```matlab
function handles = dataImportBatch( handles )
% This function imports all data in the specified directory into the tables in
% GUI_fileLoad2.m based on a standardized file format(see
% getChannelMapBatchLoad.m and getFileNameFormat. After the files from
% the folder have been imported, users can import additional data
% individually or submit the imported data for use in the main GUI.

%% Get all file names from the directory
batchFiles = handles.batchFileNames; %all files from folder structure starting with parent folder, including all operating conditions
numberBatchFiles = length( batchFiles );
batchSignalOperationNamesOnly = handles.batchSignalNamesOnly ; %operating conditions associated with each signal file
numberBatchSignalsOnly = handles.numberBatchSignalsOnly ; %number of signals in the batch directory (excludes ambients)
batchSignalFileNamesOnly = handles.batchSignalFileNamesOnly ;% file names for each signal in batch directory (excludes ambients)
batchSignalTrialNumber = handles.batchSignalTrialNumber ;% trial number inferred from file name

%% Variable assignments based on File Format

switch handles.fileFormat
    case 'Version 1' %Currently only one file format version, but the switch has been created in anticipation of future versions
        %Channel mappings from getChannelMapBatchLoad.m
        batchDriverChannel = handles.batchDriverChannel ;
        batchPassengerChannel= handles.batchPassengerChannel ;
        batchDriverPreAmbientChannel = handles.batchDriverPreAmbientChannel ;
        batchDriverPostAmbientChannel = handles.batchDriverPostAmbientChannel ;
        batchPassengerPreAmbientChannel = handles.batchPassengerPreAmbientChannel ;
        batchPassengerPostAmbientChannel = handles.batchPassengerPostAmbientChannel ;

        % Variable assignments based on optype
        for ink = 1 : numberBatchSignalsOnly

            tempFileName = batchSignalFileNamesOnly( ink ) ;

            if strcmp( batchSignalOperationNamesOnly( ink ), '10Passby')
                sampleID = handles.batchSignalTrialNumber( ink ) ; %Trial number from file name assigned to sample ID
                batchDriverSignalFileName = tempFileName ;
                batchPassengerSignalFileName = tempFileName ;
                batchTriggerSignalFileName = tempFileName ;
                batchPreAmbientFileName = handles.fileNamePreAmbient10 ;
                batchPostAmbientFileName = handles.fileNamePostAmbient10 ;
                batchPathName = handles.path10kmh ; %Path for the operating condition
                batchOpCond= '10 km/h' ;

                %Cell variable for table display
                dat10kmh( ink, : ) = [ sampleID , batchDriverSignalFileName, batchPreAmbientFileName, batchPostAmbientFileName, ...
                    batchPassengerSignalFileName , batchPreAmbientFileName, batchPostAmbientFileName, ...
                    batchTriggerSignalFileName ];
```

```
            end

            if strcmp( batchSignalOperationNamesOnly( ink ), '20Passby')
                sampleID = handles.batchSignalTrialNumber( ink ) ;
                batchDriverSignalFileName = tempFileName ;
                batchPassengerSignalFileName = tempFileName ;
                batchTriggerSignalFileName = tempFileName ;
                batchPreAmbientFileName = handles.fileNamePreAmbient20 ;
                batchPostAmbientFileName = handles.fileNamePostAmbient20 ;
                batchPathName = handles.path20kmh ;
                batchOpCond = '20 km/h';

                %Cell variable for table display
                dat20kmh( ink, : ) = [ sampleID , batchDriverSignalFileName, batchPreAmbientFileName,
                batchPostAmbientFileName, ...
                    batchPassengerSignalFileName , batchPreAmbientFileName, batchPostAmbientFileName, ...
                    batchTriggerSignalFileName ];
            end

            if strcmp( batchSignalOperationNamesOnly( ink ), '30Passby')
                sampleID = handles.batchSignalTrialNumber( ink ) ;
                batchDriverSignalFileName = tempFileName ;
                batchPassengerSignalFileName = tempFileName ;
                batchTriggerSignalFileName = tempFileName ;
                batchPreAmbientFileName = handles.fileNamePreAmbient30 ;
                batchPostAmbientFileName = handles.fileNamePostAmbient30 ;
                batchPathName = handles.path30kmh ;
                batchOpCond = '30 km/h' ;

                %Cell variable for table display
                dat30kmh( ink, : ) = [ sampleID , batchDriverSignalFileName, batchPreAmbientFileName,
                batchPostAmbientFileName, ...
                    batchPassengerSignalFileName , batchPreAmbientFileName, batchPostAmbientFileName, ...
