Batch Manufacturing Process Data Visualization
A Configurable JMP® Scripting Language Application to Generate Elapsed-Time Overlay Plots

Gwen Tennant, Alkermes, Inc., 265 Olinger Circle, Wilmington, Ohio 45177
gwen.tennant@alkermes.com

ABSTRACT
For batch manufacturing, the ability to simultaneously review process control system data from multiple batches is critical for troubleshooting process upsets and examining the impact of process changes. Manufacturing support personnel often choose to generate simple time-series overlay plots to visualize batch data because these charts can reveal nuances that might otherwise be obscured by summary statistics or mathematical models. In order to create multi-batch overlay plots, individual Supervisory Control and Data Acquisition (SCADA) data timestamps must first be converted to elapsed times with respect to fixed reference points in the manufacturing process. Performing these data manipulations manually is cumbersome and time-consuming when working with large data sets. Fortunately, the JMP Scripting Language (JSL) has enabled us to develop a custom application that directly interfaces with the data historian and automatically generates configurable multi-batch overlay plots. This nimble tool has dramatically improved data visualization capabilities and has reduced chart generation time to mere seconds.

MODEL BATCH PROCESS

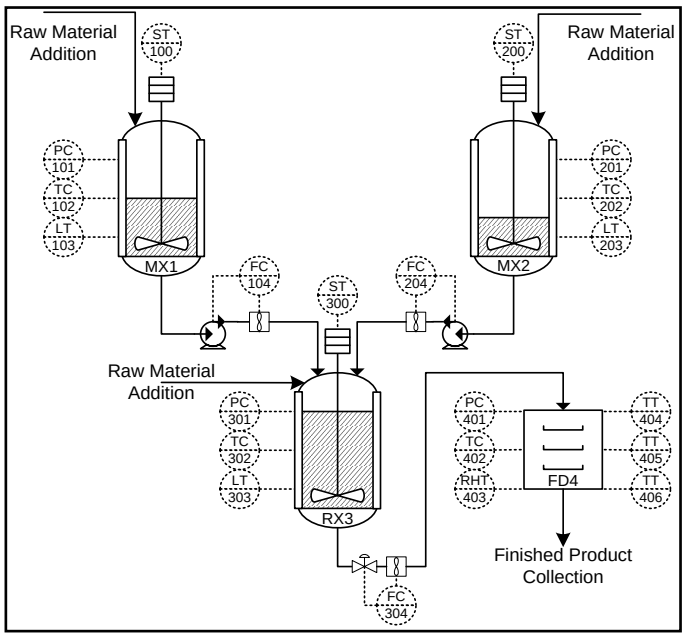

Figure 1: Batch Process P&ID

In order to provide a common framework for this analysis, consider a generic batch process consisting of two mixing tanks (MX1 and MX2), a reactor (RX3), and a filter dryer (FD4).

Raw materials are added to MX1, MX2, and RX3 at various points in the manufacturing process. These three vessels require pressure, temperature, and flow control systems. Instrumentation is also available to measure agitator speed and tank level.

The filter dryer contains temperature and pressure control loops in addition to a probe for measuring the relative humidity in the unit. Product temperature is measured at three independent locations in the filter dryer as well.

The basic piping and instrumentation diagram (P&ID) presented in Figure 1 provides an overview of this process. A complete list of inputs and outputs (I/O) is provided in Appendix A.

HISTORICAL DATA
It is common for data from modern process control systems to be periodically recorded in a historical archive. The readings from process instrumentation are stored along with timestamps that identify precisely when the data were obtained. This information is critical to manufacturing support personnel for troubleshooting process upsets and examining the impact of process changes. For example, data recorded from FD4 batch drying operations have been provided in Table 1.

time_stamp	PC_401_CV	PC_401_PV	PC_401_SP	RHT_403	TC_402_CV	TC_402_PV	TC_402_SP	TT_404	TT_405	TT_406
12/09/2011 6:18:37 PM	38.949	26.284	25	11.351	40.512	3.568	3	8.615	8.095	8.439
12/09/2011 6:19:36 PM	39.439	26.006	25	10.662	38.980	3.671	3	8.101	8.075	7.884
12/09/2011 6:19:37 PM	39.448	26.035	25	10.645	38.969	3.671	3	8.101	8.054	7.822
12/09/2011 6:20:36 PM	39.784	25.619	25	10.118	36.769	3.855	3	7.998	7.684	7.740
12/09/2011 6:20:37 PM	39.790	25.649	25	10.101	36.748	3.855	3	7.957	7.705	7.740

Table 1: Sample FD4 Process Data from SCADA System

BASIC TIME-SERIES PLOTS

Manufacturing support personnel often choose to generate simple time-series plots to visualize batch data. These charts are quick and easy to generate because they do not require any data manipulation. Also, time-series charts can reveal nuances that might otherwise be obscured by summary statistics or mathematical models. For instance, the plot provided in Figure 2 shows the relative humidity recorded in FD4 during the drying operations that took place between 09-Dec-2011 and 12-Dec-2011.

