Introduction to machine learning in Hydrology

Lazaro J. Perez & Marc Berghouse

contact: lazaro.perez@dri.edu

Outline

- Getting into Matlab
- Why use Matlab
- Matlab desktop environment
- Variables and assignments
- Vector and matrices

The big picture

MATLAB → MAT(rix)LAB(oratory) → Written in C & Fortran

MATLAB is a very powerful software package that has many built-in tools for solving problems and developing graphical illustrations

MATLAB has built-in functions to perform many operations, and there are toolboxes that can be added to augment these functions (e.g., for signal processing)

Why use MATLAB

Easy to code: Develop the computational codes and debug easily

```
variablename = expression
>> mynum = 6
```

Supports GPU: Accelerate your code using basic GPU computing

Standalone compiler: No need to call libraries or compile codes/functions

Graphical User Interface: Inexperienced programmers can design/refine data-analysis programs

Toolboxes: Machine learning, image analysis, robotics, communications, etc.

Work with external code: C, C++, Python, Fortran

SB LAB DRI

File types MATLAB

Algorithm, scripts, and functions

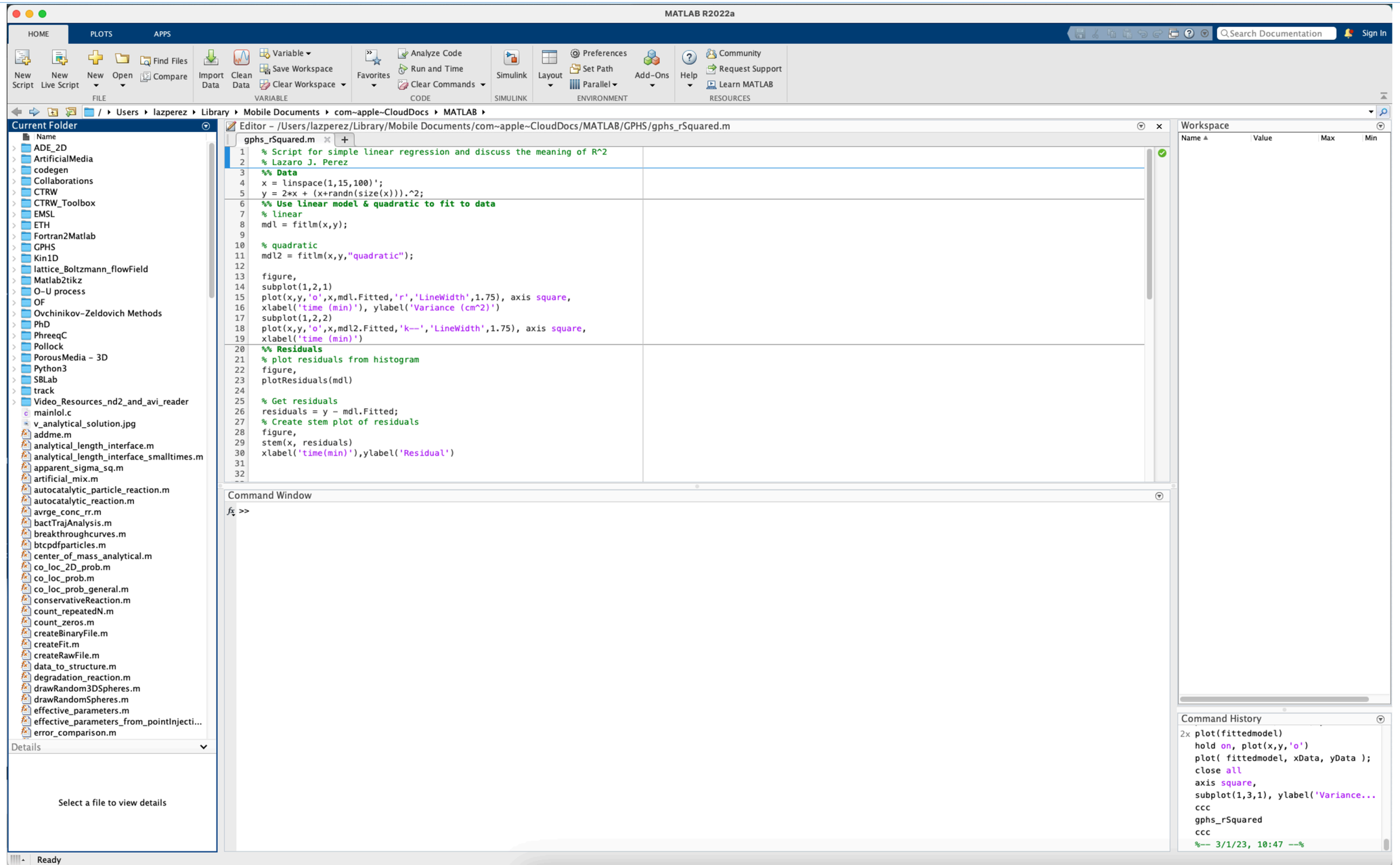
An algorithm is the sequence of steps needed to solve a problem

- Get the input: the radius
- Calculate the result: the area
- Display the output

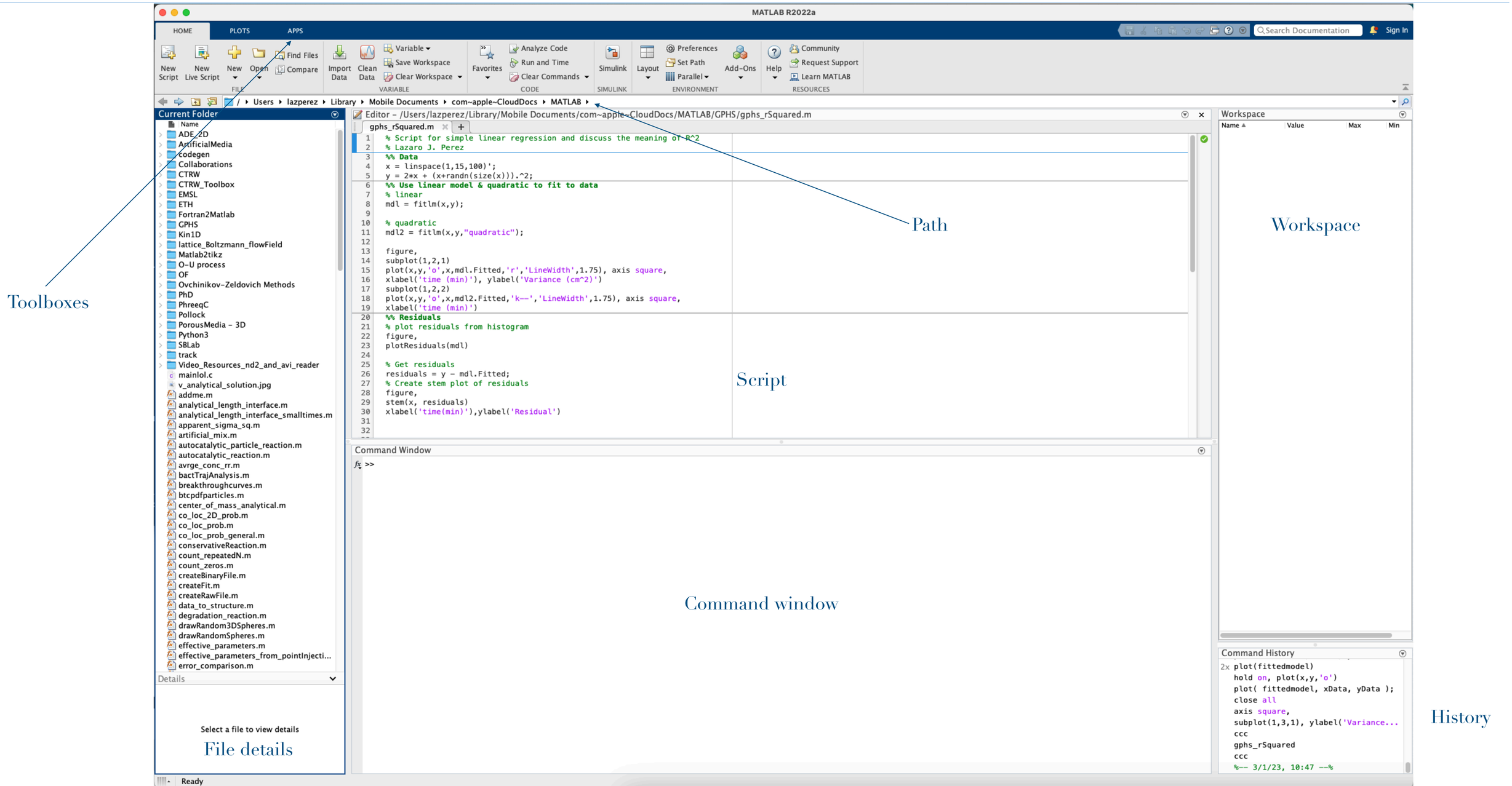
A script is a sequence of MATLAB instructions that are stored in an M-file and saved

