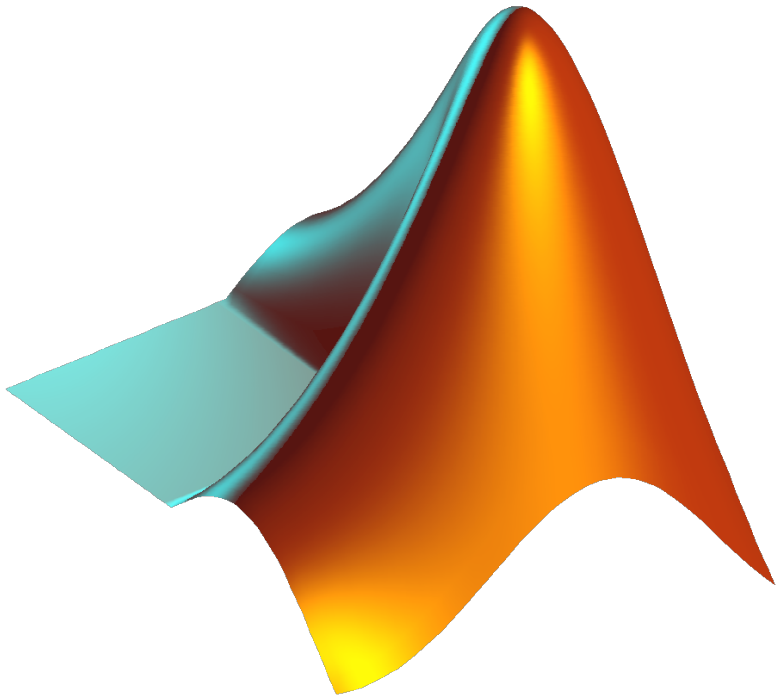
Lecture 1: Welcome to Matlab

NENS 230: Data Analysis for the Biosciences using MATLAB
Autumn 2015

Outline

1. What is Matlab and why should I care?
2. Administrative things
3. Getting started: the “integrated development environment” (IDE) or Matlab desktop
4. Matlab basics
5. Looking ahead

What is MATLAB?



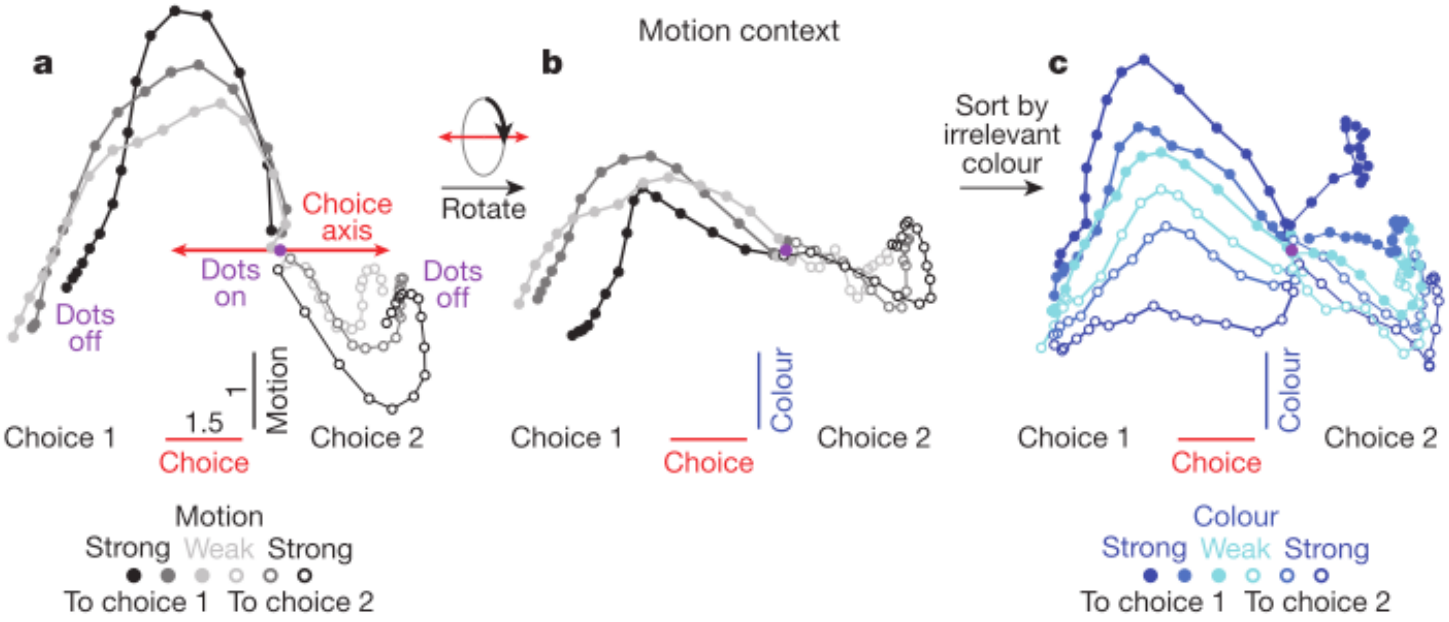
- **MAT**rix **LAB**oratory
- An general-purpose environment for doing *scientific computing*
- Allows you to acquire, process, digest, visualize, model, and communicate data
- This is done by writing code in the *MATLAB language*

