心理のための Matlab チュートリアル

Matlab for Psychologists: A Tutorial

Andrea Hamilton 著, 根本 清貴訳[1]

[1]このドキュメントは包括型脳科学研究推進支援ネットワーク活動の一環として、根本清貴が Andrea Hamilton 女史から承認を得て翻訳しました。原文は、http://antoniahamilton.com/matlab_for_psychologists.pdf にあります。なお、読みやすい日本語にするために多くのアドバイスをいただいた山口大学精神科の松尾幸治先生に深謝いたします。

イントロダクション

Matlab は言語です。ほかの同じ言語と同じように学ぶ最善の方法は使うことです。このチュートリアルは Matlab の基本的な文法を学ぶことからはじめ、次に実例を用いたたくさんの練習にうつります。最初の方はなかなか進まない様に見えますが、継続して行うことがとても大事です。なぜならいったん Matlab の使い方がわかると、次の様なことができる様になるからです。

- Excel よりもデータを素早く、柔軟に、かつエラーを少なく解析できます。
- 実験を行うために Cogent を使うことができます。
- 正確な設定で様々な呈示刺激（画像、音声、動画）を作成することができます。
- SPM のためのスクリプトを書いて、画像データを素早く効率的に解析できます

はじめるまえにいくつかのことを覚えておいてください。Matlab で何かをするときに、正しい方法はひとつだけではありません。たくさんの様々なコードは同じ結果をもたらすでしょう。しかし、Matlab に上達するにつれて コードをより巧くする方法を探すのに価値を見いだし、それによりそのコードの処理速度は速くなり、デバッグも簡単になるでしょう。どんなプログラムを書くにしても、ゼロからはじめるのはとても苦痛なことです。Matlab をはじめるにあたってのいい方法は、他の人が書いたスクリプトを持ってきて、あなたの必要に応じて書き換えることです。このチュートリアルの中にあるスクリプトは役に立つかもしれませんし、もしくは、他の人にその人のスクリプトについて尋ねてみましょう。（特に Cogent と SPM の場合はそうです。）

はじめるにあたって

このセクションは、読者が Windows を使用していて、プログラミングの経験がないと仮定しています。Linux / Unix ユーザーや C 言語の経験がある方は、少し我慢してください。

まず、Matlab はファイル名にスペースが入ることを嫌うということを覚えていてください（ちなみに私も嫌いです）。そのため、もし Matlab を使おうと思ったら、ファイル名やディレクトリ名にスペースを使うのではなく、_（アンダースコア）を使うことに慣れてください。また、ファイルを「マイ ドキュメント」やデスクトップに置くことを避けてください。C:ドライブや D:ドライブなどの直下にフォルダを作り、それを Matlab 解析用に使ってください。そうすることで、不必要なエラーが減りますのであなたの人生は楽になるでしょう。

デスクトップ上の Matlab のアイコンをクリックして Matlab を立ち上げてください。様々なウィンドウが起動します。一番大きい空白のウィンドウがコマンド ウィンドウで、>>というプロンプトがあるのがわかると思います。そこに、Matlab への指示を入力します。私はたいていコマンド履歴以外の他のウィンドウは消してしまいます。あまり役に立たないからです。

カレントディレクトリはコマンドウィンドウの一番上に示されています。`ls`, `cd`, `dir`, `pwd` などのコマンドを使ってディレクトリを移動することができます。

このチュートリアルを読み進むにあたって、例をひとつひとつ Matlab に入力してみてください。もし何をしているかわからなかったら、理解できるまでいろいろなところをあたってみてください。Matlab を習熟する唯一の方法は使うことですから、とにもかくにも試してみてください。本チュートリアルでは、Matlab に入力するものは**緑字**で示され、Matlab に出てくる結果は**青字**で示されます。そして赤字でハイライトされているところは原則として巻末の用語集にその定義が記載されています。

助けが必要だったら

Matlab には詳細なヘルプが用意されており、クエスチョンマークをクリックすることでヘルプを見ることができます。いつでも `help` とタイプすることで、コマンドのカテゴリーリストを得ることができます。たとえば `help general` とタイプすると、一般的なコマンドのリストを得ることができます。`help` の後

にコマンド名をタイプすると、そのコマンドに対する詳細なヘルプを見ることができます。たとえば、

```
>> help length

   length   Length of vector (ベクトルの長さ)
   length(X) は、ベクトル X の長さを出力します。length(X) は、空でない配列に
   対しては、MAX(SIZE(X)) と等価で、空配列に対しては 0 です。
```

レッスン 1 – Matlab の基本

Matlab はコマンドラインが基本で、そこで様々な変数をみたり操作したりすることができます。コマンドはプロンプト>>の後に入力し、Matlab は結果を返します。

値 10 をもつ変数 A をつくるためには次のようにします。(注:=の前後のスペースは必須ではありません。)

```
>> A=10
A = 10
```

今、A は Matlab のメモリもしくはワークスペースに値 10 をもつ変数として保存されています。これを使って合計などを求めることができます。

```
>> A+A
ans = 20
```

足し算の結果を直接変数に代入することもできます。

```
>> B = 5+8
B = 13
```

行末にセミコロンをつけると、Matlab はコマンドの結果を出力しません。しかし、変数の値は変更されます。

```
>> A = 15;
```

新しい値に変わったかどうか確認するために、変数をタイプしてみましょう。

```
>> A
A = 15
```

変数名はアルファベットから始まらなければなりませんが、その後は数字を使うことができます。変数の名前は大文字と小文字を区別し、32 文字の長さまで大丈夫ですが、スペースや句読点(ピリオド、コンマ、クエスチョンマークなど)は使えません。もし、スペースを使いたいと思ったらアンダースコア(_)を使うことができます。また、変数名にはすでに Matlab 用に指定されている言葉があります(if など)。これらの単語は変数名に使うことができません。もし、コマンド名を変数名に使ってしまったらそのコマンドが動かないかもしれません。そのような場合は変数をクリアしてください。

ある変数を取り除くには、clear と取り除きたい変数名をタイプします。

```
>> clear B
```

ワークスペースにある変数をすべてクリアしたいときには、`clear all` とタイプします。

コマンド `whos` はワークスペースにある変数を示します。以下の様な情報を見ることができます。

Name: 変数名

Size: 行数(1 番目の数)と列数(2 番目の数)

Bytes: その変数が使用するメモリ量。非常に大きな変数を使う以外は気にする必要はありません。

Type: 以下の例ではすべての変数がダブル配列ですが、text, cell, logical も使うことができます。(後述します)

```
>> whos
     Name        Size        Bytes      Class
     A           1x1         8          double array
     ans         1x1         8          double
```

コマンド `clear` で作った変数をクリアすることができます。 `clear var` は変数 var だけをクリアします。変数すべてをクリアしたいときには、`clear all` を使うことができます。

```
>> clear all
```

レッスン 2　行列と記号

Excel に慣れていたら、データをグリッド(格子)に配置し、解析をするときには行や列を足しあわせていけばいいという概念には慣れていることでしょう。Matlab では好きなだけのグリッドもしくは配列を作ることができ、それぞれが何らかの名前を持つ変数となります。その配列をいじりたいときには、その変数名を使えばいいのです。配列には以下の様なものがあります。

スカラー(scalar) –ひとつの数字です。

```
A = 10
```

ベクトル(vector) – 1 行か 1 列の数字の集まりです。

```
B = 1  2  3
C =  4
     3
     8
```

行列(matrix) – グリッドに配置された数字の配列です。その大きさは行数と列数であらわされます。D は 3 行 4 列の行列です。

```
D = 5  6  7  9
    8  3  5  3
    5  6  3  2
```

要素(element) – 行列やベクトルの中のひとつの成分をさします。

括弧

たいていの場合、Matlab にデータを入力するには、角括弧[ ]を使う必要があります。

データを行ベクトルに入力するときには、値を角括弧の中にいれ、それぞれの値をスペースかコンマ（もしくはその両方）で区切ります。

```
>> C = [6, 5, 8, 10]
C = 6 5 8 10
```

データを列ベクトルに入力するときには、値を角括弧の中にいれ、それぞれの値をセミコロンで区切ります。ここでのセミコロンは、「新しい行をはじめなさい」という意味です。

```
>> D = [3; 1; 6; 5]
D = 3
    1
    6
    5
```

データを行列に入力するときには、行のデータはコンマで区切り、新しい行を始めるためにセミコロンを使います。行ベクトルと列ベクトルを一度に書く感じです。

```
>> E = [1, 2, 3; 4, 5, 6]
E = 1 2 3
    4 5 6
```

一般的に、角括弧は何かをつなげたいときにいつでも使うことができます。詳しくは、`Vertcat` と `horzcat` を御覧ください。

丸括弧( )は行列から何かを取り出したいときか、行列の一部を参照したいときに使います。例えば、E(2,3) は行列 E の 2 行 3 列目にある値をさします。

```
>> E(2,3)
ans = 6
```

もし、E の行列の範囲外にあるものを参照しようとすると、エラーメッセージが出ます。

```
>> E(4,3)
??? Index exceeds matrix dimensions (インデックスが行列の次元を超えています。)
```

丸括弧は、行列の一部を変更したいときにも使えます。行列 E の 1 行 3 列目の値を 10 に変更するには、次のようにタイプします。

```
>> E(1,3) = 10
  E =     1    2   10
          4    5    6
```

もし、行や列を行列に追加したかったら、追加したい位置と内容を指定することで新しい行列が作成されます。注意しなければならないのは、追加する場合はもともとの行列と同じ大きさの行や列を指定しなければいけないということです。もし違うサイズの行や列を追加しようとするとエラーメッセージが出ます。

```
>> E(3,:) = [7, 8, 9]
  E =     1    2   10
          4    5    6
          7    8    9
```

