Spatial Data and Analysis

Discussion 3

Felipe González

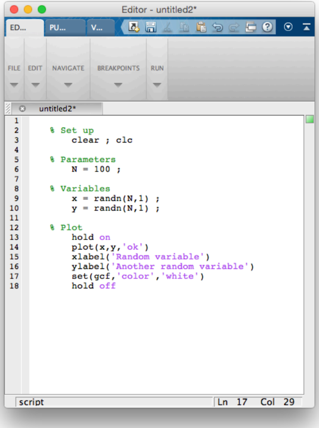
UC Berkeley

September $18th$

Outline

1. Miscellaneous
2. Summations
3. Functions
4. Waiting
5. Figures
6. Additional

Miscellaneous — bar colors when coding



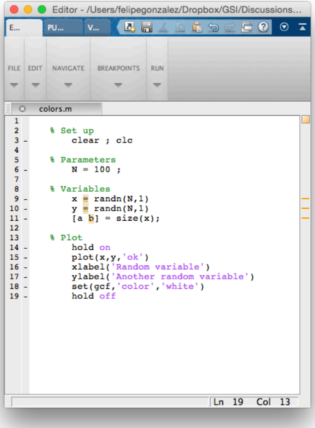
```
% Set up
    clear ; clc

% Parameters
    N = 100 ;

% Variables
    x = randn(N,1) ;
    y = randn(N,1) ;

% Plot
    hold on
    plot(x,y,'ok')
    xlabel('Random variable')
    ylabel('Another random variable')
    set(gcf,'color','white')
    hold off
```

(a) no mistakes



```
% Set up
    clear ; clc

% Parameters
    N = 100 ;

% Variables
    x = randn(N,1)
    y = randn(N,1)
    [a b] = size(x);

% Plot
    hold on
    plot(x,y,'ok')
    xlabel('Random variable')
    ylabel('Another random variable')
    set(gcf,'color','white')
    hold off
```

(b) room for improvement

Double summations

- Calculate the following expression:

$$\sum_{i=1}^{50} \sum_{j=1}^{100} (4i + 5j)$$

Double summations

- Calculate the following expression:

$$\sum_{i=1}^{50} \sum_{j=1}^{100} (4i + 5j)$$

- One way to do it:

```
x = NaN(100,50);
for i=1:50
    for j=1:100
        x(j,i) = 4*i + 5*j;
    end
end
disp('Answer:') ; disp(sum(sum(x,2)));
```

- Another (shorter) way to do it:

```
x = NaN(100,50);
for i=1:50
    x(:,i) = 4*i + 5*(1:100)' ;
end
disp(sum(sum(x,2))) ;
```

- Another (shorter) way to do it:

```
x = NaN(100,50);
for i=1:50
    x(:,i) = 4*i + 5*(1:100)' ;
end
disp(sum(sum(x,2))) ;
```

- Another (even shorter) way to do it:

```
x = 4*repmat(1:50,100,1) + 5*repmat((1:100)',1,50);
disp(sum(sum(x,2))) ;
```

- Another (shorter) way to do it:

```matlab
x = NaN(100,50);
for i=1:50
    x(:,i) = 4*i + 5*(1:100)' ;
end
disp(sum(sum(x,2))) ;
```

- Another (even shorter) way to do it:

```matlab
x = 4*repmat(1:50,100,1) + 5*repmat((1:100)',1,50);
disp(sum(sum(x,2))) ;
```

- Shortest way to do it:

```matlab
sum(sum(4*repmat(1:50,100,1)+5*repmat((1:100)',1,50)))
```

Efficiency

- If you want to study the efficiency of your code, you can use the `tic` and `toc` commands in the following way:

```
tic
% series of commands
toc
```

Efficiency

- If you want to study the efficiency of your code, you can use the `tic` and `toc` commands in the following way:

```
tic
% series of commands
toc
```

- The four previous options to calculate the double summation give the following times:
  - `0.0016` seconds
  - `0.0013`
  - `0.0010`
  - `0.0007`

Efficiency

- This might no look like much, but if we replace 100 by 10,000 and 50 by 5,000 we get the following times:
  - `3.5` seconds
  - `0.4`
  - `0.7`
  - `0.6`
- Efficient coding can reduce the time your code takes to run significantly

