Laboratorium 2

Statystyka

Funkcje statystyczne są dostępne w pomocy: **help datafun**.

Zestawy danych zapisujemy w formie wektorów o n wartościach: $x = [x_1, x_2, \cdots, x_n]$. W Matlabie jest on reprezentowany jako wierszowy wektor x.

Funkcje statystyczne

(1) **min** i **max**

Obie te funkcje zwracają też indeks najmniejszej lub największej wartości. Jeśli jest więcej niż jedno wystąpienie, zwraca to pierwsze. Proszę wypróbować komendę:

```
>> x = [9 10 10 9 8 7 3 10 9 8 5 10];
>> [maxval, maxind] = max(x)
```

W przypadku macierzy, funkcje **min** i **max** operują domyślnie na kolumnach.

Proszę sprawdzić, co zwraca:

```
>> mat = [9 10 17 5; 19 9 11 14]
mat =
9 10 17 5
19 9 11 14
>> [minval, minind] = min(mat)
```

Te funkcje porównują też pary wektorów danych. Co robi poniższe polecenie?

```
>> x = [3 5 8 2 11];
>> y = [2 6 4 5 10];
>> min(x,y)
```

(2) **mean**

To średnia arytmetyczna. Tu nieoczywisty jest przypadek macierzy, gdyż funkcja **mean** działa na kolumnach. Trzeba więc podać jako drugi argument, po przecinku, wymiar 2, aby znaleźć średnią w każdym wierszu.

```
>> mat = [8 9 3; 10 2 3; 6 10 9]
mat =
       8 9 3
      10 2 3
      6 10 9
>> mean(mat)
ans =
      8 7 5
>> mean(mat,2)
ans =
```

```
        6.6667
        5.0000
        8.3333
```

(3) **var**, **std** Czyli wariancja i odchylenie standardowe. Obie funkcje sa wbudowane.

```
>> shortx = [2 5 1 4];
>> myvar = var(shortx)
myvar =
        3.3333
>> sqrt(myvar)
ans =
        1.8257
>> std(shortx)
ans =
        1.8257
```

(4) **mode**

Czyli moda, najczęściej występująca wartość.

```
>> x = [9 10 10 9 8 7 3 10 9 8 5 10];
>> mode(x)
ans =
        10
```

(5) **median**

Czyli mediana, środkowa wartość, uporządkowanego wektora.

```
>> median([1 4 5 9 12 ])
>> median([9 4 1 5 12])
```

Operacje na zbiorach

Niech:

```
>> v1 = 6:-1:2
     6 5 4 3 2
>> v2 = 1:2:7
v2 =
     1 3 5 7
```

(1) **union**

Czyli mnogościowe sumowanie - zwraca wszystkie elementy z obu wektorów bez powtórzeń.

Może sortować lub nie. Co zwrócą poniższe funkcje?

```
>> union(v1,v2)
???
```

```
>> union(v1,v2, 'sorted')
???
>> union(v1,v2,'stable')
???
>> union(v2,v1,'stable')
???
```

(2) **intersect**

Zwraca przecięcie, czyli część wspólną obu wektorów.
Na przykład:

```
>> intersect(v1,v2)
ans =
      3 5
```

(3) **setdiff**

Czyli różnica zbiorów. Oczywiście, zależy od kolejności podania argumentów.

```
>> setdiff(v1,v2)
ans =
      2 4 6
>> setdiff(v2,v1)
ans =
      1 7
```

(4) **setxor**

Zwraca wektor składający się z wartości nie będących w przecięciu wektorów v_1, v_2. Jest to więc suma wyników funkcji **setdiff** z argumentami podanymi w obu kolejnościach.

```
>> setxor(v1,v2)
ans =
      1 2 4 6 7
>> union(setdiff(v1,v2), setdiff(v2,v1))
ans =
      1 2 4 6 7
```

(5) **unique**

Zwraca niepowtarzające się wartości w wektorze:

```
>> v3 = [1:5 3:6]
v3 =
       1 2 3 4 5 3 4 5 6
>> unique(v3)
ans =
       1 2 3 4 5 6
```

UWAGA 0.1. Każdą z funkcji **union, intersect, unique, setdiff, setxor** można wywołać z dodatkowym argumentem **'stable'**, by wynik był zwrócony zgodnie z wyjściowym uporządkowaniem wartości.

Funkcja **intersect** zwraca, oprócz wektora wartości wspólnych, wektory indeksów do wektorów v_1 i v_2.

```
>> [outvec, index1, index2] = intersect(v1,v2)
outvec =
          3 5
index1 =
          4 2
index2 =
          2 3
```

I dalej, możemy otrzymać:

```
>> v1(index1)
ans =
      3 5
>> v2(index2)
ans =
      3 5
```

(6) **ismember**

Dostaje jako argumenty dwa wektory i zwraca wektor prawd (1) i fałszy (0) o długości pierwszego argumentu. W wyniku mamy na n-tym miejscu 1, jeśli oba podane wektory mają w tym miejscu ten sam wyraz i 0 w pozostałych przypadkach.

```
>> v1
v1 =
    6 5 4 3 2
>> v2
v2 =
    1 3 5 7
>> ismember(v1,v2)
ans =
    0 1 0 1 0
>> ismember(v2,v1)
ans =
    0 1 1 0
```

(7) **issorted** Zwraca prawdę, czyli 1, jeśli podany w argumencie wektor jest posortowany rosnąco i 0 w pozostałych przypadkach.

```
>> v3 = [1:5 3:6]
```

```
v3 =
  1 2 3 4 5 3 4 5 6
>> issorted(v3)
ans =
  0
>> issorted(v2)
ans =
  1
```

Powyżej widać, że spacja to też znak. Jest wliczana w długość słowa zarówno przed jak i po nim.

