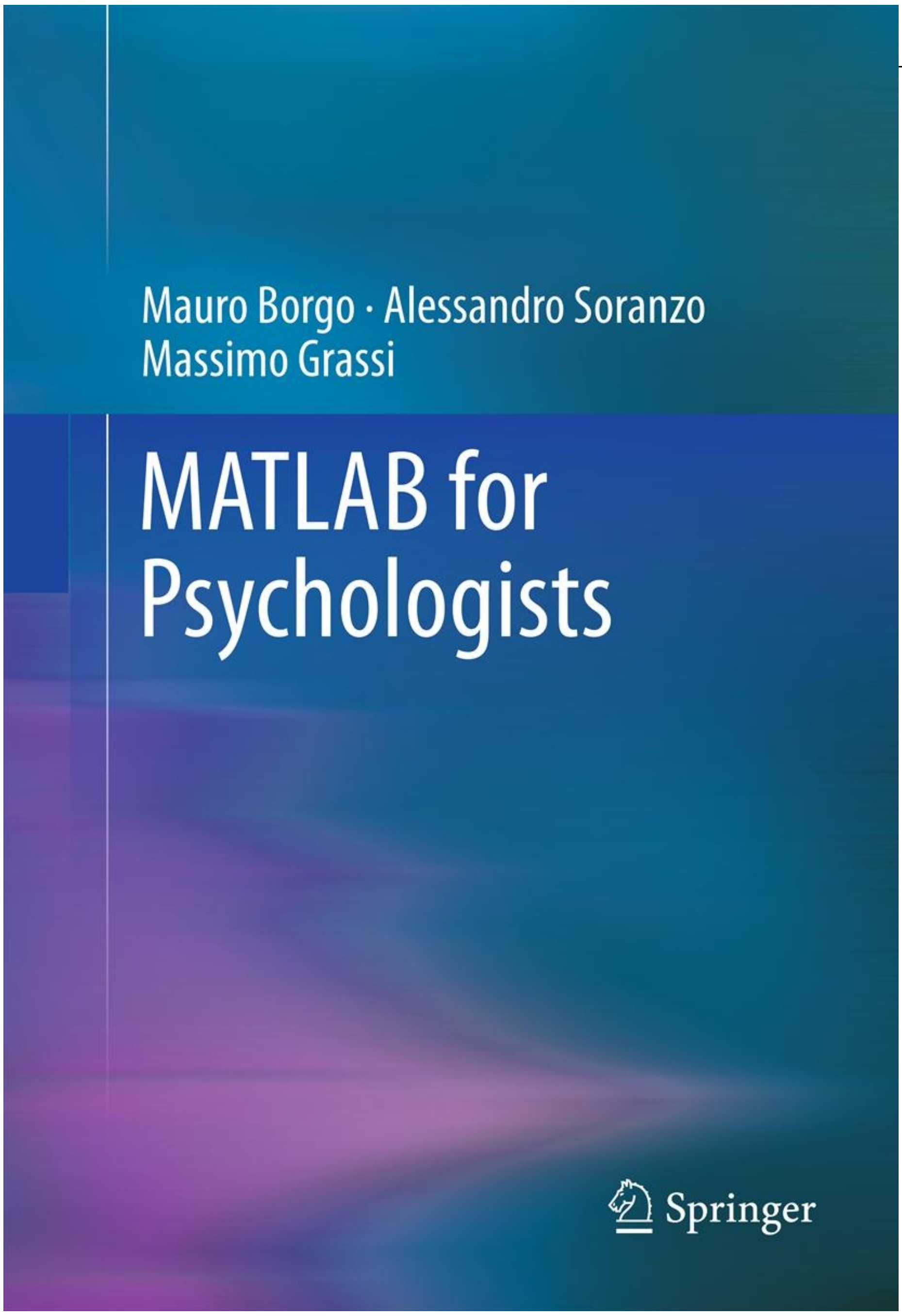


Mauro Borgo · Alessandro Soranzo
Massimo Grassi

MATLAB for Psychologists

Springer

Mauro Borgo, Alessandro Soranzo and Massimo Grassi, *MATLAB for Psychologists*, 2012, DOI: 10.1007/978-1-4614-2197-9, © Springer Science+Business Media, LLC 2012