Functions provide more flexibility than a script, primarily because you can pass input values and return output values. In addition, functions avoid storing temporary variables in the base workspace and can run faster than scripts

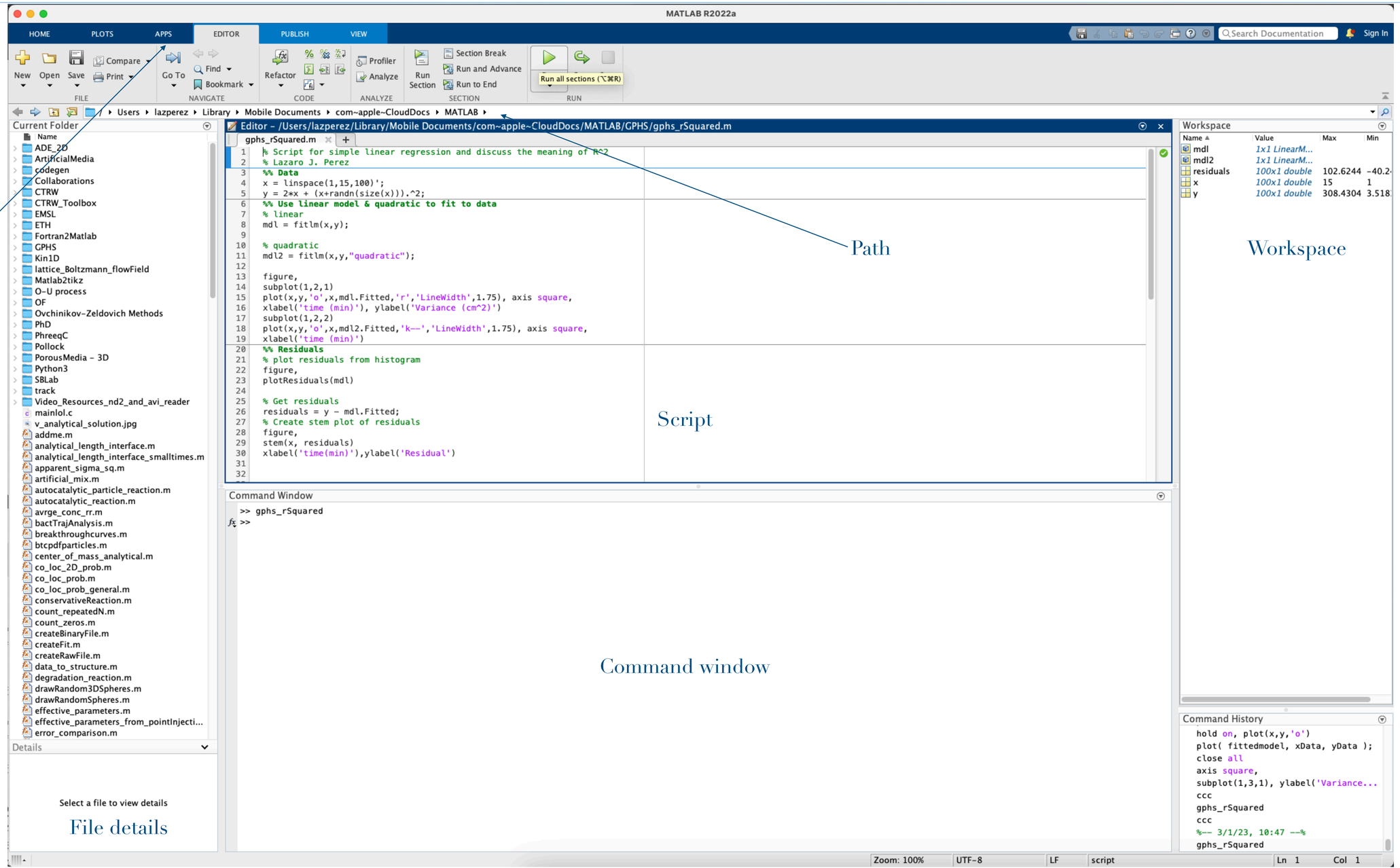
MATLAB: Desktop Environment



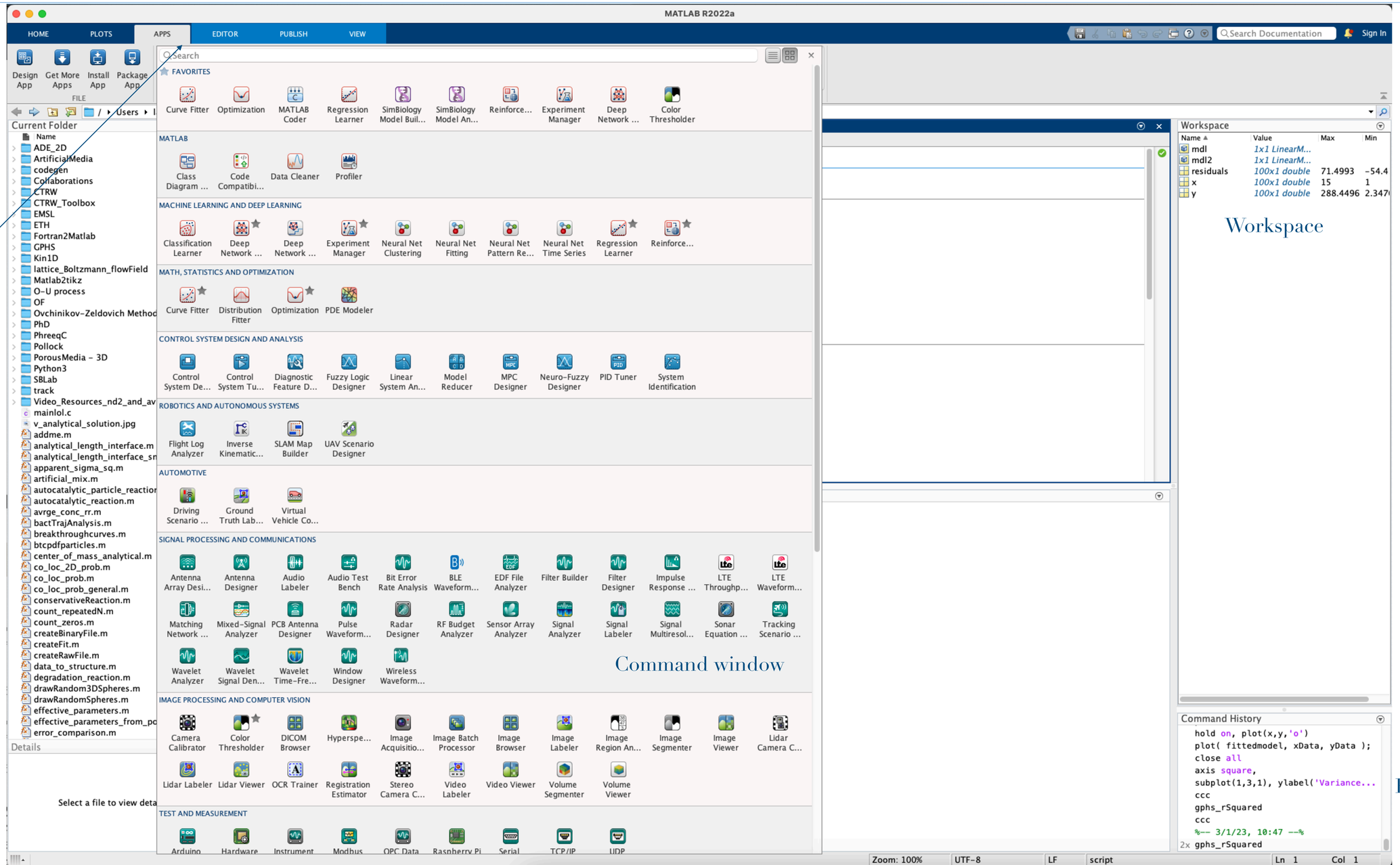
MATLAB: Desktop Environment



MATLAB: Desktop Environment



MATLAB: Desktop Environment



