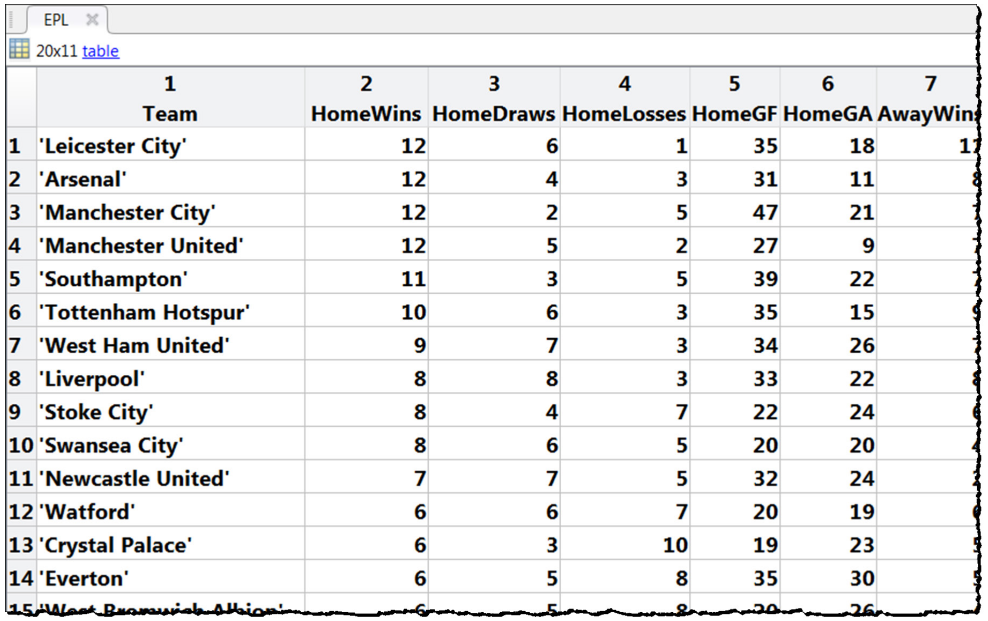
Tables of Data

MATLAB® Fundamentals for Aerospace Applications

Outline

- Storing data in tables
- Working with tables
- Indexing into tables
- Exporting from tables



EPL

20x11 table

	1 Team	2 HomeWins	3 HomeDraws	4 HomeLosses	5 HomeGF	6 HomeGA	7 AwayWin
1	'Leicester City'	12	6	1	35	18	1
2	'Arsenal'	12	4	3	31	11	8
3	'Manchester City'	12	2	5	47	21	7
4	'Manchester United'	12	5	2	27	9	7
5	'Southampton'	11	3	5	39	22	7
6	'Tottenham Hotspur'	10	6	3	35	15	9
7	'West Ham United'	9	7	3	34	26	7
8	'Liverpool'	8	8	3	33	22	8
9	'Stoke City'	8	4	7	22	24	6
10	'Swansea City'	8	6	5	20	20	4
11	'Newcastle United'	7	7	5	32	24	2
12	'Watford'	6	6	7	20	19	6
13	'Crystal Palace'	6	3	10	19	23	5
14	'Everton'	6	5	8	35	30	5
15	'West Bromwich Albion'	6	5	8	20	26	[illegible]

Chapter Learning Outcomes

The attendee will be able to:

- Import data into a MATLAB® table.
- Access and manipulate the data stored in a table.

Course Example: Premier League Football

The Excel® spreadsheet `EPLresults.xlsx` contains a table of data for the 2015-2016 season of English Premier League football (soccer). This table includes textual information (team name) and numeric data (home and away wins, draws, losses, goals for, and goals against).

Consider some simple common applications of working with this kind of tabular data, such as sorting by a particular statistic (e.g., home wins) and calculating derived information (e.g., league standing).

Storing this data as a table in MATLAB allows you to perform this kind of analysis easily.

Try

Open `EPLresults.xlsx` in Excel.