Mauro Borgo, Alessandro Soranzo and Massimo Grassi

MATLAB for Psychologists

Springer

Mauro Borgo
Via Marosticana 168, Dueville, VI, Italy

Alessandro Soranzo
School of Social Science & Law, University of Teesside, Middlesbrough, UK

Massimo Grassi
Department of General Psychology, University of Padova, Padova, Italy

ISBN 978-1-4614-2196-2 e-ISBN 978-1-4614-2197-9

Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2012931943

© Springer Science+Business Media, LLC 2012

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission o information storage and retrieval, electronic adaptation, computer software, or by similar or dissimil methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of th work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyrigh Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility fo any errors or omissions that may be made. The publisher makes no warranty, express or implied, wit respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

To ~~my three women: my wife, Tatiana, my mother, Angelina, and my grandmother Emilia~~

–Mauro Borgo

To my father

–Alessandro Soranzo

To Viola and Ruggero

–Massimo Grassi

Preface

Psychological researchers should possess several skills, and one of them is surely creativity. Creativity is needed at several key points of the research process, such as in creating experimental stimuli and planning and designing an experiment. Creativity drives good data analysis, so that numbers can reveal their full potential.

Much of this creativity is now expressed through a computer program. For example, in planning and designing a psychological experiment and in analyzing data, we use specific software that has been dedicated to that particular job. This software might, however, be a hindrance to creativity, preventing it from permeating research. This is because in the majority of cases, software is designe
to satisfy the average user and it is not flexible enough to meet specific needs.

In this sense, MATLAB is exactly the other side of the coin. When we first open the software, the
lack of a graphical interface may be frustrating: at a first glance, the program may seem difficult to use. This book is aimed at helping users in their first approaches to this software, to aid them in programming their psychological experiments and consequently in liberating their creativity. And th
is MATLAB's major advantage: we do not have to adapt our needs to the software; it is the software that adapts to our needs.

MATLAB is an extremely powerful research tool. By means of this single software tool we can control every step of our research. We can create stimuli of any kind (e.g., pictures, sounds), and we can program psychological experiments, calculate statistics, run simulations, and do any kind of signal or biosignal processing. In brief, the flexibility of this software lets us to control and customiz
every conceivable step of our research requiring a computer program. Moreover, knowledge of MATLAB will help you to find a postdoc in experimental psychology after completing the Ph.D. In many cases, research groups look for researchers with good MATLAB programming skills.

The current text is written to help the newcomer in using MATLAB for research in experimental psychology. However, the content can be transferred to any application. The reader can find the scrip
written in this book at the following web page: http://www.psy.unipd.it/~grassi/matlab_book.html

A final recommendation for the reader: do not begin to work with MATLAB without a goal. Our teaching experience suggests that having a goal greatly accelerates your learning. Therefore, think immediately about the amazing custom code you need to complete your state-of-the-art research. Th
code is here in this book, waiting to be written by you.

Mauro Borg

Alessandro Soran

Massimo Gras

Acknowledgments

Thanks to Professor Silvano Pupolin for his support.

Contents

Mauro Borgo, Alessandro Soranzo and Massimo Grassi, *MATLAB for Psychologists*, 2012, DOI: 10.1007/978-1-4614-2197-9_1,
© Springer Science+Business Media, LLC 2012

1. Basic Operations

Mauro Borgo[1], Alessandro Soranzo[2] and Massimo Grassi[3]