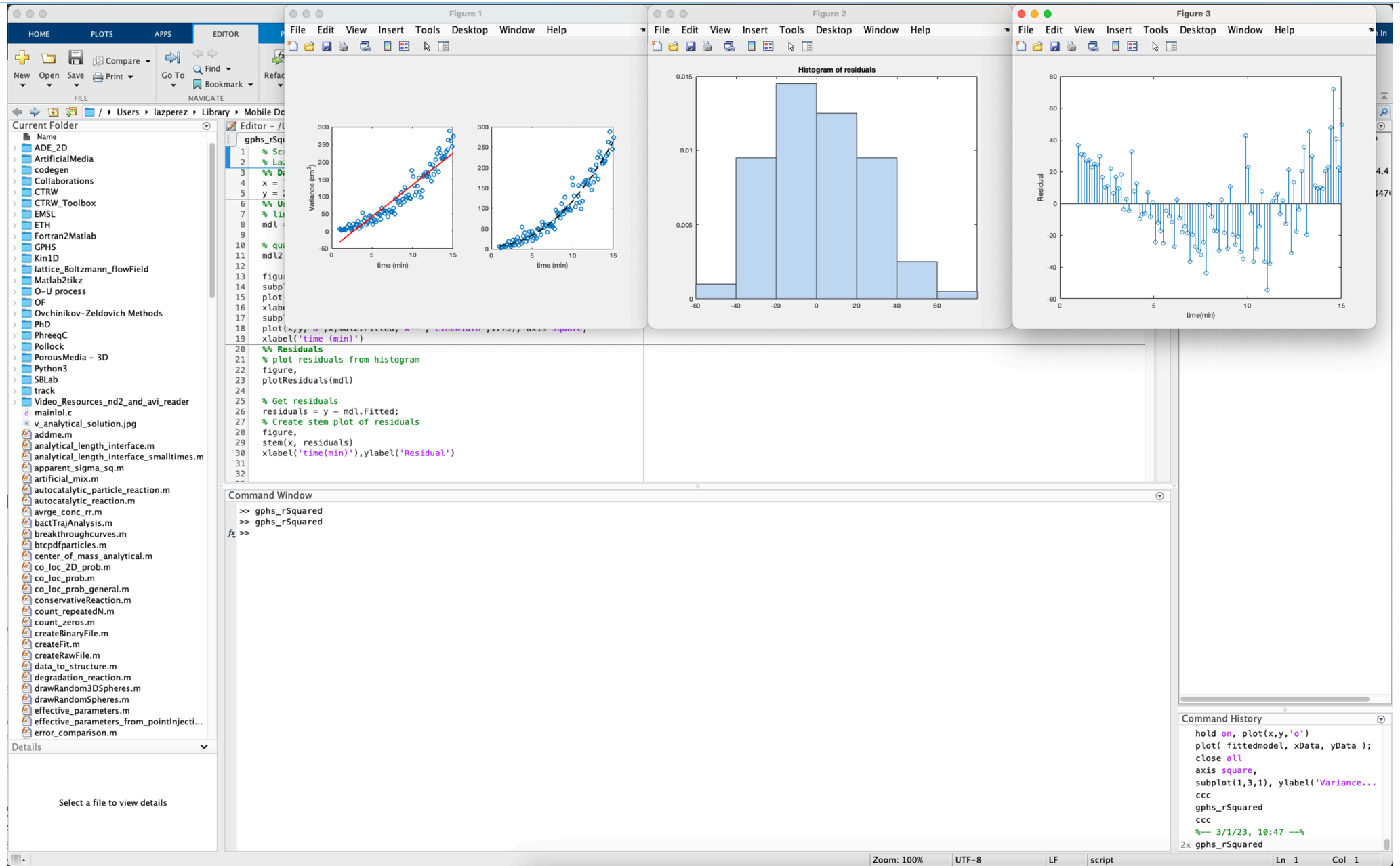
Toolboxes

Command window

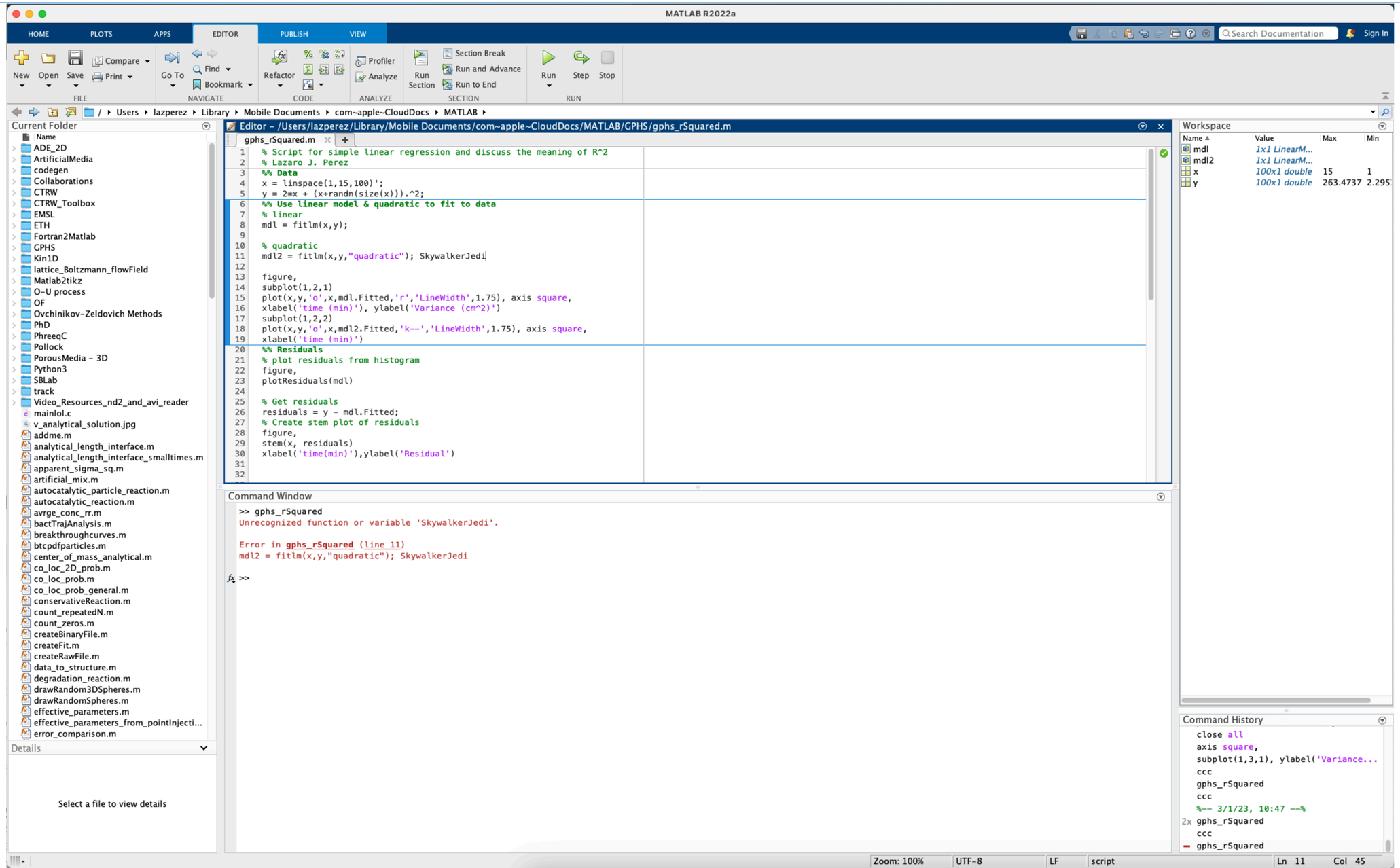
Workspace

History

MATLAB: Desktop Environment



MATLAB: Desktop Environment



Variables and assignments

Variable names are an example of identifier names

The name must begin with a letter of the alphabet

After that, the name can contain letters, digits, and the underscore character (e.g., value_1), but it cannot have a space