コロン

コロン記号：は Matlab では様々な意味を持ちます。

丸括弧の中では、コロンは行や列すべてを意味し、ふつうは行列からデータを抽出するときに使われます。

E(:,2) は行列 E の 2 列目すべてという意味です。「行列 E のすべての行、第 2 列」と読んでください。

```
>> E(:,2)
ans =  2
       5
       8
```

E(2,:) は行列 E のすべての列を意味します。「行列 E の第 2 行、すべての列」と読んでください。

```
>> E(2,:)
ans = 4  5  6
```

丸括弧の中にコロンだけあると、行列のすべての数字を列ベクトルに変換します。E(:)は行列 E のすべての値を長い 1 列のベクトルにします。

```
>> E(:)
ans =  1
       4
       7
       2
       5
       8
       10
       9
```

2 つの数字の間にあると、コロンは数字 A から B まで整数で 1 ずつ増えていくということを意味します。

```
>> F = 5:10
F = 5  6  7  8  9  10
```

これを使って行列からデータを抽出することができます。たとえば、F(:,3:5) は列 3, 4, 5 の

すべての値ということを意味します。

```
>> F(:,3:5)
ans = 7 8 9
```

コロンでまとめられた 3 つの数字のまとまりは、どの間隔で数字が増えていくのかを示します。たとえば、G = 3:4:20 は G が 3 から 20 まで 4 刻みで増えていくことを意味します。

```
>> G = 3:4:20
ans = 3 7 11 15 19
```

コマンドウィンドウから `help punct`, `help colon`, `help paren` とタイプしていただければ、Matlab の行列における記号について詳細な情報を得ることができます。

配列エディタ

私は Matlab で配列エディタを使ったことは一度もありませんが、もし Excel から移行するのだったら、便利と思うかもしれません。メニューからレイアウト – 表示 – ワークスペースにチェックを入れると Matlab は whos のようにすべての変数とその大きさを表示します。変数をクリックすると配列エディタが起動します。Excel のようにグリッドに入っているデータが表示され、ここで値を変更するとコマンドウィンドウから指示したと同じように値を変更することができます。また、Excel などの他のプログラムからここにデータを貼り付けることもできます。

レッスン 3 – インデックス

インデックスとは、行列のうちの自分が欲しい部分を得ることを意味し、Matlab に欠かせない機能です。自分が欲しいデータを得る一番簡単な方法は、添字インデックス subscript を使うことです。すなわち、Matlab に自分が欲しい行と列を伝えるということです。

例として、魔方陣（訳注：横・縦・斜めに数えてもその和が常に等しい数字配列表）の行列を作ってみましょう。コマンド `magic` で魔方陣を作ることができます。

```
>> clear all
>> A = magic(5)
A = 17  24   1   8  15
    23   5   7  14  16
     4   6  13  20  22
    10  12  19  21   3
    11  18  25   2   9
```

行列 A から 4 行 3 列にある値を、丸括弧を使って取り出すことができます。

```
>> B = A(4,3)
B = 19
```

もしくは、4 行目だけを行列 C として取り出すこともできます。この場合、C は「行列 A の第 4 行、すべての列」と読めます。

```
>> C = A(4,:)
```

```
C = 10  12  19  21  3
```

行列 A から 2 行目と 3 行目を取り出すこともできます。今、行列 D は「行列 A の第 2 行と第 3 行、すべての列」と読めます。角括弧とセミコロンが欲しい行を結合するために使われていて、丸括弧が行列 A から行を取り出すために使われています。

```
>> D = A([2;3],:)

D = 23   5   7  14  16
     4   6  13  20  22
```

さらに行列 D の 1-3 列を取り出してみましょう。行列 E は「行列 D のすべての行、1-3 列」と読めます。

```
>> E = D(:,1:3)
E = 23   5   7
     4   6  13
```

インデックス機能を用いて値を行列 A に入れることもできます。たとえば、行列 A の 2 行 2 列目を 100 にしたいならば、次のようにします。

```
>> A(2,2) = 100

A = 17  24   1   8  15
    23 100   7  14  16
     4   6  13  20  22
    10  12  19  21   3
    11  18  25   2   9
```

行列 A の 4 列目をすべて 0 にしたいならば次のようにします。

```
>> A(:,4) = 0

A = 17  24   1   0  15
    23 100   7   0  16
     4   6  13   0  22
    10  12  19   0   3
    11  18  25   0   9
```

これで練習 A をするのに十分な知識を得られたはずです。

レッスン 4 – 基本的な計算

このセクションをはじめる前に `clear all` とタイプして、そして E = [1, 2, 3; 4, 5, 6] と A=10 と入力してください。

Matlab では、数字や変数を四則演算することができます。変数は数字（もしくは数字の組み合わせとして）扱うことができます。たいていのことは予想できるかと思います。

+ は足し算です。

- は引き算です。
' は転置を意味します。(行を列に、列を行に変換します)
( ) は BODMAS(訳注:Brackets of division, multiplication, addition, subtraction の略。演算の際の括弧のルール)に従って計算の順番を指定できます。

しかし

* は行列の掛け算を意味します。
/ は行列の割り算を意味します。
(従来の)掛け算や割り算をしたいとき、ドットをつけて行います。
.* は行列の要素ごとに掛け算をすることを意味します。
./ は行列の要素ごとに割り算をすることを意味します。
.^ はべき乗を意味します。(A.^2 は A の 2 乗ということです)

数字の四則演算は単にコマンドラインにタイプすることで行うことができます。記号の前後のスペースは必須ではありません。

```
>> 174 .* 734
ans =  127716
>> (A*2) / 5
ans = 4
```

スカラーと行列を計算することもできます。この場合、行列のすべての要素に対して演算が行われます。

```
>> J = E*A
J =  10    20    30
     40    50    60
```

もし、行列の大きさが同じであれば、行列の足し算や引き算もできます。

```
>> K = J - E
ans =     9    18    27
         36    45    54
```

2 つの行列の要素を掛けあわせたり割ったりするためには、.* と ./ を使います。行列のサイズは同じでなければなりません。さもなければエラーとなります。

```
>> L = [3 2 1; 9 5 6]
L =   3     2     1
      9     5     6
```

K の各々の値を L の(行列の観点で)同じ場所にある値で割るためには、以下のようにします。

```
>> K./L
ans = 3     9    27
      4     9     9
```

E の各々の値を L の同じ場所にある値でかけるためには、以下のようにします。

```
>> E.*L
ans =

     3     4     3
    36    25    36
```

注意しなければならないことは、.を打たずに* や / を使うと、行列の掛け算と割り算になってしまうということです。この場合、すべての行と列が掛け合わされて合計が算出されます。このことは線形代数の教科書に書いてありますが、単純なデータ解析には必要ないでしょう。もし掛け算をして、大きさの異なる行列が結果として出てきたら、たぶん行列の掛け算をしてしまったのだと思ってください。行列が正方行列の場合（行と列の数が同じとき）には、特に注意してください。なぜなら、正方行列では行列の掛け算ができてしまい、明らかなエラーを生じないからです。

Matlab での基本的な演算のより詳しい情報については、`help elfun` (elementary functions), `help ops` (operators), や `help arith` (arithmetic) とタイプしてみてください。

レッスン 5 – 基本的な関数

Matlab には多くの関数が組み込まれており、それを用いることで簡単な計算をしたり、行列を素早く簡単に作ったりすることができます。関数は（あなたあるいは Matlab あるいは他の誰か）によって書かれたちょっとしたコードであり、何かを入力することで何らかの出力を得ることができます。関数は丸括弧を用い、Matlab にもともと用意されている関数には使用方法を記載したヘルプがついています。その関数について詳しく知りたい場合は、`help` に続いてその関数名をタイプしてください。

すべての関数にはひとつかそれ以上の入力もしくは引数が必要で、ひとつ以上の結果が出力されます。もしある関数の出力が一つしかないのであれば、出力結果はそのまま変数に代入されます。もし出力が複数あるのであれば、出力変数をグループ分けするために丸括弧が必要です。

すべての関数は以下の形式をとります。

```
[出力 1, 出力 2, ...] = 関数 (引数 1, 引数 2, ...)
```

単一の値を用いて計算する関数ならば、あらゆる関数が利用できます。たとえば、三角関数（sin, cos, tan）、平方根(sqrt)、log、pi などです。Matlab には行列の行や列に用いることのできる様々な有用な関数もあります。それらは、合計 (sum), 平均(mean), 標準偏差(std), 最大値(max) そして最小値(min)です。

デフォルトでは、これらの関数は行列の列に対して作用します。しかし、行に対して作用するように次元を変更することができます。その方法を知るためには、`help` の後に関数名をタイプしてください。

行列 K の合計を出しましょう。

```
>> sum(K)
ans =    45    63    81
```

行列 K の行の合計を出すためには、sum 関数に 2 つめの次元で計算をするように指示しなければいけません。（第 1 次元＝列、第 2 次元＝行です。）

```
>> sum(K,2)
ans =    54
        135
```

行列 J の平均を求めてそれを変数 mj に入れます。

```
>> mj = mean(j)
mj =  25    35    45
```

アポストロフィーは行列やベクトルを転置させるのに便利です。すなわち、すべての行が列になり、列が行になります。

```
>> K'
ans =   9    36
        18    45
        27    54
```

その他にもベクトルや行列を生成する様々な関数があります。それらは、rand, ones, zeros, repmat, ndgrid, linspace, logspace, magic などです。

たとえば、3 行 5 列の乱数表を作りたいとします。(R は乱数を生成するので、あなたの結果は異なるでしょう。)

```
>> R = rand(3,5)
R =
    0.6154    0.7382    0.9355    0.8936    0.8132
    0.7919    0.1763    0.9169    0.0579    0.0099
    0.9218    0.4057    0.4103    0.3529    0.1389
```