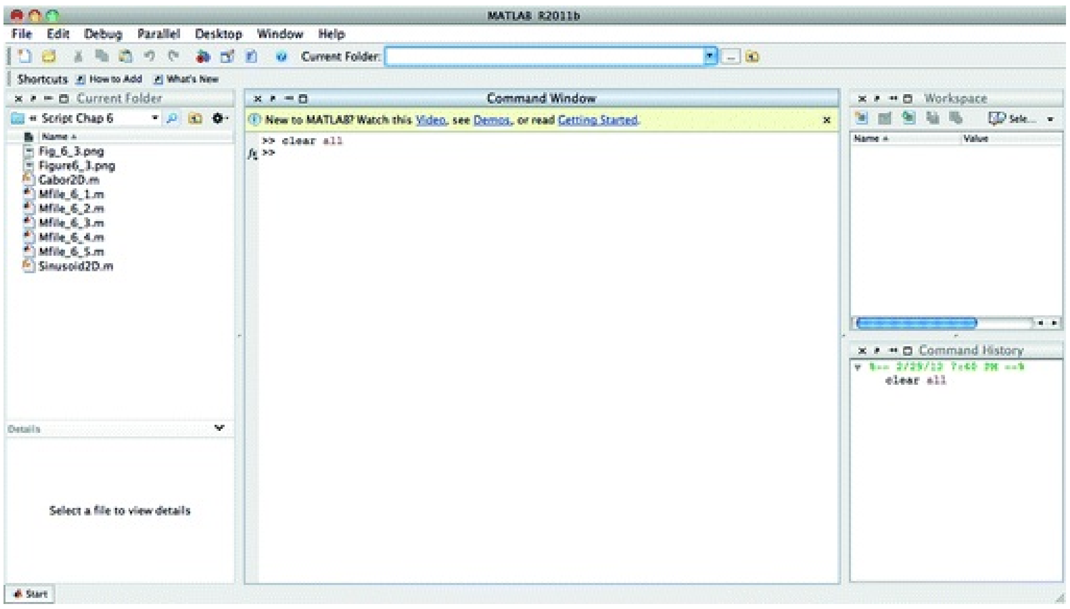
(1) Via Marosticana 168, Dueville, VI, Italy
(2) School of Social Science & Law, University of Teesside, Middlesbrough, UK
(3) Department of General Psychology, University of Padova, Padova, Italy

Abstract

This chapter gives an overview of MATLAB and compares MATLAB with a scientific calculator. Th chapter gives also an overview of basic arithmetic operations and functions as well as a short introduction to matrices and matrix manipulation.

This chapter gives an overview of MATLAB and compares MATLAB with a scientific calculator. The chapter gives also an overview of basic arithmetic operations and functions as well as a short introduction to matrices and matrix manipulation.

It is supposed that you have already installed MATLAB on your computer. When you start MATLAB, the MATLAB desktop opens, as shown in Fig. 1.1 (or something similar, depending on your MATLAB version). In this first chapter we refer only to the Command Window, where the special >> prompt appears. The other windows have the following meaning:



```
>> clear all
>>
```

```
%-- 2/29/12 7:40 PM --%
   clear all
```

Fig. 1.1 The MATLAB desktop. MATLAB release 2011b

- The Workspace Window contains a list of variables that are in use in the working session.
- The Command History contains the list of all commands you have typed in the command window.

- The Current Folder window shows the list of the files contained the folder you are working on.

When the prompt >> is visible, this means that MATLAB is waiting for a command. You can qui MATLAB at any time in either of the following ways:

1. Select Exit MATLAB from the desktop File menu.

2. Enter `quit` or `exit` after the command window prompt `>>`, and press the Enter key.

Alternatively, select File with the mouse from the top menu bar, and then exit MATLAB.

Observe that the tab above the Workspace shows the Current Directory Window. For example, in the Windows operating system, the path might be as follows: C:\MATLAB\Work, indicating that directory “Work” is a subdirectory of the main directory “MATLAB,” which is installed in drive C. Clicking on the arrow in the Current Directory Window shows a list of recently used paths. Clicking on the button to the right of the window allows the user to change the current directory. Knowing which is the current path is fundamental: from the Command Window you have access to the file stored in the given directory. It is, of course, possible to change your working directory.

Before continuing our introduction to MATLAB, we want to highlight a very useful window: the HELP Window. This window is the most useful window for beginning MATLAB users—and for expert users as well: select Help ► PRODUCTHELP from the top bar menu. The Help Window has most of the features you would see in any web browser, including clickable links, a back button, and search engine. All MATLAB commands and functions are explained with examples: you have simply to search for the desired word.