There is a limit to the length of the name; the built-in function namelengthmax tells what this maximum length is

MATLAB is case-sensitive

Putting a semicolon at the end of a statement suppresses the output

```
>> res = 9 - 2;
>>
```

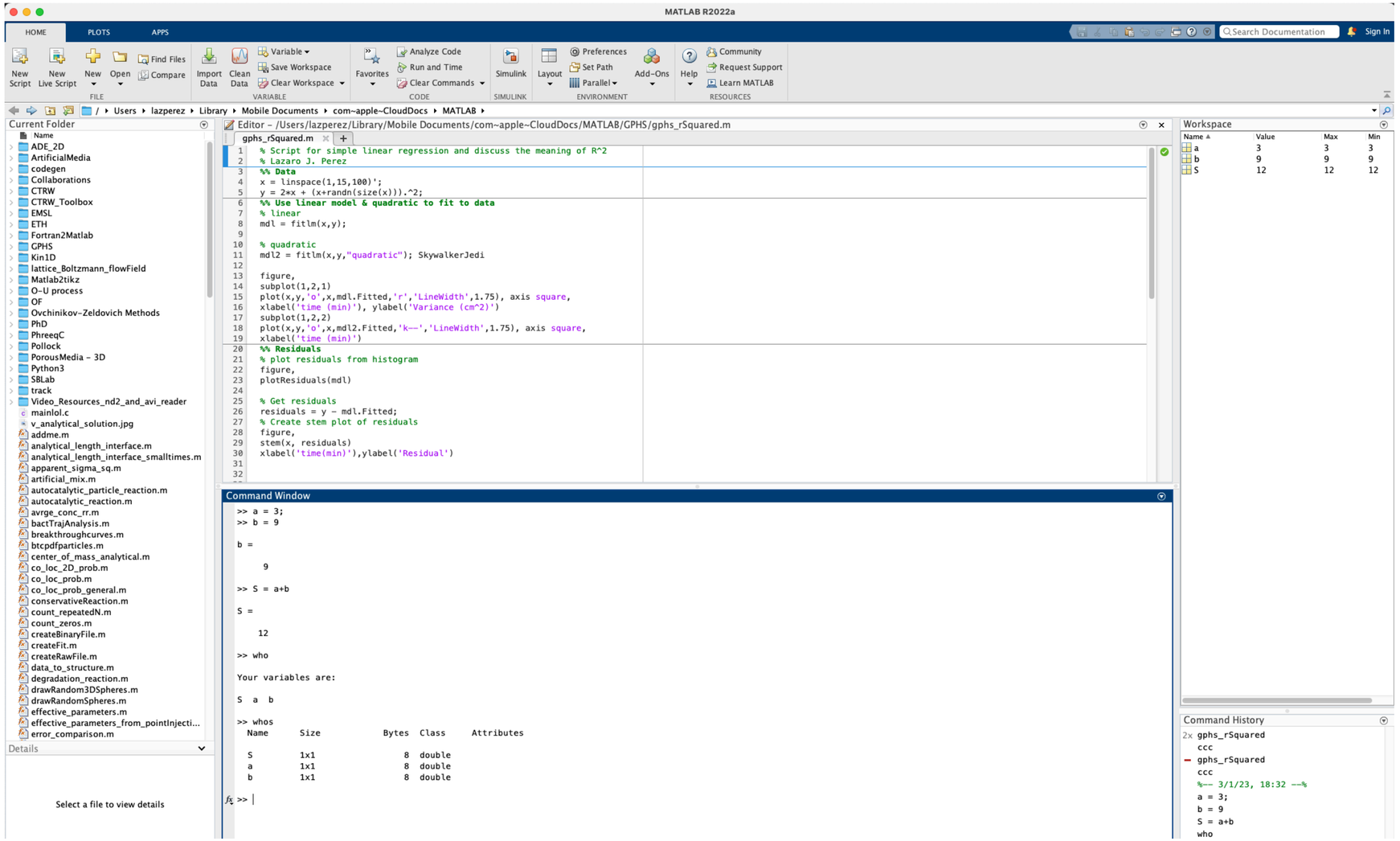
MATLAB uses a default variable named *ans* if an expression is typed at the prompt and it is not assigned to a variable

```
>> 6 + 3
ans =
     9
```

```
>> result = 6 + 3
result =
     9
```