他にも多くの有用な関数があり、それらは用語集に列挙されています。各々がどんな働きをするかを知るには、help 関数名とタイプしてください。

これで練習 B をするのに十分な知識を得られたはずです。

レッスン 6 – 論理演算

このセクションをはじめる前に clear all をタイプし変数をクリアしてください。

論理演算子はあるステートメントが真か偽か評価するために使います。偽は常に 0 で表され、Matlab は 0 でない値を真とします。わかりやすくするために、1 を真とするのがいいでしょう。論理演算子はスクリプトのどの部分を次に実行するかを決めたり、行列のどの要素を計算に使うかを決めたりするのに用いることができます。主に 4 つの論理演算子があります。

>	～より大きい
<	～より小さい
==	～と等しい
~=	～と等しくない

これらの論理演算子は AND, OR, NOT と丸括弧を使うことで連結できます。

```
&     論理積 logical AND
|     論理和 logical OR
~     否定 logical NOT
```

Matlab では、論理演算子をすべての配列に適用することもできますし、個々の値に適用することもできます。そして他の(添字)インデックスを得るための論理配列を得ることもできます。

以下の例では、行列 B は行列 A のどの値が 2 より大きいか示し、行列 C は行列 A のどの値が 5 より小さいかを示しています。

```
>> A = [1 5 3 4 8 3];
>> B = A>2
B = 0 1 1 1 1 1
>> C = A<5
C = 1 0 1 1 0 1
```

行列 D は行列 B と行列 C がともに真である値がどれかを示しています。

```
>> D = B & C
D = 0 0 1 1 0 1
```

ここで whos をタイプすると、 class に logical とあることから列 B、C、D が論理値 (logical) であることがわかります。これが意味するところは、Matlab はこれらの行列を単に 0 か 1 かの数字の集まりなのではなく、真 True か 偽 False の値をもつ行列と認識しているということを意味します。

それでは、行列 D を論理インデックスとして A に適用してみましょう。

```
>> E = A(D)
E = 3 4 3
```

行列 E は行列 D で真であった値、つまり 2 より大きく 5 より小さい値だけから構成されています。これらの値に対応する行列 D の値は 1(真)であり、それら以外の行列 A の値は落ちています。

論理インデックスは上述した添字インデックスと似た感じでよく用いられます。ともに配列からある特定の値を取り出します。論理インデックスを添字インデックスに変換するには find を使います。これによって論理値がゼロでない値を教えてくれます。

```
>> F = find(D)
F = 3 4 6
```

これは行列 D の 3 番目、4 番目、6 番目の値がゼロでないという意味です。この添字インデックス行列 F を行列 A に用いると、論理インデックス行列 D を行列 A に用いたと同じ結果を得ることができます。

```
>> A(F)
ans = 3 4 3
```

論理インデックスがもっとも有用な時はある変数が他の変数を分類するために使われる時です。3つのカテゴリーにわけられるデータがあると想像してみてください。

```
>> data = [4 14  6 11  3 14  8 17 17 12 10 18];
>> cat =  [1  3  2  1  2  2  3  1  3  2  3  1];
```

変数 cat の値が 2 である場所を見つけましょう。

```
>> cat2 = cat==2
cat2 = 0  0  1  0  1  1  0  0  0  1  0  0
```

変数 cat の値が 2 であるところに相当する変数 data の値を見つけましょう。

```
>> data2 = data(cat2)
data2 =     6     3    14    12
```

今、カテゴリー2 のデータの平均を求めることができます。

```
>> mdat2 = mean(data2)
mdat2 = 8.75
```

今説明した 3 つのステップを 1 行で行うことができます。

```
>> mdat2 = mean(data(cat==2))
mdat2 = 8.75
```

レッスン 7 – 欠損値

現実世界において心理学の実験をするとき、しばしば欠損値に苦しむことがあります。応答速度がある限界値を超えてしまっていたり、被験者が何らかの間違いをしてしまったりしたデータは、実験データから除きたくなるかもしれません。Matlab では `NaN` を用いることによりこれが可能となります。NaN は Not A Number（数字でない）の略で、行列やベクトルでデータが欠損しているところで使うことができます。一般的に、欠損値があるデータは、データを全部削除してしまうより欠損値を NaN で置き換える方がずっとよいでしょう。NaN を使うと sum や mean のような基本的な関数が使えなくなってしまいますが、Matlab には NaN を無視してデータの合計、平均、標準偏差を求めることのできる代わりの関数、`nansum`, `nanmean`, `nanstd` があります（訳注：これらの関数は標準では提供されていません。Statistics toolbox が必要です。しかし、ネットで Nan Suite を検索することで、これらの関数をダウンロードして使用することが可能です）。

レッスン 6 のデータセットを用いて説明しましょう。被験者が間違いをしたときにその場所を 1 とする err というベクトルを準備します。

```
>> err = [1 0 0 0 0 0 0 1 0 1 0 0];
```

この err をインデックスとしてすべての間違いを NaN で置き換えます。次の一行は、err が 1 のところは NaN に置き換えられると読むことができます。

```
>> data(err==1) = NaN
data = [NaN 14 6 11 3 14 8 NaN 17 NaN 10 18]
```

ここで、カテゴリー2 のデータを抽出すると、数字の 1 つが NaN になります。

```
>> data2 = data(cat2)
data2 =     6     3    14    NaN
```

ですから、平均を求めるには、nanmean を使わないといけません。

```
>> mdat2 = nanmean(data2)
mdat2 = 7.6667
```

NaN を使う他の便利な関数に isnan があります。これは行列の中に NaN があるとそこを 1 とし、それ以外をゼロにします。逆に isfinite は行列の要素が数字（NaN や Inf でない）であれば 1 を返し、それ以外はゼロを返します。

これで練習 C, D をするのに十分な知識を得られたはずです。

レッスン 8 – グラフ基本編

データの視覚化はとても大事なことですが、まず最初にデータを準備しなければなりません。このセクションをはじめる前に clear all をタイプしてください。linspace はある長さの数列を作るのに便利な関数です。何をしているかは help linspace を調べてみてください。

```
>> x = linspace(0,2*pi,30);
>> y = sin(x);
>> z = 0.5+cos(x/2);
```

グラフの基本

Plot はベクトルや行列を線や点でプロットするための基本的な関数です。データx、データ y、線の種類を指定します。どのようなグラフの線があるかは、help plot とタイプしてみてください。まずは、x と y の関係をプロットしてみましょう。

```
>>plot(x,y,'b-o')
```

Figure 1 というウィンドウが出てくるはずです。続けて別のグラフを同じ座標軸に描いてみましょう。

```
>>hold on
```

x と z の関係をプロットします。

```
>>plot(x,z,'r--')
```

軸ラベル、グラフタイトル、凡例もつけましょう。ラベルやタイトルにテキストを使う際には、シングルクォートでくくる必要があることに注意してください。

```
>>xlabel('x')
>>title('a couple of lines')
>> legend('y=sin(x)','z=0.5+cos(x/2)')
```

Axis を使うことで軸の範囲を指定することができます。たとえば、x 軸を0から10、y 軸を5から15にするためには、axis([0, 10, 5, 15])とします。

データをプロットする

Lesson2.mat がカレントディレクトリにあることを確認し、次のようにタイプしてください。

```
>> load lesson2
```

このことで、プロットするためのデータが読み込まれます。もし、ファイルが読み込まれないようであれば、ls とタイプし、カレントディレクトリの中にファイルが見えるか確認してください。もし、ls とうって lesson2.mat が見えないようであれば、ファイルを見つけ、カレントディレクトリの中に置かなければなりません。(もしないようであれば、私のウェブサイトにファイルが置いてあります。Google で Antonia Hamilton と検索してください。) カレントディレクトリは Matlab ウィンドウの一番上に表示されています。

今、どんなデータが読み込まれたか確認するために、whos とタイプしてください。data は 3 人の異なる被験者に対して行ったテストの得点を行列にしたもので各々10 回試行しています。

新しいグラフをプロットするために、前のグラフをクリアしましょう。すべての図ウィンドウを閉じるためには、close all を使います。一つの図ウィンドウを閉じるためには、clf を使います。図中の軸をひとつだけクリアしたいときには、cla を使います。

```
>> close all
```

最初にデータをみてみましょう。引数をひとつだけでプロットすると、Matlab はデータの数が x 軸に用いられます。

```
>>plot(data,'bo')
```

データに何らかの傾向があるかどうかを見るためには、平均をプロットするとよいでしょう。

```
>> hold on
>>plot(mean(data),'r-^')
```

もしくは平均とエラーバーをプロットするために errorbar 関数を使うこともできます。

```
>> errorbar(mean(data),std(data),'g-')
```

ごちゃごちゃしてきていますので、新しい図ウィンドウにグラフを描きましょう。図ウィンドウ figure windows はいくらでもつくることができ、各々の図ウィンドウの中に複数のグラフ multiple plots を描くことができます。

図 2 として新たな図をつくりましょう。

```
>> figure(2)
```

ひとつの図に複数のグラフを配置するには、subplot コマンドを使います。(タイル状に配列するので) サブグラフを縦 (行) にいくつ配列するか、横 (列) にいくつ配列するのか、そして何番目のグラフで作業をするのかを指定しなければなりません。

```
subplot(行，列，作業グラフ)
```

ひとつのウィンドウに 2 行 1 列に配列された (訳注:つまり縦に2つならんだ) グラフをつくり、最初のグラフ (左上から 1,2…とカウントしていきます) に対して作業してみましょう。そして、データをプロットし、タイトルをつけます。

```
>> subplot(2,1,1)
```

```
>> plot(data,'b*')
>> title('Raw data')
>> subplot(2,1,2)
>> bar(mean(data),'r')
```

これは、データの平均を示していますが、折れ線でなく棒グラフを用いています。

```
>> hold on
>> errorbar(mean(data),std(data),'k')
>> title('Mean data')
```

グラフをつくるのに便利な関数は glossary にあげてあります。Matlab で利用できるグラフ関数のリストは、`help graph2d` とタイプすることで見ることができます。