                    batchTriggerSignalFileName ];
            end

            if  strcmp( batchSignalOperationNamesOnly( ink ), 'Stationary')
                sampleID = handles.batchSignalTrialNumber( ink ) ;
                batchDriverSignalFileName = tempFileName ;
                batchPassengerSignalFileName = tempFileName ;
                batchFrontSignalFileName = tempFileName ;
                batchPreAmbientFileName = handles.fileNamePreAmbientStat ;
                batchPostAmbientFileName = handles.fileNamePostAmbientStat ;
                batchPathName = handles.pathStationary ;
                batchOpCond = 'Stationary' ;

                %Cell variable for table display
                datStationary( ink, : ) = [ sampleID , batchDriverSignalFileName, batchPreAmbientFileName ,
                batchPostAmbientFileName,...
```

```
                batchPassengerSignalFileName , batchPreAmbientFileName,batchPostAmbientFileName, ...
                batchFrontSignalFileName , batchPreAmbientFileName, batchPostAmbientFileName ];

            end

            if  strcmp( batchSignalOperationNamesOnly( ink ), 'Reverse')
                sampleID = handles.batchSignalTrialNumber( ink ) ;
                batchDriverSignalFileName = tempFileName ;
                batchPassengerSignalFileName = tempFileName ;
                batchPreAmbientFileName = handles.fileNamePreAmbientRev ;
                batchPostAmbientFileName = handles.fileNamePostAmbientRev ;
                batchPathName = handles.pathReverse ;
                batchOpCond = 'Reverse' ;

                %Cell variable for table display
                datReverse( ink, : ) = [ sampleID , batchDriverSignalFileName, batchPreAmbientFileName,
                batchPostAmbientFileName,...
                    batchPassengerSignalFileName , batchPreAmbientFileName, batchPostAmbientFileName ];
            end

            % Import data into handles for any operating condition
            %(if statement used for consisitency, check probably not needed)
            if strcmp( batchSignalOperationNamesOnly( ink ), '10Passby') || strcmp( batchSignalOperationNamesOnly(
            ink ), '20Passby') ||...
                    strcmp( batchSignalOperationNamesOnly( ink ), '30Passby') || strcmp(
                    batchSignalOperationNamesOnly( ink ), 'Stationary') ||...
                    strcmp( batchSignalOperationNamesOnly( ink ), 'Reverse')

                % Map variables for data import (driver/passenger)
                opCond = batchOpCond ;%defined in previous conditional statement within the loop
                sampleID = handles.batchSignalTrialNumber{ ink };

                %remapping variables for consistency with individual file load
                driverChannel = batchDriverChannel ;
                driverPreAmbientChannel = batchDriverPreAmbientChannel ;
                driverPostAmbientChannel = batchDriverPostAmbientChannel ;
                passengerChannel = batchPassengerChannel ;
                passengerPreAmbientChannel = batchPassengerPreAmbientChannel ;
                passengerPostAmbientChannel = batchPassengerPostAmbientChannel ;
                driverSignalFileName = char( batchDriverSignalFileName ) ;
                passengerSignalFileName = char( batchPassengerSignalFileName ) ;
                preAmbientFileName = batchPreAmbientFileName ;
                postAmbientFileName = batchPostAmbientFileName ;

                %Dataset variable holds all data for import
                dataImport = dataset( { { opCond; opCond; opCond; opCond; opCond; opCond }, 'opCond' }, ...
                    { { 'driver'; 'driver'; 'driver'; 'passenger'; 'passenger'; 'passenger' }, 'micSide' }, ...
                    { { sampleID; sampleID; sampleID; sampleID; sampleID; sampleID }, 'sampleID' }, ...
                    { [ driverChannel; driverPreAmbientChannel; driverPostAmbientChannel; passengerChannel;
                    passengerPreAmbientChannel; passengerPostAmbientChannel ], 'channel' }, ...
```

```
140                     { { 'signal'; 'preAmbient'; 'postAmbient'; 'signal'; 'preAmbient'; 'postAmbient' }, 'signalType'
                        }, ...
141                     { { batchPathName; batchPathName; batchPathName; batchPathName; batchPathName; batchPathName },
                        'pathName' }, ...
142                     { { driverSignalFileName; preAmbientFileName; postAmbientFileName; passengerSignalFileName;
                        preAmbientFileName; postAmbientFileName }, 'fileName' }, ...
143                     { { []; []; []; []; []; [] }, 'TriggerThreshold' }, ...