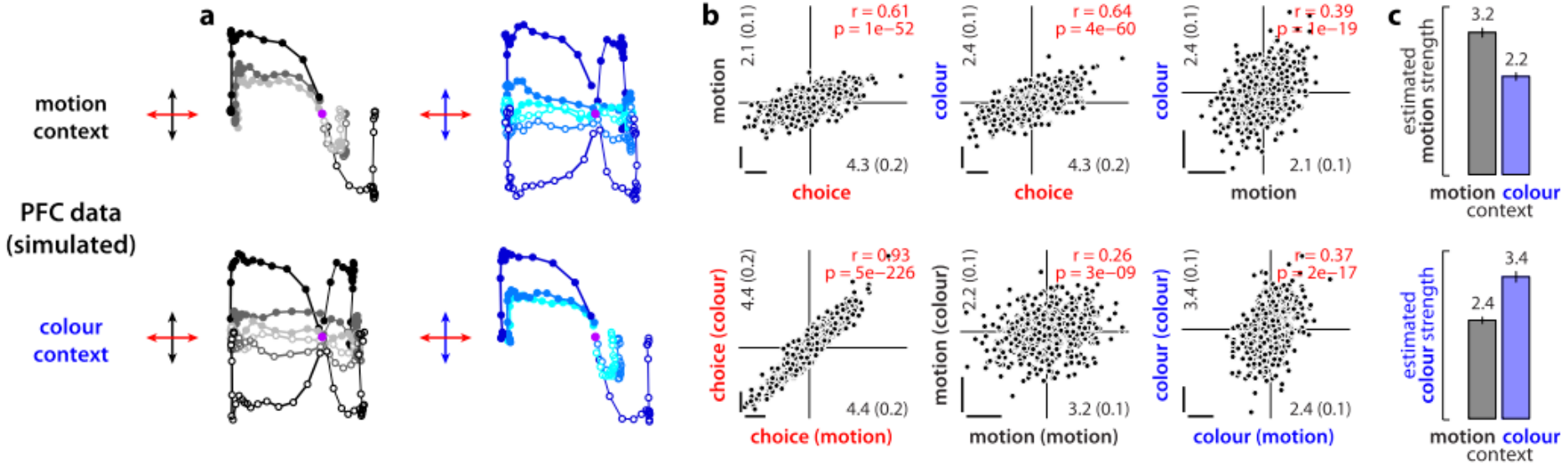
Why should I care?

- Many of your colleagues are using it! Has become **the de facto standard for analyzing/visualizing data** in scientific environments
- Analyzing data via code is both **more principled** and **orders of magnitude faster** than doing things by hand or with specialized programs
- **Learn one language** for doing everything in your pipeline: processing data, running statistics, making figures, etc.
- Learning a programming language **will change the way you think** about data and data analysis