Team	Home Wins	Home Draws	Home Losses	Home GF	Home GA	Away Wins	Away Draws	Away Losses	Away GF	Away GA
Arsenal	12	4	3	31	11	8	7	4	34	25
Aston Villa	2	5	12	14	35	1	3	15	13	41
Bournemouth	5	5	9	23	34	6	4	9	22	33
Chelsea	5	9	5	32	30	7	5	7	27	23
Crystal Palace	6	3	10	19	23	5	6	8	20	28
Everton	6	5	8	35	30	5	9	5	24	25
Leicester City	12	6	1	35	18	11	6	2	33	18
Liverpool	8	8	3	33	22	8	4	7	30	28
Manchester City	12	2	5	47	21	7	7	5	24	20
Manchester United	12	5	2	27	9	7	4	8	22	26
Newcastle United	7	7	5	32	24	2	3	14	12	41
Norwich City	6	5	8	26	30	3	2	14	13	37
Southampton	11	3	5	39	22	7	6	6	20	19
Stoke City	8	4	7	22	24	6	5	8	19	31
Sunderland	6	6	7	23	20	3	6	10	25	42
Swansea City	8	6	5	20	20	4	5	10	22	32
Tottenham Hotspur	10	6	3	35	15	9	7	3	34	20
Watford	6	6	7	20	19	6	3	10	20	31
West Bromwich Albion	6	5	8	20	26	4	8	7	14	22
West Ham United	9	7	3	34	26	7	7	5	31	25

What Is a Table?

Many data sets conform to a particular tabular arrangement with the following properties:

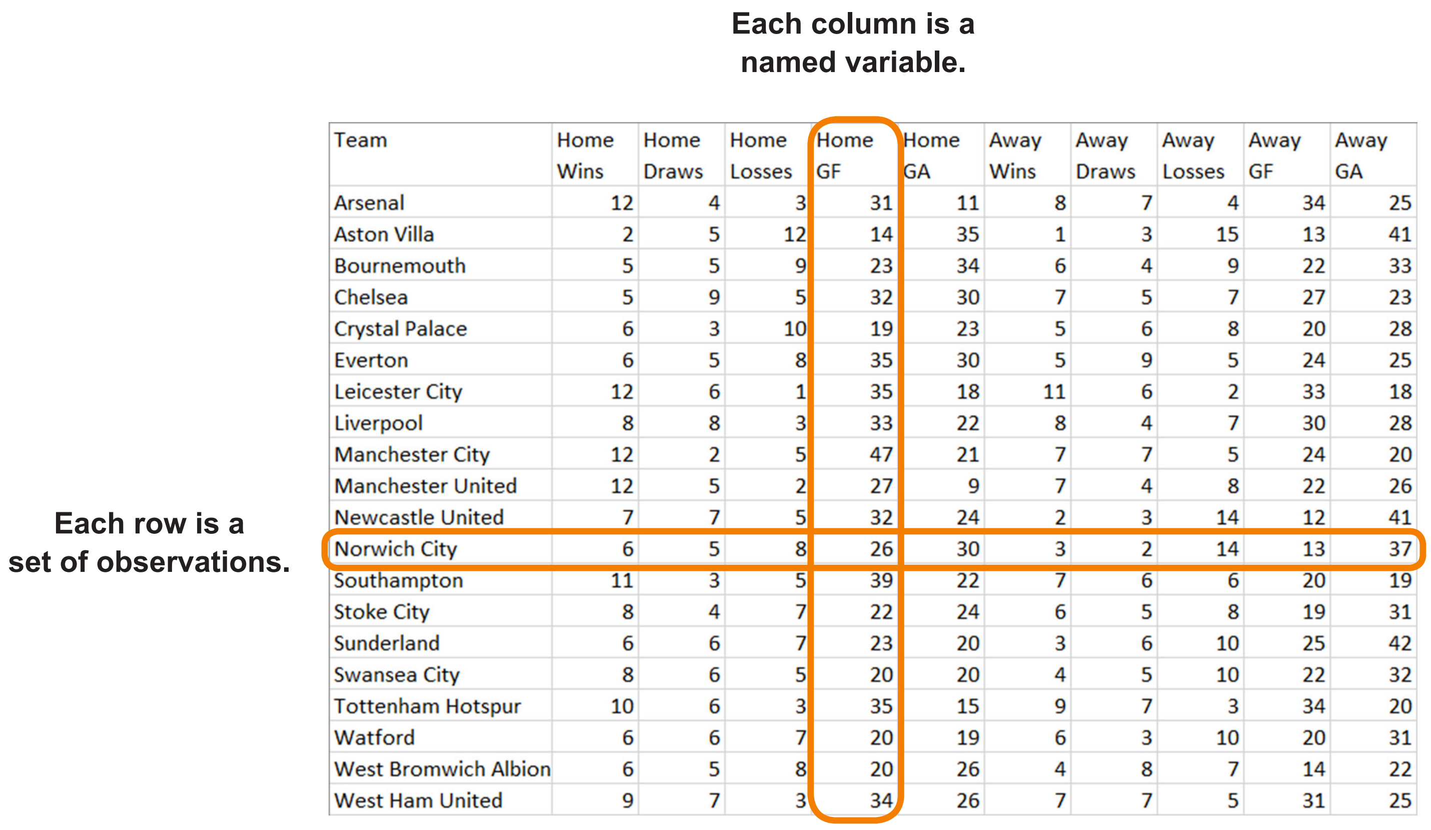
- Each column of data has the same type (text, numeric, logical (T/F), etc.).
- Different columns, however, can have different types.
- Each column typically has a unique name.
- Each column has the same number of rows.