144                     { { []; []; []; []; []; [] }, 'idx' } );
145
146                 %Casting to nominal format required for later code
147                 dataImport.opCond = nominal( dataImport.opCond );
148                 dataImport.micSide = nominal( dataImport.micSide );
149                 dataImport.sampleID = nominal( dataImport.sampleID );
150                 dataImport.signalType = nominal( dataImport.signalType );
151                 dataImport.pathName = nominal( dataImport.pathName );
152                 dataImport.fileName = nominal( dataImport.fileName );
153
154                 %Create/append row to aggregate data
155                 if isempty( handles.data )
156                     handles.data = dataImport;
157                 else
158                     handles.data = [ handles.data; dataImport ];
159                 end
160
161                 % Create trigger dataset for passbys
162                 if strcmp( batchSignalOperationNamesOnly( ink ), '10Passby' ) || strcmp(
                    batchSignalOperationNamesOnly( ink ), '20Passby' )|| strcmp( batchSignalOperationNamesOnly( ink ),
                    '30Passby' )
163                     triggerChannel = handles.batchTriggerChannel ;
164                     triggerFileName = char( batchTriggerSignalFileName );%Defined in if statements above
165
166                     triggerImport = dataset( { { opCond }, 'opCond' }, ...
167                         { { 'all' }, 'micSide' }, ...
168                         { { sampleID }, 'sampleID' }, ...
169                         { triggerChannel, 'channel' }, ...
170                         { { 'trigger' }, 'signalType' }, ...
171                         { { batchPathName }, 'pathName' }, ...
172                         { { triggerFileName }, 'fileName' }, ...
173                         { { [20] }, 'TriggerThreshold' }, ...
174                         { { [] }, 'idx' } );
175
176                     triggerImport.opCond = nominal( triggerImport.opCond );
177                     triggerImport.micSide = nominal( triggerImport.micSide );
178                     triggerImport.sampleID = nominal( triggerImport.sampleID );
179                     triggerImport.signalType = nominal( triggerImport.signalType );
180                     triggerImport.pathName = nominal( triggerImport.pathName );
181                     triggerImport.fileName = nominal( triggerImport.fileName );
182
183                     handles.data = [ handles.data; triggerImport ];
184                 end
```

```
                %Create front dataset for stationary
                if strcmp( batchSignalOperationNamesOnly( ink ), 'Stationary' )

                    frontChannel = handles.batchFrontChannel ;
                    frontPreAmbientChannel = handles.batchFrontPreAmbientChannel;
                    frontPostAmbientChannel = handles.batchFrontPostAmbientChannel;
                    frontSignalFileName = char( batchFrontSignalFileName );

                    frontImport = dataset( { { opCond; opCond; opCond }, 'opCond' }, ...
                        { { 'front'; 'front'; 'front' }, 'micSide' }, ...
                        { { sampleID; sampleID; sampleID }, 'sampleID' }, ...
                        { [ frontChannel; frontPreAmbientChannel; frontPostAmbientChannel ], 'channel' }, ...
                        { { 'signal'; 'preAmbient'; 'postAmbient' }, 'signalType' }, ...
                        { { batchPathName; batchPathName; batchPathName }, 'pathName' }, ...
                        { { frontSignalFileName; preAmbientFileName; postAmbientFileName }, 'fileName' }, ...
                        { { []; []; [] }, 'TriggerThreshold' }, ...
                        { { []; []; [] }, 'idx' } );

                    frontImport.opCond = nominal( frontImport.opCond );
                    frontImport.micSide = nominal( frontImport.micSide );
                    frontImport.sampleID = nominal( frontImport.sampleID );
                    frontImport.signalType = nominal( frontImport.signalType );
                    frontImport.pathName = nominal( frontImport.pathName );
                    frontImport.fileName = nominal( frontImport.fileName );

                    handles.data = [ handles.data; frontImport ];
                end
            end
        end
end

%% Delete empty rows for table display
datStationary(all(cellfun('isempty',datStationary),2),:) = [];
dat10kmh(all(cellfun('isempty',dat10kmh),2),:) = [];
dat20kmh(all(cellfun('isempty',dat20kmh),2),:) = [];
dat30kmh(all(cellfun('isempty',dat30kmh),2),:) = [];
datReverse(all(cellfun('isempty',datReverse),2),:) = [];

%% Update handles
handles.datStationary = datStationary ;
handles.dat10kmh = dat10kmh ;
handles.dat20kmh = dat20kmh ;
handles.dat30kmh = dat30kmh ;
handles.datReverse = datReverse ;

setappdata( 0, 'data', handles.data);
end
```