• Il y a 10 types de personnes dans le monde. Lesquels?

<table>
<thead>
<tr>
<th>Decimal pattern</th>
<th>Binary numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
</tr>
</tbody>
</table>

• Ceux qui comprennent le système binaire et les autres
• What is Matlab?
  – Numerical computing environment
  – Fourth-generation programming language (4GL)
    • First-generation language (1GL):
      – No translator to compile or assemble the 1GL. Instructions given through the front panel switches
        » Program written as binary instructions.
        • Disadvantages:
          • Extremely difficult to interpret and learn by the human programmers
          • Poor portability
          • Difficult to debug
        • Advantages:
          • Instructions are executed directly by the CPU
            • Fast and efficient
      – Still used for programming lower level functions (drivers, interfaces with firmware and hardware). Modern tools, such as native-code compilers are used to produce machine level from a higher-level language.
      – Sometimes called: Low-level programming language (little or no abstraction) or assembly language together with the 2GL
• 2GL
  – Similar to 1GL but with logical structure
• 3GL
  – More programmer-friendly, more interpreted, more portable. The computer take care of non-essential details, not the programmer.
  – High-level languages
    » 1950s: Fortran, ALGOL, COBOL
    » C++, C#, Java: object languages
    » C, BASIC and Pascal
  – Structured programming
    » Extensive use of subroutines, block structures and loops (for, if, while)
• **4GL:**
  – Very high-level programming languages
  – 4GL (and 5GL) projects are more oriented toward problem solving and systems engineering. More domain-specific (not really domain but approach)
    » LabView, Mathematica, SPSS, R, WinDev, **Matlab**

• **5GL:**
  – Programming languages based on solving problems using constraints given to the program, rather than using an algorithm written by a programmer.
  – Mainly used in artificial intelligence research.
  – Starting from a set of constraints defining a particular problem, deriving an efficient algorithm to solve it is a very difficult problem in itself. This crucial step cannot yet be automated and still requires the insight of a human programmer.
• What does Matlab do?
  – Numerical computing
    • Matrix manipulations
    • Plotting of functions and data
    • Implementation of algorithms
    • Creation of user interfaces
    • Interfacing with programs written in other languages, including C, C++, Java, and Fortran.
  • Simulink: block diagram environment

• Several millions users across industry and academia.
  – Lot of toolboxes, codes, forums,…
  – Various approaches and topics
What can you do with Matlab?

< MATrix LABoratory >

- Stimulus presentation & data acquisition (Psychtoolbox-3 [http://psychtoolbox.org/] )
- Data analysis: from simple calculations to complex computations on brain imaging data.
- Data visualisation

\[
\text{ERP}_{t,e} = XB + \varepsilon \\
\text{ERP}_{t,e} = \beta_0 + \text{cat}_1 \beta_1 + \text{cat}_2 \beta_2 + \ldots + \text{cat}_8 \beta_8 + \phi_{G-ID} \beta_9 + \phi_{G-CO} \beta_{10} + \phi_{L-ID} \beta_{11} + \phi_{L-CO} \beta_{12} + \varepsilon
\]
• Why Matlab for cognitive scientists?
  – Automatisation of data transformation:
    • prevent hours of boring job in excel (like transposing vectors, copy/paste,...)
    • Avoid human errors
  – Same software for stimulus presentation, data collection, data processing, results rendering
    • Skill transfer between research steps (previous ones) and topics
    • Code transfer between research steps and topics
• Where do I find information?
  – Help in Command Window
  – Matlab Documentation Center: http://www.mathworks.co.uk/help/matlab/
  – Matlab tutorials and learning resources http://www.mathworks.com/academia/student_center/tutorials/launchpad.html
  – Internet, forums
  – Books
Where do I find functions?