Functions

- A MATLAB function is a program that performs a sequence of operations specified in a m-file.
- A function accepts one ore more variables as inputs, operates in them in some way, and then returns one or more variables as outputs and may also generate plots
- To create a function just open a new m-file

Functions

- A function has the following structure:

```
function [y1,..,yN] = myfunction(x1,..,xM)

end
```

Functions

- A function has the following structure:

```
function [y1,..,yN] = myfunction(x1,..,xM)

end
```

- Example of a simple function:

```
function y = average(x)
if ~isvector(x)
    error('Input must be a vector')
end
y = sum(x)/length(x);
end
```

- Which you would use as: `y = average(1:100)`

Functions — double summation

- Function for the double summation:

```matlab
function Y = ssum(a,b,N,M)
Y = sum(sum(a*repmat(1:N,M,1)+b*repmat((1:M)',1,N)));
disp('Answer:') ; disp(Y)
end
```

- Which we would then use simply as:

```matlab
ssum(4,5,50,100)
```

Functions — where to locate them

- There are two folders in which MATLAB automatically looks for functions:

  1. Your working directory

  2. Default folder `'~/Documents/MATLAB/'`

Waiting

- Some processes take a long time to run

- There are several ways to keep track of the waiting time:

  1. Estimate waiting time

  2. Display iterations

  3. Use a function to create waiting bars

Waiting — estimate

- Before running a code completely, always run it in a subset and make sure it is working fine
- When running the subset of code you can keep track of how much time it takes to finish using the `tic toc` commands
- Then, it is easy to estimate how much it will take for the full code to run.
- Example: if 10% of data takes 1 minute to run, then full code will take 10 minutes to run

Waiting — display iterations

- Keep track of progress in long loops

- You can do that using this type of code:

```
disp('Iteration:')
for i = 1:100
    disp(i)
end
```

Waiting — bars

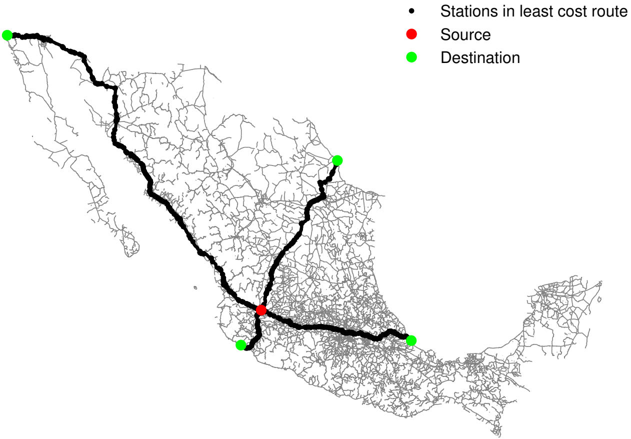
- You can also keep track of progress using a waiting bar

- The code to display a waiting bar is as follows:

```
h = waitbar(0,'Please wait...');
steps = 1000;
for step = 1:steps
    % computations take place here
    waitbar(step / steps)
end
close(h)
```

- Approximation of π (download `approxpi.zip` in website)

Waiting — example



Figures

1. Display
2. Legends
3. Subplots
4. Text

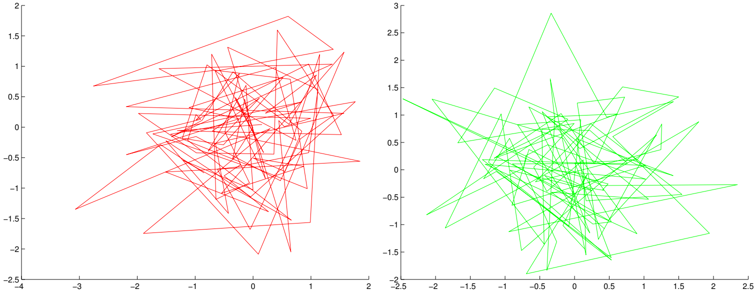
Figures — display

- Display separate figures:

```
figure
plot(randn(100,1),randn(100,1),'r')
     box off
     set(gcf,'color','white')
     export_fig Figures/plot_1.pdf

figure
plot(randn(100,1),randn(100,1),'g')
     box off
     set(gcf,'color','white')
     export_fig Figures/plot_2.pdf
```

