MATLAB® @ Work

Working with Nan

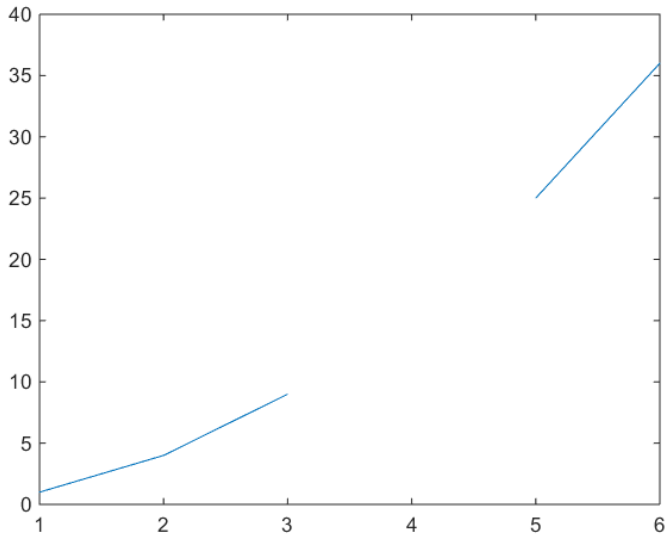
Richard Johnson

NaN is used several ways in MATLAB. Working with NaN requires some attention to detail.

Missing data

NaN is the standard placeholder for missing data, and it is sometimes used as a replacement for "bad" data. A nice feature of nan is that MATLAB does not plot it, so that it can appear as a gap in a curve.

```
d = [1 2 3 4 5 6];
h = [1 4 9 nan 25 36];
plot(d,h)
```



Test for and locate nan values using `isnan`

```
isnan(h)
```

```
ans =
     0     0     0     1     0     0
```

Testing for equality to nan does not work.

```
h==nan
```

```
ans =
     0     0     0     0     0     0
```

You might consider using the `isfinite` function, which will catch both NaN and Inf.

```
isfinite(h)
```

```
ans =
     1     1     1     0     1     1
```

It can be helpful to use a consolidator function.

```
any(isnan(h))
```

```
ans =
     1
```

You can replace NaN values with code like

```
index = find(isnan(h));
cleanH = h;
cleanH(index) = mean([h(index-1) h(index+1)])
```

```
cleanH =
     1     4     9    17    25    36
```

Of course code that deals with end effects and multiple NaN values will be more complicated.

NaN is limited to double and single representations.

```
uint16([0 1 nan])
```

```
ans =
      0      1      0
```

For integer types, you will need to develop your own value to represent missing data. Zero is usually not a good choice.

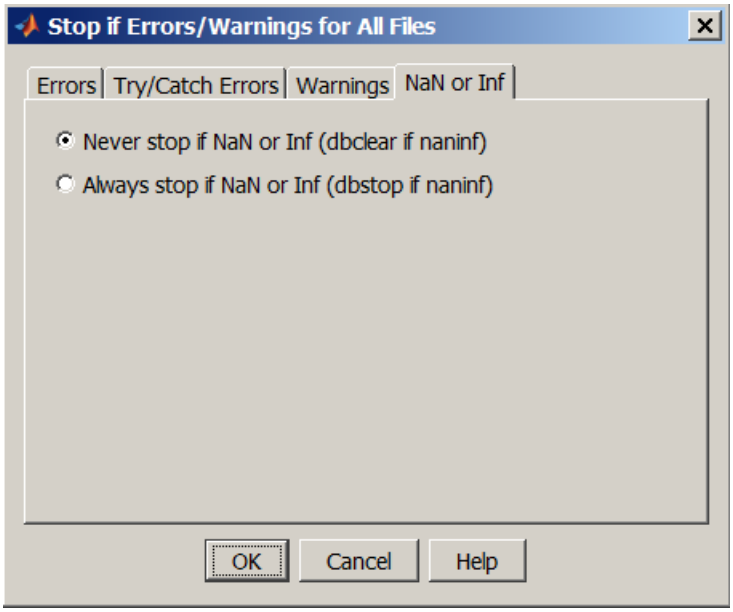
Result of some computations

Nan is the defined result for an indeterminate operation. When this occurs, MATLAB does not issue an error or warning. NaN can result from operations on Inf, but it most commonly occurs as a result of an operation on an array that contains a NaN.

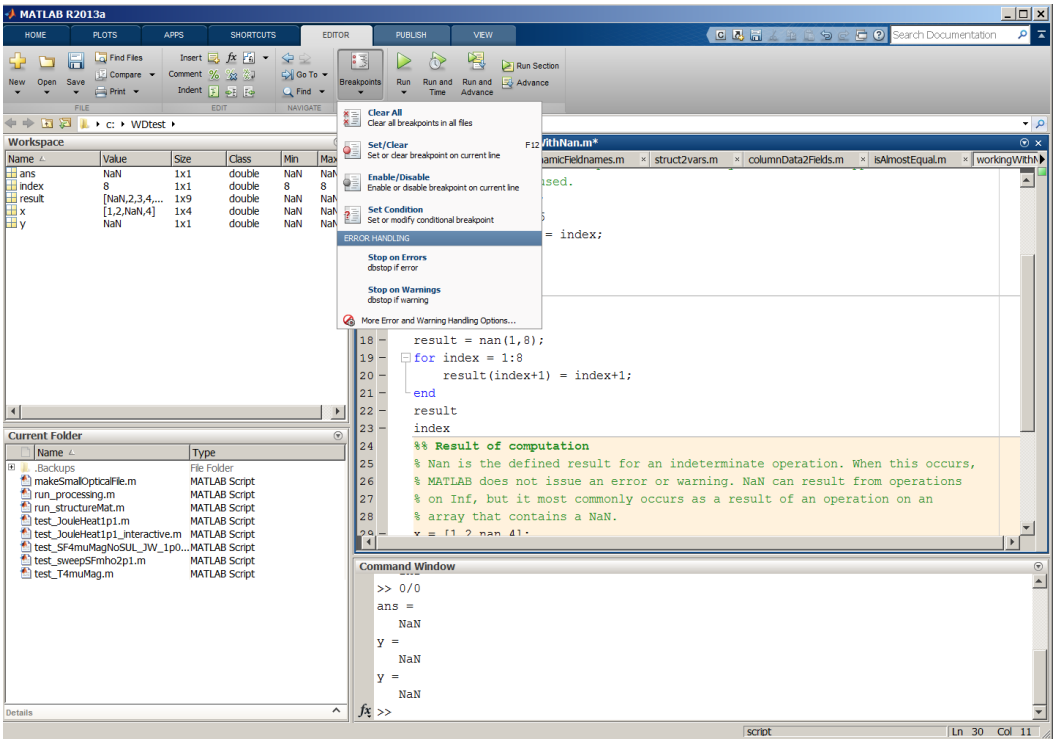
```
x = [1 2 nan 4];
y = sum(x)
```

```
y =
   NaN
```

You can have MATLAB stop in the Debugger in these cases. Select the option on the Stop if Errors window.



This feature is accessed through the More Error and Handling Options on the Breakpoints pull down menu.



Statistics and NaN

Some of the statistical functions ignore NaN.

```
y = max([0 1 nan])
```

```
y =
     1
```

```
y = min([0 1 nan])
```

```
y =
     0
```

Many statistical and aggregator functions return NaN when applied to values that contain NaN.

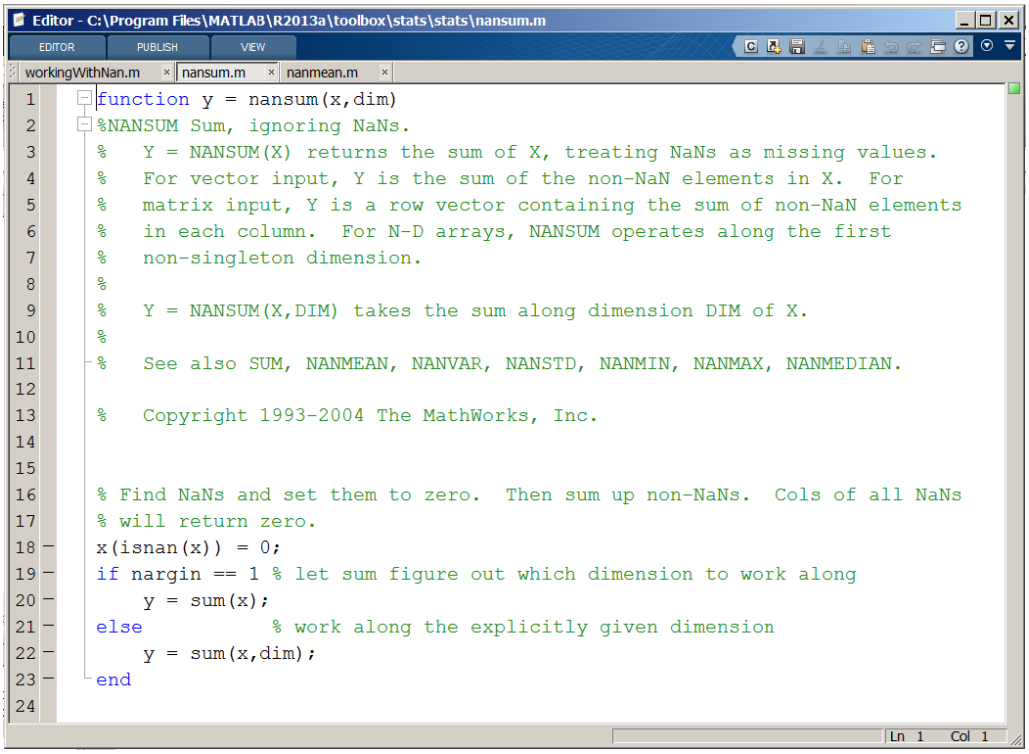
```
y = sum([0 1 nan])
```

```
y =
   NaN
```

```
y = median([0 1 nan])
```

```
y =
   NaN
```

Several of these functions have alternatives in the Statistical Toolbox that ignore NaN values. They have names like `nansum`, `nanmedian`, etc. You can also write similar versions.



Pre-allocation

Using nan to pre-allocate an array can be better than the traditional use of zeros because it can be easier to spot incomplete filling or incorrect elements.

This can happen with a while or for loop that does not have enough iterations to fill the pre-allocated array. It can also happen if the iterator is misused.

```
result = nan(1,8);
for index = 1:2:16
    result(index) = index;
end
result
index
```

```
result =
  Columns 1 through 13
     1   NaN     3   NaN     5   NaN     7   NaN     9     0    11     0    13
  Columns 14 through 15
     0    15
```

```
index =
    15
```

or

```
result = nan(1,8);
for index = 1:8
    result(index+1) = index+1;
end
result
index
```

```
result =
   NaN     2     3     4     5     6     7     8     9
index =
     8
```

Issues

nan does not equal nan

```
nan==nan
```

```
ans =
     0
```

So arrays that might appear to be the same or equivalent are not formally equal

```
isequal([1 2 nan], [1 2 nan])
```

```
ans =
     0
```

You can get around this behavior by using `isequaln`

```
isequaln([1 2 nan], [1 2 nan])
```

```
ans =
     1
```

The older function `isequalwithequalnans` is available but not recommended.

```
isequalwithequalnans([1 2 nan], [1 2 nan])
```

```
ans =
     1
```

nan is neither true

```
nan==true
```

```
ans =
     0
```

nor false.

```
nan==false
```

```
ans =
     0
```

So you can't convert an array to logical if it contains a NaN.

```
logical([1 0 nan])
Error using logical
NaN's cannot be converted to logicals.
```

Patterns for dealing with NaN

The way that you process data that includes NaN values depends on what software you have available, and of course what you want to do.

If you already have a function that does what you want then

```
y = eff(x);
```

An example is mean, if you want to have nan impact the result. Or perhaps

```
y = nanEff(x);
```

if you want to ignore the NaN values. The function nanmean is an example. Of course you can also write a version like this yourself.

Another way to ignore NaN values is something like

```
y = x;
y(~isnan(x)) = eff(x(~isnan(x)));
```

or

```
y = x;
y(isnan(x)) = nanEff(x(isnan(x)));
```

Exercises

Replace nan with an neighbor value.

Use the values

```
t = linspace(0, 2*pi, 5);
z = sin(t);
z(3) = nan;
```

Plot the values. Replace the NaN value with the previous value (hold). Add a curve using this point to the plot. Highlight the replacement point. Write the code to be general, not just for the third point.

Replace nan with an average value.

Plot the values. Replace the NaN value with the average of the previous and following values. Add a curve using this point to the plot. Highlight the replacement point. Write the code to be general, not just for the third point.

Replace nan with an interpolated value.

Plot the values. Replace the NaN value with an interpolated value based on other values. Add a curve using this point to the plot. Highlight the replacement point. Write the code to be general, not just for the third point.

Write a function `nanMeanOf`.

Write a function that works like mean but ignores nan values. Design it to work with arrays of one or two dimensions.