Now let us begin with a description of the MATLAB language. The word “MATLAB” is the concatenation of the words MATrix LABoratory, meaning that MATLAB is an interactive software system for numerical computation, especially designed for computations with matrices. Before going into the details of matrix computations, let us first see how to use MATLAB to do simple arithmetic operations: Type 1+1 after the >> prompt, followed by Enter; that is, press the Enter key, as indicated by <ENTER>

```
>> 1+1 <ENTER>
```

MATLAB gives its quick answer by displaying the following message:

```
ans =
2
```

You can perform other arithmetic operations, such as multiplication, subtraction, and division, an MATLAB always returns the correct result. If such is not the case, there is certainly something wron with what you typed. For example, you can try the following operations (type the operation after the >> prompt followed by Enter):

To TYPE after prompt >> followed by Enter	MATLAB answer	Meaning of the operation
`35*12`	`ans = 420`	Multiplication
`2/45`	`ans = 0.0444`	Division
`4−1`	`ans = 3`	Subtraction
`2^3`	`ans = 8`	Exponentiation

Note that to type numbers such as the Avogadro’s number 6.023×10^{23}, you can either write the *expression* `6.023*10^23` or you can *represent* the number in *scientific notation.* To enter Avogadro’s number in scientific format, type `6.023e23`,where 6.023 is the mantissa and 23 is the exponent. Mantissa and exponent must be separated by the letter e (or E):

>> 6.023*10^23 <ENTER> ans = 6.0230e+023	>> 6.023e23 <ENTER> ans = 6.0230e+023

Such numbers are also defined to be *floating point*.

MATLAB warns you in the case of in invalid operation or "unexpected" results. What do you thi
MATLAB will show us if we type 12/0 or 0/0? Let's try it:

To TYPE after prompt >> followed by Enter	**MATLAB answer**	**Meaning of the answer**
`12/0`	`ans = Inf`	You should not divide by zero, but if you do, the result is **Infinity**
`0/0`	`ans = NaN`	Unable to find the answer, so the result is **NaN** = **Not a Number**

`11+`	`??? 11+`
	`  |`
	`  Error: Expression or statement is incomplete or incorrect`
		If you want to perform this operation, you must complete the expression with another term

As you can see, MATLAB is unable to "stay quiet." It quickly answers your commands by displaying something in the command window. In the previous cases, the answer was a special value such as `Inf` *(Infinity)* or `NaN` *(Not a Number)*. You can use these special values on their own, typing for example:

```
>> 12/Inf <ENTER>
ans =
0
```

MATLAB can be used as a scientific calculator, combining several operations in one expression. The computation of the expression is based on well-known mathematical rules. In particular, some operations are performed before others, based on precedence rules, which are given in the following table:

Precedence	**Operator**
1	Parentheses (round brackets)
2	Exponentiation, left to right
3	Multiplication and division, left to right
4	Addition and subtraction, left to right

If you want to know the result of the operation $\{2+[5*3/(7-5)^2]/3\}$ you have to type:

```
>> (2+(5*3/(7-5)^2)/3) <ENTER>
ans =
3.2500
```

In this example, MATLAB first calculates (7−5) = 2, then it squares 2^2 = 4, then it performs th
multiplication 5*3 = 15 (multiplication left to right), and then divides the result by the previously computed result, i.e., 15/4 = 3.75. The result in brackets is divided by 3 (3.75/3 = 1.25) and then added to 2, giving the result. Note that in MATLAB, parentheses are always given by round brackets.

MATLAB was developed for scientists, and for this reason you can find built-in operations and functions that are more advanced than the ones we have just looked at. Considering MATLAB as a sort of scientific calculator, you can engage the “cosine button” simply by typing:

```
>> cos(36) <ENTER>
ans =
-0.1280
```

Other common functions are reported in the following table. Type the operation after the >> prompt followed by Enter:

To TYPE after prompt >> followed by Enter	MATLAB answer	Meaning of the operation
`cos(12)`	`ans = 0.8439`	Cosine of the element in parentheses
`sin(12)`	`ans = −0.5366`	Sine of the element in parentheses
`tan(4)`	`ans = 1.1578`	Tangent of the element in parentheses
`exp(3)`	`ans = 20.0855`	Exponential of the element in parentheses
`log(10)`	`ans = 2.3026`	Natural logarithm of the element in parentheses
`log10(12)`	`ans = 1.0792`	Base-10 logarithm of the element in parentheses

We will see in the rest of the book the possibility using many other functions. Just to introduce some: statistical functions, interpolation functions, linear-algebraic functions, functions for images and sound elaboration, and last but not least, your own custom-created functions!

We conclude by giving some hints on creating and editing command lines:

- You can select (and edit) previous commands you have entered using the up-arrow and down-arrow keys. Remember to press Enter to execute the command.
- MATLAB has a useful editing feature called *smart recall*. Just type the first few characters of th command you want to recall, e.g., type the characters `lo` and press the up-arrow key—this recal the most recent command starting with `lo`.The result might be, for example, `log(10)` or `log10(12)`.

Variables

Thus far, we have seen the use of MATLAB as a scientific calculator. However, MATLAB is much more than a calculator, and the main difference is the possibility to use “variables.” In a scientific calculator we can save and recall a single number only. In MATLAB, in contrast (as in other programming languages), we can store and recall virtually an infinity of different values called *variables*. A *variable* is a sort of box, having a certain shape, a certain dimension, with a label namin it. In such a box you can put the (virtual) item you need, for example a number, an image, and so on.

Suppose you want to save a number representing your age. You can create your own variable and store it by simply typing the following command:

```
>> age=22 <ENTER>
age =
22
```

The symbol `age` is the variable name (the box name), which contains the number 22. Each time you recall (type) such a name, the content of the variable is used; in this simple case, it is displayed. Type again the *variable* name:

```
>> age <ENTER>
age=
22
```

You can define other variables, for example the number of your friends. Just type:

```
>> Nfriends = 132 <ENTER>
Nfriends =
132
```

At this stage, you may wonder about the shape of the box or its volume. The answer is not straightforward. However, by typing the `whos` command, MATLAB prompts all the variables currently active in the working session:

```
>> whos <ENTER>
```

Name	Size	Bytes	Class	Attributes
age	1 × 1	8	double	
Nfriends	1 × 1	8	double	

The `whos` command gives you a list of all the variables created in the workspace together with their characteristics. In order to understand the meaning of such characteristics, consider the analogy between variable and box, as presented in the following table:

Variable	Box	Visual interpretation
Name	Name of the box	
Size	Number of objects you have put in (in the previous case, just one object, i.e., 1 × 1).	
Bytes	Total volume of the box. This is the number of objects multiplied by the dimension of each (to store a number you need 8 bytes)	
Class	Type of object you can put inside the box (in the previous case, it is a number stored with *double precision*)	
Attributes	Other information	

Note that the variables list obtained using the command `whos` can readily be seen in the Workspace Window (see Fig. 1.1).

One nice thing that MATLAB does when you create a variable is that it automatically selects the most suitable type of box for the variable. You need, however, to know a few simple rules about variable names:

1. The variable name must start with a letter.
2. It may consist only of the letters a–z, the digits 0–9, and the underscore (_). You cannot have a name with spaces or others symbols (such as +, ^, *).
3. MATLAB is case-sensitive, which means that it distinguishes between upper- and lowercase letters. So `age` is different from `AgE` or `Age`.

Try to create the following variables by typing them after the >> prompt followed by Enter: `N-friends = 12`, `$aDay = 60`, `3rd_classified = 11`. What happens, and why? MATLAB gives you the following error:

```
??? $aDay=60
```

```
|
~~Error: The    input    character    is not valid in MATLAB~~
statements or expressions.
```

Obviously, in these examples we didn't follow the aforementioned rules (use of the character— and $, beginning the name with a number).

MATLAB has a few predefined variable names. Some of these are presented in the following table:

To TYPE after prompt >> followed by Enter	MATLAB answer	Value contained in the variable
`Pi`	`ans = 3.1416`	π
`Esp`	`ans = 2.2204e-016`	Floating-point relative accuracy, i.e., the distance from 1.0 to the next larges double-precision number
`j`	`ans = 0 + 1.0000i`	Imaginary unit, i.e., sqrt(−1), used to enter complex numbers
`I`	`ans = 0 + 1.0000i`	Imaginary unit, i.e., sqrt(−1), used to enter complex numbers
`NaN`	`ans = NaN`	Not a number
`Inf`	`ans = inf`	Infinity

You can redefine a variable by simply assigning it a new value:

```
>> pi=12 <ENTER>
pi =
-12
>> pi <ENTER>
pi =
-12
```

Once you have inserted a new value, you cannot recall the previous one. However, in the special case of predefined variables, you can clear the redefined variable, and the predefined variable is restored. To clear variables you use the command `clear` followed by the variable name (or a list of them). Let's try:

```
>> clear pi <ENTER>
```

MATLAB doesn't give you an answer. However, the command has been executed. Type the *pi* variable again, and MATLAB will return the value of π:

```
>> pi <ENTER>
ans =
3.1416
```

The command `clear` can be followed by the specification `all`, and all the variables stored in th workspace are deleted. To test whether this is indeed the case, type the `whos` command:

```
>> clear all <ENTER>
>> whos <ENTER>
>>
```

Note that you receive no answer from MATLAB because there is nothing to display. At the same time, you can see that the Workspace Window (Fig. 1.1) is empty.

With variables you can type complex expressions and store the result. Let's try:

```
>> number=13; <ENTER>
>> a=14; <ENTER>
>> c=pi*((number+a/2)/10); <ENTER>
```

~~MATLAB doesn't give an answer because you ended the command with the semicolon (;) which~~ prevents the value of `number` from being echoed on the screen. However, `number` still has the valu 13, as you can see by entering its name without a semicolon (or looking at the Workspace Window):

```
>> number <ENTER>
number =
13
>> c <ENTER>
c =
6.2832
```

Thinking in a Matrix Way

Our first question about matrices is, "What is a Matrix?" We are not talking about the film, the sequels, the comic books, or the video games. For us, a matrix isn't a complex computer simulation that you have to do battle with to save humanity. However, you can choose to continue to read the book (in analogy to the blue pill in the film that allows you to lead your normal life) to learn how use MATrix LABoratory to create innovative experiments, thereby changing the world with your discoveries.

In MATLAB, a matrix is a rectangular array of numbers, as shown in the following Fig. 1.2.

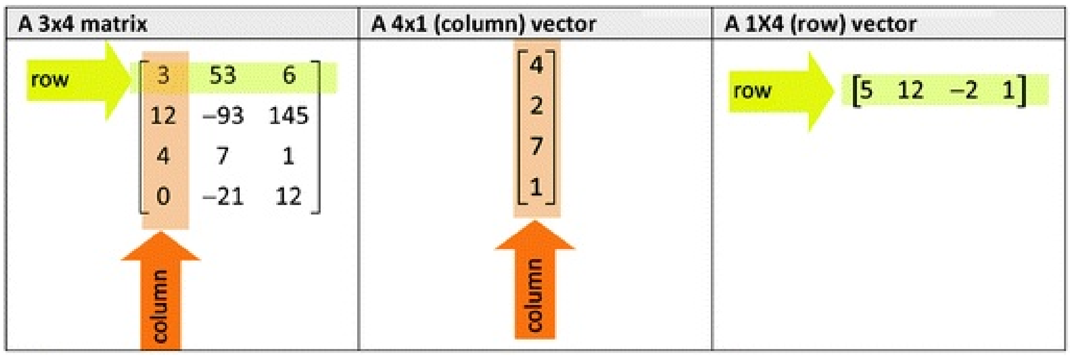

A 3x4 matrix
row
column
A 4x1 (column) vector
column
A 1X4 (row) vector
row
$\begin{bmatrix} 3 & 53 & 6 \\ 12 & -93 & 145 \\ 4 & 7 & 1 \\ 0 & -21 & 12 \end{bmatrix}$
$\begin{bmatrix} 4 \\ 2 \\ 7 \\ 1 \end{bmatrix}$
$\begin{bmatrix} 5 & 12 & -2 & 1 \end{bmatrix}$