Figures — display



(a) two random variables

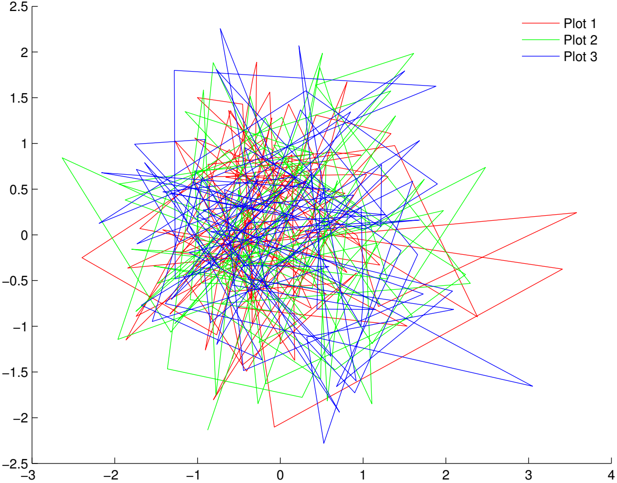
(b) more randomness

Figures — legends

- Control of legends:

```
figure
hold on
h1 = plot(randn(100,1),randn(100,1),'r') ;
h2 = plot(randn(100,1),randn(100,1),'g') ;
h3 = plot(randn(100,1),randn(100,1),'b') ;
    legend([h1 h2 h3],'Plot 1','Plot 2','Plot 3')
    legend boxoff
    set(gcf,'color','white')
hold off
```

Figures — legends

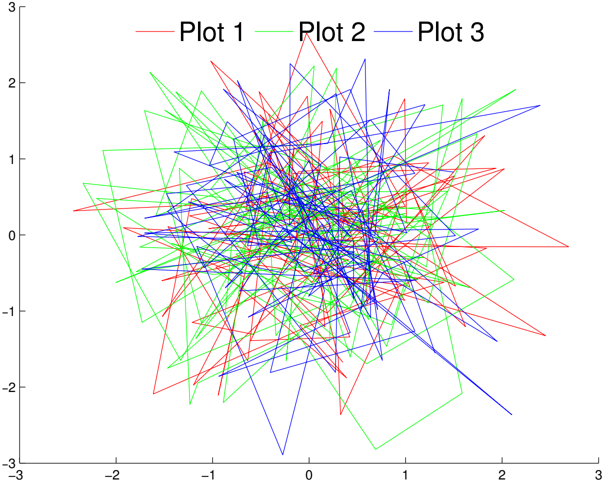


Figures — legends

Control font in legends:

```
figure
hold on
h1 = plot(randn(100,1),randn(100,1),'r') ;
h2 = plot(randn(100,1),randn(100,1),'g') ;
h3 = plot(randn(100,1),randn(100,1),'b') ;
L  = legend([h1 h2 h3],'Plot 1','Plot 2','Plot 3',...
         'Orientation','horizontal',...
         'Location','North');
    legend boxoff
    set(gcf,'color','white')
    set(L,'FontSize',20)
    export_fig Figures/plot_legend_2.pdf
hold off
```

