Overview

- Four major facets of Toolbox Q/C framework
  - Data model for storing Q/C rules, qualifiers, Q/C metadata
  - Tools for performing Q/C (assigning qualifier flags)
  - Tools for managing qualifier flag assignments
  - Tools for visualizing and managing qualified values

- Components work together seamlessly
  - Q/C framework integrated with data processing, analysis
  - Supports Q/C throughout the entire data lifecycle

- Q/C strategy important
  - Q/C is an iterative process
  - Operations done in wrong order can nullify flags, waste effort
  - Information can also be lost
Data Model

- GCE Data Structure designed to store data, Q/C info, metadata together
- Dedicated Q/C criteria field
- Paired value/flag arrays
- Q/C fields in doc metadata
  - Data/Codes (flag definitions)
  - Data/Anomalies (Q/C summary)
  - Data/ValueCodes (definitions for instantiated flag columns)
  - Processing history (all Q/C operations)
Data Model – Q/C Principles

- Data and structure validation happens before Q/C analysis
  - No need to add Q/C checks for valid data type, ragged tables, text in numeric fields, ...

- Basic Q/C operations are documented automatically
  - Edits to rules, evaluation of rules, manual/visual Q/C, flagged value removal, ...

- No qualifier vocabularies or semantics are assumed
  - Qualifiers are defined in the data set metadata (or interactively)
  - Just as valid to assign qualifiers for “good” values as “bad” (e.g. “reviewed”)
  - Subtle bias for “bad” interpretation in some tools, but not hard and fast

- Multiple qualifiers can be assigned to a data value
  - Multiple Q/C criteria matched
  - Multiple qualifiers assigned/copied manually
  - Major exception is graphical assignment (1 flag set or all flags cleared)
Assigning Qualifiers

- Qualifiers can be assigned automatically
  - Assigned based on criteria expressions for each column ("rules")
  - Assigned by specialized flag functions in rules or run in workflows
  - Assigned by tools when values revised, calculated (e.g. gap-filling)

- Qualifiers can be assigned manually/interactively
  - Typed into a data grid or added using workflow functions
  - Graphically assign with the mouse on data plots

- Qualifiers can be copied/imported
  - Copy composite flags from one or more columns to other column(s)
  - Convert text columns to flags to import pre-assigned or 3rd party flags
Automatic Assignment (Rules)

- **Basic syntax:** `[logical expression]=’[flag code]’`

- **Logical Expression =**
  - MATLAB expression that evaluates each data value against some criteria and returns true/false (true mean matches criteria)

- **Flag Code =**
  - Alphanumeric character to assign when expression true (I, q, 9, *)
  - Flag Codes defined in the dataset metadata (I = invalid value, Q = questionable ...)

- **Examples**
  - Conditional statements: `x<0=’Q’`
  - Mathematical/statistical comparisons: `x<mean(x)-2*std(x)=’Q’`
  - Functions for set-based comparisons: `flag_notinlist(x,’A,B,C’)=’I’`
Automatic Assignment (Rules)

Rule Characteristics

- Column values are referenced in rules using “x” or “col_[name]”
- Rules are evaluated for every value in the column independently
- Rules can reference corresponding values in other columns
- Multiple expressions can be combined using Boolean operators (&/|) and include parentheses to create complex criteria
- Can define any number of rules per column (semi-colon separator)
  - If multiple rules are matched, multiple flags are assigned (e.g. IQQ)
- Can use mathematical statements, MATLAB (or toolbox) functions to create derived arrays for comparison with column values
  - mean(), max(), std(), sum()
- Can call functions to retrieve reference data, run algorithms, etc (open-ended)
Automatic Assignment (Rules)

- Rules can be created, edited, deleted, re-ordered using GUI forms
Automatic Assignment (Rules)

- Syntax help displayed for custom flag functions to guide use.

Example of flag function:
```
flag_valuechange(vals, lowlimit, highlimit, frametime, iterations)
```
- Parameters:
  - `vals`: array of values
  - `lowlimit`: low limit change criteria (flag value < mean - lowlimit) (default = 2% value range)
  - `highlimit`: high limit change criteria (flag value > mean + highlimit) (default = 2% value range)
  - `frametime`: number of preceding values to use for mean calculation (default = 1) (more than 1) each iteration removes the first flagged value in each respectively flagged group and re-analyzes the remaining values.
  - `iterations`: number of flag-check iterations to perform to minimize inappropriate flagging of in-bounds values following out-of-bounds values (minimum = 1, max = 10) each iteration removes the first flagged value in each respectively flagged group and re-analyzes the remaining values.
- Returns an index of values above or below the mean of preceding values by the specified limits.