Variables and assignments

The Workspace Window shows the variables that have been created in the current Command Window and their values



Variables and assignments

Operators

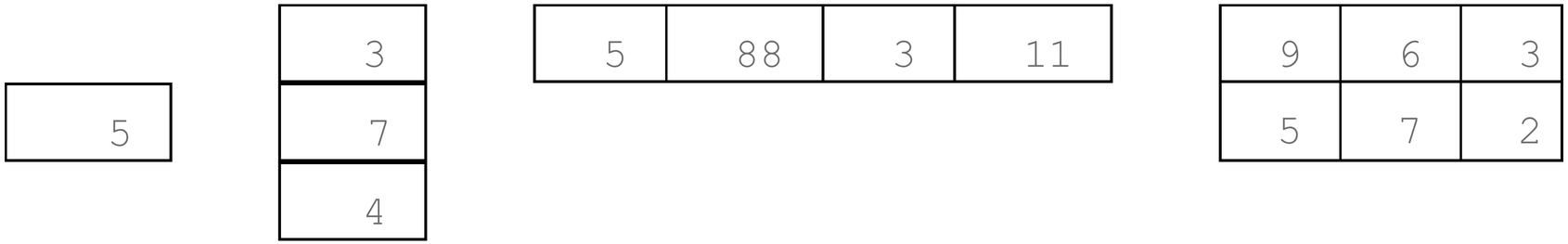
- `+` Addition
- `-` Negation, subtraction
- `*` Multiplication
- `/` Division (divided by, e.g., 10/5 is 2)
- `\` Division (divided into, e.g., 10\5 is 0.5)
- `^` Exponentiation

Matrices

Vectors and matrices store sets of values, all of which are the same type

```
>> num = 6 + 3;
>> numi = int32(num);
>> whos
  Name      Size    Bytes    Class     Attributes
  num       1 x 1       8    double
  numi      1 x 1       4    int32
```

A vector can be either a row vector or a column vector



Vectors and scalars are just special cases of matrices

Matrices

Vectors and matrices store sets of values, all of which are the same type

```
>> v = [1    2    3    4]
v =
     1     2     3     4
```

```
>> v = [1,2,3,4]
v =
     1     2     3     4
```

```
>> vec = 1:5
vec =
     1     2     3     4     5
```

Colon operator, linspace, and concatenating

```
>> nv = 1:2:9
nv =
     1     3     5     7     9
```

```
>> ls = linspace(3,15,5)
ls =
     3     6     9    12    15
```

```
>> newvec = [nv ls]
newvec =
     1     3     5     7     9     3     6     9    12    15
```

Indexing: starts at (1,1)

newvec

1	2	3	4	5	6	7	8	9	10
1	3	5	7	9	3	6	9	12	15

Matrices

Colon operator, linspace, and concatenating

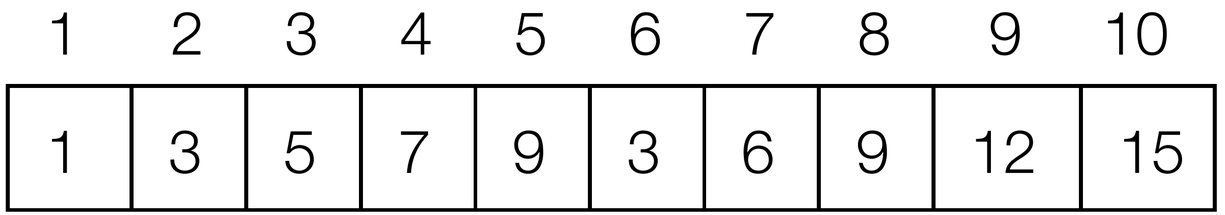
```
>> nv = 1:2:9
nv =
     1     3     5     7     9
```

```
>> ls = linspace(3,15,5)
ls =
     3     6     9    12    15
```

```
>> newvec = [nv ls]
newvec =
     1     3     5     7     9     3     6     9    12    15
```

Indexing: starts at (1,1)

newvec



Specific index

```
>> newvec(5)
ans =
     9
```

Index sequence

```
>> b = newvec(4:6)
b =
     7     9     3
```

Specific index vector

```
>> newvec([1  5  10])
ans =
     1     9    15
```

Matrices

Column vectors

```
>> c = [1; 2; 3; 4]
c =
     1
     2
     3
     4
```

One way to create a column vector is to explicitly put the values in square brackets, separated by semicolons (rather than commas or spaces)

Any row vector created using any method can be transposed to result in a column vector

```
>> r = 1:3;
>> c = r'
c =
     1
     2
     3
```

For vectors, transposing a row vector results in a column vector, and transposing a column vector results in a row vector

A built-in operator, the apostrophe, in MATLAB will transpose

Matrices

Creating matrix variables

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

Creating a matrix variable is simply a generalization of creating row and column vector variables
That is, the values within a row are separated by either spaces or commas, and the different rows are separated by semicolons

There must always be the same number of values in each row.
If you attempt to create a matrix in which there are different numbers of values in the rows, the result will be an error message, such as in the following:

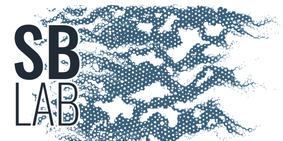
```
>> mat = [3 5 7; 1 2]
??? Error using ==> vertcat
CAT arguments dimensions are not consistent.
```

Matrix variables

```
>> rand(2)
ans =
    0.2311    0.4860
    0.6068    0.8913

>> rand(1,3)
ans =
    0.7621    0.4565    0.0185
```

```
>> zeros(3)
ans =
     0     0     0
     0     0     0
     0     0     0
```

```
>> ones(2,4)
ans =
     1     1     1     1
     1     1     1     1
```



Matrices

Referring to matrix elements

```
>> mat = [2:4; 3:5]
mat =
     2     3     4
     3     4     5
```

Indexing

```
>> mat(2,3)
ans =
     5
```

Subscript indexing

```
>> mat(1:2,2:3)
ans =
     3     4
     4     5
```

Row indexing

```
>> mat(1,:)
ans =
     2     3     4
```

Column indexing

```
>> mat(:, 2)
ans =
     3
     4
```

Single index: If a single index is used with a matrix, MATLAB unwinds the matrix column by column

```
>> intmat = [100 77; 28 14]
intmat =
   100    77
    28    14
```

```
>> intmat(1)
ans =
   100
```

```
>> intmat(2)
ans =
    28
```

```
>> intmat(4)
ans =
    14
```

