LESSON 10: Vector logic

FOCUS QUESTION: How can I extract the rows and columns of an array based on data characteristics?

This lesson demonstrates how to use relational and logical operators to extract data for analysis.

In this lesson you will:

- Use relational operators (>, <, >=, <=, ==, ~=) to pick out pieces of the data.
- Use logical operators (&, |, ~) to combine tests.
- Extract two groups based on a condition.
- Work with realistic data.

Contents

DATA FOR THIS LESSON

File	Description
	▪ The data set contains sleep diary data for a cohort in MATLAB variables. ▪ The arrays have a column for each person. ▪ The vectors have an element for each person. ▪ The values in column *n* correspond to the same person as the value in position *n* of each vector. ▪ The file contains the following variables:

diaries.mat	- bedTimes - array of bed times in decimal-date format. - dayCaffeine - array of daytime caffeine indicators. - gender - vector of male/female gender designators. - nightCaffeine - array of evening caffeine indicators. - section - vector of section indicators. The possible section numbers are 0, 1, 2, and 3. Section 0 contains only a single instructor. The remaining values correspond to course section numbers. - toSleepMinutes - an array of number of minutes to fall asleep. - useAlarm - array of alarm use indicators. - wakeTimes - array of wakeup times in decimal-date format. - The data was originally gathered by students taking CS 1173 in the fall 2009 semester and anonymized and randomized to be unidentifiable. - The first column of each array represents the instructor's values, the rest of the columns represent individual students. - Diaries were recorded for 21 days (from September 23, 2009 to October 13, 2009).

SETUP FOR LESSON 10

- Set the Current Directory to `V:\VectorLogic`. (You will need to make a new directory for `VectorLogic`.)
- Download `diaries.mat` from the resources link on Blackboard.
- Create a new script called `VectorLogicLesson.m` in your `VectorLogic` directory. Enter each of the examples in a new cell in this script.

EXAMPLE 1: Load the consolidated sleep diary data (load)

Create a new cell in which you type and execute:

```
load diaries.mat;  % Load the sleep diaries
```

You should see 8 variables in the Workspace Browser:

- `bedTimes` - an array with the bedtimes of individual students in the columns
- `dayCaffeine` - a logical array with columns indicating daytime caffeine use for individual students
- `gender` - a vector of strings containing 'male' or 'female' designations for each student
- `nightCaffeine` - a logical array with columns indicating caffeine use after 6 pm for individual students
- `section` - vector containing sections numbers of the individual studens
- `toSleepMinutes` - an array with the number of minutes to fall asleep each night for the individual students
- `useAlarm` - a logical array with indications of alarm use for individual students in the columns.
- `wakeTimes` - an array with the wake times of individual students in the columns.

NOTE: All of the times are represented as doubles, which are real numbers. The integer part of the time gives the number of days since a reference day (in our case Jan 1, 0 AD) and the fractional part gives time on the current day represented as a fraction of 24 hours. You can use the `datestr` function to find out what date and time this double corresponds to (e.g., `datestr(x)` gives a string with the readable form of the date and time corresponding to the value `x`).

EXERCISE 1: Diagram some arrays
Draw a diagram of the `bedTimes` and `gender` arrays. Label the rows and columns of each array. Explain the correspondence between rows and columns in the two arrays.

EXAMPLE 2: Calculate the number of students in section 2 (==)

Create a new cell in which you type and execute:

```
sect2 = (section == 2);      % sect2 has 1's corresponding to section 2 students
totalSect2 = sum(sect2);     % Add up the true's (1's) to find number of students
fprintf('%g students in section 2\n', totalSect2);
```

You should see 2 variables in the Workspace Browser:

- `sect2` - a vector with 1's corresponding to the students in section 2
- `totalSect2` - the total number of students in section 2 (a single value)

You should also see the following output in the Command Window:

```
46 students in section 2
```

EXERCISE 2: Define a logical vector identifying the students in section 1.

EXAMPLE 3: Find the average number of minutes students in section 2 took to fall asleep (indexing)

Create a new cell in which you type and execute:

```
minutesSect2 = toSleepMinutes(:, sect2);  % Pick columns of section 2
averMinutes2 = mean(minutesSect2(:));     % Find overall average of section 2
fprintf('Average minutes to sleep for section 2 students = %g\n', ...
    averMinutes2);
```

You should see 2 variables in your Workspace Browser:

- `minutesSect2` - array whose columns are minutes to fall asleep for section 2 students
- `averMinutes2` - the average number of minutes to fall asleep for students in section 2.

You should also see the following output in the Command Window:

```
Average minutes to sleep for section 2 students = 16.9482
```

EXERCISE 3: Define a variable for alarm use in section 2.

EXERCISE 4: Output the percentage of times that students in section 2 used an alarm.
(Hint: the `mean` function might be useful here.)

EXAMPLE 4: Calculate the number of men in the cohort (strcmp)

Create a new cell in which you type and execute:

```
men = strcmp(gender, 'male');      % 1's in positions corresponding to males
totalMen = sum(men);           % Add to find number of men
fprintf('%g men in the cohort\n', totalMen);
```

You should see 2 variables in your Workspace Browser:

- `men` - a logical vector with 1's corresponding to male students
- `totalMen` - variable holding number of male students in cohort

You should also see the following output in the Command Window:

```
70 men in the cohort
```

EXERCISE 5: Define a variable identifying the women in the study.
The variable, `women`, should hold a logical vector that is the same size as `gender` and has 1's in positions in which `gender` corresponds to female students.