Some examples

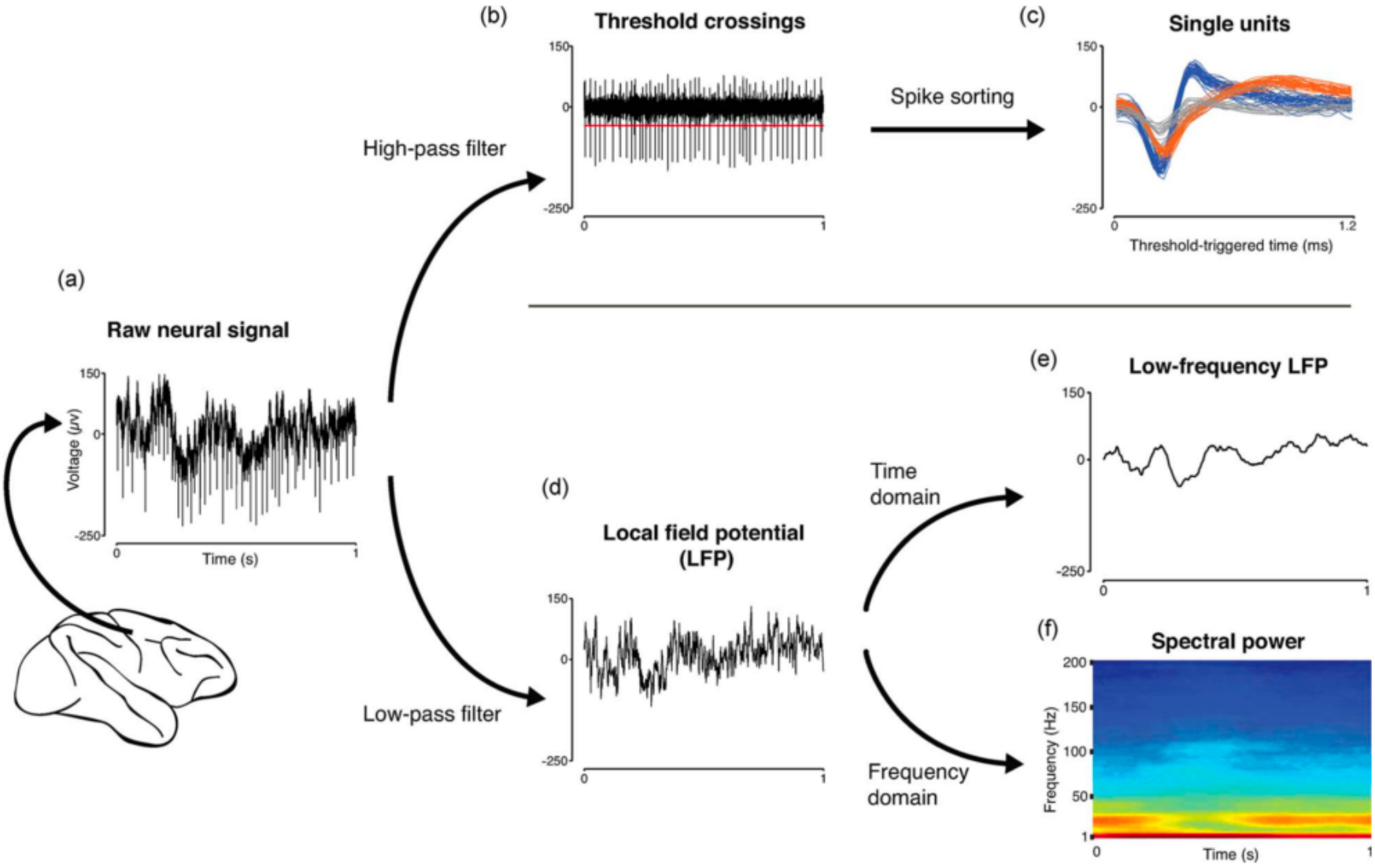
Data Analysis & Statistics





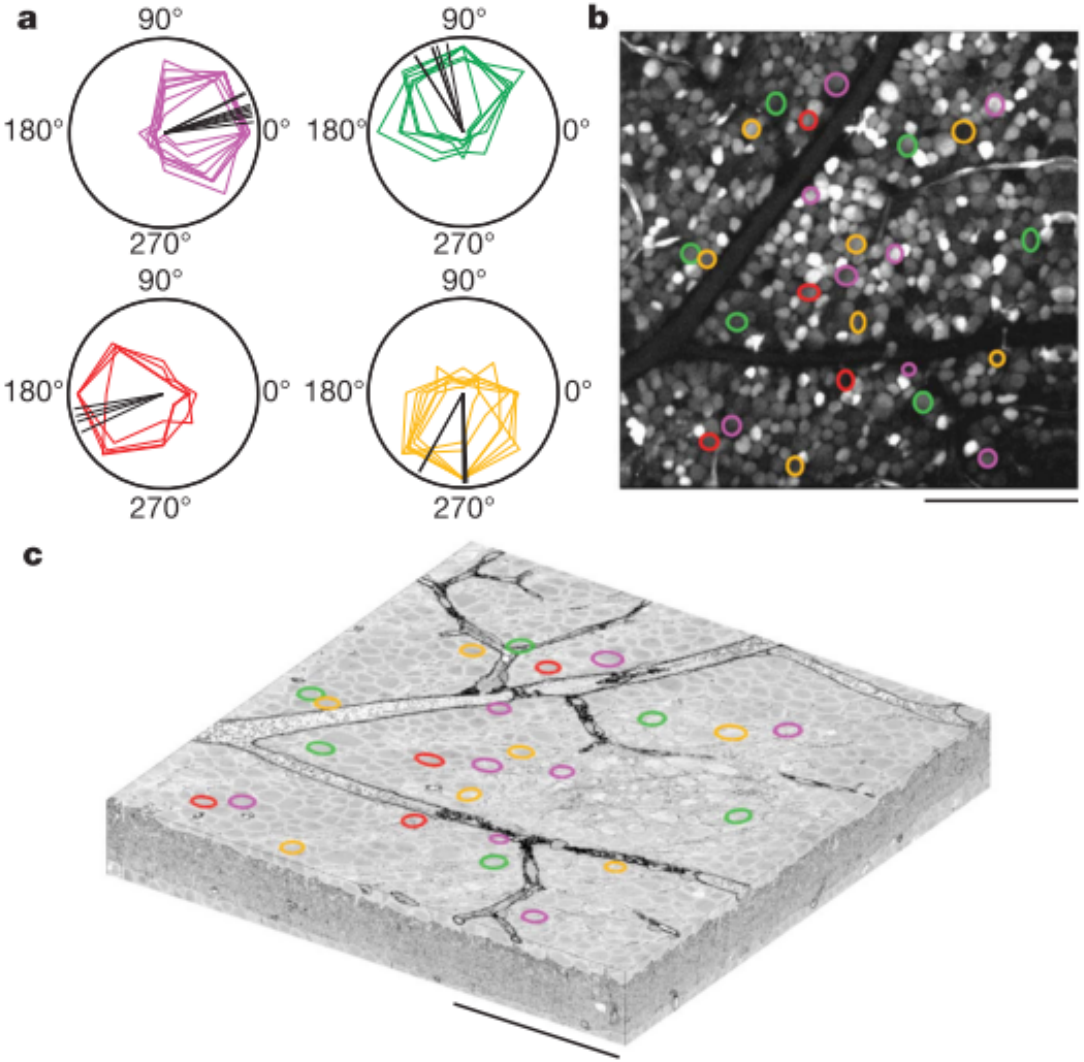
Mante*, Sussillo*, et al 2013

Signal Processing

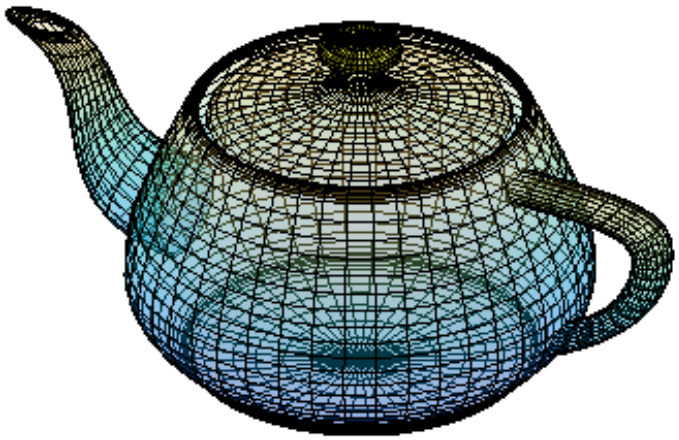


Kao et al. 2014

Visualization

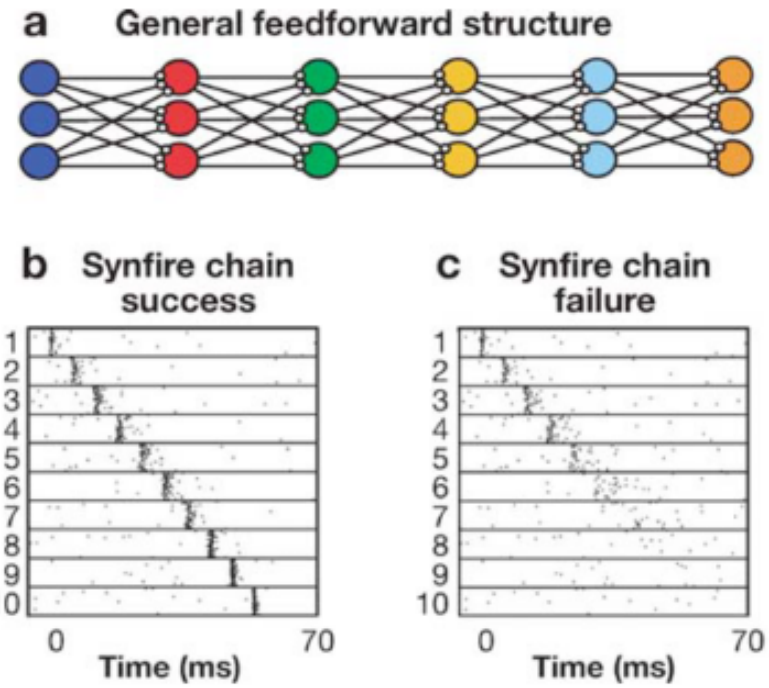


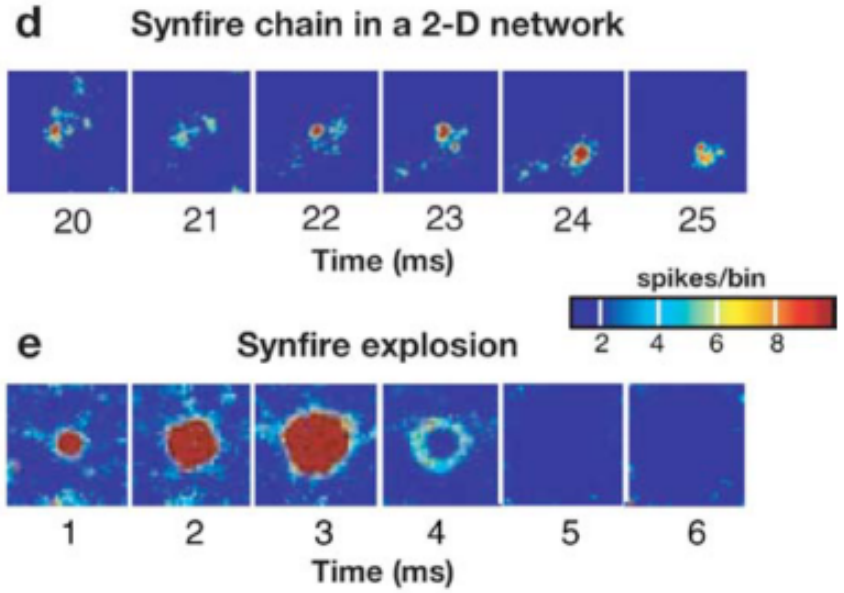
Briggman et al. 2011

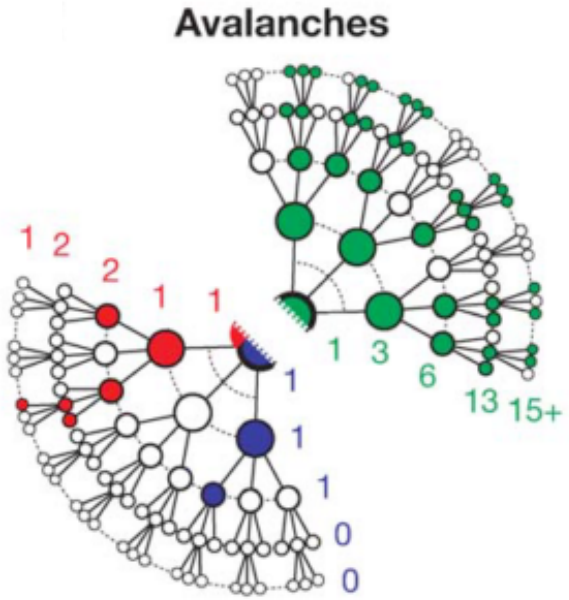


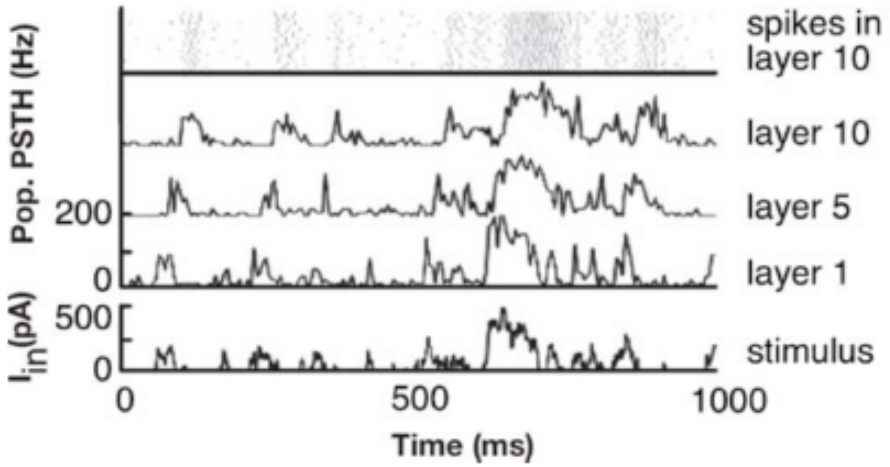
built-in teapot demo

Modeling



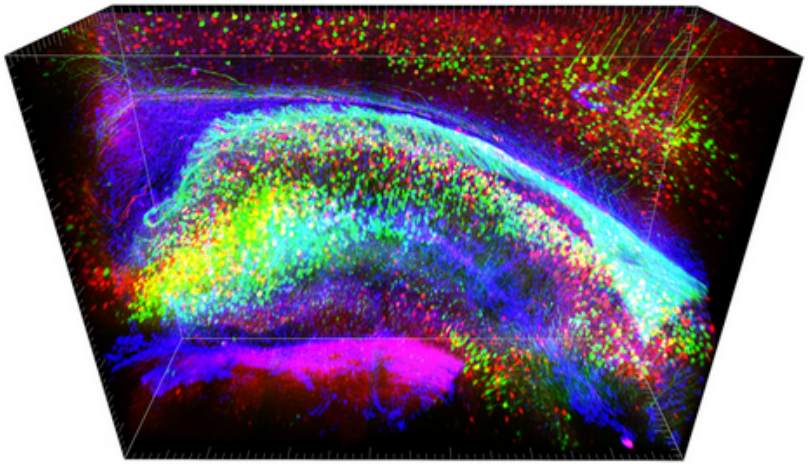


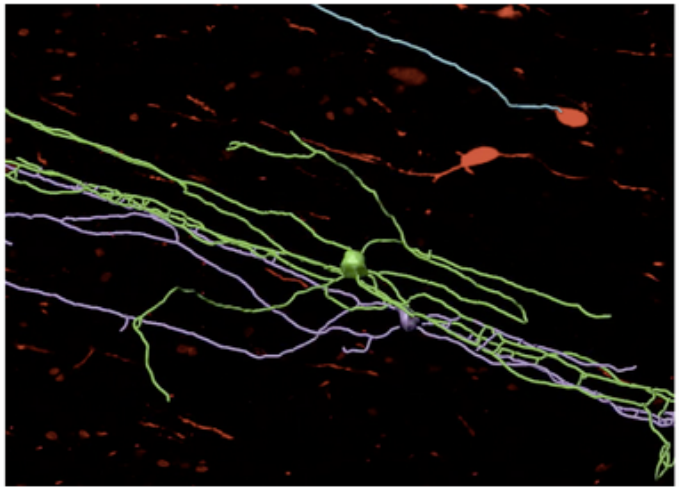


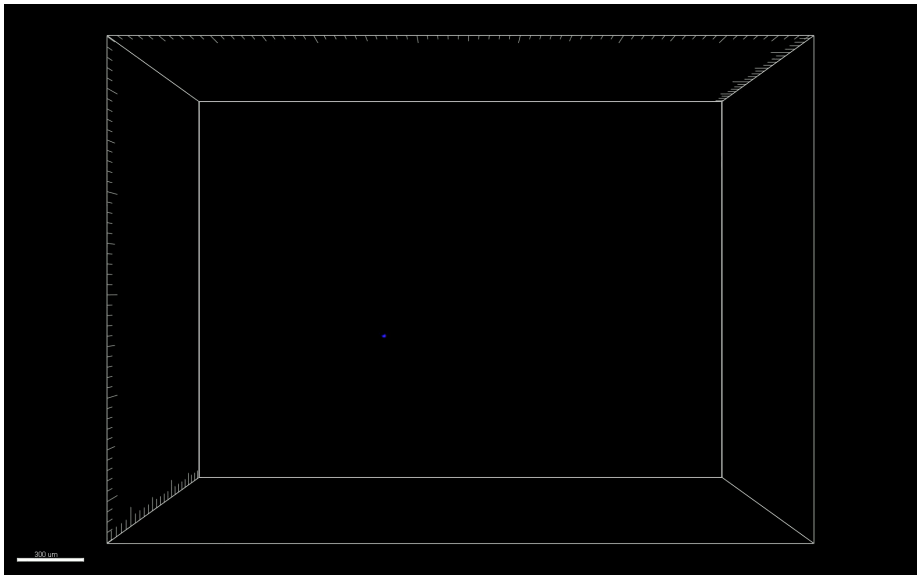


Vogels et al. 2005

Image processing







Chung et al. 2013

Closed-loop experiments

Nuyujukian et al. 2014

Course Aims

- Become proficient with programming in Matlab