Matrices

Deleting elements in vectors

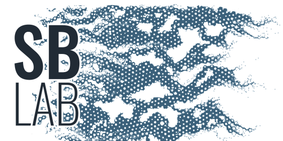
```
>> vec = 1:5
vec =
     1     2     3     4     5

>> vec(3) = []
vec =
     1     2     4     5
```

```
>> vec = 1:8
vec =
     1     2     3     4     5     6     7     8

>> vec(2:4) = []
vec =
     1     5     6     7     8
```

Deleting elements in matrices

```
>> mat = [7 9 8; 4 6 5]
mat =
     7     9     8
     4     6     5

>> mat(1,2) = [];
??? Indexed empty matrix assignment is not allowed.
```

```
>> mat(:,2) = []
mat =
     7     8
     4     5
```



Matrices

Deleting elements in matrices

```
>> mat = [7 9 8; 4 6 5]
mat =
     7     9     8
     4     6     5
```

```
>> mat(1,2) = [];
??? Indexed empty matrix assignment is not allowed.
```

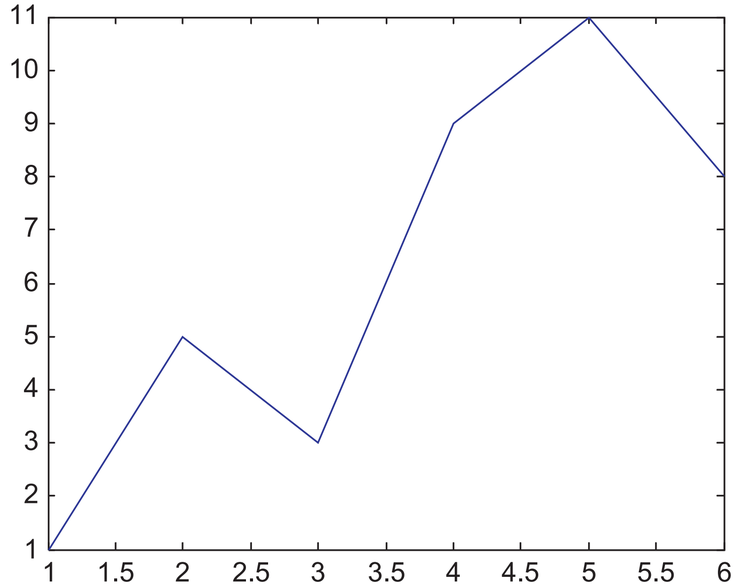
```
>> mat(:,2) = []
mat =
     7     8
     4     5
```

```
>> mat = [7 9 8; 4 6 5]
mat =
     7     9     8
     4     6     5
```

```
>> mat(3) = []
mat =
     7     4     6     8     5
```

Plotting

```
>> x = 1:6;
>> y = [1 5 3 9 11 8];
>> plot(x,y)
```



Simple related plot functions

clf clears the Figure Window by removing everything from it.

figure creates a new, empty Figure Window when called without any arguments. Calling it as *figure(n)* where n is an integer is a way of creating and maintaining multiple Figure Windows, and of referring to each individually.

hold is a toggle that freezes the current graph in the Figure Window, so that new plots will be superimposed on the current one. Just hold by itself is a toggle, so calling this function once turns the hold on, and then the next time turns it off. Alternatively, the commands hold on and hold off can be used.

legend displays strings passed to it in a legend box in the Figure Window, in order of the plots in the Figure Window.

Tables

Tables are a data type suitable for column-oriented or tabular data that is often stored as columns in a text file or in a spreadsheet

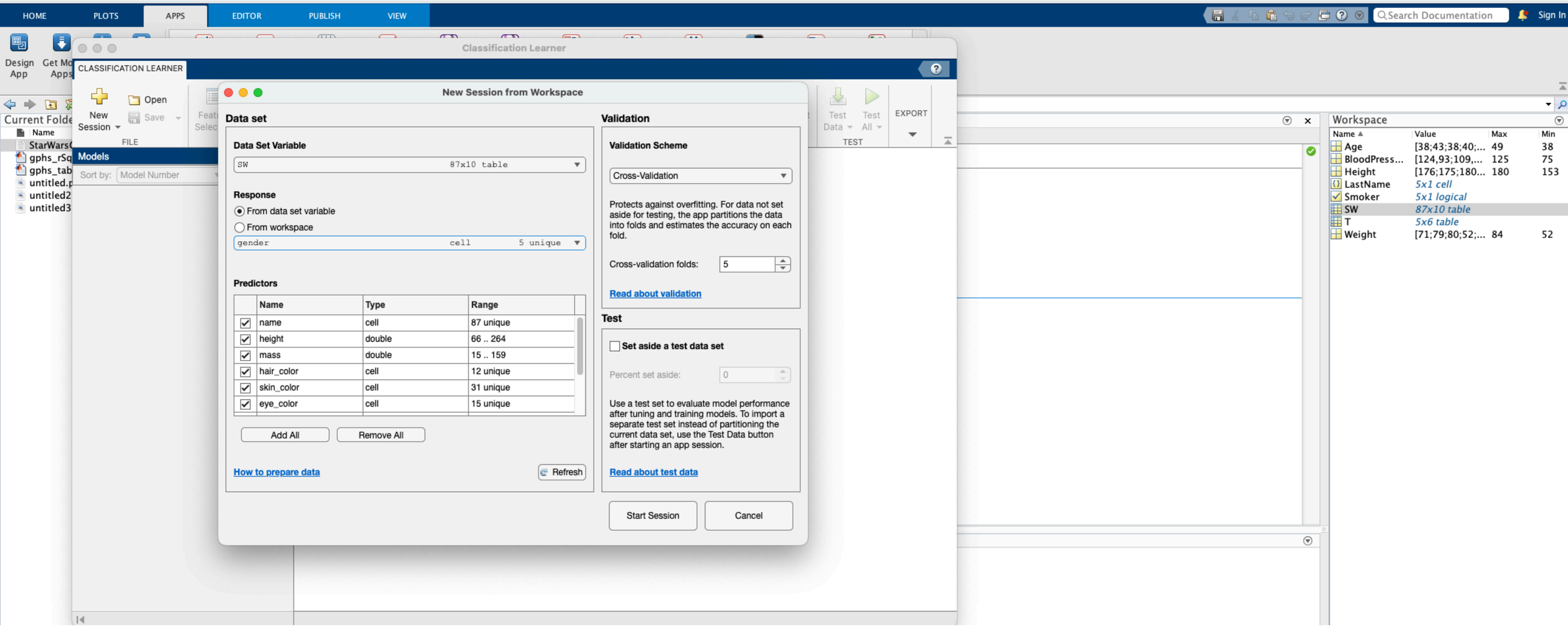
Tables consist of rows and column-oriented variables

Each variable in a table can have a different data type and a different size, with the one restriction that each variable must have the same number of rows