Figures — legends

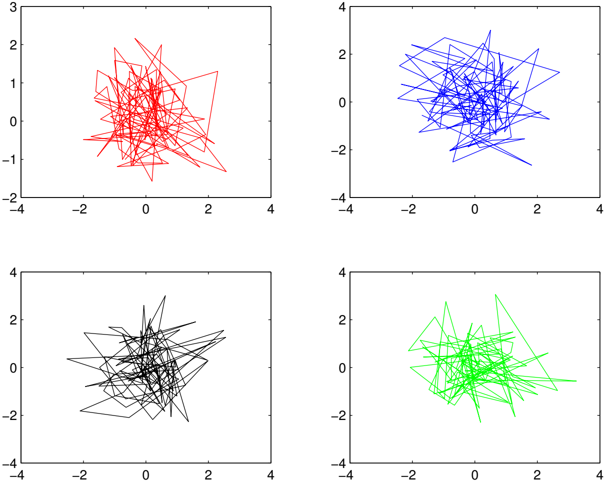


Figures — subplots

- Subplots

```
figure
hold on
subplot(2,2,1) ; plot(randn(100,1),randn(100,1),'r')
subplot(2,2,2) ; plot(randn(100,1),randn(100,1),'b')
subplot(2,2,3) ; plot(randn(100,1),randn(100,1),'k')
subplot(2,2,4) ; plot(randn(100,1),randn(100,1),'g')
    set(gcf,'color','white')
    export_fig Figures/plot_subplot_1.pdf
hold off
```

Figures — subplots



Figures — subplots

- Suppose you think space between subplots is too big

Figures — subplots

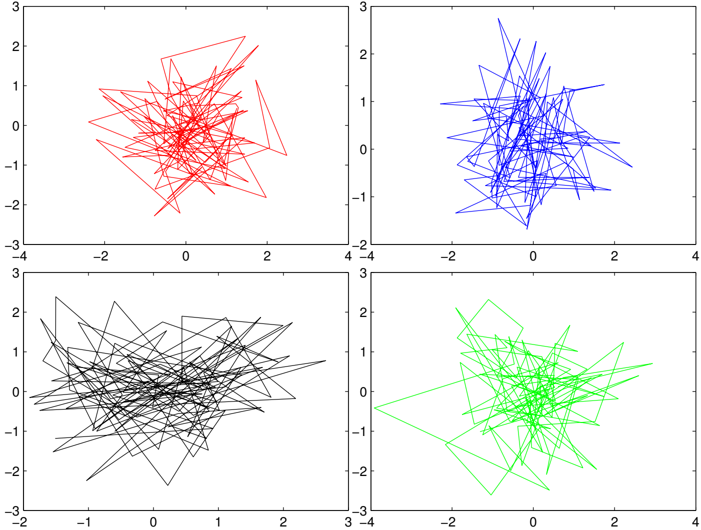
- Suppose you think space between subplots is too big
- Solution: find a function that changes that!
  - `tight_subplot` is one option, but there are others

Figures — subplots

- Suppose you think space between subplots is too big
- Solution: find a function that changes that!
  - `tight_subplot` is one option, but there are others
- Changing space between figures

```
ha = tight_subplot(2,2,[.05 .03],[.05 .05],[.05 .05]);
axes(ha(1)) ; plot(randn(100,1),randn(100,1),'r')
axes(ha(2)) ; plot(randn(100,1),randn(100,1),'b')
axes(ha(3)) ; plot(randn(100,1),randn(100,1),'k')
axes(ha(4)) ; plot(randn(100,1),randn(100,1),'g')
    set(gcf,'color','white')
    export_fig Figures/plot_subplot_2.pdf
```

Figures — subplots

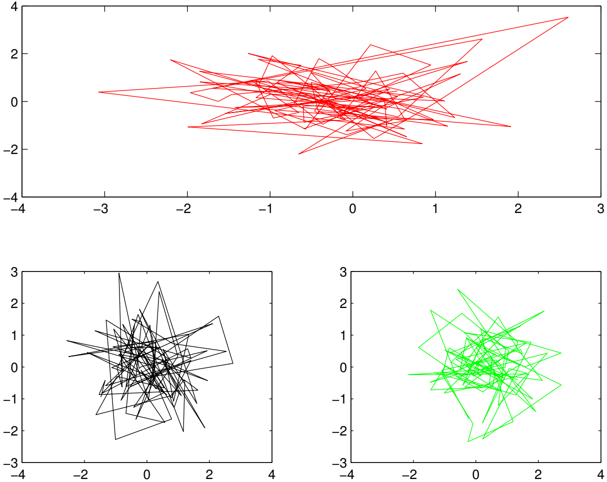


Figures — subplots

Subplots can have different sizes:

```
figure
hold on
subplot(2,2,[1 2]); plot(randn(100,1),randn(100,1),'r')
subplot(2,2,3)     ; plot(randn(100,1),randn(100,1),'k')
subplot(2,2,4)     ; plot(randn(100,1),randn(100,1),'g')
    set(gcf,'color','white')
    export_fig Figures/plot_subplot_3.pdf
hold off
```