- Teach you to recognize when/how Matlab can improve your workflow
- Learn how to think through implementing scientific analyses programmatically
- Develop practical data visualization skills
- Understand what resources are available for you to learn on your own

Course Structure

First 2/3 of the course:

- Data wrangling: how to load data into Matlab and get it into a useful format; variables and data types
- Basic programming: control flow, logic, and manipulation
- Data visualization: 2D and 3D plots and charts, customizing figures

Last 1/3 of the course (flexible):

- Specific algorithms/tools for analyzing data
- Signal and image processing, regressions and model fitting, and class' choice advanced features

Course Logistics

Weekly programming assignments

The best and only way to learn programming is by doing it

Assignments are designed to apply what was taught *and teach additional MATLAB commands*

No background about the example subject matter is needed

You'll look back at these as a MATLAB reference

Will be posted on website, **due at the start of the next class.** Turn in by email to nens230@gmail.com

Look at solutions when posted: good example code and "pro tips"

Final project

Do something **you** find useful with MATLAB (e.g. analyze some of your data, program a model)

Individual projects, but you can work with one other person on two separate but related projects (check with us first)

One page project proposal due 11/17. Feel free to run ideas by us before then

Grading

Weekly assignments graded on a 0, ✓, ✓+ basis:

0	Submission is broken or only a cursory attempt
✓	Submission mostly works, but has some flaws
✓+	Submitted code works and is well written

Sample solutions to the programming assignments will be posted after they are due. You can resubmit an assignment to get a ✓+

Final project that serves as your assignment for the final two weeks of the quarter. It will be graded on the same basis.

Course grading is **satisfactory** or **no credit**. You must receive a ✓+ **on all but two** assignments to pass the course
(and you must have a ✓ on those two. No zeros!)

Resources (help!)

- Your classmates
- Post questions to Piazza. This is preferred, we'll check it very often. piazza.com/stanford/fall2015/nens230
- Your instructors:

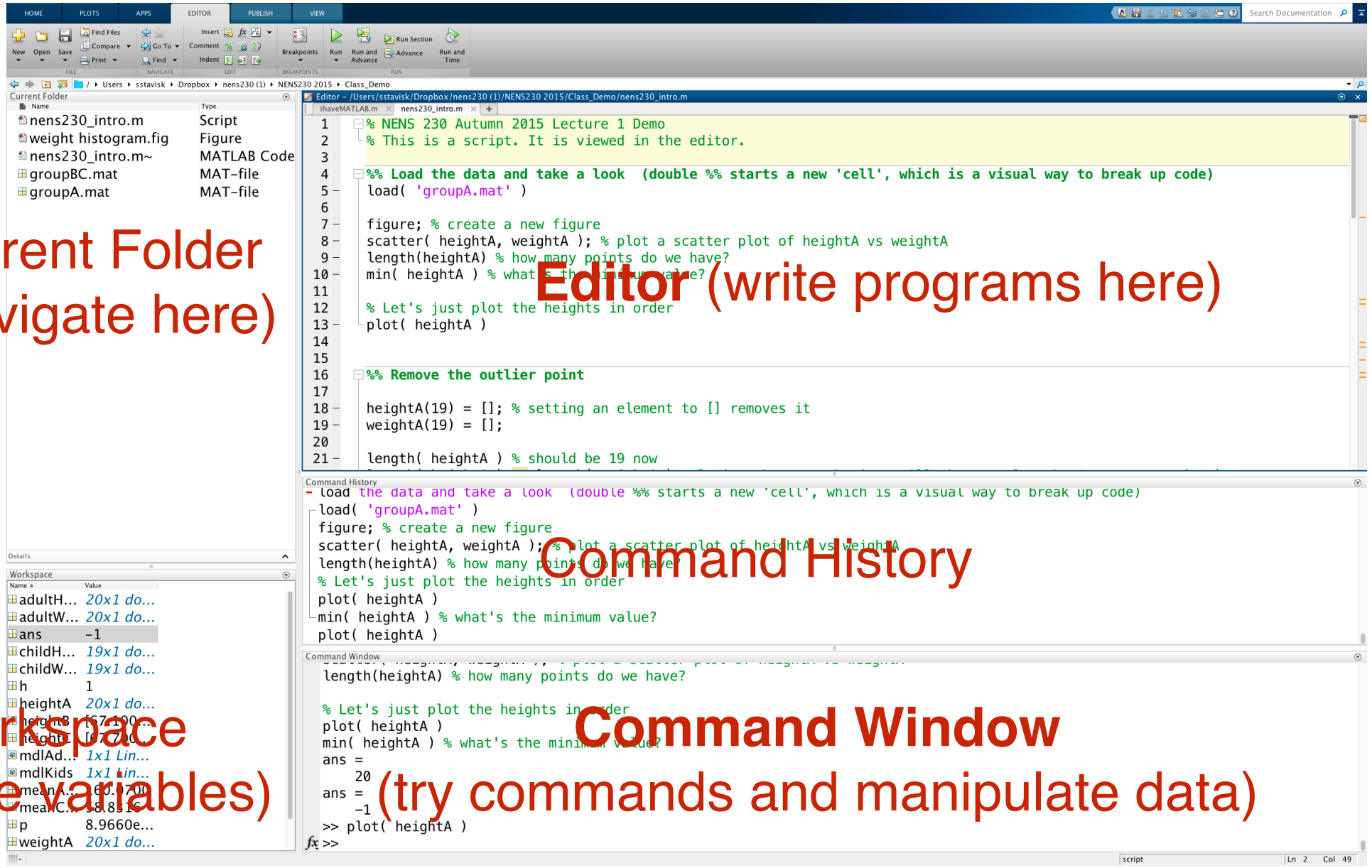
  Send emails to nens230@gmail.com

- Office Hours! When & where will be posted for the following week by the end of the weekend.
- **This week:** Thursday 9/24, 1-3 pm, Peet's at Clark Center top floor (Sergey)
- Course website: http://nens230.stanford.edu/

Let’s get started

Demo: Arrays and Functions

The MATLAB Application



Current Folder (navigate here)

Editor (write programs here)

Command History

Workspace (see variables)

Command Window (try commands and manipulate data)

Data Manipulation

In programming languages, we use a single equals sign to mean ‘assignment’:

```
x = 3;
```

The line above creates a new **variable**, which is named **x.** We can use this variable in subsequent calculations:

```
x + 1
x*x
```

What if we do the following?

```
x = 5;
y = x;
x = 10;
```

Then what is the value of `y`?

