

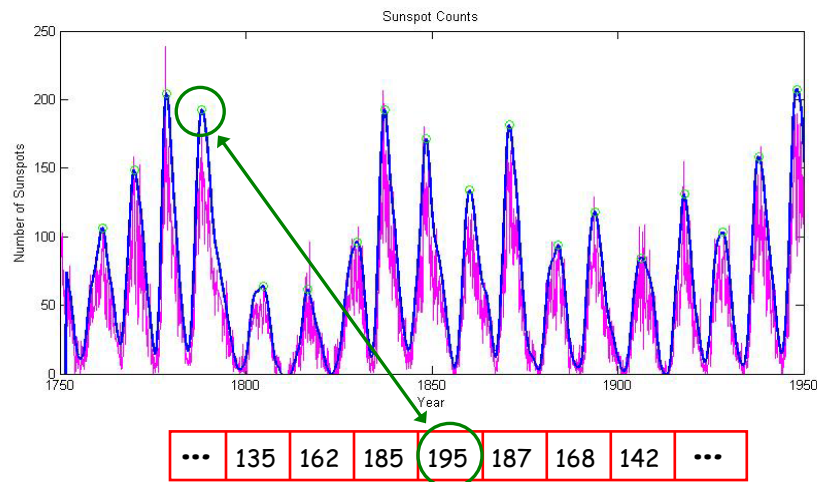
# Vectors and indexing

Tools of the trade



**CS112 Scientific Computation**  
Department of Computer Science  
Wellesley College

The length of the sunspot cycle: ~ 12 years

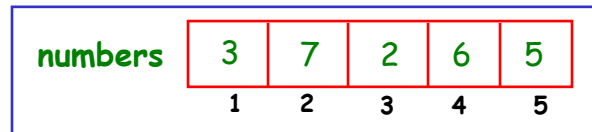


Indexing 5-2

## Indexing

Each location of a vector stores a **value** and has an **index** that specifies its position within the vector

**numbers = [3 7 2 6 5]**



The first location has index 1, and indices increment by 1 for successive locations

Indexing 5-3

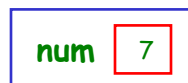
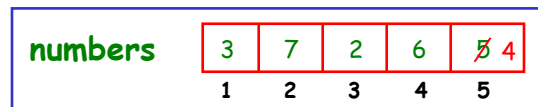
## Reading, Riting, & Rithmetic

We can **read or change** the contents of a location by referring to its index

**num = numbers(2)**

**numbers(5) = 4**

**sumNum = numbers(2) + numbers(5)**



Indexing 5-4

## The end game

The keyword **end**, when provided as an index of a vector, refers to the last index of the vector

```
sumNum = numbers(2) + numbers(5)
```

```
sumNum = numbers(2) + numbers(end)
```

<b>numbers</b>	3	7	2	6	5
	1	2	3	4	5



**Exercise:** Write a sequence of assignment statements that exchange the contents of the first and last locations of **numbers**, assuming you don't know the length of **numbers**

Indexing 5-5

## Out of bounds



An attempt to **read** the contents of a location whose *index is outside the range of indices* is **not good**

```
>> numbers = [3 7 2 6 5];
```

```
>> num = numbers(8)
```

??? Index exceeds matrix dimensions.

```
>> num = numbers(end + 1)
```

??? Index exceeds matrix dimensions.

<b>numbers</b>	3	7	2	6	5
	1	2	3	4	5

Indexing 5-6

## MATLAB is more forgiving than the emperor 😊

However, MATLAB allows you to *add a new value* at a location *beyond the range of indices*

```
>> numbers = [3 7 2 6 5];
```

```
>> numbers(6) = 9;
```

numbers	3	7	2	6	5	9
	1	2	3	4	5	6

```
>> numbers(9) = 4;
```

numbers	3	7	2	6	5	9	0	0	4
	1	2	3	4	5	6	7	8	9

Indexing 5-7

## Starting from scratch

We can create an initial vector of 0's from scratch

```
>> numbers = zeros(1,6);
```

numbers	0	0	0	0	0	0
	1	2	3	4	5	6

```
>> numbers(1) = input('first number');
```

```
>> numbers(2) = input('second number');
```

Indexing 5-8

## Referring to multiple locations

We can refer to multiple locations all at once using a vector of indices

<b>numbers</b>	3	7	2	6	5	9	0	0	4
	1	2	3	4	5	6	7	8	9

>> **newNumbers = numbers( [2 4] )**

<b>newNumbers</b>	7	6
	1	2

\* What is the value of **newNumbers** after executing:  
**newNumbers = numbers( [9 4 7 2] )** ?