EXAMPLE 5: Calculate the % of men in the cohort

Create a new cell in which you type and execute:

```
totalStudents = length(gender);  % gender has an entry for each student
percentMen = 100.*totalMen./totalStudents;
fprintf('%g%% of the students in the cohort are men\n', percentMen);
```

You should see 2 variables in your Workspace Browser:

- `totalStudents` - the total number of students in the cohort
- `percentMen` - the percentage of students in the cohort that are male

You should also see the following output in the Command Window:

```
48.6111% of the students in the cohort are men
```

EXERCISE 6: Find the fraction of the cohort that is female.

EXAMPLE 6: Calculate the number of men in section 2 (&)

Create a new cell in which you type and execute:

```
menSect2 = men & sect2;            % 1's in positions of men in section 2
totalMen2 = sum(menSect2);         % Add up the trues (1's)
fprintf('%g men in section 2\n', totalMen2);
```

You should see 2 variables in your Workspace Browser:

- `menSect2` - logical vector with 1's in positions of men in section 2
- `totalmen3` - total number of men in section 2

You should also see the following output in the Command Window:

```
25 men in section 2
```

EXAMPLE 7: Calculate total students in sections 2 or 3 (|)

Create a new cell in which you type and execute:

```
sect2or3 = (section == 2) | (section == 3);  % 1's for section 2 or 3
fprintf('%g students in sections 2 or 3\n', sum(sect2or3));
```

You should see 2 variables in your Workspace Browser:

- `sect2or3` - logical vector with ones corresponding to students in either section 2 or section 3

You should also see the following output in the Command Window:

```
98 students in sections 2 or 3
```

EXAMPLE 8: Calculate the number of wake-ups that were 8:30 am or later (>=)

Create a new cell in which you type and execute:

```
wakeupHours = (wakeTimes - floor(wakeTimes))*24; % Fractional part
wakeGE830 = (wakeupHours >= 8.5);                % Which are >= 8:30 am?
totalWakeGE830 = sum(wakeGE830(:));              % Wake-ups after 8:30 am.
fprintf('%g wake-ups are 8:30 am or later\n', totalWakeGE830);
```

You should see 3 variables in your Workspace Browser:

- `wakeupHours` - an array with the wake-up time of day for the cohort
- `wakeGE830`- logical array with 1's corresponding to wake-ups 8:30 or later
- `totalWakeGE830` - total times members of cohort got up at 8:30 am or later

You should also see the following output in the Command Window:

```
1484 wake-ups are 8:30 am or later
```

EXAMPLE 9: Calculate % of wake-ups between 7:30 am and 9:45 am (&)

Create a new cell in which you type and execute:

```
wakeBetween = (7.5 <= wakeupHours) & (wakeupHours <= 9.75);  % & means both
betweenPercent = 100*mean(wakeBetween(:));
fprintf('%g%% of the wake-ups are between 7:30 am and 9:45 am\n', betweenPercent);
```

You should see 2 variables in your Workspace Browser"

- `wakeBetween` - logical array with ones corresponding to wakeups in [7:30, 9:45]
- `betweenPercent` - percentage of wakeups in [7:30, 9:45]

You should also see the following output in the Command Window:

```
34.7222% of the wake-ups are between 7:30 am and 9:45 am
```

EXERCISE 7: Find the percentage of students who used caffeine at least once during the study.

SUMMARY OF SYNTAX

MATLAB syntax	Description
`Y = floor(X)`	Return an array `Y` that is the same size as the array `X`. Each element of `Y` is the largest integer that is less than or equal to the corresponding element of `X`. For example, floor(3.5) is 3, while floor(-3.5) is -4.
`n = length(A)`	Return the number of elements in the longest dimension of `A`.
Logical element-wise operators: **&, |, ~**	Combine arrays based on the logical element-wise operators AND, OR, and NOT. The logical value `true` corresponds to the value 1, and the logical value `false` corresponds to the value 0. If you apply a logical element-wise operator to an array of numerical values rather than logical values, MATLAB first creates a logical array with `true` corresponding to the nonzero elements and `false` corresponding to the zero elements. ▪ **and**: `A & B` is true only in positions where both `A` and `B` are true. ▪ **or**: `A | B` is true in the positions where `A` or `B` or both are true. ▪ **not**: ~A is true only in the positions where `A` is false.
Relational element-wise operators:	Combine arrays based on a comparison of their values. The result of applying a relational operator is a logical array of the same size as the input operands. The result has true in the positions where the relationship is true and false elsewhere. ▪ **greater than**: `A > B` ▪ **greater than or equal**: `A >= B` ▪ **less than**: `A < B`

`>, <, >=, <=, ==, ~=`	▪ **less than or equal**: `A <= B` ▪ **equal**: `A == B` ▪ **not equal**: `A ~= B`
`[rows, col] = size(A)`	Return the number of elements in the first two dimensions of `A`.
`mySize = size(A, dim)`	Return the number of elements along the specified dimension of `A`.
`strcmp(s1, s2)`	Return true if the string `s1` is equal to the string `s2` and false otherwise.
`strcmp(A, s1)`	Return a logical array of the same size as the cell array of strings `A`. The result has true in the positions corresponding to the entries of `A` that match `s1` and false elsewhere.

This lesson was written by Kay A. Robbins of the University of Texas at San Antonio and last modified March 23, 2015. Please contact kay.robbins@utsa.edu with comments or suggestions. The image is a photograph of a nocturnal instrument photographed by Michael Daly on 8/22/2009. The image is available on Wikipedia as http://en.wikipedia.org/wiki/Nocturnal_%28instrument%29.

Published with MATLAB® 8.3