Command Line basics

List files in your current directory: `ls`

List of variables in your workspace: `who`

More info on variables in the workspace: `whos`

Clear all variables from the workspace: `clear`

Clear the text in the command window: `clc`

Change your current file directory: `cd`

Open the Matlab documentation: `doc`

Help for a specific function: `help myfunction`

Arrays: sets of numbers

```
x = [1 3 9 77 55]
```

indexing

```
x(5)== 55
```

length

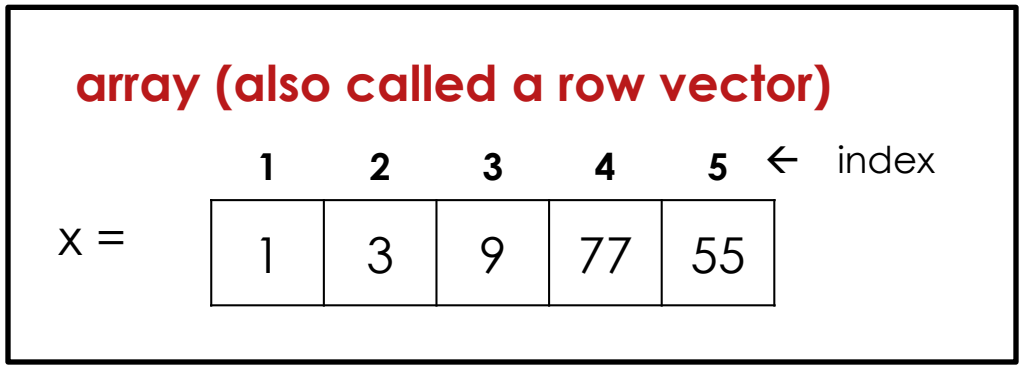
```
length(x) == 5
```

colon operator

```
x(1:3) == [1 3 9]
```

end indexing

```
x(4:end) == [77 55]
```



`length()` is a **function**

*The object in parentheses is called the **argument** of the function*

<- Ways to extract parts of an array: “index into an array”

How to “build” an array

colon operator

```
x = 1:10
x = [1 2 3 4 5 6 7 8 9 10]
```

returns values from 1 to 10, incrementing by 1

```
linspace() is a function that makes an array
x = linspace(0,1);
```

returns 100 (default) equally spaced values from 0 to 1

```
x = linspace(0,1,6)
x = [0 0.2 0.4 0.6 0.8 1]
```

returns 10 equally spaced values from 0 to 1

Some functions (like `linspace`)
can have a variable number of arguments!

Combining arrays

square brackets ‘concatenate’

```
x = 1:5
y = 6:10
z = [x y]
```

returns values from 1 to 10 because it combined x and y

Deleting elements

Set an element to the empty bracket 'deletes' it:

```
x = 1:6
x(2:3) = []
```

returns [1 2 4 5 6]

You can also re-assign a variable as a subset

```
x = 'strings are arrays of characters'
x = x(1:7)
```

sets x to returns 'strings'

`end` **is a special keyword meaning the end of the array**

```
x = x(end-2:end)
```

returns 'gs'

Many useful functions operate on arrays

Statistics:

```
mean( x )
std( x )
fitlm( x, y )
```

Plotting:

```
plot( x )
scatter( x, y )
histogram( x )
```

A script is a bunch of MATLAB commands

Contained in a .m file, it's just some text. You find it in the **Current Folder** window and edit it in the **Editor** window.

Commands do the same thing when called in the **Command Window** as in a **script**

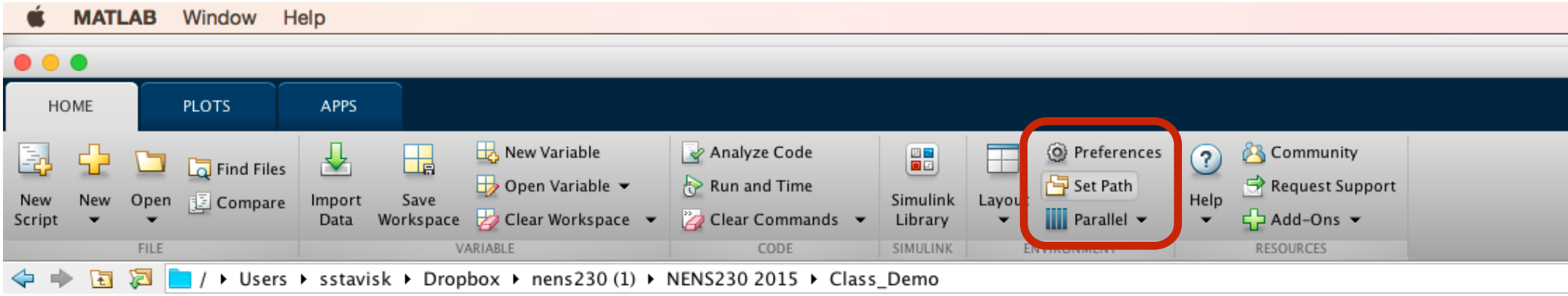
Wrote a Matlab script? You've "coded" a "program". Nice!

The MATLAB Path

Matlab needs to know what functions and scripts it can call. All the .m files that are in folders in the **path** are accessible. If it's not on the path, it's as if it doesn't exist.

The folder you're in (check with `pwd`, change by navigating in Current Folder or `cd` command) is always on the path

Path can be changed here:



Demo, ROUND 2!

For next week

- Get MATLAB. A how-to guide is on the website
- **Assignment 1** is a warmup to make sure you're set up with MATLAB and can navigate around it a bit and call some commands. It will be posted later today.
- Feel free to play around with MATLAB, look at the built-in tutorials, etc.

One way to access MATLAB

corn MATLAB

Use a “terminal” program (Terminal on MacOS)

```
ssh -X sstavisk@corn.stanford.edu
```

(but use your SUNET id)

```
module load matlab
```

```
matlab
```

`exit` to quit

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

`length`

`[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