That is, the data can be thought of some number of *observations* (rows) of a set of different *variables* (columns).

In this configuration, you typically want to index into columns by name, and index into rows across all columns (i.e., keep observations together).

You can store each column as a separate MATLAB variable, but this makes it harder to index into multiple variables for the same row (observation). A matrix makes it easy to index into entire rows or columns, but does not allow indexing by name, or different data types in different columns.

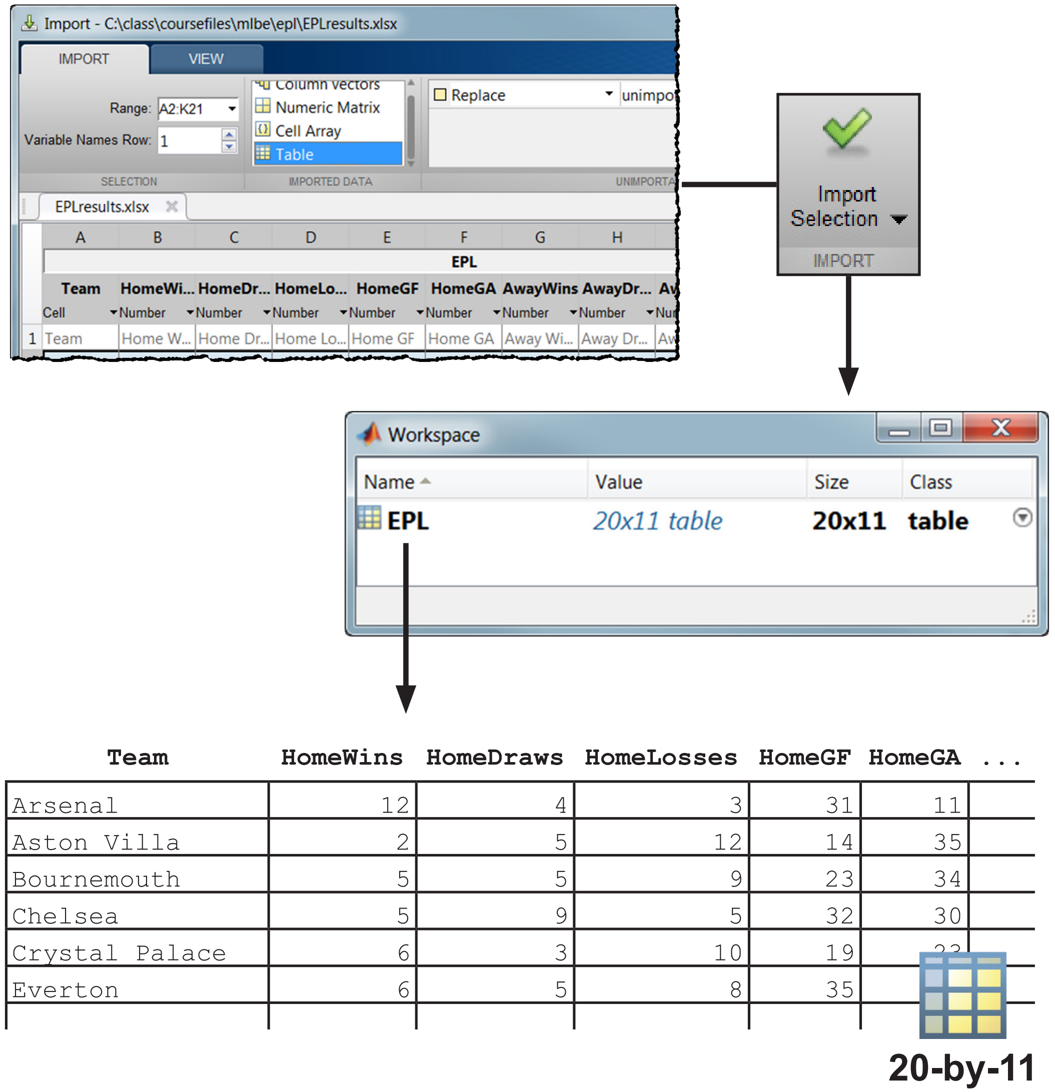
MATLAB provides the `table` data type for storing and manipulating tabular data.