Help for the 'flag_valuechange' function:
```
Syntax: iflag = flag_valuechange(vals, lowlimit, highlimit, frametime, iterations)
```
- Inputs:
  - `vals`: array of values
  - `lowlimit`: low limit change criteria (flag value < mean - lowlimit) (default = 2% value range)
  - `highlimit`: high limit change criteria (flag value > mean + highlimit) (default = 2% value range)
  - `frametime`: number of preceding values to use for mean calculation (default = 1) (more than 1) each iteration removes the first flagged value in each respectively flagged group and re-analyzes the remaining values.
  - `iterations`: number of flag-check iterations to perform to minimize inappropriate flagging of in-bounds values following out-of-bounds values (minimum = 1, max = 10) each iteration removes the first flagged value in each respectively flagged group and re-analyzes the remaining values.
- Returns:
  - `iflag`: logical index of values outside of the range: mean - lowlimit < value < mean + highlimit.
## Automatic Assignment (Rules)

### Basic Examples (MATLAB conditionals, built-in functions)

<table>
<thead>
<tr>
<th>Q/C Goal</th>
<th>Rule Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limit/range check</td>
<td>simple conditionals</td>
<td><code>x&lt;0='I'; x&gt;37='Q'</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>col_Salinity&lt;0='I'; col_Salinity&gt;37='Q'</code></td>
</tr>
<tr>
<td>Valid value check</td>
<td>null check function</td>
<td><code>isnull(x)='M'</code> (numeric or string column)`</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>isnan(x)='M'</code> (numeric only)`</td>
</tr>
<tr>
<td>Sanity/consistency check</td>
<td>algebraic equations</td>
<td><code>col_WetWeight&lt;=col_DryWeight='Q'</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>(col_SpartinaPct+col_JuncusPct+col_BorrichiaPct)&gt;100='I'</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>(col_NO2+col_NO3)&gt;col_NOX='I'</code></td>
</tr>
<tr>
<td>Outlier detection</td>
<td>statistical tests</td>
<td><code>x&gt;mean(x)+3*std(x)='Q'</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>x&gt;mean(no_nan(x))+3*std(no_nan(x)) = 'Q' (NaN-safe)</code></td>
</tr>
<tr>
<td>Condition check</td>
<td>multi-column rules</td>
<td><code>col_Depth&lt;=0.2='I'; col_BatteryVolts&lt;=9='Q'</code> (in Salinity)</td>
</tr>
</tbody>
</table>

LTER All Scientists Meeting - Sep2015
## More Complex Examples (custom flag functions)

<table>
<thead>
<tr>
<th>Q/C Goal</th>
<th>Rule Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code check</td>
<td>set-based function</td>
<td>flag_notinlist(col_Site,'GCE1,GCE2,GCE3,GCE4')='I'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>flag_notinlist(col_Site, 'plant_list.mat</td>
</tr>
<tr>
<td></td>
<td></td>
<td>flag_notinarray(col_Plot,[1 5 10 15 20])='I'</td>
</tr>
<tr>
<td>Pattern check</td>
<td>moving window function</td>
<td>flag_valuechange(col_AirTemp,5,5,3)=‘Q’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>flag_nsigma(col_Humidity,3,3,10)=‘Q’</td>
</tr>
<tr>
<td>Location check</td>
<td>geographic lookup fnc</td>
<td>flag_locationcoords(x,'Longitude','Latitude',0.5)=‘Q’</td>
</tr>
<tr>
<td>Stuck/fouled sensor</td>
<td>inverse change function</td>
<td>flag_novaluechange(col_Salinity,0.2,0.2,3)=‘Q’</td>
</tr>
<tr>
<td>Derived property</td>
<td>custom function</td>
<td>flag_o2saturation(col_O2Conc,col_WaterTemp,col_Salinity,100,30,'mg/L')='Q'</td>
</tr>
<tr>
<td>Conditional checks</td>
<td>compound rules</td>
<td>flag_valuechange(col_AirTemp,5,5,3)&amp;col_Prec=-0='Q'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>col_PAR&gt;0&amp;flag_timeofday(col_Date,{'00:00','06:00','20:00','23:59:59'})='Q'</td>
</tr>
</tbody>
</table>
Automatic Assignment (Rules)