Usuwanie elementów z wektora

Załóżmy, że mamy wektor $a = [1, 2, 3, 4, 5]$. Aby usunąć zeń wszystkie wystąpienia liczby 3, tworzymy nowy wektor b bez trójek następująco:

```
b = a(a~=3);
```

Jeśli chcemy usunąć z a trzeci element, wektor b tworzymy:

```
b = a;
b(3) = [];
```

lub w jednej linijce:

```
b = a([1:2, 4:end]);
```

Tak więc, by usunąć z wektora najmniejszy i największy element piszemy tak:

```
>> xwithbig = [9 10 10 9 8 100 7 3 10 9 8 5 10];
>> newx = xwithbig(xwithbig w= min(xwithbig) & ...
xwithbig w= max(xwithbig))
```

Proszę sprawdzić, jak to działa. Tu trzy kropki oznaczają złamanie linii - rozmieszczenie jednego polecenia na dwie linie.

Dopasowanie krzywej do danych

(1) `https://www.mathworks.com/help/matlab/data_analysis/descriptive-statistics.html`
(2) Rozkład temperatur w ciągu sześciu godzin:

```
>> x = 1:6;
>> y = [77 72 65 55 70 68];
>> plot(x,y,'bo')
```

Funkcja **polyfit** zwraca współczynniki wielomianu danego stopnia, który najlepiej pasuje do danych, używając metosy najmniejszych kwadratów.

Np. chcąc dopasować prostą do powyższych danych (czyli wieloman stopnia 1), dostaniemy:

```
>> coeff = polyfit(x, y, 1)
coeff =
   -1.7429   73.9333
```

Czyli najlepszą prostą jest $y = -1.7429x + 73.9333$.
Spróbujmy funkcji kwadratowej:

```
>> coefs = polyfit(x,y,2)
coefs =
    1.8393  -14.6179   91.1000
```

Używamy funkcji **polyval**, by wyznaczyć wartości wielomianu w zadanych punktach.

```
>> curve = polyval(coefs, x)
curve =

   78.321   69.221   63.800   62.057   63.993   69.607
```

Poniżej pełen skrypt dopasowujący kwadratową krzywą do zbioru temperatur:

```
x = 1:6;
y = [77 72 65 55 70 68];
coefs = polyfit(x,y,2);
curve = polyval(coefs,x);
plot(x,y,'ro',x,curve)
xlabel('Time')
ylabel('Temperatures')
title('Temperatures one afternoon')
axis([0 7 60 80])
```

Funkcji **polyval** nie trzeba używać do całego wektora x. Można ograniczyć się do zgodności w jednym tylko punkcie. Proszę wypróbować np.:

```
polyval(coefs,2.5)
```

Ponadto, możemy też ekstrapolować poza zbiór danych. Na przykład oszacować temperaturę o godzinie 7.

```
polyval(coefs,7)
```

Zadania

(1) Wyraź następujące wielomiany jako wektory współczynników:

$$2x^3 - 3x^2 + x^5 3x^4 + x^2 + 2x + 4$$

(2) Znajdź wartości wielomianu $3x^3 + 4x^2 + 2x - 2$ w punktach $4, 6$ i 8.
(3) Przeanalizuj działanie poniższego skryptu:

Tablica 1. My caption

Time	0	3	6	9	12	15	18	21	24
Flow rate	800	980	1090	1520	1920	1670	1440	1380	1300

```
x = 2:6;
y = [65 67 72 71 63];
morex = linspace(min(x),max(x));
for pd = 1:3
        coefs = polyfit(x,y,pd);
        curve = polyval(coefs,morex);
        subplot(1,3,pd)
        plot(x,y,'ro',morex,curve)
        xlabel('Time')
        ylabel('Temperatures')
        title(sprintf('Degree %d',pd))
        axis([1 7 60 75])
end
```

(4) Napisz skrypt, który wygenreuje 10 losowych liczb całkowitych z przedziału $[0, 100]$. Jeśli są one równo rozłożone w tym przedziale, to ułożone rosnąco, powinny wylądować na jednej prostej. Aby to sprawdzić, dopasuj prostą do wylosowanego zbioru punktów i nanieś punkty i prostą na wykresie i dodaj legendę.

(5) Przepływ wody w Mystical River zmierzony pewnego sierpniowego dnia jest pokazany w poniższej tabeli. Czas odmierzany jest w godzinach, a przepływ w stopach sześciennych na sekundę.

Napisz skrypt, w którym dopasujesz wielomiany stopnia 3 i 4 do poniższych danych. Stwórz podwykres dla obu wielomianów. Nanieś też oryginalne dane w postaci czarnych kółek. Nazwy podwykresów powinny zawierać stopnie wielomianów. Nanieś też odpowiednie nazwy dla osi x i y.