Each column is a named variable.
Each row is a set of observations.

Team	Home Wins	Home Draws	Home Losses	Home GF	Home GA	Away Wins	Away Draws	Away Losses	Away GF	Away GA
Arsenal	12	4	3	31	11	8	7	4	34	25
Aston Villa	2	5	12	14	35	1	3	15	13	41
Bournemouth	5	5	9	23	34	6	4	9	22	33
Chelsea	5	9	5	32	30	7	5	7	27	23
Crystal Palace	6	3	10	19	23	5	6	8	20	28
Everton	6	5	8	35	30	5	9	5	24	25
Leicester City	12	6	1	35	18	11	6	2	33	18
Liverpool	8	8	3	33	22	8	4	7	30	28
Manchester City	12	2	5	47	21	7	7	5	24	20
Manchester United	12	5	2	27	9	7	4	8	22	26
Newcastle United	7	7	5	32	24	2	3	14	12	41
Norwich City	6	5	8	26	30	3	2	14	13	37
Southampton	11	3	5	39	22	7	6	6	20	19
Stoke City	8	4	7	22	24	6	5	8	19	31
Sunderland	6	6	7	23	20	3	6	10	25	42
Swansea City	8	6	5	20	20	4	5	10	22	32
Tottenham Hotspur	10	6	3	35	15	9	7	3	34	20
Watford	6	6	7	20	19	6	3	10	20	31
West Bromwich Albion	6	5	8	20	26	4	8	7	14	22
West Ham United	9	7	3	34	26	7	7	5	31	25

Storing Data as a Table

When importing data from a file with the Import Tool, you can specify the type of the output. Choosing the **Table** option will bring the data into the MATLAB workspace as a single `table` variable. By default the name of the table will be derived from the file name.


Import - C:\class\coursefiles\mlbe\epl\EPLresults.xlsx
IMPORT
VIEW
Range: A2:K21
Variable Names Row: 1
Column vectors
Numeric Matrix
Cell Array
Table
Replace
unimpo
SELECTION
IMPORTED DATA
UNIMPORTA
EPLresults.xlsx
A B C D E F G H
EPL
Team HomeWi... HomeDr... HomeLo... HomeGF HomeGA AwayWins AwayDr... Av
Cell Number Number Number Number Number Number Number Nu
1 Team Home W... Home Dr... Home Lo... Home GF Home GA Away Wi... Away Dr... Aw
Import Selection
IMPORT
Workspace
Name Value Size Class
EPL 20x11 table 20x11 table
20-by-11

Team	HomeWins	HomeDraws	HomeLosses	HomeGF	HomeGA	...
Arsenal	12	4	3	31	11	
Aston Villa	2	5	12	14	35	
Bournemouth	5	5	9	23	34	
Chelsea	5	9	5	32	30	
Crystal Palace	6	3	10	19	[illegible]	
Everton	6	5	8	35	[illegible]	

Try

Use the Import Tool to import data from `EPLresults.xlsx` as a table.

You may want to rename the table from `EPLresults` to something shorter (such as `EPL`). You can do this either before or after importing.

After importing, open the table in the Variable Editor. Check that the variable names have been correctly inferred from the file.

Like a matrix, the table will be *m*-by-*n*, where *m* is the number of data observations (rows) and *n* is the number of variables (columns). The *n* variables will be named, using header information in the file, if possible.