- Core program
- Toolboxes (commercial or free)
- File exchange (http://www.mathworks.com/matlabcentral/fileexchange/)

Entropy based Saliency detection by Konda Reddy Mopuri
This function finds the salient regions in a video using entropy measurement. (entropy, saliency, salient regions)

- jog.avi
- park.avi

`fx` EntSalincy(inputVideo,th,w)

images2movie - convert images into movie by Todd Karin
Convert images stored on your computer to a movie. (picture, image, convert)

- exampleImages.mp4

`fx` getAllFilesInFolder(folderName)
`fx` imageFolder2mpeg(folderName, varargin)

Traffic Warning Sign Recognition Matlab Code by Muhammad Ammad
Traffic Warning Sign Recognition Matlab Code using matlab demo from the matlab 7.6 in video processe (blob analysis, traffic warning sign ..., recognition)

- Yield.avi
- stoprd.avi

`fx` main
• How does it look like?

Possibilities to change the layout
To come back to this layout:
Destop/Desktop Layout/Default
• File formats created by Matlab
  – .fig
    • MATLAB figure
  – .m
    • MATLAB code (function, script, or class)
  – .mat
    • MATLAB data (binary file for storing variables)
  – .mex... (.mexw32, .mexw64, .mexglx, ...)
    • MATLAB executable MEX-files (platform specific, e.g. ".mexmac" for the Mac, ".mexglx" for Linux, etc.[44])
  – .p
    • MATLAB content-obscured .m file
• Where does the data come from and where does it go?
  – Loading / reading:
    • load: load mat or ascii files
    • importdata:
      – Import the file, specifying the space delimiter and the header
      – Import and display the image
    • imread
    • xlsread
    • From txt files
      – filetoopen=['test' num2str(sbj) '.txt'];
      – filetxt=fopen(filetoopen);
      – data=textscan(filetxt,'%s%s%n%n%n%n%n%n');
      – procedure=(data{1});
      – session=(data{2});
      – eyex=(data{3});
– Saving / wriitng

• save: save some or all the variables of the workspace
• xlswrite: write excel files
• dlmwrite: write txt files
  – datatowrite=[situation echantillon list(i,1) randx Randy time
eyex eyey diffx diffy windowsize visualcondition];
  – dlmwrite(datatxt, datatowrite, '-append','delimiter', '\t','
precision', 7);
• Languages fundamentals

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ans</code></td>
<td>Most recent answer</td>
</tr>
<tr>
<td><code>clc</code></td>
<td>Clear Command Window</td>
</tr>
<tr>
<td><code>diary</code></td>
<td>Save Command Window text to file</td>
</tr>
<tr>
<td><code>format</code></td>
<td>Set display format for output</td>
</tr>
<tr>
<td><code>iskeyword</code></td>
<td>Determine whether input is MATLAB keyword</td>
</tr>
<tr>
<td><code>clear</code></td>
<td>Clear specified variable</td>
</tr>
<tr>
<td><code>clear all</code></td>
<td>Clear all specified variables</td>
</tr>
<tr>
<td><code>clearvars -except A</code></td>
<td>Clear specified variable except A</td>
</tr>
</tbody>
</table>

• Matlab is case sensitive
  – A ≠ a

• Editor
  – Comments, Cells, Breakpoints, automatic indentation
    • Evaluate Current Cell: ctrl+enter
    • Evaluate selection: F9
    • Navigate from cell to cell: ctrl + up/down arrow
• Entering statement
  – Single elements

```matlab
>> a=1
a =
  1
>> b=1;
>> c=a+b;
>> d=sin(a)
```

– look at your workspace
  • Name
  • Value
  • Size
  • Format
  • Class...

– try up and down arrows in the Command Window
– try “whos”
• Formats in Matlab
  – Classes of the values that can be in single element variables, vectors, matrices, structures, cells,…
  – The variables do not need to be declared (and a class specified) before being used.
  – The class of a variable can be changed

– Classes
• $a = 7$;
• $b = 'string' $;
• $c = '7' $;
• $d = \text{str2num}(c)$;

• Empty cell: NaN, not a number
• Vectors

  – Creating a vector
    – $A = [1 \ 3 \ 5 \ 4 \ 8]$; $B = [10 \ 6 \ 4 \ 9 \ 2]$; \% create row vector by default

• Vector by range
  – $x = [1:100]$;