Figures — subplots

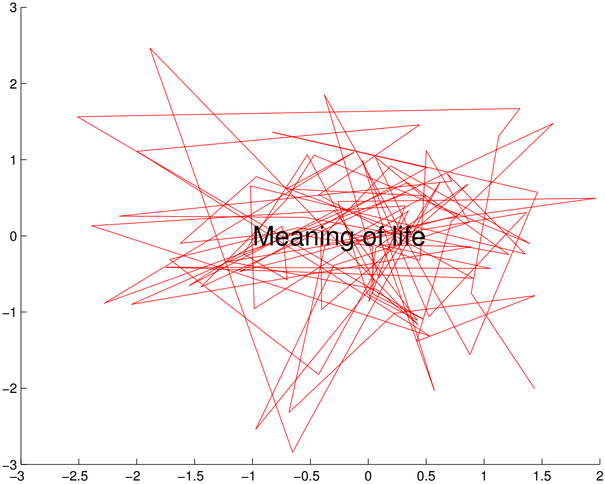


Figures — text

- Display text within figure:

```
figure
plot(randn(100,1),randn(100,1),'r')
    text(-1,0,'Meaning of life','FontSize',20)
    box off
    set(gcf,'color','white')
    export_fig Figures/plot_text.pdf
```

Figures — text



Figures — colors

- Colors can be specified manually instead of using the defaults that are already coded
- Colors in MATLAB are a $[0, 1]$ RGB triplet:
  - Yellow is `[1 1 0]` but already coded as `'y'`
  - Red is `[1 0 0]` but already coded as `'r'`

Figures — colors

- Colors can be specified manually instead of using the defaults that are already coded

- Colors in MATLAB are a $[0, 1]$ RGB triplet:
  - Yellow is `[1 1 0]` but already coded as `'y'`
  - Red is `[1 0 0]` but already coded as `'r'`

- You'll find that sometimes people use a different scale for RGB colors. For example, $(0, 255)$ scale in which red is coded as `[255 0 0]`

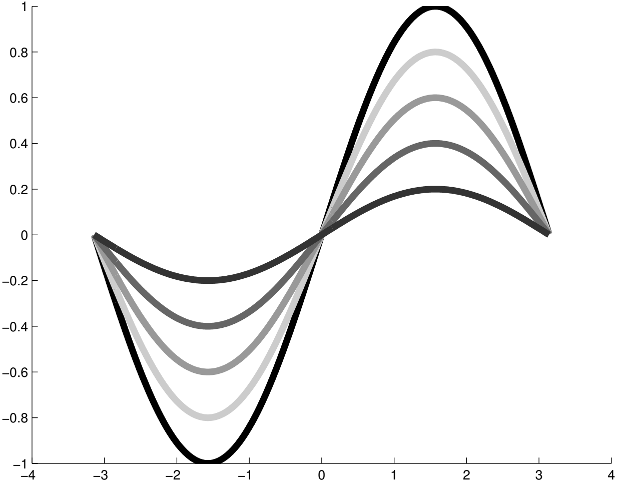
- If that is the case just convert that color to a $(0, 1)$ scale dividing by 255

Figures — colors

Plot lines in gray scale

```
x = -pi:0.01:pi;
figure
hold on
plot(x, 1*sin(x),'LineWidth',5,'Color',[ 0  0  0])
plot(x,.8*sin(x),'LineWidth',5,'Color',[.8 .8 .8])
plot(x,.6*sin(x),'LineWidth',5,'Color',[.6 .6 .6])
plot(x,.4*sin(x),'LineWidth',5,'Color',[.4 .4 .4])
plot(x,.2*sin(x),'LineWidth',5,'Color',[.2 .2 .2])
    set(gcf,'color','white')
    export_fig Figures/plot_color.pdf
hold off
```

Figures — colors



Additional resources

1. Check the document `MATLAB_Functions.pdf` that I uploaded to bCourses for more about functions
2. In this link you can find many many functions that people have written and shared for free
3. Take a look at a function which produces another (more flexible) waiting bar, including multiple bars, time estimates for long-running tasks, etc.