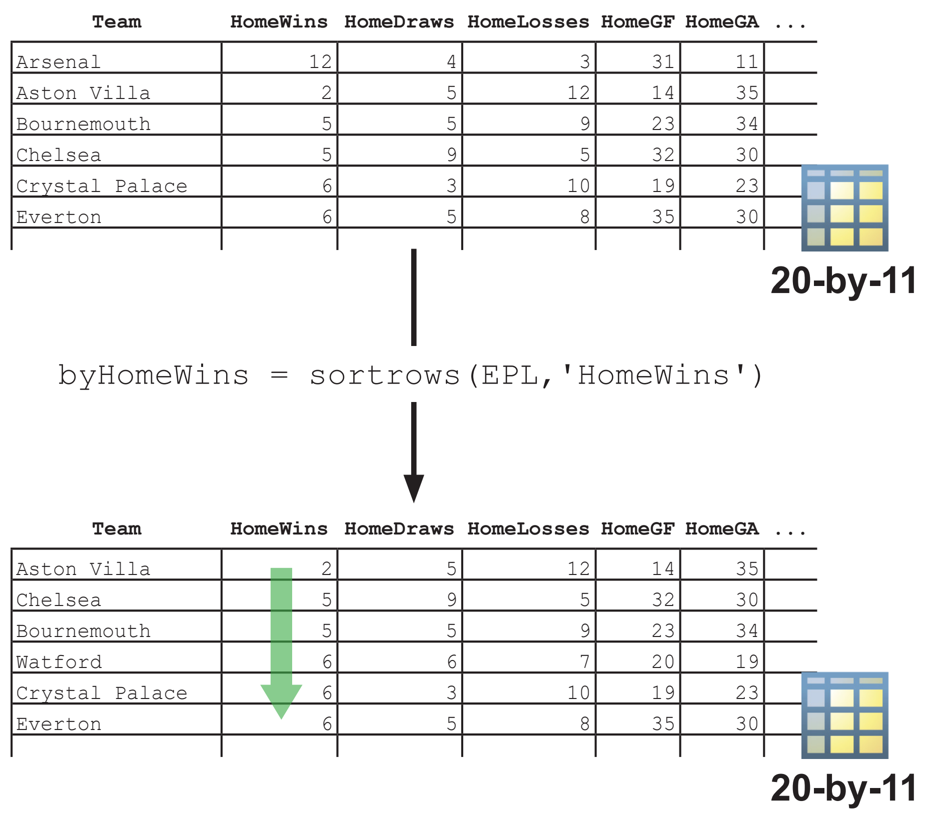
The names of the variables in the table must follow the same rules for names of variables in the workspace. If necessary, the Import Tool will modify the header information to make valid variable names.

You can create a table directly from other variables in the workspace using the `table` function. Alternatively, you can convert data stored in other data types to tables using conversion functions such as `array2table` and `cell2table`:

```
x = rand(5,3);
y = array2table(x,'VariableNames',{'A','B','C'})
```

Operating on Tables

Some operations are intended to work on tables as a single unit. For example, the `sortrows` function will sort an entire table according to a given variable:



Team	HomeWins	HomeDraws	HomeLosses	HomeGF	HomeGA	...
Arsenal	12	4	3	31	11	
Aston Villa	2	5	12	14	35	
Bournemouth	5	5	9	23	34	
Chelsea	5	9	5	32	30	
Crystal Palace	6	3	10	19	23	
Everton	6	5	8	35	30	

```
byHomeWins = sortrows(EPL,'HomeWins')
```

Team	HomeWins	HomeDraws	HomeLosses	HomeGF	HomeGA	...
Aston Villa	2	5	12	14	35	
Chelsea	5	9	5	32	30	
Bournemouth	5	5	9	23	34	
Watford	6	6	7	20	19	
Crystal Palace	6	3	10	19	23	
Everton	6	5	8	35	30	

Try

Create a table of the results sorted by home wins. Note the size and type of the result.

```
  byHomeWins = sortrows(EPL,'HomeWins')
```

Sort by home wins in descending order.

```
  byHomeWins = sortrows(EPL,'HomeWins','descend')
```

Sort `EPL` by home wins, then by away wins.

```
  EPL = sortrows(EPL,...
      {'HomeWins','AwayWins'},'descend');
```

View summary statistics for all the numeric specifications.

```
  summary(EPL)
```

By default, the rows are sorted in ascending order. Sort in descending order by passing the option `'descend'`.

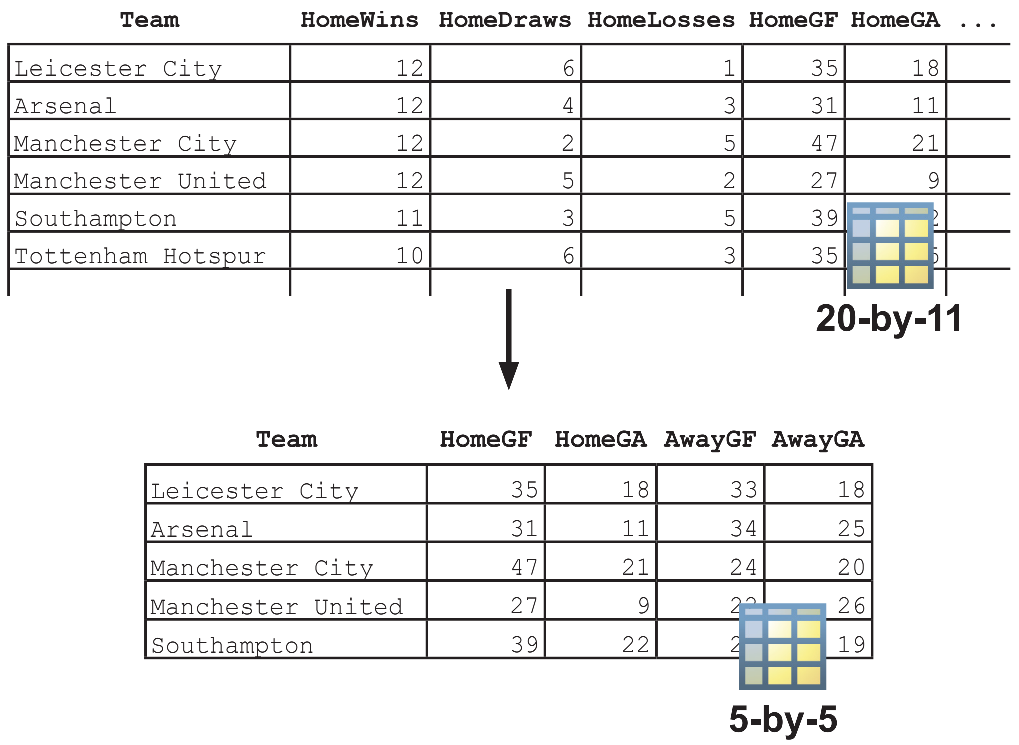
```
byHomeWins = sortrows(EPL,'HomeWins','descend')
```

Extracting Portions of a Table

You can index into a table using regular array indexing with parentheses:

```
top5scoring = EPL(1:5,[1,5:6,10:11])
```

As with any array indexing, the result is a subset of the original data. In this case, `top5scoring` will be a 5-by-5 table.



Team	HomeWins	HomeDraws	HomeLosses	HomeGF	HomeGA	...
Leicester City	12	6	1	35	18	
Arsenal	12	4	3	31	11	
Manchester City	12	2	5	47	21	
Manchester United	12	5	2	27	9	
Southampton	11	3	5	39	[illegible]	
Tottenham Hotspur	10	6	3	35	[illegible]	

20-by-11

Team	HomeGF	HomeGA	AwayGF	AwayGA
Leicester City	35	18	33	18
Arsenal	31	11	34	25
Manchester City	47	21	24	20
Manchester United	27	9	[illegible]	26
Southampton	39	22	[illegible]	19

5-by-5

One benefit of storing data as a table is that you can index into the variables using either a number or a string corresponding to a variable name:

```
top5HGF = EPL(1:5,'HomeGF')
```

Try

View the results for the top 10 teams. Note the size and type of the result.

```
EPL(1:10,:)
```

Use both numeric and named indexing to create a new table containing just the home wins. Note the size and type of the result.

```
hwTable = EPL(:,2)
hwTable = EPL(:,'HomeWins')
```

Use numeric indexing to create a new table containing the home and away wins for the top 10 teams.

```
Top10Wins = EPL(1:10,[1,2,7]);
```

Create the same table using named indexing.

```
vars = {'Team','HomeWins','AwayWins'};
Top10Wins = EPL(1:10,vars);
```

To index into multiple variables, use a cell array of character vectors:

```
v = {'Team','HomeGF','HomeGA',AwayGF','AwayGA'};
top5scoring = EPL(1:5,v)
```

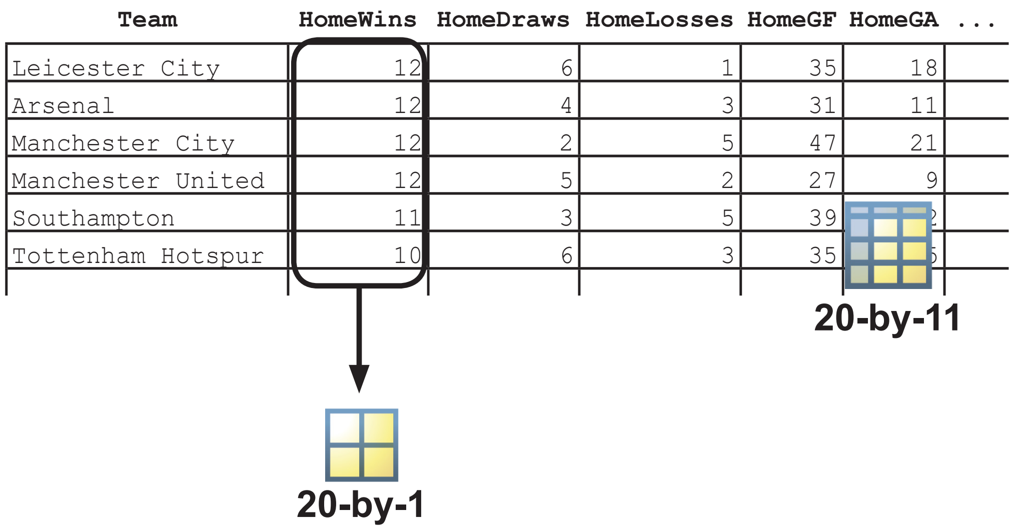
Extracting Data from a Table

To perform analysis on the data contained in a table, it is typically necessary to extract the data from the table before passing it to another function (such as `plot`).

You can index into variables in a table using dot (`.`) notation to reference the variables by name:

```
homeWins = EPL.HomeWins
```

This extracts the actual variable that was stored in the table. The result will therefore be of the same size and type as that variable.



Team	HomeWins	HomeDraws	HomeLosses	HomeGF	HomeGA	...
Leicester City	12	6	1	35	18	
Arsenal	12	4	3	31	11	
Manchester City	12	2	5	47	21	
Manchester United	12	5	2	27	9	
Southampton	11	3	5	39	2	
Tottenham Hotspur	10	6	3	35	5	

Try

Use dot referencing to extract home wins. Note the size and type of the result.

```
homeWins = EPL.HomeWins;
```

Repeat with numeric and named indexing.

```
homeWins = EPL{:,2};
homeWins = EPL{:,'HomeWins'};
```

Is there a home field advantage?

```
awayWins = EPL.AwayWins;
scatter(homeWins,awayWins)
```

Use numeric indexing to extract home wins, draws, and losses as a matrix.

```
homeRecord = EPL{:,2:4}
```

Use named indexing to extract home wins, draws, and losses as a matrix.

```
vars = {'HomeWins','HomeDraws','HomeLosses'};
homeRecord = EPL{:,vars};
```

You can also use curly braces (`{}`) to index into variables in a table. You can index numerically

```
wins = EPL{:,[2,7]}
```

or by name

```
gf = EPL{:,{'HomeGF','AwayGF'}}
```

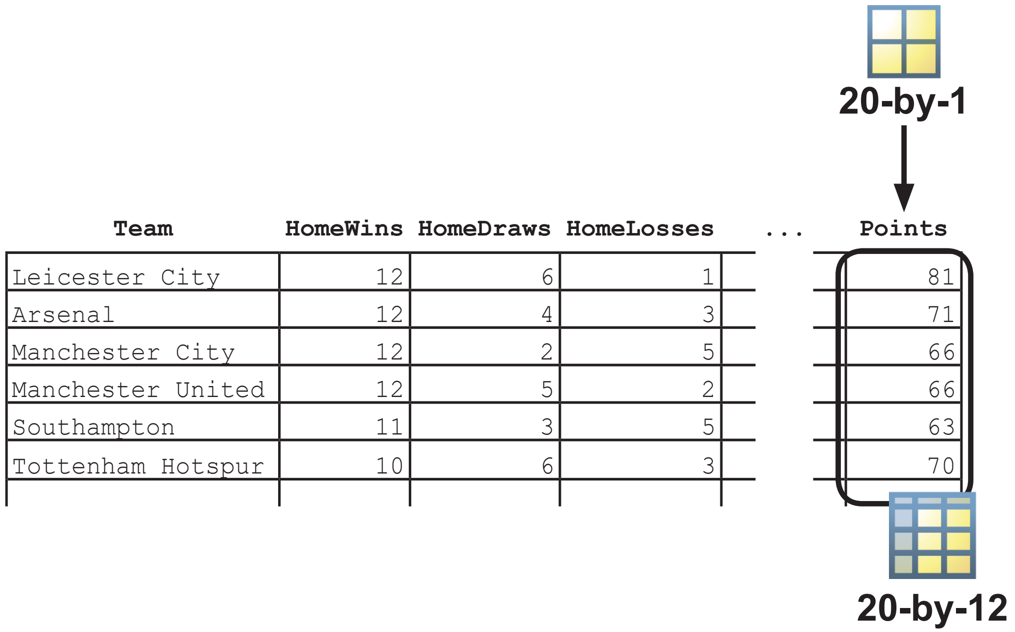
Note that, when you extract multiple variables, all variables must be of the same type.

Modifying Tables

As with any array in MATLAB, you can modify tables using indexed assignment:

```
EPL.Points = points
EPL{:,'Points'} = points
EPL{:,12} = points
```

If the specified variable exists in the table, it will be overwritten. If not, it will be added.



Team	HomeWins	HomeDraws	HomeLosses	...	Points
Leicester City	12	6	1		81
Arsenal	12	4	3		71
Manchester City	12	2	5		66
Manchester United	12	5	2		66
Southampton	11	3	5		63
Tottenham Hotspur	10	6	3		70

Try

Calculate the total points for each team, where wins are worth 3 points, draws worth 1 point, and losses worth 0 points. Add the total points to the table as a new variable.

```
wins = EPL.HomeWins + EPL.AwayWins;
draws = EPL.HomeDraws + EPL.AwayDraws;
points = 3*wins + draws;
EPL.Points = points;
```

Calculate the goal differential by finding the difference between goals for (GF) and goals against (GA).

```
gf = EPL.HomeGF + EPL.AwayGF;
ga = EPL.HomeGA + EPL.AwayGA;
EPL.GD = gf - ga;
```

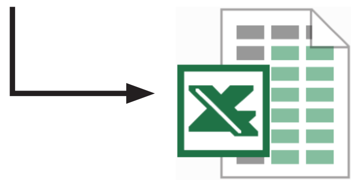
Find the league standings. Sort the table by points, and then by goal differential.

```
standings = EPL(:,{'Team','Points','GD'});
standings = sortrows(standings,...
    {'Points','GD'},'descend');
```

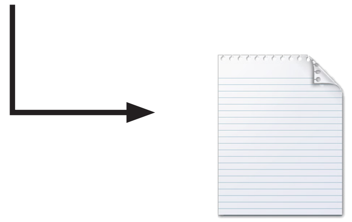
Exporting Tables

You can use the `writetable` function to export a table to a delimited text file or Excel spreadsheet. By default, `writetable` uses the file name to determine the appropriate output format:

```
writetable(EPL,'filename.xlsx')
```



```
writetable(EPL,'filename.txt')
```



Use extra arguments to specify options such as the spreadsheet range or the text delimiter:

```
writetable(EPL,'filename.xlsx',...
    'Range','C5:O24')

writetable(EPL,'filename.txt',...
    'Delimiter','\t')
```

Try

Save `standings` to an Excel spreadsheet and to a delimited text file.

```
writetable(standings,'EPLstandings.xlsx')
writetable(standings,'EPLstandings.txt')
```

Summary

- Storing data in tables
- Working with tables
- Indexing into tables
- Exporting from tables

Try

Open and run `EPLtable.mlx`.

Function	Use
`table` `array2table`	Create a table from workspace variables
`sortrows`	Sort a table by given variables
`summary`	Display summary statistics of data in a table

`.`
`{}`
	Index into table variables

`writetable`	Export table to spreadsheet or delimited text file

Test Your Knowledge

Name:____

1. If `x` is a 20-by-5 table with variables “A”, “B”, “C”, “D”, and “E”, which are all numeric vectors, the command `y = x.C` will return:

   A. A 20-by-1 numeric vector

   B. A 20-by-1 table

   C. A 20-by-5 numeric array

   D. A 1-by-1 table

   E. An error message

2. If `x` is a 20-by-5 table with variables “A”, “B”, “C”, “D”, and “E”, which are all numeric vectors, the command `y = x(:,'C')` will return:

   A. A 20-by-1 numeric vector

   B. A 20-by-1 table

   C. A 20-by-5 numeric array

   D. A 1-by-1 table

   E. An error message