• Specifying steps
  – $x2 = [1:2:100]$;
    – $x3 = [352:-8:12]$;

• Empty vector then copy/paste in variable editor from other software
  – $X4 = []$;

  – Transposing a vector: ‘
    • $A = A'$;
    • $D = [8 \ 5 \ 6 \ 14 \ 7]'$; \% create column vector
– Accessing elements within a vector
  • A(2)
  • A(3:5)
  • A(1:2:5)
  • A(2:end)
  • A(:)

• A==5
• B(A==5)
– Basic operations on vectors (or matrices)

• Vector and scalar
  – C=3+A
  – C=2*A

• Vector and vector
  – Additions/subtractions on vectors
    » C=A+B
    » Transpose A, try again D=A+B and transpose back
  – Multiplications/divisions on vectors
    » try A*B
    » try A’*B  % matrix product
    » try A*B’  % matrix product
    » try A.*B  % element-wise product
  – Power
    » G = A.^2

• Combining
  » E = -2*(D(3:5)’+A([1 2 4])+B(1:2:5))
• Functions on vectors (or matrices)
  – max(A)
  – A == max(A)
  – [maxA, location] = max(A)

  – Examples of simple function: max, min, mean, median, mode, corrcoef, var, std, cov, isnan, is*, sort, cat, repmat

• Information about vectors
  – length(A)
  – ndims(A)
  – size(A)
• Matrices
  – Creating 2D matrices
    • \( F = [A' \ B'] \) % combining vectors
      compare with \( G = [A \ B] \)
    • \( A = [1 \ 2 \ 3; \ 3 \ 4 \ 5; \ 6 \ 7 \ 8] \) % column of row vectors
    • \( B = [[1 \ 2 \ 3]' \ [2 \ 4 \ 7]' \ [3 \ 5 \ 8]'] \) % row of column vectors
    • \( C=[] \) % creating empty matrix then filling via the
      variable editor
  – Creating special matrices (also valid on vector, scalar,...)
    • Identity matrix
      – \( G = \text{eye}(4,3) \)
    • Ones matrix
      – \( H = \text{ones}(4,3) \)
• Zero matrix
  – \( I = \text{zeros}(4,3) \)
  – \( L = \text{zeros}(5) \)
  – Create \( L \) in order to get a five 0 values vector

• Random matrix
  – \( J = \text{rand}(4,3) \)
  – Check \( \text{rand} \)

• Random matrix with normally distributed random numbers (Gaussian noise)
  – \( K = \text{randn}(4,3) \)
  – Check \( \text{randn} \)
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ones</td>
<td>Create a matrix or array of all ones.</td>
</tr>
<tr>
<td>zeros</td>
<td>Create a matrix or array of all zeros.</td>
</tr>
<tr>
<td>eye</td>
<td>Create a matrix with ones on the diagonal and zeros elsewhere.</td>
</tr>
<tr>
<td>accumarray</td>
<td>Distribute elements of an input matrix to specified locations in an output matrix, also allowing for accumulation.</td>
</tr>
<tr>
<td>diag</td>
<td>Create a diagonal matrix from a vector.</td>
</tr>
<tr>
<td>magic</td>
<td>Create a square matrix with rows, columns, and diagonals that add up to the same number.</td>
</tr>
<tr>
<td>rand</td>
<td>Create a matrix or array of uniformly distributed random numbers.</td>
</tr>
<tr>
<td>randn</td>
<td>Create a matrix or array of normally distributed random numbers and arrays.</td>
</tr>
<tr>
<td>randperm</td>
<td>Create a vector (1-by-n matrix) containing a random permutation of the specified integers.</td>
</tr>
</tbody>
</table>
– Information about matrices
  • ndims
  • \([m,n] = \text{size}(X)\)
  • \(m = \text{size}(X,\text{dim})\)

– Creating 3D matrices (and so on)
  • \(M(:,:,1) = K;\)
  • \(M(:,:,2) = J;\)
  • \(M(:,:,3) = I);\)
```
>> round(randn(5))

ans =

-1   2  -1   0  -1
 0  -1   0  -1   1
-1   0   1  -1   0
-1   0   1  -1   1
 0   1   2   2  -1
```
**2D matrix indexation**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>-1</td>
</tr>
</tbody>
</table>