これで練習 E をするのに十分な知識を得られたはずです。

レッスン９ – スクリプトの基本

たくさんのコマンドをうたなければいけない場合、コマンドをひとつひとつタイプしているとわけわからなくなってしまうこともあるでしょう。これを避けるために、Matlab のコマンドウィンドウにタイプするコマンドはすべてスクリプトに書き込むことができます。スクリプトは基本的にコマンドのリストで、コマンドはスクリーンに表示されている順にひとつずつ実行されていきます。

スクリプトは自分の好きなエディタ（メモ帳や emacs など）で編集することができますが、Matlab エディタには利点がいくつかあります。特に、スクリプトの中の特定の単語をハイライトしたり、ループのインデントを自動で設定して視認性を高めることができたりします（ファイル→設定→エディター/デバッガー→タブ→Emacs 形式の Tab キースマートインデントにチェックをいれてください）。また、スクリプトの一部分を選択して F9 を押すと、Matlab はその部分だけ実行します。

スクリプトを作成するには `edit` とタイプします。そうすると、Matlab エディタが起動します（Windows の場合）。このセクションでは、**深緑色** または **紫色** で表示されている内容をエディタに書いていってください（エディタでは紫色には表示されません）。そして、スクリプトを走らせるには、それらを選択して、F9 を押します。もし F9 が機能しないようならば、`edit` とタイプせずにエディタを立ち上げたのかもしれません（もしくは Windows をお使いでないのかもしれません）が、コマンドラインにカット＆ペーストすることによりスクリプトを走らせることもできます。スクリプトは Word と同じように保存したり読み込んだりすることができます。また、コマンドウィンドウにそのスクリプト名をタイプすることでスクリプト全体を走らせることもできます。

スクリプトを書くとき、（あえて前の変数を残しておきたいときを除いて）最初に `clear all` を書いておくことと、そのスクリプトが何かを記載したコメントを書く習慣をつけるとよいでしょう。コメント行は `%` からはじまり、スクリプトを走らせるときに、Matlab はその行を無視します。従って、自分が書くコードに注釈や説明をつけたり、デバッグするときにうまく走らない部分を一時的に機能させなくしたりするときに便利です。コメントはその行が終わるまで書くことができます。

セミコロン `;` はスクリプトを書くときにとても便利な記号です。通常、Matlab にコマンドをうつと、Matlab は計算結果や指定した変数の値を返します。しかしスクリプトを走らせて計算を何百回もするような時、毎回計算結果を見る必要はないかもしれません。行末にセミコロンをつけることによって、Matlab は計算をきちんとしますが、出力が表示されないようにすることができます。

その他のスクリプトを書くときに有用なコマンドに `pause` があります。これは画面の出力を見たいときにスクリプトを一時停止できるというものです。（再開するには何かキーを押してください。）もうひとつ大事なのが **Crtl-C**（すなわち、コントロール Ctrl キーを押しながら C を同時に押すということです）で

す。これによってスクリプトが暴走してしまった時にスクリプトを止めることができ、プロンプトがまた表示されるようになります。

レッスン 6 の最後に使ったデータを使って、3 つのカテゴリーの各々の平均を求め、それらの平均を変数 mdat に入れ、結果をグラフにするというスクリプトを書いてみましょう。以下の行を Matlab エディタに入力してみてください。

```
%% このスクリプトは各カテゴリのデータの平均値をプロットします。
clear all
data = [4 14 6 11 3 14 8 17 17 12 10 18];
cat =  [1  3 2  1 2  2 3  1  3  2  3 1];
mdat(1) = mean(data(cat==1))
mdat(2) = mean(data(cat==2))
mdat(3) = mean(data(cat==3))
figure(1), clf
plot(1:3,mdat,'r-*')
title('Mean of data for each category')
```

Matlab エディタに入力し終わったらすべての行を選択して、F9 を押すことにより実行できます。このようにして簡単な方法で何度も繰り返し解析を行うことができます。Matlab を使ってこのようにグラフを描いたならば、いつでもスクリプトを見直すことで、グラフを描くためにどのようなことをしたのかすぐに確認できます。そして新しいデータセットを入手したとき、数字をちょっと変えるだけで同じ解析がすぐにできてしまいます。こんなわけでデータ解析の時に Matlab でスクリプトを書くのがとても効率がよいわけです。今回のスクリプトを `lesson8.m` として保存してください。後にこのスクリプトを編集します。

レッスン 10 – フロー制御

スクリプトはどのような組み合わせでも表示されている順番にコマンドを走らせることができますが、スクリプトをより便利にするには、スクリプトがいろいろなところで動作するようにしたり止まったりすることができるようになる必要があります。Matlab には 5 つの基本的なコマンドがあります。

If

`If` はもっとも基本的なコントロール要素です。もし、If の後に続く式が真（もしくは数値がゼロ以外）であれば、次のコマンドが実行されます。`If` の後には `else` が続くことができ、If の後に続く式が偽もしくはゼロだと実行します。そして、コマンドを終えるには必ず `end` をつけなければなりません。

```
>> A = 10;
if (A>5)
B = 1
else
B = 0
end
B = 1
```

A の値を変えてみながらこのスクリプトを試してみてください。

`If` 構文はエラーをつかまえるのにしばしば有用です。たとえば、前のレッスンで作成した

lesson8.m に以下のような行を加えてみましょう(これは cat と data を定義した直後に挿入してください)。

```
if( length(cat) ~= length(data) )
        disp('ERROR: Data and categories are not the same length')
        return  %%return はスクリプトを止めることを意味します
end
```

このようにすることでこのスクリプトはもし間違ったデータが入力されると、エラーメッセージを出してくれるようになります。

For

For は一連のコマンドを複数回繰り返します。For にはループを何回繰り返すかを指定するカウンタ counter があります。カウンタはループの中でインデックスとしても変数としても使うことができますが、**ループの中で変更してはいけません。**

下の例では、最初のループは i=1 で走ります。ですから、A(1) = 2 となります。次に i=2 としてループが走りますので、A(2) = 4 となり、以降同様に続いていきます。ここで i の値は連続した数値をもつベクトルを作る時と同じ要領で指定されています(コロンのセクションを確認してください)。For ループはいつもループの終わりを指定する end がなければいけません。なお、break を使うことで強制終了することもできます。

```
for i=1:4
   A(i) = i*2
end
```

```
A = 2
A = 2 4
A = 2 4 6
A = 2 4 6 8
```

2 つ目の例として、再び lesson8.m をみてください。その中の mdat を計算する 3 行を for ループで置き換えてみましょう。各行で変化する数字のところにカウンタ i を置くことによって、各カテゴリーの平均を求め、その結果を mdat にいれていきます。今は 3 カテゴリーしかないので、そんなに時間の節約にはなりませんが、もし、カテゴリーが 100 あったら、時間の節約になることがわかるかと思います。

```
for i=1:3
end
mdat(i) = mean(data(cat==i))
```

Switch

Switch ループは、if を次々に何回も行うようなものです。選択肢が複数あるときに便利です。Switch の後に変数を丸括弧の中にいれます。各々の case は可能性のある選択肢で、その値が適合すれば、次のステートメントが実行されます。どれにもあてはまらなければ、otherwise の後のステートメントが実行されます。switch 構文は最後に end がなければなりません。(disp はテキストを表示するという意味です) Switch ループはあるテキストが他のテキストと合致しているかどうかを決定するのにも用いることができて、テキストの分類にも役立ちます。

```
>> A = 3
switch(A)
  case 1
    disp('A is one')
  case 3
    disp('A is three')
  case 5
    disp('A is five')
otherwise
    disp('A is not one or three or five')
end
'A is three'
```

While

while は連続した for ループのようなもので、値が偽 false になるまで続きます。下の例では、ループ中の y は 1 より大きくなるまで増え続けます。その後、x は 2 にセットされ、ループが終了します。ここでもそうですが、ループを終わらせるには end が必要です。while ループは長さの分からないテキストファイルを読み込むときにもっとも便利です。

```
x = 1;
y = -5;
while(x==1)
  y = y+1
  if(y>1)
    x = 2
  end
end
y = -4
y = -3
y = -2
y = -1
y =  0
y =  1
y =  2
x =  2
```

Try ... Catch

これは実際には制御文ではなく、エラーを見つけてスクリプトがクラッシュするのを防ぐために用いられます。Matlab に何かを try するように指示し、もしエラーが出るようだったら、別の指示のもとにエラーを catch させます。これは変則的なテキストやデータが入っているようなファイルを処理するときに便利です。以下の例をみてみましょう。

```
>> A = 1:10;
>> B = 1:5;
```

```
for i=1:length(A)
  try
        C(i) = A(i) + B(i) %% i が B の長さより大きいとエラーになります。
  catch
        disp('B is too small')
  end
end
C = 2
C = 2 4
C = 2 4 6
C = 2 4 6 8
C = 2 4 6 8 10
B is too small
```

レッスン 11 – 関数

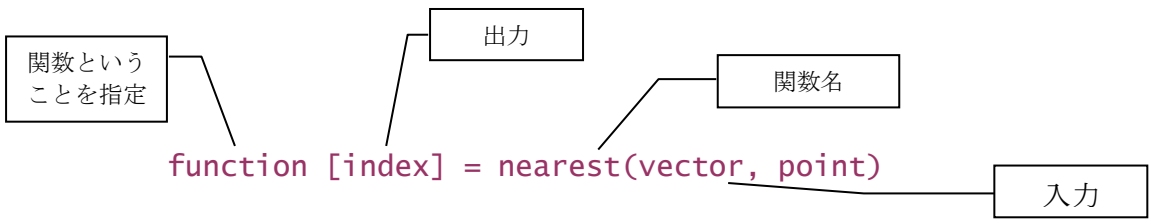
スクリプトは、たくさんのコマンドを一気にコマンドウィンドウにタイプしていくようなもので、長く複雑にすることができます。もし頻繁に用いるシンプルで小さなタスクがあるなら、関数 function を書いておくと、もっと簡単になります。関数に入力すると、出力を得ますが、関数の中で起きていることはメインワークスペースから隠されている、もしくは、被包 encapsulated されています。これにより、関数内で用いられる変数とワークスペースの変数が同じ名前であったとしても、お互いに干渉しあったり、お互いの価が上書されたりすることはありません。