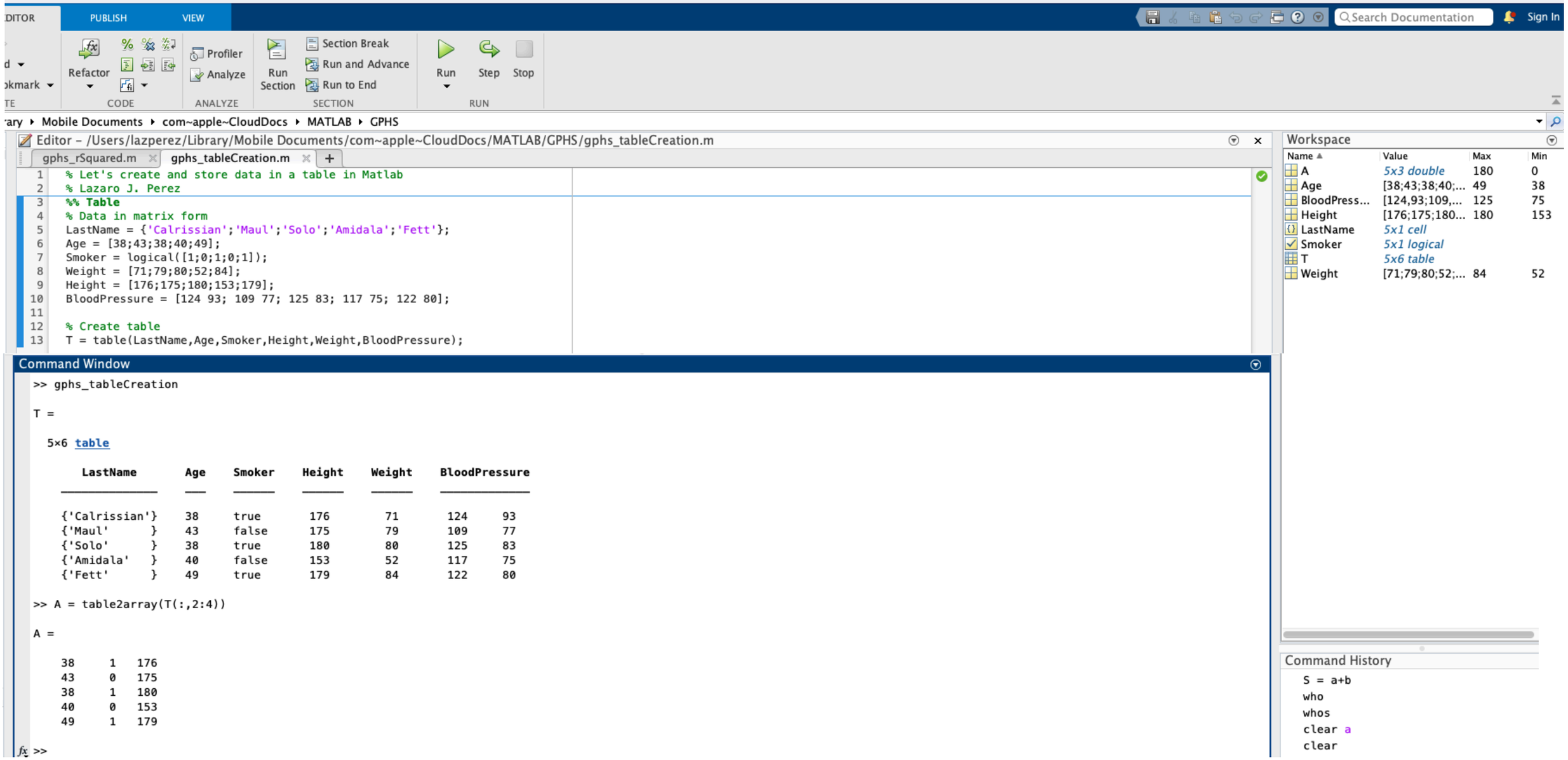
Key Matlab functions to work with tables

Function	Description
table	Table array with named variables that can contain different types
array2table	Convert homogeneous array to table
table2array	Convert table to homogeneous array

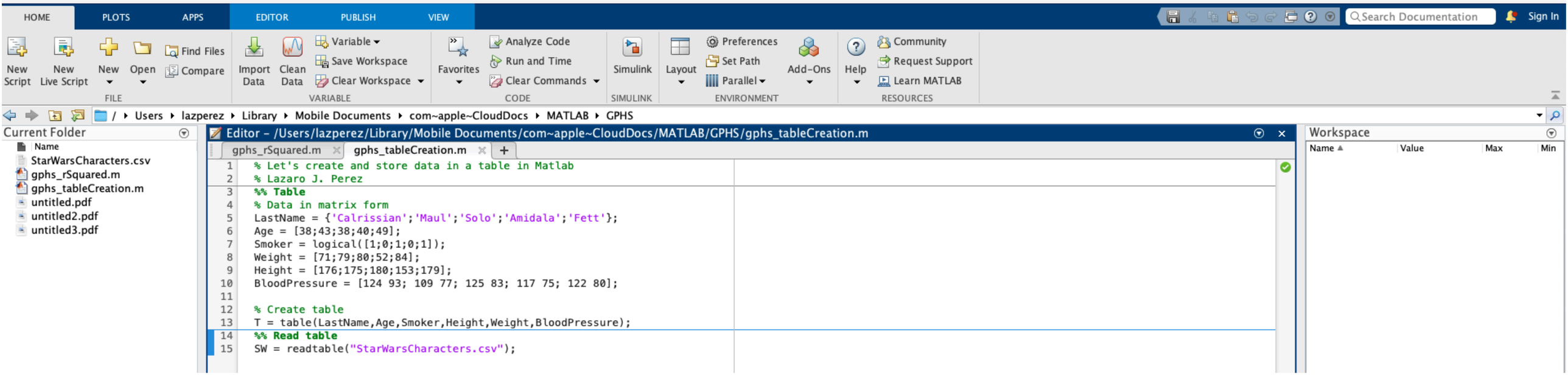
Tables


HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
Design App
Get More Apps
Classification Learner
CLASSIFICATION LEARNER
New Session
Open
Save
FILE
Test Data
Test All
TEST
EXPORT
Models
Sort by: Model Number
New Session from Workspace
Data set
Data Set Variable
SW 87x10 table
Response
From data set variable
From workspace
gender cell 5 unique
Predictors
Name Type Range
name cell 87 unique
height double 66 .. 264
mass double 15 .. 159
hair_color cell 12 unique
skin_color cell 31 unique
eye_color cell 15 unique
Add All
Remove All
How to prepare data
Refresh
Validation
Validation Scheme
Cross-Validation
Protects against overfitting. For data not set aside for testing, the app partitions the data into folds and estimates the accuracy on each fold.
Cross-validation folds: 5
Read about validation
Test
Set aside a test data set
Percent set aside: 0
Use a test set to evaluate model performance after tuning and training models. To import a separate test set instead of partitioning the current data set, use the Test Data button after starting an app session.
Read about test data
Start Session
Cancel
Workspace
Name Value Max Min
Age [38;43;38;40;... 49 38
BloodPress... [124,93;109,... 125 75
Height [176;175;180... 180 153
LastName 5x1 cell
Smoker 5x1 logical
SW 87x10 table
T 5x6 table
Weight [71;79;80;52;... 84 52