**Examples:**
- \( m(3,3) \)
- \( m(5,1) \)
3D matrix indexation

columns = dimension 2

rows = dimension 1

dimension 3
3D matrix indexation
3D matrix indexation

\[ m(2,3,1:3) \]
Strings

3 x 6 char matrix

\[
\begin{align*}
\text{string} & \rightarrow \text{matlab} & 1 \\
\text{string} & \rightarrow \text{volume} & 2 \\
\text{string} & \rightarrow \text{absent} & 3
\end{align*}
\]

\[
\begin{align*}
>> a &= ['matlab'; 'volume'; 'absent'] \\
>> b &= ['123456']
\end{align*}
\]

\[
\begin{align*}
a &= \\
b &= 123456
\end{align*}
\]
variable initialisation

```matlab
>> a = ['matlab';'volume';'absent'];
tmp = 21;
rmat = randn(2,3);
```

‘=’ means ‘is set to’
# variable initialisation

<table>
<thead>
<tr>
<th>Name DO’S</th>
<th>Name DONT’S’S</th>
</tr>
</thead>
<tbody>
<tr>
<td>call it something useful</td>
<td>include a space</td>
</tr>
<tr>
<td>start with a letter</td>
<td>Variable_Name VariableName</td>
</tr>
<tr>
<td>short</td>
<td>start with a number</td>
</tr>
<tr>
<td>unique</td>
<td>use an existing name</td>
</tr>
<tr>
<td></td>
<td>whos name</td>
</tr>
<tr>
<td></td>
<td>type as a string</td>
</tr>
</tbody>
</table>
# Variable Initialisation

<table>
<thead>
<tr>
<th>Default Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ans</td>
<td>last results</td>
</tr>
<tr>
<td>pi</td>
<td>ratio of any Euclidean circle's circumference to its diameter</td>
</tr>
<tr>
<td>inf</td>
<td>infinity</td>
</tr>
<tr>
<td>NaN</td>
<td>Not-a-Number (e.g. i / 0)</td>
</tr>
<tr>
<td>i / j</td>
<td>imaginary units ( \sqrt{-1} )</td>
</tr>
<tr>
<td>realmin</td>
<td>smallest positive real number</td>
</tr>
<tr>
<td>realmax</td>
<td>largest real number</td>
</tr>
</tbody>
</table>

In doubt, type `help` name (e.g. `>>help realmin`)
<table>
<thead>
<tr>
<th>variable initialisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>a=[1 2 3 4]</td>
</tr>
<tr>
<td>1,2,3,4</td>
</tr>
<tr>
<td>1 2 3 4</td>
</tr>
<tr>
<td>b=[1;2;3;4]</td>
</tr>
<tr>
<td>c=[a';b];</td>
</tr>
<tr>
<td>a'</td>
</tr>
<tr>
<td><strong>clear</strong> all</td>
</tr>
<tr>
<td><strong>clear</strong> a b</td>
</tr>
</tbody>
</table>
Exercises

• intro_1_1.m
## Indexing

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a(1:3)</td>
<td>select a specific portion</td>
</tr>
<tr>
<td>()</td>
<td>refer to part of a matrix</td>
</tr>
<tr>
<td>a(:)</td>
<td>select everything</td>
</tr>
<tr>
<td>a(1:2:10)</td>
<td>select alternate items</td>
</tr>
<tr>
<td>a(1:2)=[ ]</td>
<td>delete elements</td>
</tr>
</tbody>
</table>
2D matrix indexation

columns = dimension 2

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>-1</td>
</tr>
</tbody>
</table>

rows = dimension 1

Examples:

>> m(3,3)
>> m(5,1)
2D matrix indexation

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>-1</td>
</tr>
</tbody>
</table>

Examples:

- `m(1,1:5)` or `m(1,:)`
- `m(3:5,3)`
2D matrix indexation

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>-1</td>
</tr>
</tbody>
</table>