このルールの例外は、グローバル変数です。変数が関数の中でグローバル変数と宣言され（つまり、関数の一番最初に `global var` ではじまる行が含まれている）、かつ、他の関数やスクリプトでも同様にグローバル変数として宣言されると、その変数はもはや関数の中に隠されることなく共有される値となります。つまり、ある場所で変数の値を変更すると、他のところに影響を与えることとなります。例として、SPM を使っている場合、変数 `defaults` はグローバル変数です。これが意味するところは、SPM を起動後、コマンドウィンドウで `global defaults` とタイプすると `defaults` 変数の中を見ることができるということです。これを見ながらデータ処理のためのデフォルトの設定を編集することができます。

関数はスクリプトのように書きますが、`function` ではじめなければならず、続いて、`[outputs] = function_name(inputs)` の形式をとらねばなりません。入力や出力はどのような形（配列、文字列など）でもとることができます。

関数名の後に有用な情報を記載しておく習慣をつけておくのがいいでしょう。関数名の直後に% ではじまるすべての行はその関数のヘルプファイルとなります。従ってその関数が必要とする入出力の形式を記載しておくと便利でしょう。

以下の例は、あるベクトルで、特定の値(point)にもっとも近いインデックスを見つける関数です。Matlab エディタで新規作成にて以下の内容を記載し、nearest.m として保存してください。



```
function [index] = nearest(vector, point)
%% [index] = nearest(vector, point)
%% この関数はベクトルの数値インデックスで、絶対値が point に最も近いものを見つけます。
%% もし、合致するものが一つ以上あるならば、最初の一つだけが返されます。
df = abs(vector-point);
ind = find(df==min(df));
index = ind(1);
```

コマンドウィンドウからこの新しい関数を試してみましょう。どのように動くかを確認するには、`help nearest` とタイプしてみてください。`r=rand(50,1)` とタイプして乱数を生成し、0.5 に近いデータを見つけてみましょう。(訳注: `nearest(r,0.5)`となります。)それが正しいかプロットして確認しましょう。

パス

スクリプトや関数を走らせる時、あるいは、誰か他の人の関数を使用するとき、Matlab はどうやってどのプログラムを走らせるかわかるのでしょうか？スクリプトが少ししかない時はさして重要でないようにみえるかもしれませんが、すぐに大変重要なことになります。原則として、Matlab は最初にカレントディレクトリを探しに行き、次に**パス**の中にあるディレクトリを上から順に探していきます。Matlab のパスを確認するには、`path` とタイプします。パスを編集するには、**ファイル→パスの設定**を選択します。Matlab がどのコマンドを走らせているのかすぐに確認したい場合には、`which` に続けてコマンド名をタイプするのが便利です。

これで練習 F をするのに十分な知識を得られたはずです。

レッスン 12 – 変数についてもう少し

データの保存と読み込み

Matlab はどんな変数でも.mat ファイルに保存することができますので、後でいつでもそれらの変数を読み込むことができます。ファイル名はどんなに長くても大丈夫ですが、ファイル名の先頭に数字がついたり、ファイル名にスペースを使うことはできません。`save`, `load`, `clear` コマンドは括弧ではなく、スペースを用いてファイル名などを指定します。コマンド名に続いて保存、クリア、読み込みたいファイルをタイプするだけです。

変数 x, y, z と data を somestuff.mat に保存したいときには、以下のようにタイプします。

```
>> save somestuff x y z data
```

すると、somestuff.mat と名前のついたファイルがカレントディレクトリに保存されます。このファイルには変数 x, y, z, data が入っています。

次に、変数をすべてクリアしてみましょう。

```
>> clear all
```

変数はすべて取り除かれます。`whos` とタイプすれば変数が何もないことがわかります。

```
>> load somestuff
```

こうすると、変数 x, y, z, data が元に戻っています！

保存する際には SPSS や Excel で読み込むことができるように異なる形式で保存することができます。コマンド dlmwrite は行列をタブ区切りのテキストファイルとして保存します。このファイルは SPSS や Excel やその他のどんなソフトでも読み込むことができます。

タブ区切りのテキストファイルを書き出しましょう。'\t' はデリミタ（データの区切り文字）で、他の文字を使うこともできますが、これまでのところタブが他のソフトウェアともっとも互換性の高い区切り文字だと思います。

```
>>  dlmwrite('mydata.txt',data,'\t')
```

こうすると、mydata.txt というファイルがカレントディレクトリに保存されます。

入出力関数（つまり、データを読み込んだり保存したりする関数）の一覧は、help iofun とタイプすることで見ることができます。

文字列とセル配列

これまで扱ってきたすべての変数は数字でしたが、他のデータ形式を操作できると便利なこともあります。テキスト（すなわち文字列）は数字と同じように変数に割り当てることができます。その場合、テキストはシングルクォートでくくらなければなりません。

```
>> firstname = 'Joe';
>> surname = 'Bloggs';
```

文字列は 2 つのベクトルと同じように結合することができます。たいていの場合、その間に' '（シングルクォートの中にスペースがひとつ入っています）を使ってスペースをいれてあげると見栄えがよくなります。

```
>> fullname = [firstname,' ',surname]
'Joe Bloggs'
```

数字を文字列に変換して表示させるには、num2str が使えます。

```
>> age = 25;
>> agestring =[fullname,' is ',num2str(age),' years old']
'Joe Bloggs is 25 years old'
```

文字列を配列にいれるのはあまり都合がよくありません。なぜなら、Matlab は、配列の要素が同じ長さでないとうまく動かないからです。しかし、セル配列ならば行列と同じように扱うことができ、各要素の大きさを気にする必要がありません。セルは行列と同じように指定することができますが、丸括弧 () でなく、波括弧 {} でくくります。

```
>>   stimuli={'dog','cat','horse','rat';'car','train','hammer','van'}
stimuli =
    'dog'  'cat'   'horse'    'rat'
    'car'  'train' 'hammer'   'van'
```

同じ行の要素を区切るのにコンマを使っていて、新しい行をはじめるためにセミコロンを使っていることに注意してください。これはふつうの行列を扱う時と同じです。各要素を参照する時は波括弧を使います。

```
>> stimuli{1,3}
ans =
     horse
```

行や列を移動するのは行列と全く同じです。ここでは丸括弧を使っています。そうすることで各要素が4つの異なる要素として移動するのではなく、配列として移動することになります。その結果は animals という名前の新しいセル配列となります。

```
>> animals = stimuli(1,:)
animals =
    'dog'  'cat'   'horse'    'rat'
```

セル配列には文字列と数字を混在させることができます。セル配列に他の行列を配置することすらできます。しかし、もっとも有用なのは文字列だと思います。Matlab には文字列を操作する様々な関数が用意されています。その中には、セル配列の中の特定の文字列を探すようなものがあります。

```
>> strmatch('cat',animals)
ans =
2
```

セル配列から得られる数字を数値配列に変換するには、cat を使います。

```
>> nums = {10, 15, 8, 12};    %% セル配列としての数字
>> realnums = cat(1,nums{:}); %% 数値配列に変換
```

より詳しい情報は help strfun および help cell とタイプしてください。

構造体配列

解析をやっていると、whos とタイプすれば 30 ぐらいの変数が表示され、その変数が何なのか、そしてどの変数が大事なのか思い出せなくなってきます。こんなとき、いちばんの解決方法は構造体配列 structure を使って変数を整理することです。構造体配列はデータを整理するために階層構造となっていて、ひとつの変数が複数のフィールドを持つことができます。それらのフィールドはドット(.)からはじまります。たとえば、データが言語タスクと絵画タスクの得点から構成されているとするならば、word_data と pic_data という 2 つのベクトルデータを準備しているかもしれません。これらの変数を扱うために lesson3.mat を読み込んでください。これらの変数を data という構造体配列に被験者の名前と年齢とともにいれることができます。

```
>> data.word = word_data;
>> data.pic = pic_data;
>> data.subjectname = 'Joe Bloggs';
>> data.subjectage = 25;
```

構造体配列に何が入っているか確認するために、data とタイプしてみてください。

フィールドの名前をタイプすることで data の中のどんなフィールドも参照することができます。構造体配列は異なるタイプのデータをまとめるときにもっとも便利で、大事な変数をすぐに見たり保存したりできます。構造体配列は多重階層にできます（各フィールドがサブフィールドをもつことができます）し、複数のデータを配置できます。たとえば、先ほどの構造体配列に 2 人めの被験者のデータを入れることができます。その場合、data(2)として指定します。

```
>> data(2).word = word_data2;
>> data(2).pic = pic_data2;
>> data(2).subjectname = 'John Doe';
>> data(2).subjectage = 26;
```

すべての被験者のデータを同一のプロセスで解析するために同じフォーマットで保存したいようなときにこのような方法は便利です。

システムコマンド

これはデータ形式の話ではありませんが、知っていると便利なトピックです。Matlab のコマンドウィンドウから、コンピュータを制御することができます。システムコマンドである dir , ls , pwd , cd は UNIX や DOS コマンドプロンプトのようにカレントディレクトリのファイルを表示したり、ディレクトリを変更したりできます。さらに system (Windows の場合) または unix (UNIX の場合) コマンドはコマンドをシステムに送ります。たとえば、

```
>> system('copy data.txt processed_data.txt')
```

このコマンドは data.txt ファイルを processed_data.txt の名前でコピーします。これらのコマンドを for ループで使うことによってファイルをあるディレクトリから別のディレクトリに移動したり、ファイル名を変更したりできます（特に画像データの場合に便利です。ファイルが何百もありますので）。

他の便利なコマンドは eval です。任意の文字列を Matlab 上の数式として評価して実行する関数です。なので、複数の数式（異なる名前をもつ変数など）を文字列として記載して、Matlab に実行させることができます。

これで練習 G, H をするのに十分な知識を得られたはずです。

レッスン 13 – グラフ応用編

Matlab のグラフ機能でたいていのことはできますが、ときどきもっといろいろしたくなることがあります。Matlab では、コマンドラインから図の細かい設定をすることができます。しかしこれは非常に複雑です。詳しくは Matlab のグラフィックに関するマニュアルかヘルプ、もしくは Mathworks のウェブサイト www.mathworks.com を御覧ください。図に関するたいていのことは編集したいと思う線やテキストをクリックするか、図のウィンドウの編集・ツールメニューからできます。ここではいくつか基本的なことを述べたいと思います。

グラフィックスのハンドル

すべての図、サブグラフ、軸、グラフの線などにはすべてハンドル（訳注：ハンドル handle には名前という意味があります。さまざまな設定をするために各々の機能に固有の名前がついているととらえてください。）がついており、このハンドルを使って Matlab に図のどこを編集したいかを指示することが

できます。たいていの場合、ユーザーからハンドルは全く見えませんが、知る必要も気にする必要もありません。ただし、いくつかのハンドルは知っていると便利です。

gcf は現在扱っている図に対するハンドルです。コマンド get(gcf) により、現在の図のすべてのプロパティを見ることができます。このプロパティを見れば、何を編集すればいいのかわかります。たとえば、'Position' プロパティを使うと、Matlab に画面のどこに図を表示させるかを指定することができます。

```
set(gcf,'Position',[50,50,600,800])
```

これが意味するところは、現在の図の左下の角を画面の左下の角からみて横に 50 ピクセル、縦に 50 ピクセルの場所に配置し、図の大きさを横 600 ピクセル、縦 800 ピクセルにするという意味です。

gca は現在の軸に対するハンドルです。先ほどと同じように get(gca) を使えば編集できるすべてのプロパティが表示されます。軸の目盛り上に表示するラベルを変えるには次のようにします。

```
set(gca,'XTick',1:4','XTickLabel',{'cat','dog','car','boat'})
```

これによって x 軸に 4 つの目盛りがふられ、各々の目盛りに'cat', 'dog' などの言葉がついたのわかると思います。

グラフの線に対するハンドル Line handles はグラフ関数を出力に指定することによって得られます。たとえば、

```
h = plot(1:10, rand(1,10))
```

この場合、h はグラフの線に対するハンドルになります。この h を使って線の太さや色を変えることができます。

```
set(h,'LineWidth',3)
```

グラフィックスのハンドルについての詳細は Matlab のグラフィックスマニュアルに記載されています。

画像

グラフの線だけでなく、Matlab は行列を画像として表示することができます。基本となるコマンドは image と imagesc です。ともに行列をカラーパターンとしてプロットしますが、imagesc は見栄えが良くなる様に色合いを自動で調整してくれます。colorbar コマンドは画像の横にカラースケールを表示し、colormap によって使われる色を変更できます。使うことのできるカラーマップの一覧は Help graph3d を見てください。

```
x = linspace(0,4*pi, 100);
y = linspace(0,2*pi,100);
c = sin(repmat(x,100,1))+cos(repmat(y',1,100));
figure(1), clf
imagesc(x,y,c)
colorbar
```

コマンドウィンドウから変数 c を見るだけでは、ただの数の羅列にしか見えませんが、imagesc を使うことによってパターンが見えてきます。たくさんのデータを概観するときには便利な機能です。ときにこのようにしてデータを見るだけでデータがおかしい被験者を見つけることができます。

3 次元のグラフ

Matlab は 3 次元でのグラフを描くこともでき、そのグラフを拡大したり動かしたり自由にできます。3 次元のグラフ機能ついての詳細は help plot3d で見ることができます。一番便利なのは、surf で、3 次元のグラフの表面を描きます。たとえば、先ほどのカラーで表示した図は 3 次元で描くことができます。

```
x = linspace(0,4*pi, 100);
y = linspace(0,2*pi,100);
c = sin(repmat(x,100,1))+cos(repmat(y',1,100));
figure(2), clf
surf(x,y,c)
```

図ウィンドウの上に、丸で囲まれた矢印のアイコンがあります。これをクリックすると、グラフが回転したり移動したりすることができ、見栄えを良くすることができます。その他にも、視点、光のあたり具合や表面の反射など、仮想世界を創るのに必要なことはどんなことでも幅広く調整することができます。詳しくは Matlab のグラフィックスマニュアルを見てみてください。

写真

データを図示できるように、Matlab は　jpeg 形式、gif 形式、avi 形式といった写真や動画を読み込み表示することができます。この機能は写真を表示・編集したりこじゃれた画像をつくったりするのに有用です。以下のスクリプトはビットマップ画像を読み込み、注視点を追加し、別名で保存します。このようなスクリプトを使うことで、実験のための呈示刺激をさっと作ることができるようになります。スクリプトでは dog.bmp という画像ファイルを使うことを想定しています。当然ながらスクリプトのファイル名を変更したり、このファイルを作成してください。

```
clear all, close all
picname = 'dog.bmp'
%% 画像を読み込む
dog = imread(picname)
[m,n,p] = size(dog)
%% 画像を表示する
figure(1), clf
image(dog)
%% 図のプロパティを適切に設定する
set(1,'Position',[50,50,m,n])
set(gca,'Position',[0,0,1,1],'XLim',[0,m],'YLim',[0,n])
axis off
hold on
%% 注視点のための＋を描く
h=text(m/2,n/2,'+')
set(h,'FontSize',20,'FontWeight','Bold', ...
'HorizontalAlignment', 'center')
frame = getframe(gca);
%% 図から画像を抽出する
dog_fix = frame2im(frame);
```

```
%% 新しいファイル名で画像を書き出す
imwrite(['fix_',picname],dog_fix)
```

このスクリプト全体を for ループにいれることでたくさんの画像に対して一度に同じ処理をすることができます。ですから最初にスクリプトを書くのに多少の時間がかかったとしても、処理しなければならない画像が何百枚もあるならば、スクリプトを書くほうがいちいち Photoshop で加工するよりもずっと早くかつ信頼性の高い結果を得ることができるわけです。

レッスン 14 – Matlab へのデータファイルの読み込ませ方

「どのようにしたら Matlab に自分のデータファイルを読みこませることができますか？」はしばしば質問されることの一つです。答えは…「あなたのデータファイルによります」。データをいろんなプログラムで使えるようにする一番簡単な方法は、タブ区切りテキストを準備することです。タブ区切りテキストでは行列は以下のようにあらわせます。

```
数 タブ 数 タブ 数 … 改行
数 タブ 数 タブ 数 … 改行
... etc
```

タブ はそこにタブ記号が入ることを意味し、改行のところで行が改行されます。
各行には同じ数の列がなければならず、文字列が入ってはいけません。この形式は、Excel, Matlab, SPSS など、どんなプログラムでも互換性がありますので、もしデータをこの形式で準備できるならばあなたの人生はずっと楽になることでしょう。Matlab に読み込むには、dlmread を使い、行列をタブ区切りテキストとして書き出すには dlmwrite を使います。

しかしながら、たくさんのプログラムは、データをとてもごちゃごちゃにした形にして返してきます。そこには、それぞれ意味が異なる単語や数字がテキストファイルのそれぞれの列に並んでいます。このような場合、これらのファイルを読み込むための独自のスクリプトを書かなければなりません。試行錯誤を繰り返さなければなりませんが、一度うまく読み込めるスクリプトが書けたら、データ処理はずっと早くできるようになります！

データファイルを読み込むための基本的なスクリプトは以下のようになります。このスクリプトでは、各行が TIME, STIM, KEY のどれかの識別子からはじまり、その後に何らかのデータあると仮定しています。

```
%%% 自分のデータファイルを読み込むためのスクリプト
fname = input('Enter the file name ')   %% ファイル名を尋ねるプロンプト

fid = fopen(fname)   %% ファイルを開き、ファイル ID を取得する
fline = fgets(fid)   %% ファイルの行を読む
w = 1;       %% カウンタ
while(fline~=-1) %% ファイルが終わると fline が-1 になる
                 %% そうなるまで処理を続ける
      switch fline(1:4)   %% 各行の最初の数文字に基づいた switch ループ
      case 'TIME'
            trial_time(w) = str2num(fline(6:10))   %% 時間の読み込み
      case 'STIM'
```

```
            stimulus{w} = fline(6:length(fline));   %% 刺激の読み込み
        case 'KEY '
            key(w) = str2num(fline(6)) %% キーヒットの読み込み
            w = w+1;   %% カウンタを1増やす
        end
end
fclose(fid)   %% ファイルを閉じる
```

たいていのデータファイルはこの例よりもずっと複雑ですが、これからはじめれば、自分の必要に応じてそれをあてはめていくことができるでしょう。テキストを扱う際の便利な機能には num2str と str2num があり、これらは数字と文字列を交互に変換します。isspace は空白のスペースを見つけます。 isletter は文字を見つけます。strmatch, strfind, strcmp, findstr はある行の中にある文字列を見つけるためのものです。ファイルの入力と文字列の取り扱いについては、help iofun や help strfun をタイプしてみてください。

レッスン 15 – 解析を最初から最後まで通して行うスクリプトの例

は、いつの日かここに記されるでしょう…。

心理のための Matlab チュートリアル: 練習

練習 A: データ操作

はじめる前に `clear all` とタイプすることを忘れないでください。

1) 次の行列とベクトルを入力してください。

```
a =    9  12  13   0
      10   3   6  15
       2   5  10   3
b =    1   4   2  11
       9   8  16   7
      12   5   0   3
```

2) `a` と `b` を使って以下の行列を作ってください。
   a) `c` は行列 `a` の第 3 行第 3 列にある要素です。
   b) `d` は行列 `a` の第 3 列です。
   c) `e` は行列 `b` の第 1 行と第 3 行です。
   d) `f` は行列 `a` と行列 `b` を上下に結合したものです。(訳注: a, b の順に 6 行になります)
   e) `g` は行列 `b` の第 4 列 の隣に行列 `a` の第 1 列がくる行列です。

3) 行列のいくつかの要素を変更してください。
   a) 行列 `e` の要素 (2,2) を 20 にしてください。
   b) 行列 `a` の第 1 行をすべてゼロにしてください。
   c) 行列 `f` の第 3 列を 1 から順に 6 までしてください。
   d) 行列 `a` の第 1 列を行列 `b` の第 2 列にしてください。

練習 B: 計算と関数

はじめる前に `clear all` とタイプすることを忘れないでください。

1) 次の行列とベクトルを入力してください。

```
A = 1  5  6
    3  0  8
B = 7  3  5
    2  8  1
C = 10
D = 2
```

2) 以下の計算を行なってください。

```
E = A - B
F = D*B
G =
A.*B
H = A'
J = B/D
```

3) 行列の部分に対して計算を行いましょう。
   a) 行列 A の第 1 列を行列 M に代入してください。
   b) 行列 G の第 2 列を行列 N に代入してください。
   c) M と N を足してください。
   d) 行列 A の第 3 列のみを C でかけあわせ、結果を行列 A の第 3 列に戻してください。いくつかの段階を経てもかまいませんが、一行でこの計算を行うことができます。
   e) 行列 H の第 D 行（すなわち第 2 行）を見つけ、その行のすべての要素の合計を出してください。
   f) 最初の 2 行は行列 A、次の 2 行は行列 B でできた行列 K をつくってください。

4) 行列の要素の掛け算と行列の掛け算の違いをみて見ましょう。
   a) `A*B` をしてみてください。（エラーとなります）
   b) Try `A*B'`（これは、行列 A と行列 B を転置したものを掛けあわせるということです）と `A.*B`（これは行列 A と行列 B の要素を各々掛け合わせるということです）の結果を比べてみてください。

5) 行列 J の各列の最大値と行列 B の各行の最小値を見つけてください。必要であれば、`help max` および `help min` とタイプしてみてください。

練習 C: 論理値

はじめる前に `clear all` とタイプし、データセット ex_C.mat を読み込んでください。このデータセットには 3 つのベクトルが含まれています。
`rt` は一人の被験者の視覚的注意タスクを行った時の反応時間のベクトルです。
`cue` はキューが適切(1)か、不適切(2)か、キューそのものがなかった(3)かをあらわすベクトルです。
`side` はターゲットが画面の左(1)か右(2)にあらわれたかを意味するベクトルです。

1) 適切なキューのもとで行われた試行を定義するための論理ベクトル `valid` を作ってください。
2) これを用いて適切なキューがあったときの反応時間の平均値と標準偏差を求めてください。
3) 反応時間が 100 msec より短いか 1000 msec より長い場合はその試行はエラーとします。エラー試行を定義する論理ベクトル `error` を作ってください。
4) `error` と `valid` を使って適切な試行の平均および標準偏差を求めてください。
5) 呈示刺激を左右どちらにするかを決めるときに使える論理ベクトル `side` を作ってください。
6) すべての試行において反応時間は左右で変わるでしょうか？適切なキューのもとで行われた試行だけではどうでしょうか？

練習 D:　実際のデータセット

フレッドは、魅力の知覚についてのパイロット研究を行なっています。彼は友人に 20 の顔を見て魅力的かどうか判断してもらうようにお願いしました。各々の顔の性別と対称性は前もってわかっています。変数をクリアしてデータセット ex1.mat を読み込んでください。このデータセットには 3 つのベクトルと 1 つの行列が入っています。

`stimno`（刺激の番号）
`gender`（1 = 男性, 2 = 女性）
`symm`（1-5 のスケールで評価されている対称性）
`data`（3 人の被験者が各々1-100 のスケールで評価した魅力の点数）

1) フレッドは被験者 2 の呈示刺激 3, 5, 17 を書き間違えました。正しい値は 30, 62, 35 です。修正してください。
2) 被験者 3 のつけた魅力のスコアの平均、最小、最大値を求め、各々の顔に対してつけた値の範囲を求めてください。
3) 被験者 1 は男性の顔と女性の顔のどちらをより魅力的と評価しているでしょうか？
4) 被験者 3 は、対称性が高い（4 か 5）顔を低い（1 か 2）顔よりも魅力的と評価しているでしょうか？
5) 被験者 2 と 3 の評価の一致しているところはありますか？どれくらいあるでしょうか？
6) 被験者 1 と 2 の不一致の平均値を求めてください。（`abs` を使いましょう）
7) 提示された顔のうち、4, 10, 12, 18 は他の顔よりもずっと年をとっています。彼らは魅力的でないと評価されているでしょうか？
8) データを刺激の番号に従って並べ替えてください。（`sortrows` を使いましょう）

練習 E: グラフ基本編

はじめる前に `clear all` と `close all` をタイプしてください。

1) グラフをいくつか描いてみましょう。
   a) 値が 1 から 10 までのベクトル X をつくってください。
   b) X を 2 乗したベクトル Y をつくってください。
   c) X を 9 倍したベクトル Z をつくってください。
   d) Figure 2 に X と Y をプロットしてください。グラフの線は赤とし、各々のデータは星印(stars)で示してください。（必要ならば `help plot` とタイプしてください）
   e) そのグラフを残したまま、X と Z をプロットしてください。グラフの線は緑で、各々のデータは四角で示してください。
   f) 図にタイトルと凡例をつけてください。

2) Exercise2.mat を読み込んでください。このファイルには 3 つのベクトルが入っています。`iq`, `scoreA`, `scoreB` の 3 つで（架空の）被験者のテストの点数です。
   a) 青の丸印を使って scoreA と iq のグラフを描いてください。
   b) 同じグラフ上に赤い四角を使って scoreB と iq のグラフを描いてください。
   c) 軸にラベルをつけてください。
   d) statistics toolbox を持っているならば、`lsline` をタイプすして最小二乗法を用い

てこのデータにもっともよくフィットする線を描いてください。

e) 新しいグラフに、棒グラフを用いて scoreA と scoreB の平均を比べてみてください。2 つの平均値をもつ新しいベクトルをつくるほうが簡単かもしれません。
f) 棒グラフにエラーバーをつけてください。
g) 2 つのサブグラフをもつ図を作りましょう。scoreA の分布を（`hist` を使って）subplot1 に描いてください。Subplot2 には scoreB の分布を描いてください。

練習 F: スクリプトと関数

1) 練習 C を振り返り、練習 C を実行するためのスクリプトを書いてください。そのスクリプトでは適切なキュー、不適切なキュー、キューなしの 3 つの条件での平均値のグラフも描くようにしてください。
2) 繰り返しの構文を書く練習をしてみましょう。
    a. For ループを使って、3 つの乱数の平均値を 20 回求め、その結果をベクトル A にいれてください。
    b. 2 番めのループには、30 の乱数の平均値を 20 回求め、その結果をベクトル B にいれてください。
    c. A と B の標準偏差を求めてください。
    d. 図 2 をクリアし、2 つの分布を比べられるようにプロット A とプロット B を異なるサブグラフに描いてみてください。（ヒストグラムを描いてみましょう！）
3) 本文の例にある switch ループを試してみましょう。
    a) A の異なる値に対してループを走らせてみてください。
    b) ループを修正して A が 0 か 1 か 2 か示すようにしてみてください。
    c) Switch 文を修正して、A を 3 でわったときの余りを出力するようにしてください。（`rem` コマンドを使ってみてください）
    d) `switch` ループ全体を `for` ループの中にいれ、A を 50 から 70 までカウントするようにしましょう。そのプログラムでは、A の各々の値に対して 3 でわったときの余りが表示されるはずです。
4) ベクトルの標準誤差を計算する関数を下記、それが実際に動くかどうか試してください。

練習 G: 構造体配列とセル配列

この練習をするためにスクリプトを書いてもらってもかまいません。

1) 次の単語をセル配置にいれてください。すべての単語が 1 列になるようにしてください：
hat coat gloves shoes socks vest jacket gown
2) セル配列の第 2 列に、各々の単語の頻度を入力してください。Hat は高頻度 (high)、coat, shoes, jacket は中頻度 (medium)、その他は低頻度 (low)です。
3) ex_F.mat を読み込んでください。このファイルには行列 ld が含まれており、これは 30 人の被験者が上の 8 つの単語を識別するのにかかった時間です。各々の単語の識別時間の平均値および標準誤差をプロットするスクリプトを書いてください。（セル配列から得た）単語を X 軸のラベルに用いてください。
4) データを高頻度、中頻度、低頻度の順にソートしてください。そのために for ループと switch ステートメントを使ってセル配列の各単語を分類してください。
5) 高頻度、中頻度、低頻度の識別時間の平均値を棒グラフにプロットし、適切な見出しと軸ラベルをつけてください。
6) 解析結果を構造体配列に入れ、保存してください。単語リストと各単語の平均値と平均

誤差を保存してください。

練習 H: 実際のデータセット パート２

ex2.mat にはある被験者のストループテストのデータが入っています。

`cong` - 1=一致 congruent, 2=中立 neutral, 3=不一致 incongruent
`stim` - 1=単語, 2=カラー
`data` – 反応時間
`err` – 誤答率, 1=エラー

1) 色に一致している反応時間の生データをプロットしてください。
2) すべてのエラー及び反応時間で 200 msec 未満もしくは 1000 msec より長いものを除いてください。
3) 除外率は条件によって異なっていますか？
4) ヒストグラムを用いてデータが正規分布かどうか確認してください。
5) for ループを用いて各条件の反応時間の平均時間および標準誤差を求め、それを構造体配列にいれるスクリプトを書いてください。
6) 上記 5 の結果をエラーバーとともにプロットしてください。
7) 2 人めの被験者のデータ(ex2b.mat) を読み込み先ほど書いたスクリプトで解析してください。

Matlab 用語集

用語集 – 定義

スカラー Scalar	単一の数字 `D = 10`

ベクトル
Vector
	1 行もしくは 1 列の数字
	`A = 1 2 3`
	`B = 4`
	`3`
	`8`

行列
Matrix
	数字を格子に配列したもので、大きさは行数×列数であらわされます。行列 C は 3 行 4 列の行列です。
	`C = 5 6 7 9`
	`8 3 5 3`
	`5 6 3 2`

要素　Element	行列やベクトル中のひとつの数字
変数 Variable	Matlab のワークスペースに保存されているものすべて。変数を確認するには whos とタイプします。
論理配列 logical array	1 と 0 からなるベクトルまたは行列。1 は真 TRUE を意味し、0 は偽 FALSE を意味します。
セル配列 cell array	波括弧{}であらわされる配列で、大きさは問いません。セル配列は特に単語リストや単語と数字がまざっている場合などに有用です。セル配列は通常の行列と同じように参照したりインデックスとして使うことができますが、当然単語に対しては計算はできません。
ワークスペース workspace	Matlab が使うメモリーで、変数が格納されています。変数を確認するには whos コマンドを使います。
引数 Argument	関数への入力の際に指定するものです。

関数
Function
	入力（引数）を受け、それに対してなんらかの操作を行い、そして出力を返すコードのことです。Matlab での関数は被包化 encapsulated することができます。つまり、関数の中の変数はユーザーから見えませんし、関数そのものもワークスペースの中の変数を見ることができません。関数の例には以下のようなものがあります。
	`A = [5 3 4 2 7]`
	`B = rem(A,3)`　%% rem は Matlab の組み込み関数で、第 1 の引数（A）を第 2 の引数（3）でわった余りを返します。
	`B = [2 0 1 2 1]`
	Matlab には多くの組み込み関数があります。Cognet や SPM にも多くの関数があります。自分自身で関数を書くこともできます。

スクリプト script	コマンドを実行順に書いていったものです。関数に似ていますが、スクリプトは被包化されていませんので、ワークスペースの変数を自在に扱うことができます。

コメント comment	スクリプトや関数の中で% からはじまる行は実行されません。それらはコメントで、そのコードが何をするか思い出すために使うことができます。
パス path	Matlab が関数を探すディレクトリのリストです。パスを見るためには、path とタイプしてください。Matlab はパスの中のディレクトリを上から順に見ていき、最初に名前が合致する関数を使います。同じ名前の関数が異なるディレクトリにある場合、which name とタイプすることでどのバージョンを Matlab が使っているか確認することができます。メニューのファイル→パス設定を使うとパスエディタが起動し、Matlab のパスをいじることができます。

用語集 – 演算子

より詳しい情報を得るには以下をタイプしてください: `help ops, help paren, help punct`

; （行末）	画面に出力が表示されるのを抑制します。（Matlab のワークスペースには保存されています。）

; （行の途中）	ベクトルや行列の新しい行をはじめます。
	例: `A = [2, 3; 5, 7]`
	`A = 2 3`
	`5 7`

, （行の途中）	行列の要素を区切ります（コンマあるいはスペースが使えます。）
[ ] 角括弧	角括弧を使って行列を作ったり、行列をつなげたりすることができます。行の要素はコンマかスペースで区切り、新しい行をつくるにはセミコロンを使います。
( ) 丸括弧	1) 丸括弧を使ってインデックスとして行列の要素を取り出すことができます。 2) 関数の引数として丸括弧を使います。 3) 計算の順序を決めるために丸括弧を使います。
{ } 波括弧	単語などをセル配列に配置するために波括弧を使います。 `B = {'dog', 'cat', 'rabbit'}`
: コロン	1) コロンが単独で使われると、行または列すべてを意味します。 `M(:,2)` は行列 M の第 2 列全てという意味です。 `N(3,:)` は行列 N の第 3 行のすべての列という意味です。 括弧のなかにコロンだけがあると、すべてのものを列ベクトルに変換します。`M(:)` は行列 M のすべてを長い 1 列のベクトルに変換します。 2) 2 つの数字の間では、コロンは A から B にカウントします。 `C = 5:8` は C = 5 6 7 8 ということを意味します。 `M(:,5:8)` は行列 M の 5, 6, 7, 8 列のすべてということを意味します。 3) 3 つの数字を 2 つのコロンでつなげると、数字をカウントしていく際の間隔を指定することができます。 `D = 3:4:20` は D は 3 から 20 まで 4 刻みでカウントするという意味です。 `D = 3 7 11 15 19`
...	行が終わることなく次の行に続いているということを意味します。コードの 1 行が非常に長いときに便利です。
. ピリオド	1) 小数点 2) 要素の計算（下記参照） 3) 構造体配列のフィールドへのアクセス
'hello' シングルクォート	シングルクォートでくくられたものはなんでも Matlab は文字列として扱います。

=	等号。ある値を変数に割り当てるときに使います。たとえば、A = 10 は Matlab のワークスペースに 10 という値をもった変数 A を保存します。
+ -	プラス、マイナス
* /	行列の掛け算、割り算
.*	要素ごとの掛け算
./	要素ごとの割り算
^	行列の累乗
.^	要素ごとの累乗（つまり、A.^2 は A の各要素を 2 乗するという意味です）
'	転置 （行を列にし、列を行にします）
==	等しい（論理演算）
~=	等しくない（論理演算）
~	否定（論理演算）
>	～より大きい（論理演算）
<	～より小さい（論理演算）
&	論理積
|	論理和
NaN	数字でない Not a number という意味です。欠損値を扱うときに便利です。欠損値を無視して統計を行うときに nanmean と nanstd を使います。
Inf	無限大
global	変数をグローバル変数として定義します。すなわち、ある関数の中のグローバル変数は被包化されず関数外からアクセスしたり変更したりできます。使う時は各関数やスクリプトでその変数がグローバル変数であることを宣言しなければいけません。

用語集 – 基本的なコマンド

詳細は help に続けてコマンド名をタイプしてください。

help *コマンド*	あるコマンドのヘルプを表示します。
lookfor *word*	ヘルプの中にある特定の単語を検索します。（結果表示に時間がかかることがあります。）
whos	現在の変数を表示します。
clear A	メモリから変数 A を取り除きます。
clear all	メモリからすべてを取り除きます。
size	行列の大きさを返します。
max / min	行列の各列の最大値/最小値
mean	列の平均値
std	標準偏差
sqrt	平方根
abs	数字の絶対値、つまり負の数を正の数に変換します。
diff	列の次に来る値との差分です。
sort	列を最大から最小に並べ替えます。
sortrows	特定の列の値に従って行列を並び替えます。
round	小数をもっとも近い整数に丸めます。

find	ベクトルの中でゼロより大きい値を見つけます。論理インデックスを添字インデックスに変換するときに便利です。
rem(a,b)	a を b で割った時の余りです。
ones(m,n)	すべての値が 1 である m 行 n 列の行列を作ります。
zeros(m,n)	すべての値が 0 である m 行 n 列の行列を作ります。
linspace(a,b,n)	最初が a、最後が b で要素が n であるベクトルを作ります。a と b の間の間隔は同じになります。
rand(m,n)	0 と 1 の間で均一に分布した乱数でできた m 行 n 列の行列を作ります。
randperm(m)	整数を 1 から m までランダムに配置します。実験の順番をランダムにしたいときに便利です。
disp(文字列)	文字列を画面に表示します。スクリプトで何が起こっているのかを伝えるときに便利です。
num2str	数字を文字列に変換します。値を表示するときに便利です。
repmat	行列を複製します。行列を特定の値で埋めるときに便利です。

用語集 – グラフに関する関数

plot	データ x とデータ y のグラフを描きます。グラフの線の種類については Matlab のヘルプを御覧ください。
title	タイトルをグラフに追加します。
xlabel / ylabel	軸ラベルを追加します。
axes	軸の限度を設定します。
figure	特定の図を作るか選択します。
clf	図をクリアします。(一番最後に選択された図です。)
subplot(m,n,p)	図を m 行 n 列のサブグラフにわけ、p 番目のサブグラフを作るか選択します。
cla	現在選択している軸しくはサブグラフをクリアします。
legend	図に凡例を追加します。
lsline	最小二乗線を追加します。
text	グラフに文字列を追加します。
grid	グラフに格子状に目盛線を追加します。
bar	棒グラフ
hist	ヒストグラム
errorbar	エラーバーとともにグラフを描きます。
image / imagesc	行列を画像や図として表示します。
colorbar	画像にカラースケールを追加します。
colormap	画像で使われている色を変更します。
get / set	グラフィックのプロパティをすべて制御します。たとえば、あるグラフの x 軸のラベルを'cat', 'dog', 'rabbit'としたい場合、次のようにします。 `set(gca,'XTick',1:3,'XTickLabel',{'cat','dog','rabbit'})` gca は *get current axis* の意味で Matlab に現在アクティブな軸のラベルを設定するように Matlab に指示します。 XTick は x 軸に表示される目盛りの数です。XTickLabel は x 軸の各目盛り上に表示される値です。x 軸上に設定できる項目をすべて見るには、`get(gca)` を使ってください。

用語集 – 統計

以下は Statistics Toolbox が必要です。すべての機能を見るには help stats とタイプしてください。

regress	線形回帰
ttest	ベクトルの平均がゼロより異なるか検定します。
ttest2	2 群の平均が異なるか検定します。
anova1	基本的な分散分析です。Matlab は基本的に複雑な ANOVA は得意でありません。タブ区切りテキストに書き出して SPSS で反復測定分散分析や二要因以上の分散分析をする方がよいでしょう。