Indexing 5-9

## Change contents of multiple locations

We can also change the values stored in multiple locations using a vector of indices

<b>numbers</b>	3	7	2	6	5	9	0	0	4
	1	2	3	4	5	6	7	8	9

>> **numbers( [1 3 5 7] ) = 2**

<b>numbers</b>	2	7	2	6	2	9	2	0	4
	1	2	3	4	5	6	7	8	9

Indexing 5-10

## Different locations get different values

Change multiple locations all at once, given equal length indices on the left & values on the right

numbers	3	7	2	6	5	9	0	0	4
	1	2	3	4	5	6	7	8	9

`>> numbers( [2 4 6] ) = [9 5 1]`

numbers	3	9	2	5	5	1	0	0	4
	1	2	3	4	5	6	7	8	9

Indexing 5-11

## Time-out exercise



Given the **numbers** vector,

numbers	3	9	2	5	5	1	0	0	4
	1	2	3	4	5	6	7	8	9

what will be the new contents of **numbers** after executing the following statements?

`>> numbers( [4 5] ) = numbers(8)`  
`>> numbers( [1 2 3] ) = numbers( [7 end 6] )`

Indexing 5-12

## Analyzing health data

What is the average cholesterol level for women in their twenties who exercise at least 30 minutes a day?

cholesterol	189	239	178	185	251	165	...
age	25	35	28	40	28	22	...
sex	'm'	'm'	'f'	'm'	'm'	'f'	...
exercise	30	15	40	25	15	60	...

Indexing 5-13

## Selecting vector contents with logical vectors

Suppose we want to refer to vector locations whose contents *satisfy a logical condition*\*

For example:

- (1) Change all negative numbers to 0
- (2) Store the even and odd numbers in separate vectors
- (3) Calculate the average of all numbers larger than 10



Indexing 5-14

## Selection using logical vectors

A *logical vector*, when supplied as an index of a vector, *selects locations where logical value is 1 (true)*

**Case 1:** Logical vector is given as the index of a vector *in an expression*

nums	6	-7	-2	6	-5	3	8
	1	2	3	4	5	6	7

`negNums = nums[nums < 0]`

negNums	-7	-2	-5
	1	2	3

Indexing 5-15

## Selection using logical vectors (again)

**Case 2:** Logical vector is given as the index of a vector on the *left side of an assignment*

nums	6	-7	-2	6	-5	3	8
	1	2	3	4	5	6	7

`nums[nums < 0] = 0`

nums	6	0	0	6	0	3	8
	1	2	3	4	5	6	7

\* This was task (1) *Change all negative numbers to 0*

Indexing 5-16

## Selection using logical vectors (again)<sup>2</sup>

Store even and odd numbers in separate vectors\*

<b>nums</b>	8	14	7	17	22	5	10
	1	2	3	4	5	6	7

**evenNums** =

**oddNums** =



\* Hint: **rem(a,b)** returns the remainder of dividing **a** by **b**

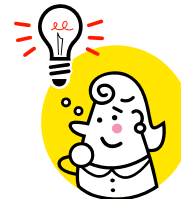
Indexing 5-17

## Selection using logical vectors (again)<sup>3</sup>

Calculate the average of all numbers larger than 10\*

<b>nums</b>	8	14	7	17	22	5	10
	1	2	3	4	5	6	7

**avgVal** =



\* Hint: The **mean()** function is your friend

Indexing 5-18

## Now for the *pièce de résistance*

What is the average cholesterol level for women in their twenties who exercise at least 30 minutes a day?

cholesterol	189	239	178	185	251	165	...
age	25	35	28	40	28	22	...
sex	'm'	'm'	'f'	'm'	'm'	'f'	...
exercise	30	15	40	25	15	60	...

Indexing 5-19