Examples:
```
>> m(1,1:2:5)
>> m(1:2:5,3)
```
### 2D matrix indexation

#### Example:

```matlab
>> m(:,[1 3 5])
```
2D matrix indexation

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

replace or delete elements:
\[ a(1, 1) = 0; \]
\[ a(3, 3) = \text{NaN}; \]
\[ a(2, 2) = 100; \]
\[ a(:, 1) = []; \]
Exercises

• intro_1_2.m
Basic Calculations

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>add</td>
</tr>
<tr>
<td>-</td>
<td>subtract</td>
</tr>
<tr>
<td>./</td>
<td>divide</td>
</tr>
<tr>
<td>.*</td>
<td>multiply</td>
</tr>
<tr>
<td>.^</td>
<td>power</td>
</tr>
</tbody>
</table>

**element-wise:**

```plaintext
>> a=1:3;
>> b=1:3;
>> a+b
ans =
     2  4  6
>> a+1
ans =
     2  3  4
```
Basic Calculations

```matlab
>> a=randn(3,3)

a =

-0.0825  -1.7947    0.1001
-1.9330    0.8404   -0.5445
-0.4390  -0.8880    0.3035

>> a=a+[1:3]
??? Error using ==> plus
Matrix dimensions must agree.

>> a=a+repmat(1:3,3,1)

a =

  0.9175    0.2053    3.1001
-0.9330    2.8404    2.4555
  0.5610    1.1120    3.3035
```
Exercises

• intro_1_3.m
functions

A Matlab function is a series of instructions stored in a plain text file with an ‘.m’ extension - hence it’s called an m-file. Functions have two parameter lists, one for input, one for output.

```
a = [1 2 3 4];
sum_a = a(1) + a(2) + a(3) + a(4);
sum_a = sum(a);
```

<table>
<thead>
<tr>
<th></th>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>cd</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>pwd</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>ls</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>edit</td>
<td></td>
<td>(✗)</td>
</tr>
</tbody>
</table>

Some functions have only input or output:

**Tip:** if you find yourself using the same lines of code over and over, it’s time to turn them into a function.
combining functions

- use brackets to order functions:
  - $b = a(\text{l,:,});$
  - $c = \text{mean}(b);$
  - $c = \text{mean}(a(\text{l,:}));$

- higher order embedding:
  - $c = \text{mad}(a(a>\text{mean}(a(:))))$;
Exercises

- intro_2_1.m
- intro_2_2.m
logical operators

Assess if a statement is true (1) or false (0)

==   equal to
~==  not equal to
<    less than
>    more than
>=   more than or equal to
<=   less than or equal to
&    AND
|    OR

All operators and special characters >> help &
logical operators

MATLAB returns a variable of class logical of the same size with either a 1 (true) or a 0 (false) corresponding to each element.

```
>> rt=990;
>> rt > 200
ans =
     1
>> rt <= 500
ans =
     0
```
logical operators

Logical operators can be combined:

```plaintext
>> rt=990;
>> rt > 200 & rt < 1000
ans =
   1
>> rt <= 500 | rt > 900
ans =
   1
```
logical operators

You can use the output of a logical operation as an index:

```matlab
>> rt=[100 200 300 200];
>> good_rt = rt==200
good_rt =
    0 1 0 1

>> rt(good_rt)
ans =
    200 200
```
Exercises

- intro_2_3.m
Loops

if (condition statement)
    (matlab commands)
end
if

  if (condition statement)
      (matlab commands)
  elseif (condition statement)
      (matlab commands)
  elseif (condition statement)
      (matlab commands)
  .
  .
  .
  else
      (matlab commands)
end
for loops

for j=1:4
    j
end
while loops

while (condition statement)
  (matlab commands)
end
Bootstrapping

• Bootstrapping is the practice of estimating properties of an estimator (such as its variance) by measuring those properties when sampling from an approximating distribution.

• One standard choice for an approximating distribution is the empirical distribution of the observed data.

• This can be implemented by constructing a number of resamples of the observed dataset (and of equal size to the observed dataset), each of which is obtained by random sampling with replacement from the original dataset.