NENS 230: Analysis Techniques for Neuroscience using MATLAB

Autumn Quarter 2011, Mondays 9-10:50am, LKSC 209

Lecture 1: [Course Overview ; MATLAB Orientation]

Daniel O'Shea, Instructor	*djoshea@stanford.edu*	Office Hours: Friday, 9:30 - 11:30am, Peet's
Sergey Stavisky, Instructor	*sergey.stavisky@stanford.edu*	Office Hours: Tuesday, 9-11am, Clark W1.3
Eric Trautmann, TA	*etrautmann@stanford.edu*	Office Hours: Thursday, 9-11am

[**Course Overview** ;
MATLAB Orientation]

Course Aims

1. Become proficient in MATLAB programming
2. Understand how to learn more MATLAB as needed
3. Learn to recognize when MATLAB would help your workflow
4. Learn how to think through implementing scientific analyses programmatically
5. Gain experience in the above by programming specific analyses and visualizations commonly encountered in neurosciences

Course Outline

Weeks 1-2:	The basics of MATLAB
Week 3:	Importing and organizing data
Weeks 4-5:	Plotting data and manipulating images
Week 6:	Statistics, Regression
Week 7:	Writing better code
Weeks 8-9:	No class (SfN and Thanksgiving)
Week 10:	Class-chosen topic
Week 11:	Looking ahead: what else one can do in MATLAB

Course Structure

Lectures on Mondays, 9-10:50am, usually in LKSC 209 but occasionally in other rooms. We will try to record the lectures.

Mix of lecture and interactive on-screen walk-throughs

Some weeks there *may* be time at the end of class to get started on assignments with the course staff available to help

Lectures posted on course website

Assignments will be posted on Monday and will be due before class on the following Monday. Email assignments to nens230@gmail.com

Assignments will be graded on a 0, ✓, ✓+ basis:
0 Not submitted, or only a cursory attempt
✓ Shows substantial effort and progress, but not everything works
✓+ Submitted code does everything it's supposed to

Sample solutions to the assignments will be posted.
Look over these! They will show best practices and helpful tricks not covered in class.

If you received a ✓, use the sample solution as a guide to fix your code and resubmit the assignment within two weeks

Don't just copy our solution; fix and extend what you'd previously submitted to make it do what it's supposed to.
Resubmitted complete assignments will be bumped up to a ✓+.

Course grading is **satisfactory** or **no credit**.
If by the end of the quarter you have a ✓+ on all but one of the assignments, and do the final project, you will pass.

Useful Resources

MATLAB Help

Your classmates

Course online Q&A forum at piazza.com/stanford/fall2011/nens230
Ask questions, answer other people's questions. Course staff will also check and respond to unanswered questions.

Many MATLAB FAQs and tutorials can be found online

Course e-mail: nens230@gmail.com

Office hours:
Dan: Friday, 9:30-11:30am, Peet's Coffee, Clark 3rd Floor
Sergey: Tuesday, 9-11am, Clark W1.3 (at his desk right behind Prof. Shenoy's office, or in the adjacent "NeuroLounge" conference room)
Eric: Thursday, 9-11am, Location To Be Determined.

[Course Overview ;

MATLAB Orientation]

What is MATLAB?

A software product made by The Mathworks, Inc (Natick, MA)

Combination of:

- Programming Language
- Compiler/Interpreter
- Desktop IDE ("Integrated Development Environment")
- Graphics Environment
- Library of useful functions ("toolboxes")

Why MATLAB?

- Ubiquitous in academic science and industry research & development
- High-level and flexible programming language
- Easy to learn development environment
- Excellent documentation and learning tools
- Subject-specific toolboxes and publicly shared code save time
- Excellent built-in linear algebra great for scientific number-crunching

"**MAT**rix **LAB**oratory"

<u>Well-suited for rapid development</u>

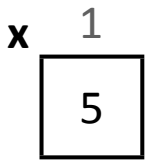
Cons:

- Proprietary
- Often slower than other languages

MATLAB Desktop Walkthrough

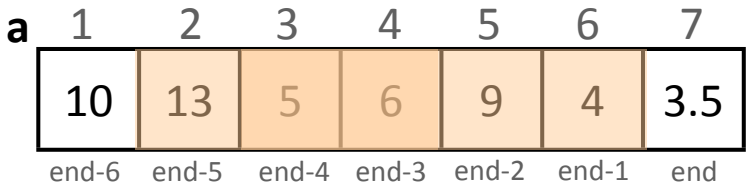
Vectors

```
x = 5
```



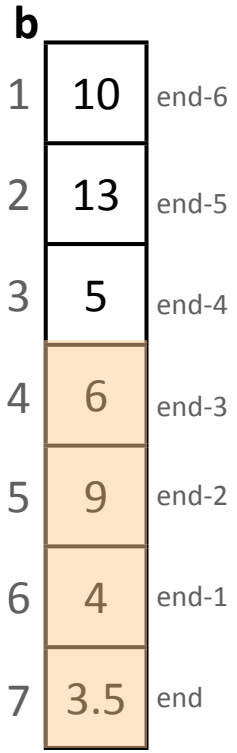
length(x) == 1

```
a = [10 13 5 6 9 4 3.5]
```



length(a) == 7

```
a(2:4)        == [13 5 6]
a(3:end-1)    == [5 6 9 4]
```



length(b) == 7

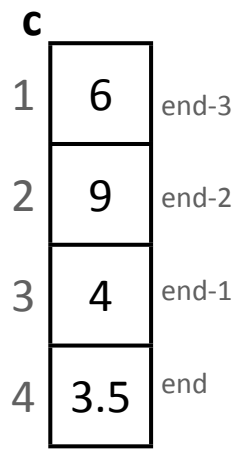
transpose

```
b = a'
```

==[10;13;5;6;9;4;3.5]

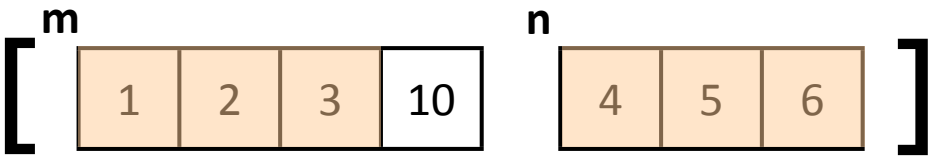
semicolon separates elements vertically

```
c = b(end-3:end)
```



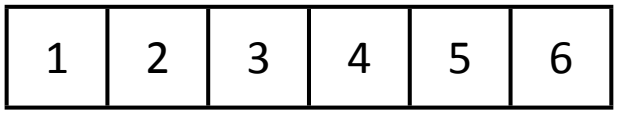
Concatenating Vectors

```
m = [1 2 3 10]
n = [4 5 6]
```



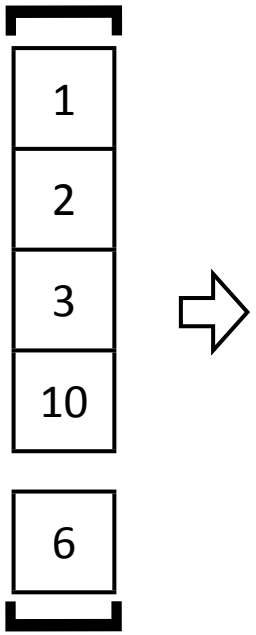

```
[m(1:3) n]
```

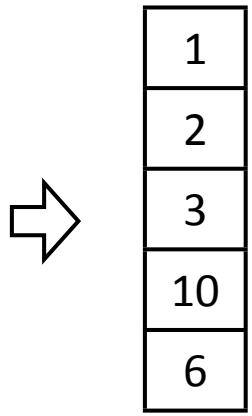
==[1 2 3 4 5 6]



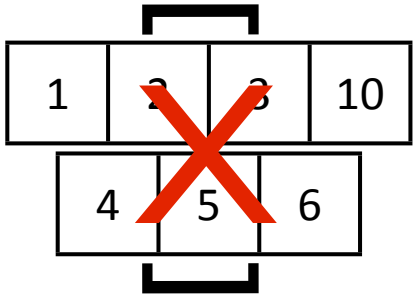

```
[m' ; n(end)' ]
```

==[1;2;3;10]






```
[m; n]
```



Excising Elements of Vectors

```
speed = [3.2 3.5 3.6 -1 -1 3.2]
```

speed	1	2	3	4	5	6
	3.2	3.5	3.6	-1	-1	3.2

length(speed) == 6

```
speed([4 5]) = []
```

==[3.2 3.5 3.6 3.2]

speed	1	2	3	4
	3.2	3.5	3.6	3.2

length(speed) == 4

input to any indexing operation is just a vector

```
keepIdx = [1 2 3 6];
speed2 = speed(keepIdx)
```

speed2 == [3.2 3.5 3.6 3.2]

```
removeIdx = [4 5];
speed2 = speed
speed2(removeIdx) = []
```

speed2 == [3.2 3.5 3.6 -1 -1 3.2]

speed2 == [3.2 3.5 3.6 3.2]

Mouse Behavior Data Example

Lecture 1 Review

Concepts

MATLAB desktop:

- **Command Window** is where you enter commands and see output
- **Current Folder** is a directory browser
- **Workspace** shows the variables currently in memory
- **Command History** shows your past commands
- **Variable Editor** lets you inspect and edit the variables in the workspace
- **Editor** lets you edit .m files such as scripts
- **Help** is your new bff

.mat data files store saved variables

.m scripts are set of commands to be executed when the script is run

.fig are saved figures that can be opened and manipulated through **plot tools**

Scripts and .mat files must be on your **path**, and subfolders must be explicitly added

Path priority works from top to bottom for files with identical names

Variables are named pieces of data; you can create, manipulate, save them

Almost all variables are matrices

Functions are the fundamental unit of computation

The same function can do different things depending on its **input**

You can **define** a variable to be equal to an existing a variable

You can define a variable to be a modified form of its current state

vectors can be indexed into using parentheses ( )

vectors and strings can be **concatenated** using square brackets [ ]

`doc topic` brings up the help page about topic

In Editor, **run** will run a whole script, or individual sections can be highlighted and run

Commands do the same thing when run from a script or from the Command Window

Functions

`load`

`=` sets LHS to RHS

`display`

`size`

`[a;b]` concatenates vertically

`[a b]` concatenates horizontally

`a(3:end-1)` indexing

`a(n) = []` excises n^{th} element

`+ - / *` arithmetic

`save`

`clear`

`clc`

`mean`

`plot`

`bar`

`hist`

`title`

`xlabel`

`ylabel`

`saveas`

`pwd`

trailing ; suppresses output