- Rules are evaluated automatically during processing
  - When data are imported and metadata template is applied
  - When data values are revised or columns added
  - When Q/C criteria edited or new template applied

- When rules are evaluated, any existing flags are overwritten

- Rules with syntax errors will be skipped, warning issued

- Rules can be “locked” & “unlocked” to control automatic assignment
  - Adding the token “manual” to the Q/C criteria locks flags and prevents recalculation
  - Rules are retained, but dataflag() will skip evaluation and never clear flags
  - Removing the token “manual” unlocks flags and triggers recalculation

- Rules are converted to column descriptions when flags are converted to text columns for export
Automatic Assignment (Rules)

- Missing (NaN) values complicate rule evaluation
  - NaN can never equal anything, even another NaN (like RDBMS Null)
  - Aggregate statistics (mean, min, max, ...) return NaN if any values are missing
  - Compound Q/C criteria will always return false if any value is NaN, e.g. \texttt{col\_Salinity>0&col\_Depth>1&col\_Temp>10&col\_Temp<35} = ‘Q’ will never assign a flag for a row where Salinity, Depth or Temp value = NaN
  - No warning will be produced if NaN prevents flag assignment (valid condition)

- Best practices for NaNs
  - Always include basic, single-variable rules for limit checks when applicable (don’t rely on multi-column rules)
  - Use \texttt{no\_nan()} to remove NaNs from data before calculating mean, min, max, ...
  - Use \texttt{nan2zero()} to convert NaNs to 0 when rules involve totals

\begin{verbatim}
>> x = [2.1,7.25,1.2,-1,NaN,2.3]
x =
  2.1000    7.2500    1.2000   -1.0000   NaN    2.3000
>> idx = x == NaN
idx =
     0     0     0     0     0     0
>> idx = isnan(x)
idx =
     0     0     0     0     1     0
>> mean(x)
ans =
   NaN
>> mean(no_nan(x))
ans =
  2.3700
\end{verbatim}
Functions that calculate or revise values include options for assigning flags:

- Interpolated values – interp_missing()
- Calculated values – add_calcexpr(), calc_missing()
- Revised values – update_data()

Useful for tracking data provenance and revisions.
Flags for Summarized Data

- Functions that aggregate or bin data include options for generating Q/C criteria to flag derived columns.

- Relevant metrics columns included automatically:
  - Num_Missing_xxx
  - Pct_Missing_xxx

- Preserves info about quality and completeness of primary data.

- Rules and flag assignments can be edited in derived data set using same tools as primary data set.
Manual Assignment (Grid)

- Flags can be manually edited in the data grid
  - Useful for fine-tuning assignments
  - Can add and remove flags
  - Can filter view to hone in on flags or values
  - Any edits “lock” rules to prevent overwriting revisions
Graphical Assignment

- Visual Q/C tool can be invoked from interactive data plots
  - Actions variable-specific to prevent inadvertent flagging of wrong values
  - Can zoom/pan to isolate points to flag
  - Left-click/drag to assign, right-click/drag to clear

- Anomaly reports can be auto-generated and edited to describe rationale in metadata
Copying Flags

- Flags can be copied from 1+ columns to 1+ other columns (many:many cardinality)
- Useful for propagating flags to dependent columns
  - After manual editing
  - To avoid complex dependency checks
- Can add to or over-write flags on target column(s)
- Can use GUI form or copyflags() function from command line
Text Columns to Flags

- Flags imported as text columns can be converted to intrinsic toolbox flags
  - Single and multi-character flags supported
  - Multi-character flags require mapping (/settings/flap_mapping.mat)
- Flag columns must have “Flag_” prefix and name matching corresponding data column (Salinity, Flag_Salinity)
- Flags can be added to or overwrite existing flags
- Can be used to cache manual flag assignments before restoring automatic evaluation
  - Instantiate flags as columns
  - Edit manual flags as necessary
  - Unlock flags and revise Q/C criteria as necessary
  - Convert text columns to flags, adding to new automatically-assigned flags
Specialized Flag Assignment Func’s

- `daterange2flags()`
  - Assigns flag to 1 or more columns based on date ranges
  - Accepts array of start/end dates (discontinuous)
  - Useful for flagging known outages or maintenance periods
  - Command-line only

- `codes2criteria()`
  - Generates Q/C criteria for coded columns based on code definitions in the metadata
  - Useful for catching typos and undefined codes
  - Command-line or Data Editor menu command
Managing Qualifiers

- Qualifiers can be cleared using the same approaches
  - Revise Q/C criteria limits or remove rules
  - Clear flags manually or graphically on plots (right-click)
  - Copy flags from other columns using the overwrite option

- Criteria can be locked/unlocked to manage auto assignment
  - Locking adds the token “manual” to criteria (but preserves other rules)
  - Unlocking removes “manual”, triggers recalculation
  - Any manually-assigned flags cleared when criteria unlocked!

- Bulk operations also supported
  - Clear all flags in selected column(s) and lock rules
  - Revise flag codes using search/replace
Managing Qualified Values

- Data values and qualifier flags paired and managed together transparently
  - Flags move with data when sorting rows/columns, merging, joining data
  - Flag column present even if no criteria defined or flags are assigned
  - Revising data value triggers criteria evaluation (unless locked)

- Data never automatically altered/deleted based on flags
  - Flags regarded as value-level metadata
  - Separation of data, flags permits flexible handling in workflows, archives

- Data analysis, plotting, export tools provide options for managing flagged values “on the fly”
  - Retain, document/display flags with values
  - Delete all or selected flagged values
  - Ignore flags
Managing Qualified Values

- Can document flagged/missing values in metadata
- Can selectively delete flagged values, rows permanently
Q/C Strategy Recommendations

- Define limit check rules for all measured variables
  - Sensor reading outside max/min specs
  - Impossible values (e.g. negative count, fraction over 100%, zero mass)
  - Improbable values based on historical data, experience (e.g. >3 SD over mean)
- Define set-based rules to check coded values (flag_inlist, ...)
- Define multi-column dependency rules for related variables
  - Redundant measurements comparable (e.g. air temp)
  - Suitable conditions when measurements made (e.g. good water depth, above freezing, suitable wind speed)
  - Reasonable time of day for diurnal variables (e.g. PAR, solar rad day/night)
  - Readings not recorded during extreme events (e.g. precip, wind)
- Define rules to check for expected disruptions or maintenance
  - Sensors disconnected or out of water for cleaning
  - Stuck/fouled/blocked sensors
Q/C Strategy Recommendations

- Define pattern check rules to identify possible contamination
  - Value changes too rapid (flag_valuechange, flag_nsigma)
  - Too little value change over time (flag_novaluechange)
  - Short-duration spikes/dips (flag_valuechange, flag_percentchange)
  - Long-duration maintenance events (flag_well_pumping)

- Save baseline Q/C rules in metadata template for re-use

- Review/refine automatic flag assignments on data plots

- Propagate flags to dependent/calculated columns to reflect edits on primary measurements

- Generate anomalies report and edit to include descriptions of rationale for manual revisions
Demo/Exercises

- Import or load a sample data set
- Define flags and rules in Q/C Criteria editor
  - Add and define one or more flags using ‘Flag’ button
  - Define limit check rules using primary form controls
  - Define multi-column rules using ‘Add/Edit Custom Criteria’
  - Define algorithmic rules using ‘Add/Edit QC Flag Function’
  - Try to implement all of the syntax examples from slides 9–10
  - Save the metadata as a template for re-use
- Refine flags on a data plot
  - Add single flags and drag to flag a range of values
  - Clear flags
- Copy flags from multiple columns to another column
- Unlock flags to restore automatic calculation
- Delete values assigned a specific flag
- Interpolate missing values and assign a flag to the imputed values
## Core QA/QC Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>addflags</td>
<td>Adds a character flag to specified records in one or more columns of a GCE Data Structure</td>
<td><code>[s2, msg] = addflags(s, cols, rows, flag, flagdef)</code></td>
</tr>
<tr>
<td>clearflags</td>
<td>Clears specified flags from a GCE Data Structure for display or export purposes</td>
<td><code>[s2, msg] = clearflags(s, flagchars, cols, lockopt, metaopt)</code></td>
</tr>
<tr>
<td>codes2criteria</td>
<td>Generates QA/QC criteria for coded columns in a GCE Data Structure based on code definitions</td>
<td><code>[s2, msg] = codes2criteria(s, cols, flagcode)</code></td>
</tr>
<tr>
<td>cols2flags</td>
<td>Converts values in specified text columns of a GCE Data Structure to QA/QC flags</td>
<td><code>[s2, msg] = cols2flags(s, flagcols, datacols, overwrite, delete, prefix)</code></td>
</tr>
<tr>
<td>cols2flags_mapped</td>
<td>Converts data set columns to QA/QC flags after mapping multi-character flags to single character</td>
<td><code>[s2, msg] = cols2flags_mapped(s, flagset_name, flagcols, datacols, overwrite, delete)</code></td>
</tr>
<tr>
<td>copyflags</td>
<td>Copies composite flags from one or more columns and adds to or replaces the existing flag arrays</td>
<td><code>[s2, msg] = copyflags(s, flagcols, datacols, opt)</code></td>
</tr>
<tr>
<td>cullflags</td>
<td>Deletes all records from a GCE Data Structure containing any values assigned specified flags</td>
<td><code>[s2, msg] = cullflags(s, flagchars, cols, metaopt, logsize)</code></td>
</tr>
<tr>
<td>dataflag</td>
<td>Evaluates Q/C criteria or assigns user-specified flags to generate or update Q/C flag arrays</td>
<td><code>[s2, msg, flagerrors] = dataflag(s, cols, manual_flags)</code></td>
</tr>
<tr>
<td>daterange2flags</td>
<td>Assigns Q/C flags by date range and locks flags to prevent automatic recalculation</td>
<td><code>[s2, msg] = daterange2flags(s, datecol, dates, cols, flag)</code></td>
</tr>
<tr>
<td>get_flagdefs</td>
<td>Retrieves definitions for selected QA/QC flags from GCE Data Structure metadata</td>
<td><code>flagdefs = get_flagdefs(s, flags)</code></td>
</tr>
<tr>
<td>harvest_check</td>
<td>Generates a harvest check email based on user-specified thresholds for missing and flagged values</td>
<td><code>msg = harvest_check(s, days, missing, flagged, daily_flagged, wrap)</code></td>
</tr>
<tr>
<td>nullflags</td>
<td>Converts values in a GCE Data Structure assigned specified flags to NaN/empty</td>
<td><code>[s2, msg] = nullflags(s, flagchars, cols, metaopt, clearflags, newflag, logsize)</code></td>
</tr>
</tbody>
</table>
QA/QC GUI Applications

ui_clearflags  GCE Data Toolbox dialog for selective deletion of QA/QC flags or flagged values, or selective conversion of flags to columns
syntax: ui_clearflags(op, s, mode, h_cb, cb)

ui_copyflags  GCE Data Toolbox dialog for copying composite flags from one or more data columns
syntax: ui_copyflags(op, s, h_cb, cb)

ui_flagdefs  QA/QC flag definition and data anomalies editor dialog called by 'ui_editor'.
syntax: ui_flagdefs(op, meta, h_cb, cb)

ui_flagfunction  GUI dialog called by 'ui_qccriteria' for adding custom function calls to a Q/C criteria string
syntax: ui_flagfunction

ui_flagpicker  Adds a Q/C flag picker popupmenu and edit button to a GUI figure at a specified position
syntax: ui_flagpicker(op, s, pos, h, cb, enable)

ui_manual_qc  Opens a GCE Data Structure in a data grid to allow data values and Q/C flags to be viewed and edited
syntax: ui_manual_qc(op, s, cols, h, cb, colwid)

ui_qccriteria  QA/QC criteria editor dialog called by 'ui_editor' and 'ui_template'
syntax: ui_qccriteria(op, s, col, flagdefs, h_cb, cb)

ui_querybuilder  GUI dialog for building row restriction queries to subselect data from a GCE Data Structure
syntax: ui_querybuilder(op, val)

ui_string_replace  Dialog called by 'ui_editor' for searching and replace text values or flags in a GCE Data Structure
syntax: ui_string_replace(op, s, col, target, h_cb, cb)

ui_visualqc  Dialog for assigning and clearing QC/QA flags visually by clicking on data points with the mouse.
syntax: ui_visualqc(op, qc_data)
**QA/QC Functions**

- **flag_locks**: Locks or unlocks Q/C flag criteria for specified columns in a GCE-LTER data structure. 
  Syntax: `[s2, msg] = flag_locks(s, lock_option, cols)`

- **flag_replace**: Performs string replacement for flags of one or more columns in a GCE Data Structure. 
  Syntax: `[s2, msg] = flag_replace(s, cols, oldstring, newstring, matchtype, caseopt)`

- **flags2cols**: Converts Q/C flags in a GCE Data Structure to coded string or integer columns in the data set. 
  Syntax: `[s2, msg] = flags2cols(s, flagcols, clearopt, missing, pos, encode, prefix, firstflag)`

- **flags2cols_selected**: Converts selected Q/C flag information in specified columns of a GCE Data Structure to coded string columns. 
  Syntax: `[s2, msg] = flags2cols_selected(s, flags, cols, encode, emptycols, prefix)`

- **flag_ctd_soak_period**: Returns an index indicating records collected during the pre-deployment soaking period of a CTD cast. 
  Syntax: `Iflag = flag_ctd_soak_period(vals, windowsize, maxdepth)`

- **flag_daterange**: Returns an index of date/time values that are within a specified time of day for time-based QA/QC flagging. 
  Syntax: `Iflag = flag_daterange(datetime, daterange, dateformat)`

- **flag_inarray**: Returns an index of numeric values that are present in a specified array. 
  Syntax: `Iflag = flag_inarray(vals, valuearray, tolerance)`

- **flag_inlist**: Returns an index of string values that are present in a specified list or file. 
  Syntax: `Iflag = flag_inlist(vals, valuelist, caseopt)`

- **flag_locationcoords**: Returns an index of location codes with coordinates that differ by more than the specified tolerance from reference coordinates. 
  Syntax: `Iflag = flag_locationcoords(location, longitude, latitude, tolerance, caseoption, ref_file)`

- **flag_locationnames**: Returns an index of location code values that are not present in the geographic database. 
  Syntax: `Iflag = flag_locationnames(locations, caseopt, emptyopt)`
QA/QC Functions (cont.)

flag_notinarray  
Returns an index of numeric values that are not present in a specified array  
syntax: Iflag = flag_notinarray(vals, valuearray, tolerance)

flag_notinlist  
Returns an index of string values that are not present in a specified list or file  
syntax: Iflag = flag_notinlist(vals, valuelist, caseopt, emptyopt)

flag_novaluechange  
Returns an index of values that do not differ from the mean of preceeding values  
syntax: Iflag = flag_novaluechange(vals, lowlimit, highlimit, framesize, iterations)

flag_nsigma  
Returns an index of values above or below the mean of preceeding values by the specified number of standard deviations  
syntax: Iflag = flag_nsigma(vals, lowlimit, highlimit, framesize, iterations)

flag_o2saturation  
Returns an index of DO concentrations that are above or below specified saturation limits  
syntax: Iflag = flag_o2saturation(o2_conc, temp, sal, maxsat, minsat, units)

flag_percentchange  
Returns an index of values above/below the mean of preceeding values by specified percentage  
syntax: Iflag = flag_percentchange(vals, lowlimit, highlimit, framesize, iterations)

flag_sitenames  
Returns an index of site code values that are not present in the geographic database  
syntax: Iflag = flag_sitenames(sitecodes, caseopt, emptyopt)

flag_timeofday  
Returns an index of date/time values that are within a specified time of day  
syntax: Iflag = flag_timeofday(datetime, timerange, dateformat)

flag_total  
Returns an index of values that exceed a limit when totalled with a specified number of preceding values  
syntax: Iflag = flag_total(vals, highlimit, framesize, iterations)

flag_valuechange  
Returns an index of values above or below the mean of preceeding values by a limit  
syntax: Iflag = flag_valuechange(vals, lowlimit, highlimit, framesize, iterations)

flag_well_pumping  
Returns an index indicating records collected during and following well pumping events based on negative spikes in pressure  
syntax: Iflag = flag_well_pumping(vals, windowsize, threshold)