Fig. 1.2 Matrices of various dimensions

The horizontal lines of a matrix are called rows, and the vertical lines are called columns. The numbers in the matrix are called entries. A matrix with m rows and n columns is called an $m \times n$ matrix. A matrix one of whose dimensions equals one is often called a vector. An m × 1 matrix (one column and m rows) is called a column vector, and a 1 × n matrix (one row and n columns) is called row vector.

If you are familiar with the spreadsheet software Excel, you can imagine each Excel worksheet as a matrix, with rows and columns.

Now let's try to define some matrices and vectors in MATLAB. Type the following statements as written:

```
>> a=[3,5,7,8] <ENTER>
a =
5 12 -2 1
>> b=  [4;2;7;1] <ENTER>
b =
4
```

```
2
~~7~~
1
>> c=  [3, 53,6;12,-93,145;4,7,1;0,-21,12] <ENTER>
c =
 353 6
12-93145
 47  1
 0  -21     12
>> whos <ENTER>
Name SizeBytes ClassAttributes
a1x432 double
b4x132 double
c 4x396 double
number 1x18 double
```

As you can see, after the command `whos`, each variable is displayed together with its size expressed in rows × columns. (Note: if you have used other variables previously, your list of variable could be different). Note that a scalar value, like the variable `number`, can be considered a 1 × 1 matrix.

Sometimes we need to know the size of the variables only, instead of the full list of their properties. In this case, the function `size(c)` returns the number of rows and columns of the variable `c`:

```
>> size(c)
ans =
4 3
```

Another useful function is `length(c)`, which returns the length of the vector c. If c is a matrix, the function `length` returns the number of rows only:

```
>> length(c)
ans=
4
```

To put data into a matrix, you must type the values within square brackets, separated by *spaces* or *commas* for different elements in a row, while the *semicolon* (;) is used to indicate the end of the row Note that the number of elements must be the same in each row:

```
>> x=[ 1 2 3; 2 5 7] <ENTER>
x =
1 2  3
2      5  7
```

If you have not put the same number of elements in each row, MATLAB displays an error:

```
>> x = [2 3; 2 5 7];
??? Error using ==> vertcat
CAT arguments dimensions are not consistent.
```

As you can see, MATLAB is not a wizard who tries putting the missing element in the right place MATLAB does not know whether you want to put the element 2 and 3 in the first and second column or in the second and third columns respectively.

You can use a matrix or a vector to implement another variable. For example, type in the following statements:

```
>> x = [3 2 1]; <ENTER>
>> y = [6,7,8]; <ENTER>
>> z1 = [x -y]; <ENTER>
>> z2 = [x; -y]; <ENTER>
```

Can you work out what `z1 and z2` will look like before displaying them? In the following tabl we present other examples showing how to use variables already implemented to create new variable

Mathematical representation	MATLAB (type after the prompt >> followed by Enter)	Dimension
$M = \begin{bmatrix} 3 & 12 & \pi \end{bmatrix}$	`M=[3,12,pi];`	1 × 3 Row vector
$N = \begin{bmatrix} 3 & 12 & \pi \\ 8 & 9 & 10 \end{bmatrix}$	`N=[3,12,pi; 8,9,10];`	2 × 3 Matrix
	Or equivalently, if you have already inserted M:	
	`N=[M; 8,9,10];`	
$P = \begin{bmatrix} 4 \\ 2 \\ -1 \end{bmatrix}$	P=[4;2;−1]	3 × 1 Column vector
$Q = \begin{bmatrix} 4 & -4 \\ 2 & -2 \\ -1 & 1 \end{bmatrix}$	Q=[4, −4;2, −2;−1, 1];	3 × 2 Matrix
	Or equivalently, if you have already inserted P:	
	Q=[P;−P];	

If you do not specify any variable content (i.e., any values inside the square brackets), MATLAB creates a variable of size zero with no value, or more precisely, a matrix of dimension 0 × 0 with no value in it.

```
>> y = [ ]; <ENTER>
>> whos y <ENTER>
NameSizeBytesClassAttributes
y 0x00double
```

The Workspace Browser in the desktop provides a handy visual representation of the workspace. By clicking a variable in the Workspace Browser, we open the Array Editor, which can be used to view and change values.

The entry that lies in the *i*th row and the *j*th column of a matrix is typically referred to as the (*i*,*j*) or (*i*,*j*)th entry of the matrix. For example, the (3,2) entry of matrix Q in the table above is 1. In mathematical format, it is usually written as $Q_{3,2}$, while in MATLAB you can access to the matrix entries in this way:

```
>> Q(3,2) <ENTER>
ans =
1
```

Note the use of parentheses. For indexing you use parentheses, whereas to define a matrix, you us square brackets; otherwise, you get an error:

```
>>Q[2,3]
??? Q[2,3]
|
Error: Unbalanced or unexpected parenthesis or bracket.
```

In the following table you can find other examples:

Mathematical representation	MATLAB (type after the prompt >> followed by Enter)

$Q_{3,1}$ is equal to −1	>>Q(3,1) ans= -1
$N_{2,2}$ is equal to 9	>> N(2,2) ans= 9
$M_{1,3}$ is equal to π	>> M(1,3) ans= 3.1416
$P_{2,1}$ is equal to 2	>> N(2,1) ans= 2

If P and M are two vectors, it is possible to refer to their entries by referencing only their single dimension, i.e., you can type M(3) instead of M(1,3), and N(2) instead of N(2,1).

If you refer to an element in a nonexistent position, MATLAB gives you an alert:

```
>> Q(3,3) <ENTER>
??? Index exceeds matrix dimensions.
```

What happens if you want to address more than one element at time? This is possible in MATLA using a vector (or a matrix) in the indexing place to express the selected rows or columns:

```
>> Q([1,3],2) <ENTER>
ans =
-4
 1
```

How many values do you expect MATLAB to display when you type Q([1,3],[1,2]) ? Two or Fou Let’s try:

```
>> Q([1,3],[1,2]) <ENTER>
ans =
 4 -4
-1  1
```

The answer is four, because MATLAB shows the values in the positions given by each combination of the specified rows and columns, i.e., Q_{11}, Q_{12}, Q_{31}, Q_{32}.

Now suppose you have a large matrix from which you want to extract elements going from the ith row to the jth row in the second column. MATLAB offers a very efficient way to this, namely the colon (:) operator. Before seeing how it works, let us generate a new matrix:

```
>> x=[1 2 3; 4 5 6; 7 8 9; 10 11 12; 13 14 15] <ENTER>
x =
 1  2  3
 4  5  6
 7  8  9
10 11 12
13 14 15
```

Now type:

```
>> i=2; j=4;<ENTER>
>> x(i:j,2)<ENTER>
ans =
 5
 8
```

sample content of MATLAB for Psychologists

- **download online Edible Cocktails: From Garden to Glass--Seasonal Cocktails with a Fresh Twist**
- *download The African Mercenary (Casca, Book 12) for free*
- The Secret Garden here
- download online Grimoires: A History of Magic Books

- http://yachtwebsitedemo.com/books/Seven-Languages-in-Seven-Weeks--A-Pragmatic-Guide-to-Learning-Programming-Languages.pdf
- http://www.khoi.dk/?books/Christian-Humanism-and-the-Puritan-Social-Order--Ideas-in-Context-.pdf
- http://thewun.org/?library/Cultural-Techniques--Grids--Filters--Doors--and-Other-Articulations-of-the-Real.pdf
- http://crackingscience.org/?library/Basic-Nutrition--Healthy-Eating--A-Guide-to-Nutrition-.pdf