Tables

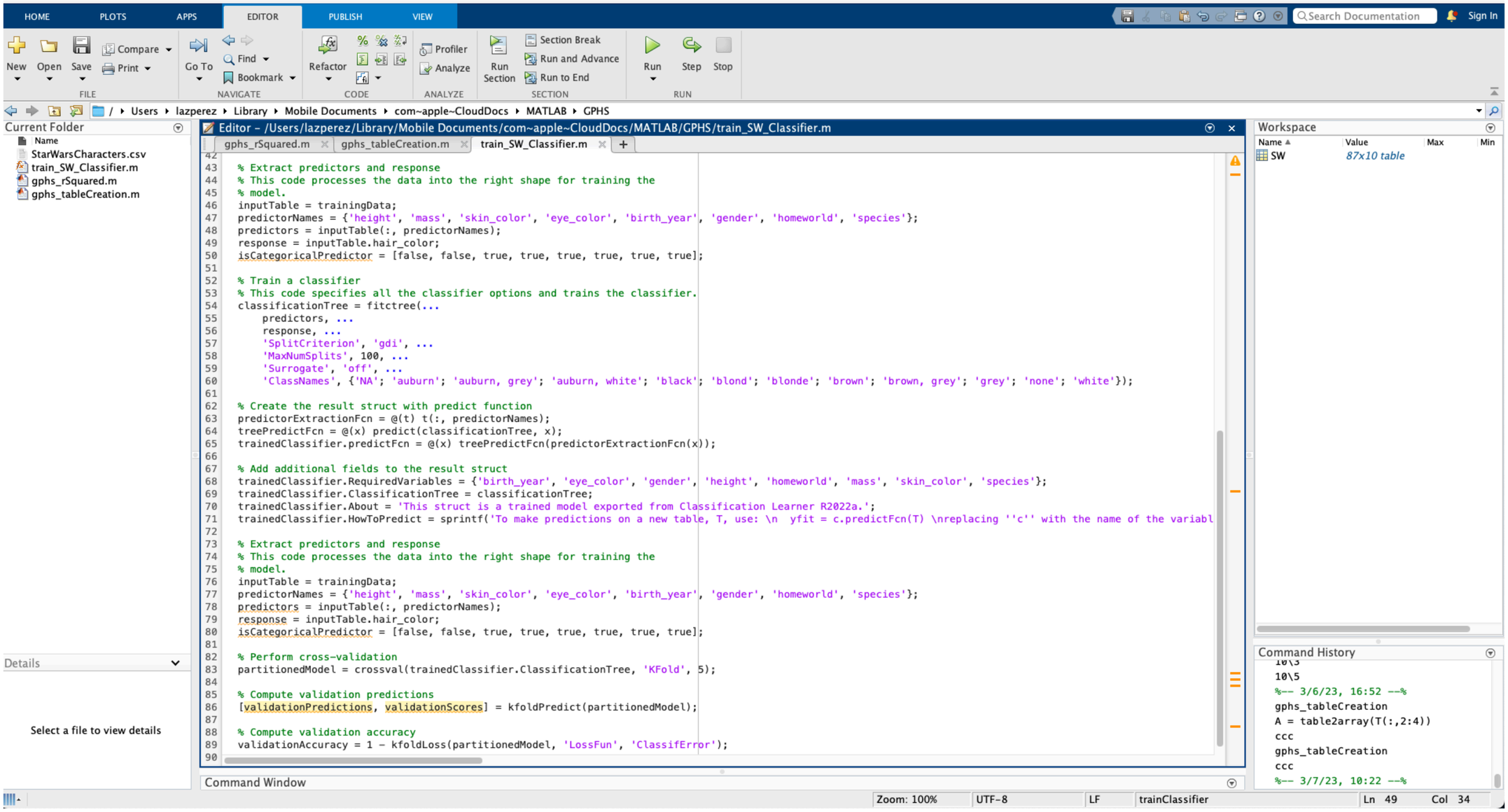

EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
Refactor
Profiler
Analyze
Run Section
Section Break
Run and Advance
Run to End
Run
Step
Stop
CODE
ANALYZE
SECTION
RUN
Mobile Documents ▸ com~apple~CloudDocs ▸ MATLAB ▸ GPHS
Editor - /Users/lazperez/Library/Mobile Documents/com~apple~CloudDocs/MATLAB/GPHS/gphs_tableCreation.m
gphs_rSquared.m
gphs_tableCreation.m
1 % Let's create and store data in a table in Matlab
2 % Lazaro J. Perez
3 %% Table
4 % Data in matrix form
5 LastName = {'Calrissian';'Maul';'Solo';'Amidala';'Fett'};
6 Age = [38;43;38;40;49];
7 Smoker = logical([1;0;1;0;1]);
8 Weight = [71;79;80;52;84];
9 Height = [176;175;180;153;179];
10 BloodPressure = [124 93; 109 77; 125 83; 117 75; 122 80];
11
12 % Create table
13 T = table(LastName,Age,Smoker,Height,Weight,BloodPressure);
Workspace
Name
Value
Max
Min
A 5x3 double 180 0
Age [38;43;38;40;... 49 38
BloodPress... [124,93;109,... 125 75
Height [176;175;180... 180 153
LastName 5x1 cell
Smoker 5x1 logical
T 5x6 table
Weight [71;79;80;52;... 84 52
Command Window
>> gphs_tableCreation
T =
5×6 table
LastName Age Smoker Height Weight BloodPressure
{'Calrissian'} 38 true 176 71 124 93
{'Maul' } 43 false 175 79 109 77
{'Solo' } 38 true 180 80 125 83
{'Amidala' } 40 false 153 52 117 75
{'Fett' } 49 true 179 84 122 80
>> A = table2array(T(:,2:4))
A =
38 1 176
43 0 175
38 1 180
40 0 153
49 1 179
>>
Command History
S = a+b
who
whos
clear a
clear

Tables


HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New Script
New Live Script
New
Open
Find Files
Compare
FILE
Import Data
Clean Data
Variable
Save Workspace
Clear Workspace
VARIABLE
Favorites
Analyze Code
Run and Time
Clear Commands
CODE
Simulink
SIMULINK
Layout
Preferences
Set Path
Parallel
ENVIRONMENT
Add-Ons
Help
Community
Request Support
Learn MATLAB
RESOURCES
/ ▸ Users ▸ lazperez ▸ Library ▸ Mobile Documents ▸ com~apple~CloudDocs ▸ MATLAB ▸ GPHS
Current Folder
Name
StarWarsCharacters.csv
gphs_rSquared.m
gphs_tableCreation.m
untitled.pdf
untitled2.pdf
untitled3.pdf
Editor - /Users/lazperez/Library/Mobile Documents/com~apple~CloudDocs/MATLAB/GPHS/gphs_tableCreation.m
gphs_rSquared.m
gphs_tableCreation.m
1 % Let's create and store data in a table in Matlab
2 % Lazaro J. Perez
3 %% Table
4 % Data in matrix form
5 LastName = {'Calrissian';'Maul';'Solo';'Amidala';'Fett'};
6 Age = [38;43;38;40;49];
7 Smoker = logical([1;0;1;0;1]);
8 Weight = [71;79;80;52;84];
9 Height = [176;175;180;153;179];
10 BloodPressure = [124 93; 109 77; 125 83; 117 75; 122 80];
11
12 % Create table
13 T = table(LastName,Age,Smoker,Height,Weight,BloodPressure);
14 %% Read table
15 SW = readtable("StarWarsCharacters.csv");
Workspace
Name
Value
Max
Min

Machine learning

Coding:



Machine learning

Toolbox: