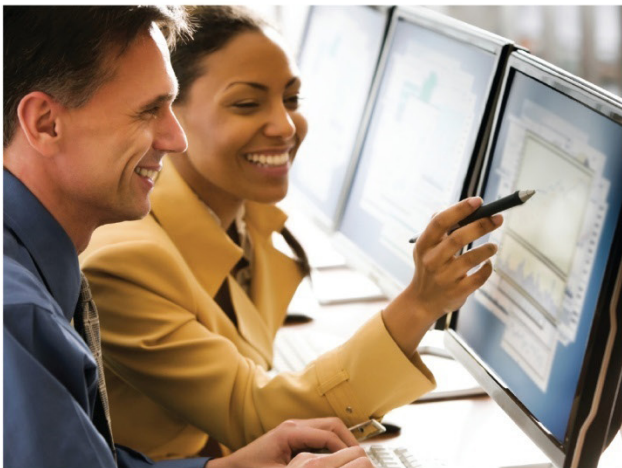


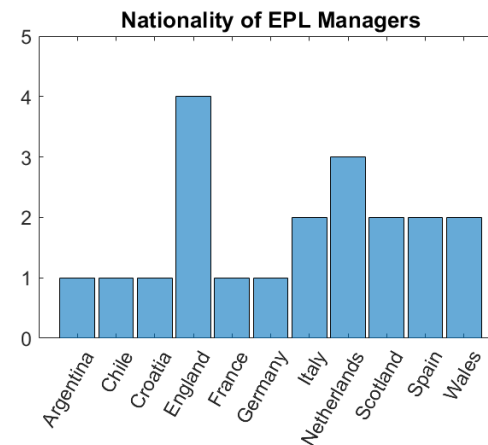
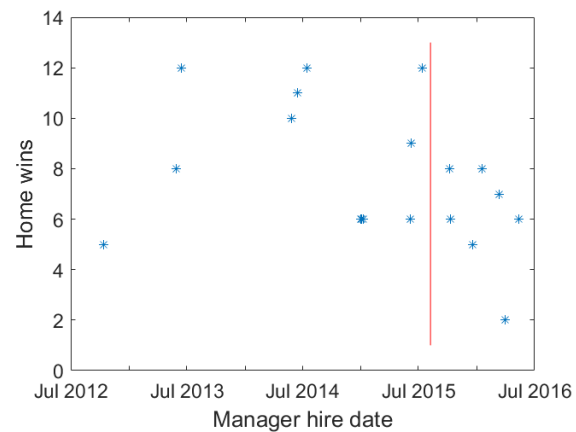
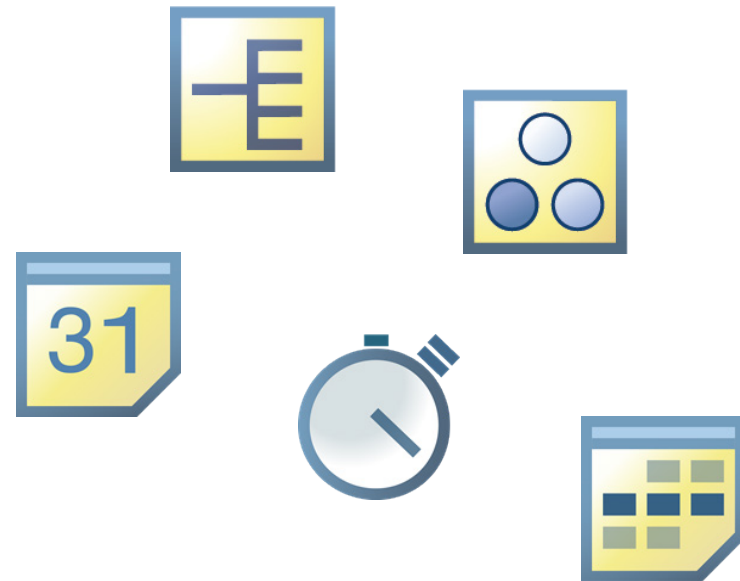
Organizing Data

MATLAB® Fundamentals for Aerospace Applications



Outline

- Modifying table properties
- Combining tables
- MATLAB® data types
- Dates and durations
- Categorical data



Chapter Learning Outcomes

The attendee will be able to:

- Query and modify table properties.
- Import and store data as an appropriate MATLAB data type.
- Access and manipulate variables representing dates and time durations.
- Display and plot dates and time durations with a specified format.

Course Example: Premier League Team Information

The spreadsheet `EPLteams.xlsx` contains information about each team in the English Premier League during the 2015-2016 season.

Some of the text information is “free text” (manager’s name), some represents dates (manager’s start date), and some represents named categories (manager’s nationality).

1	2	3	4	5
Team	Payroll_M	Manager	ManagerHireDate	ManagerNationality
'Arsenal'	192	'Arsène Wenger'	10/1/1996	'France'
'Aston Villa'	65.1000	'Eric Black'	3/29/2016	'Scotland'
'Bournemouth'	25	'Eddie Howe'	10/12/2012	'England'
'Chelsea'	215.6000	'Guus Hiddink'	12/19/2015	'Netherlands'
'Crystal Palace'	54.3000	'Alan Pardew'	1/2/2015	'England'
'Everton'	74.7000	'David Unsworth'	5/12/2016	'England'
'Leicester City'	48.2000	'Claudio Ranieri'	7/13/2015	'Italy'
'Liverpool'	152	'Jürgen Klopp'	10/8/2015	'Germany'
'Manchester City'	193.8000	'Manuel Pellegrini'	6/14/2013	'Chile'
'Manchester United'	203	'Louis van Gaal'	7/14/2014	'Netherlands'
'Newcastle United'	75.8000	'Rafael Benítez'	3/11/2016	'Spain'
'Norwich City'	37	'Alex Neil'	1/9/2015	'Scotland'
'Southampton'	59.5000	'Ronald Koeman'	6/16/2014	'Netherlands'
'Stoke City'	72.3000	'Mark Hughes'	5/30/2013	'Wales'
'Sunderland'	71	'Sam Allardyce'	10/9/2015	'England'
'Swansea City'	51	'Francesco Guidolin'	1/18/2016	'Italy'
'Tottenham Hotspur'	110.5000	'Mauricio Pochettino'	5/27/2014	'Argentina'

The goal of the chapter’s example is to combine this team information with the results of the 2015-2016 season and organize the data for analysis using native MATLAB data types.

Try

Open `EPLteams.xlsx` in Excel®.

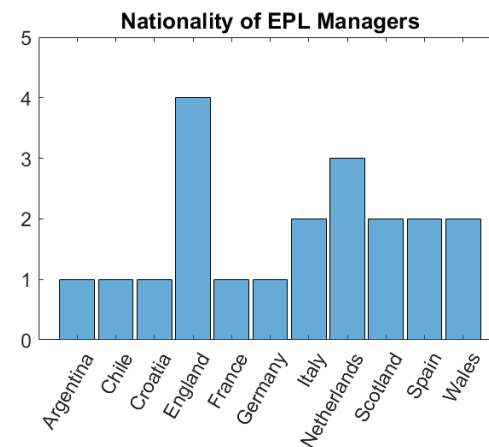
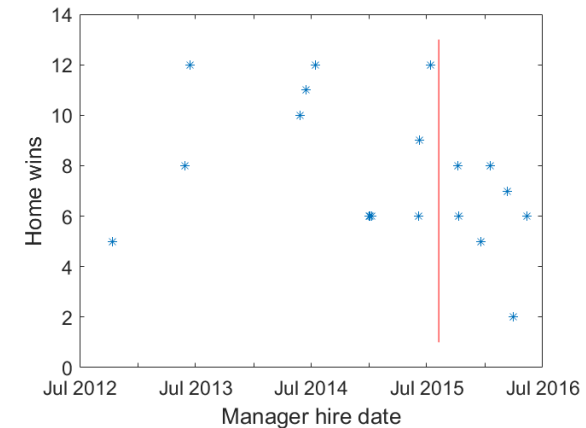


Table Properties

The set of names of the variables in a table is an editable property of the table. You can view all the *properties* (metadata) associated with a table by typing

```
teaminfo.Properties
```

This returns a *structure* of table properties. Data in structures is accessed using named referencing. You can therefore access individual properties using dot notation:

```
teaminfo.Properties.VariableDescriptions
```

These properties can also be modified using assignment:

```
teaminfo.Properties.Description = ...  
    'Information on teams in the 2015-2016 EPL.'
```

To change the variable names, use assignment and provide a cell array of variable name character vectors to change the current `VariableNames` property:

```
teaminfo.Properties.VariableNames = NewNames
```

Try

Import the spreadsheet `EPLteams.xlsx` as a table.

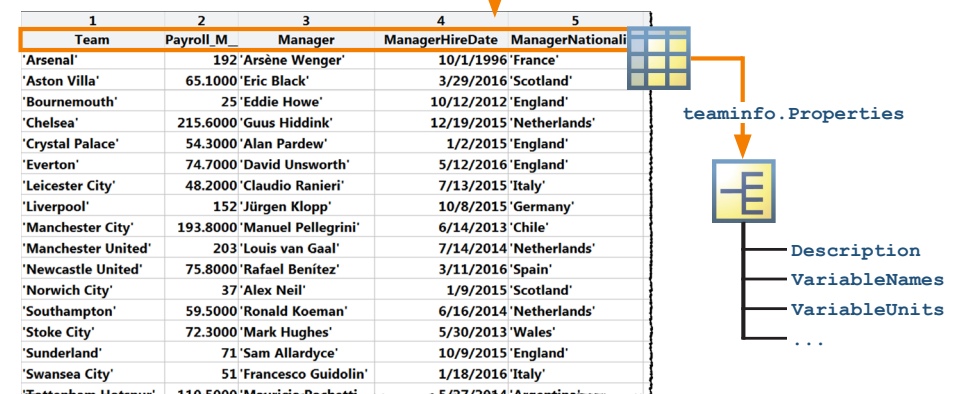
```
teaminfo = readtable('EPLteams.xlsx');
```

View the metadata associated with the table of English Premier League team information.

```
teaminfo.Properties
```

Modify the table description to provide information on the data contained in the table.

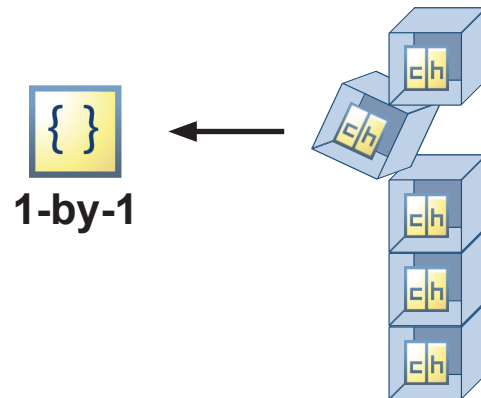
```
vars = teaminfo.Properties.VariableNames
```



Indexing into Cell Arrays

As with all other arrays in MATLAB, you can refer to a subset of a cell array using standard array indexing:

```
x = y(2)
```

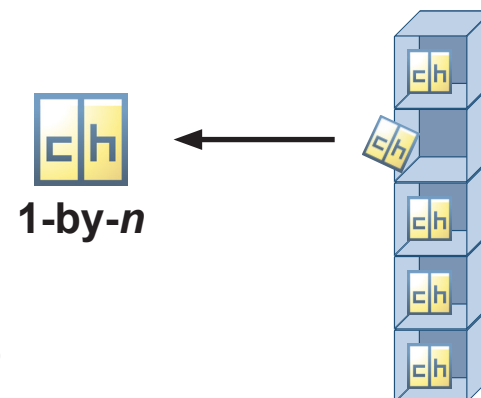


The result is another cell array. However, there are situations where a 1-by-1 cell array of text and a character vector are not interchangeable, such as when modifying the contents of a cell:

```
y(2) = 'world' ❌
```

In this situation, you need to refer to the contents of the cell. As with tables, you can do this by indexing with curly braces:

```
x = y{2}
```



```
y{2} = 'world' ✅
```

Try

Extract the managers' names from the table `teaminfo`. Extract the fourth name. Note the size and class of the output.

```
mgmt = teaminfo.Manager;  
m4 = mgmt(4)  
m4 = mgmt{4}
```

Modify the fourth element of the cell array.

```
mgmt(4) = 'Antonio Conte'
```

Why doesn't this work?

Try again, using cell indexing.

```
mgmt{4} = 'Antonio Conte'
```

Change the name of the payroll variable to remove extraneous characters.

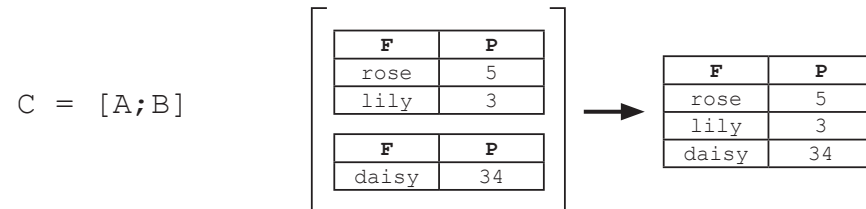
```
teaminfo.Properties.VariableNames{2} = ...  
    'Payroll';
```

Remember:
"Curly for content!"

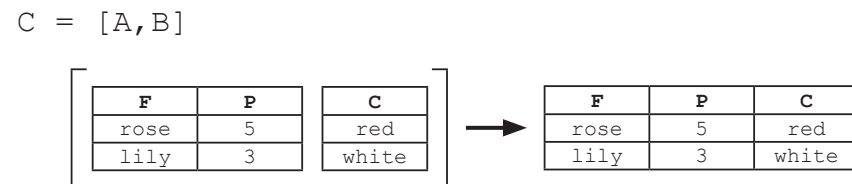


Combining Tables

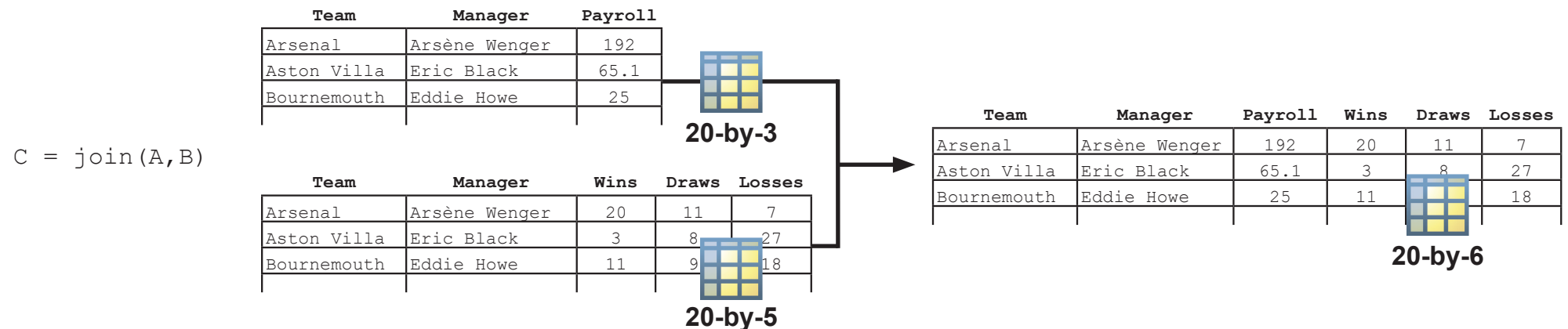
Tables containing different observations of the same variables can be combined using vertical concatenation.



Tables can also be concatenated horizontally. The data should be aligned before concatenating horizontally to ensure corresponding observations (rows) remain together.



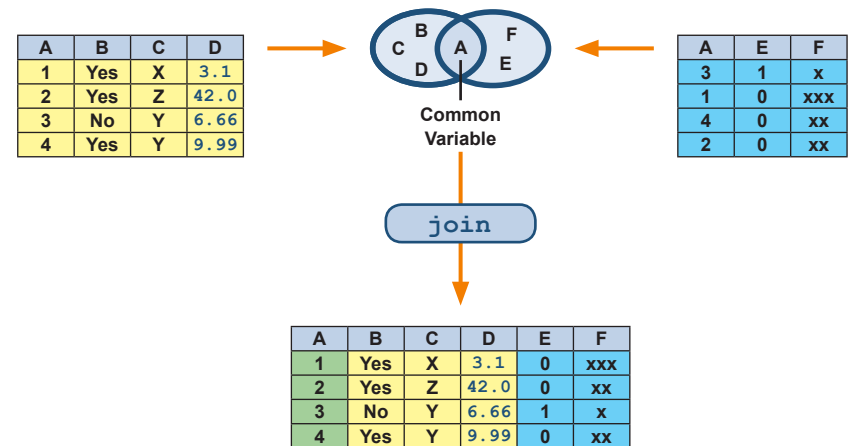
You can merge tables with different variables with the `join`, `innerjoin`, and `outerjoin` functions. MATLAB uses the variables that both tables have in common to match the rows.



Try

Load the file `EPLresults.mat`. Join the table containing season results with the table of team information.

```
load EPLresults
EPL = join(teaminfo, EPL, 'Keys', 'Team')
```



MATLAB® Data Types

Data is stored in the MATLAB environment in the form of variables. Every variable in MATLAB is an array of a particular data type. Each data type is intended to store data with a characteristic organization:

Array Type	Intended Contents
single double	Floating-point numbers with ~8 (single) or ~16 (double) decimal places of precision
intX uintX	Integer values of various ranges
char	Text
string	Textual data
datetime duration calendarDuration	Dates, times, durations between dates and times
logical	Boolean (true/false) values
categorical	Values from a finite set (often text labels)
table	Tabular arrays with named variables, which all have the same number of rows but may be of different types
function_handle	The definition of a function
cell	Heterogeneous data, indexed numerically
structure	Heterogeneous data, indexed by name

Double arrays provide the greatest support for performing calculations and visualizations. However, they can store only numeric data and do not allow for named indexing (referencing).

Statistical data is often in a tabular form, which is most conveniently stored in MATLAB as a table. The variables (columns) within a table can be of any MATLAB data type, as long as all observations of a given variable are the same type.

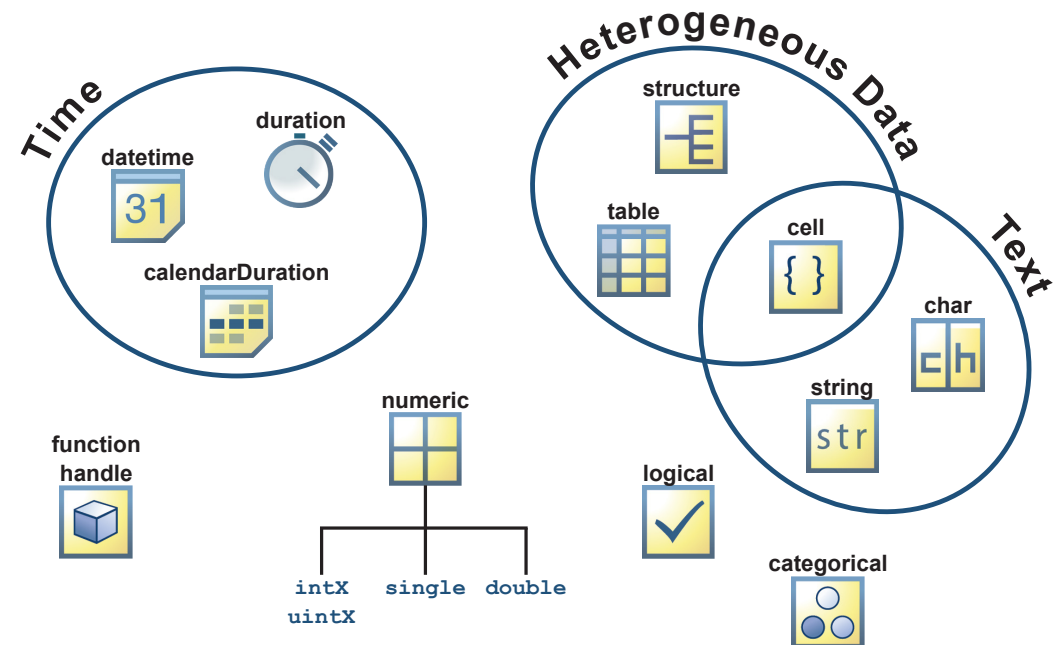
Logical arrays are useful for representing binary data (true/false or 1/0).

Categorical arrays are useful for representing a finite set of categories with text labels (such as “male”/“female” or “excellent”/“good”/“fair”/“poor”).

Use cell arrays to store text, such as names or labels. You cannot perform mathematical or graphical operations on cell arrays (even if they contain only numeric data). You can perform text manipulations on cell arrays of character vectors.




See the documentation for details on the various numeric types: **MATLAB → Language Fundamentals → Data Types → Numeric Types**. In particular, note that there are particular rules for how arithmetic is implemented when using nondouble variables. These rules are summarized on page A-15 in Appendix A (“MATLAB Reference”).

You can convert between data types: **MATLAB → Language Fundamentals → Data Types → Data Type Conversion**. Conversion functions are summarized on page A-17 in Appendix A (“MATLAB Reference”).



Representing Dates and Times

MATLAB has three data types for representing dates and times:

Type	Use
datetime 	For representing a specific instant in time, such as “January 7, 1610” or “17:10:00, October 21, 2005”.
duration 	For representing an elapsed time, such “7 hours, 14 minutes, 42.3 seconds”, without reference to a specific date. Fixed lengths are used for all time units (1 day = 24 hours, 1 year = 365.2425 days).
calendarDuration 	For representing an elapsed time in calendar units, such “3 years, 1 month, 4 days”, but without reference to a specific date. The value of a unit can change with context (e.g., a month can be 28, 29, 30, or 31 days).

You can import dates as `datetime` variables directly using the Import Tool and the `readtable` function. You can also create `datetime` variables by specifying the components:

```
t1 = datetime(2005,10,21)
t2 = datetime(2008,6,23,17,10,0)
```

Arithmetic operators and functions are defined in contexts that make sense for date variables. For example, elapsed time can be calculated by subtracting two `datetime` variables, or by using the `between` function.

```
dt = t2 - t1
dtC = between(t1,t2)
```

Try

Extract the manager hire date. Note the size and type of the result.

```
mgrHireDate = EPL.ManagerHireDate
```

Sort the table by manager hire date.

```
byMHD = sortrows(EPL,'ManagerHireDate');
byMHD(:,{'Team','Manager','ManagerHireDate'})
```

Create a `datetime` variable representing the season start date.

```
seasonStart = datetime(2015,8,8);
```

Calculate the tenure of each manager at the start of the season. Note the size and type of the result.

```
mgrTenure = seasonStart - mgrHireDate
```

Calculate the tenure in days. Note the size and type of the result.

```
mgrTenureD = days(mgrTenure)
```

Subtraction returns a `duration` variable, and the `between` function returns a `calendarDuration`.

By default, `duration` variables display in hours. To find the equivalent number of days, use the `days` function.

```
d = days(dt)
```

In this context, the `days` function accepts a `duration` variable and returns a numeric value. If instead you pass a numeric value to the `days` function, it returns a `duration`.

```
dt = days(d)
```

See the documentation for details on date and time data types: **MATLAB → Language Fundamentals → Data Types → Dates and Time.**

Displaying and Plotting Dates

All date variables have a `Format` property that controls how they are displayed. This property is set to a default format when the variable is created. You can see the current setting using dot notation:

```
t = datetime(2005,10,21)
t.Format
```

```
t =
    21-Oct-2005
```

```
ans =
    dd-MMM-uuuu
```

You can use the same notation with assignment to change the format:

```
t.Format = 'eee dd/MMM/yy'
```

```
t =
    Fri 21/Oct/05
```

See the documentation for all the available identifiers for specifying the date format:

```
doc datetime Properties
```

You can plot a time series directly with the `plot` function:

```
plot(Date,Y)
```

The dates are displayed on the axis tick marks using an automatically chosen format. If needed, use the optional `DatetimeTickFormat` or `DurationTickFormat` property to change the format:

```
plot(Date,Y,'DatetimeTickFormat','MM yy')
```

Try

Plot each team's number of home wins by their manager's hire date.

```
plot(EPL.ManagerHireDate,EPL.HomeWins,'*')
```

Adjust the x-axis limits to ignore the outlier.

```
t1 = datetime(2012,7,1);
t2 = datetime(2016,7,1);
xlim([t1,t2])
```

Add a vertical line at the season start date. Do managers hired before or after the season start date appear to have more home wins?

```
hold on
plot([seasonStart seasonStart],[1 13],'r')
hold off
```

In which months are managers commonly hired?

```
hireMonth = month(EPL.ManagerHireDate);
histogram(hireMonth)
```

Modify the x-axis labels.

```
t = datetime(2000,1:12,1);
mths = month(t,'shortname');
xticks(1:12)
xticklabels(mths)
```

The `x/y/zlim` functions accept date and time variables to control axes limits.

```
t1 = datetime(2005,10,21);
t2 = datetime(2008,6,23);
xlim([t1,t2])
```

Representing Discrete Categories

The `ManagerNationality` variable in the table of Premier League team data is recorded as a cell array of character vectors. However, the text represents *categorical* data – data that takes values from a finite set of categories. Use the `categorical` function to convert a numeric array or a cell array of character vectors to a categorical array:

```
c = categorical(z)
```

By default, the possible values and the names of the categories will be determined automatically from the data.

Categorical arrays allow you to perform logical comparisons with the standard mathematical relational operators, which is useful for logical indexing:

```
smallidx = y == 'small'
```

Because categorical arrays need to store only a limited number of possible values, they require less memory than either numeric or text arrays. Logical comparisons with relational operators are also more efficient than string comparisons.

It can be useful to rename or reorganize categories, or even add or remove categories. Use the `addcats` and `removecats` functions to add and remove categories, respectively. Use the `renamecats`, `mergocats`, and `reordercats` functions to rename, merge, and reorder categories, respectively:

```
c = {'r','b','r','g','g','b'};
cc = categorical(c);
cfull = renamecats(cc,{'r','b','g'},...
    {'red','blue','green'})
```

```
cfull =
    red    blue    red    green    green    blue
```

Try

Try to investigate the number of managers from England. Why doesn't this work?

```
nnz(EPL.ManagerNationality == 'England')
nat = EPL.ManagerNationality;
whos nat
```

Convert the countries to a categorical array.

```
EPL.ManagerNationality = ...
    categorical(EPL.ManagerNationality);
ctry = EPL.ManagerNationality;
whos ctry
```

Try again to find the number of managers from England. What set of countries are represented by the managers? How many managers are from each country?

```
nnz(ctry == 'England')
cats = categories(ctry)
summary(ctry)
```

How many managers are from the UK?

```
catsUK = {'England','Scotland','Wales'};
natUK = mergocats(ctry,catsUK,'UK');
nnz(natUK == 'UK')
```

Statistics and Machine Learning Toolbox™ also provides many functions that work with categorical data to perform statistical analysis on groups, such as `grpstats`, `gscatter`, `gplotmatrix`, and `boxplot`.

Summary

- Modifying table properties
- Combining tables
- MATLAB® data types
- Dates and durations
- Categorical data

Try

Open and run the script `EPLteamdata.mlx`.

Function	Use
<code>.Properties</code>	Access table metadata
<code>.</code>	Index into contents of a structure
<code>{}</code>	Index into contents of a cell array
<code>join</code> <code>innerjoin</code> <code>outerjoin</code>	Merge tables based on common variables
<code>datetime</code>	Create a variable representing points in time
<code>-</code> <code>between</code>	Calculate the duration between two datetime variables Calculate the calendar duration between two datetime variables
<code>years</code> <code>days</code> <code>hours</code> <code>minutes</code> <code>seconds</code>	Convert between a duration variable and its numeric equivalent
<code>calyears</code> <code>calquarters</code> <code>calmonths</code> <code>calweeks</code> <code>caldays</code>	Convert between a <code>calendarDuration</code> variable and its numeric equivalent
<code>categorical</code>	Create a categorical array
<code>categories</code>	List the categories of a categorical array
<code>mergecats</code>	Merge categories in a categorical array

Test Your Knowledge

Name: _____

1. If `t1` and `t2` are `datetime` variables representing March 13, 2000, and March 15, 2000, respectively, what is the result of the command
`dt = t1 - t2`?
 - A. A `datetime` variable representing March 11, 2000.
 - B. A scalar `duration` variable representing -48 hours.
 - C. A 2-element `duration` array with both elements representing -24 hours.
 - D. An error because `t2` is a later date than `t1`.
 - E. An error because `datetime` variables cannot be subtracted.

2. Which command renames the first variable in the table `t` from `foo` to `bar`?
 - A. `t.bar = t.foo`
 - B. `t.VariableNames(1) = 'bar'`
 - C. `t.Properties.VariableNames{1} = 'bar'`
 - D. `VariableNames(t,1) = 'bar'`

