

Preprocessing Time Series Data with MATLAB

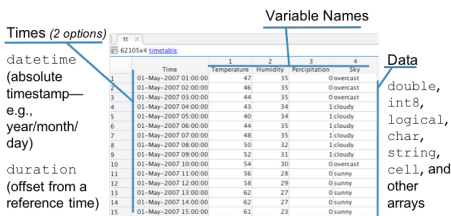
This reference shows common use cases, but is by no means comprehensive.

The **>>** icon provides links to relevant sections of the MATLAB® documentation to learn more.

Timetable

MATLAB datatype designed to organize and work with time series data.

Components of a Timetable



Times (2 options)
datetime (absolute timestamp—e.g., year/month/day)
duration (offset from a reference time)

Variable Names
1 2 3 4
Temperature Humidity Precipitation Sky

Data
double, int8, logical, char, string, cell, and other arrays

Create Timetables >>

```
tt = timetable(times, var1, var2, ... ,varN); >>
```

(All variables must have the same number of rows.)

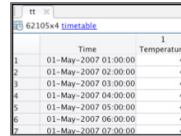
```
tt = table2timetable(t);
```

(The first datetime or duration variable in “t” becomes the row times.)

Timetable Manipulation

Access Data These return the same array:

```
tt.Temperature  
tt(:, 'Temperature')  
tt(:, 1) >>
```



Add a New Variable

```
tt.newVar = zeros(height(tt), 1); >>
```

Change Variable Names

```
tt.properties.VariableNames =  
newNames; >>
```

(Names must be valid MATLAB identifiers)

Tip: Use matlab.lang.makevalidname to create valid names from potentially invalid names.

Resample Data Using Retime

```
tt = retime(tt, newtimes, method); >>
```

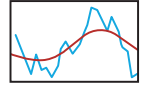
method is used to fill gaps after retimeing, and has the same options as synchronize (see “Merge Timetables”).

Data Cleaning

Smooth Data >>

```
B = smoothdata(A, method);
```

Smooth noisy data with methods:

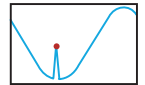


'movmean', 'movmedian', 'gaussian',
'lowess', 'loess', 'rloess',
'rloess', 'sgolay'

Detect Outliers >>

```
TF = isoutlier(A, method);
```

Identify outliers with methods:

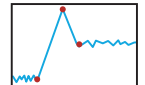


'median', 'mean', 'quartiles',
'grubbs', 'gesd'

Detect Change Points >>

```
TF = ischange(A, method);
```

Find abrupt changes with methods:



'mean', 'variance', 'linear'

Merge Timetables

Synchronize multiple timetables to a common time vector.

```
tt = synchronize(tt1, tt2, ..., ttN); >>
```

Synchronizing often results in missing data points (times at which a variable was not measured).

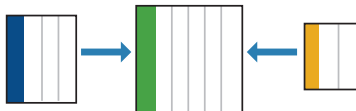
synchronize supports several methods for adjusting data to fill in gaps: >>

Fill: 'fillwithmissing', 'fillwithconstant'

Interpolation: 'linear', 'spline', 'pchip'

Nearest Neighbor: 'previous',
'next', 'nearest'

Aggregation: 'mean', 'min', 'max', @func, ...



Missing Data

Find Missing Values

```
TF = ismissing(tt); >>
```

Fill Missing Values

```
tt = fillmissing(tt, method); >>
```

Replace missing values with values calculated from nearby points with methods:

'previous', 'next', 'nearest',
'linear', 'spline', 'pchip'

Remove Rows Containing Missing Values

```
tt = rmmissing(tt);
```

Time	1 Temperature	2 Humidity
01-May-2007 01:00:00	47	35
01-May-2007 02:00:00	46	NaN
01-May-2007 03:00:00	NaN	35
01-May-2007 04:00:00	NaN	34
01-May-2007 05:00:00	40	34
01-May-2007 06:00:00	44	35
01-May-2007 07:00:00	48	35

Big Data

Tall arrays extend MATLAB functions to work on data too big to load into memory.

Create a “tall” timetable:

```
% Create a datastore that points to  
% the data
```

```
ds = datastore('*.csv');
```

```
% Create a tall table from the  
% datastore  
t = tall(ds); >>
```

```
% Convert to a timetable
```

```
tt = table2timetable(t); >>
```

