kedua ialah jumlah dari elemen pertama dan kedua dari **x**, dan seterusnya
cumsum(A)	menghitung matriks berukuran sama dengan **A** di mana kolom-kolomnya merupakan jumlah kumulatif dari kolom di **A**

Sebagai contoh, kita akan definisikan vektor **y** dan matriks **B** sebagai berikut:

```
y =[1 4 9 16 25]
B= [7 8 9;4 5 6;1 2 3]
   B= 7 8 9
      4 5 6
      1 2 3
>> y=[1:5].^2;
>> B=[1:3 ; 4:6 ; 7:9];
>> jml_y = sum(y)
      jml_y =
            55
>> jml_B = sum(B)
      jml_B =
            12 15 18
>> total_B = sum(sum(B))
      total_B =
            45
>> kumulasi_y = cumsum(y)
      kumulasi_y =
            1 5 14 30 55
>> kumulasi_B = cumsum(B)
```

```
kumulasi_B =
       1  2  3
       5  7  9
       12  15  18
```

Sementara itu, produk (perkalian elemen-elemen) vektor dan matriks bisa diperoleh dengan cara yang mirip.

prod(x)	mengalikan nilai elemen vektor **x** mengalikan nilai elemen dari setiap kolom di matriks **A**; hasilnya berupa vektor 1×n
prod(A)	mengalikan nilai semua elemen matriks **A**
prod(prod(A))	menghitung vektor berukuran sama dengan **x**
cumprod(x)	berisi produk kumulatif elemen **x**; yaitu elemen kedua ialah perkalian dari elemen pertama dan kedua dari **x**, dan seterusnya
cumprod(A)	menghitung matriks berukuran sama dengan **A** di mana kolom-kolomnya merupakan produk kumulatif dari kolom di **A** Sebagai contoh kita gunakan vektor **y** dan matriks **B** seperti sebelumnya

```
>> pdk_y = prod(y)
       pdk_y =
              14400
>> pdk_B = prod(B)
       pdk_B =
              28  80  162
>> tot_pdk_B = prod(prod(B))
       tot_pdk_B =
              362880
>> kumulasi_pdk_y = cumprod(y)
       kumulasi_pdk_y =
              1  4  36  576  14400
>> kumulasi_pdk_B = cumprod(B)
       kumulasi_pdk_B =
              1   2   3
              4   10  18
              28  80  162
```

STATISTIKA

Command untuk analisis data statistik.

mean(x)	menghitung rata-rata aritmatik dari elemen vektor **x**
mean(A)	menghitung rata-rata aritmatik dari elemen setiap kolom di matriks **A**; hasilnya berupa vektor 1×n
median(...)	menghitung median (nilai tengah)
std(...)	menghitung deviasi standar (simpangan baku)
var(...)	menghitung variansi

Contoh :

```
>> x=[175 177 173 165 160 170 174 177 168 170];
>> A=[3.3 2.8 3.3; 3.9 4.0 3.8; 3.8 3.5 2.9; 2.9 3.2 3.1];
>> rataan_IP_sem = mean(A)
       rataan_IP_sem =
              3.4750  3.3750  3.2750
>> rataan_IP_mhs = mean(A')
       rataan_IP_mhs =
              3.1333  3.9000  3.4000  3.0667
>> rataan_IP_total = mean(mean(A))
       rataan_IP_total =
              3.3750
>> nilai_tengah = median(x), deviasi = std(x),variansi = var(x)
       nilai_tengah =
              171.5000
       deviasi =
```

```
        5.4661
variansi =
        29.8778
```

SORTIR

Unutk mengurutkan data (sortir) di MATLAB dengan *command* berikut ini:

sort(x)	menghitung vektor dengan elemen **x** telah tersortir secara *ascending* (dari kecil ke besar). Jika **x** bernilai kompleks maka dihitung **sort(abs(x))**
[y,ind] = sort(x)	menghitung vektor **y** berisi sortiran elemen **x**, dan vektor **ind** berisi indeks sehingga **y = x(ind)**
[B,Ind] = sort(A)	menghitung matriks **B** berisi sortiran kolom kolom matriks **A**.

Contoh dengan menggunakan data sebelumnya, diurutkan data tinggi badan dari kecil ke besar (ascending).

```
>> sort(x)
        ans =
        160 165 168 170 170 173 174 175 177 177
```

Atau diurutkan disertai indeks yang menunjukkan nomor urut elemen pada vektor **x** sebelum disortir.

```
>> [y,ind]=sort(x)
        y =
        160 165 168 170 170 173 174 175 177 177
        ind =
        5 4 9 6 10 3 7 1 2 8
```

Untuk mengurutkan dari besar ke kecil (descending).

```
>> fliplr(sort(x))
        ans =
        177 177 175 174 173 170 170 168 165 160
```

Demikian pula untuk mengurutkan elemen matriks: secara *ascending* pada kolom per kolom:

```
>> sort(A)
        ans =
        2.9000 2.8000 2.9000
        3.3000 3.2000 3.1000
        3.8000 3.5000 3.3000
        3.9000 4.0000 3.8000
```

Atau secara descending pada kolom per kolom:

```
>> flipud(sort(A))
        ans =
        3.9000 4.0000 3.8000
        3.8000 3.5000 3.3000
        3.3000 3.2000 3.1000
        2.9000 2.8000 2.9000
```

Ataupun melakukan sortir dengan indeks. Perhatikan bahwa kolom-kolom dalam **IND** berisi nomor urut elemen pada matriks **A** sebelum disortir.

```
>> [Y,IND]=sort(A)
        Y =
        2.9000 2.8000 2.9000
        3.3000 3.2000 3.1000
        3.8000 3.5000 3.3000
        3.9000 4.0000 3.8000
        IND =
        4 1 3
        1 4 4
        3 3 1
        2 2 2
```