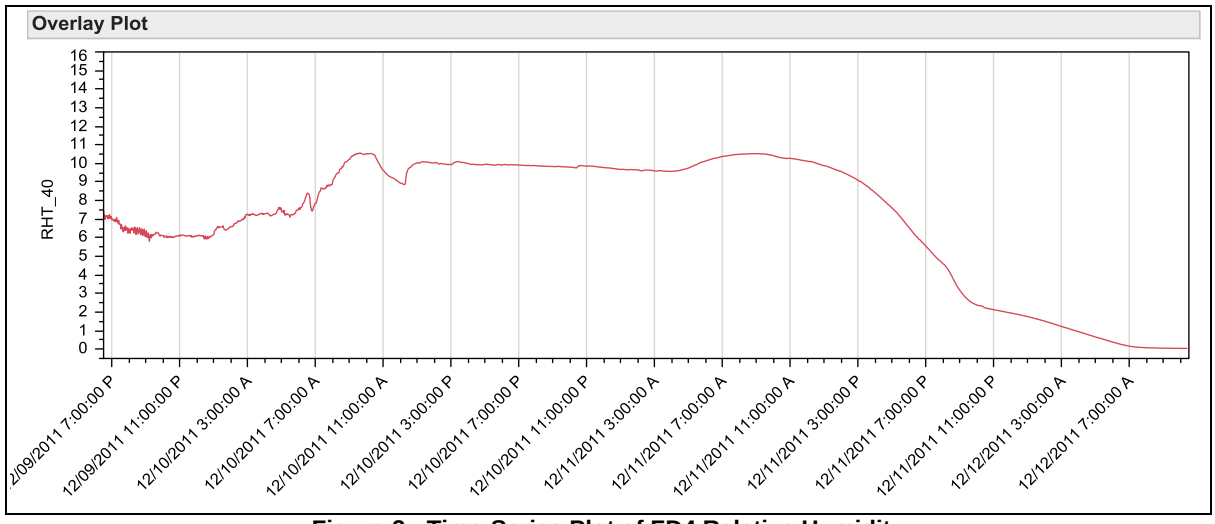

Figure 2: Time-Series Plot of FD4 Relative Humidity

Unfortunately, it is difficult to identify batch-to-batch variation when plotting timestamps on the x axis. As illustrated in Figure 3, while the basic shapes of the four separate batch profiles are apparent, it is not easy to ascertain the intricate details. When more batches are added to the plot, the individual batch trends become further compressed. Ultimately, once the time scale is sufficiently large, the individual batch trends are obscured beyond the point where any useful information can be obtained.

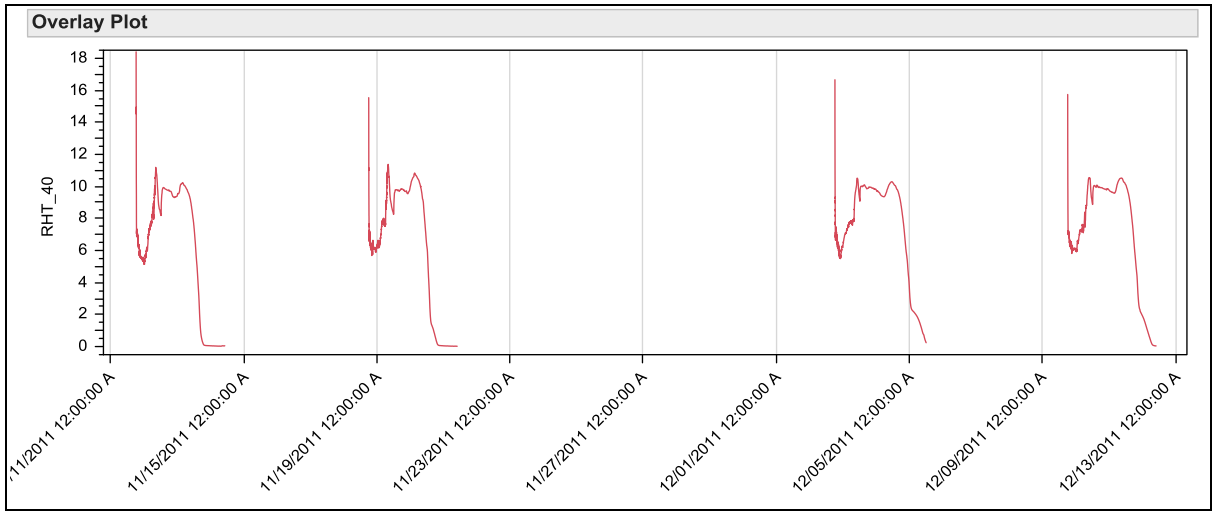

Figure 3: Visualizing Batch-to-Batch Variation with Timestamps on X Axis

Another shortcoming of these basic time-series charts is a lack of identifying information about the particular measurement being trended. No engineering units are given, and while some information about the sensor may be gleaned from the naming convention, there is no description of the instrument's functionality or link to the associated unit operation. Although some commercial data historians readily export this type of information along with the requested data, it is not unusual for these details to be omitted.

ELAPSED-TIME OVERLAY PLOTS

Elapsed-time overlay plots provide a better method of visualizing process data for multiple batches. Prior to creating these graphs, individual timestamps must be converted to elapsed times with respect to fixed reference points in the manufacturing process. Data can then be plotted for each batch individually to create the "overlay" effect, as shown in Figure 4.

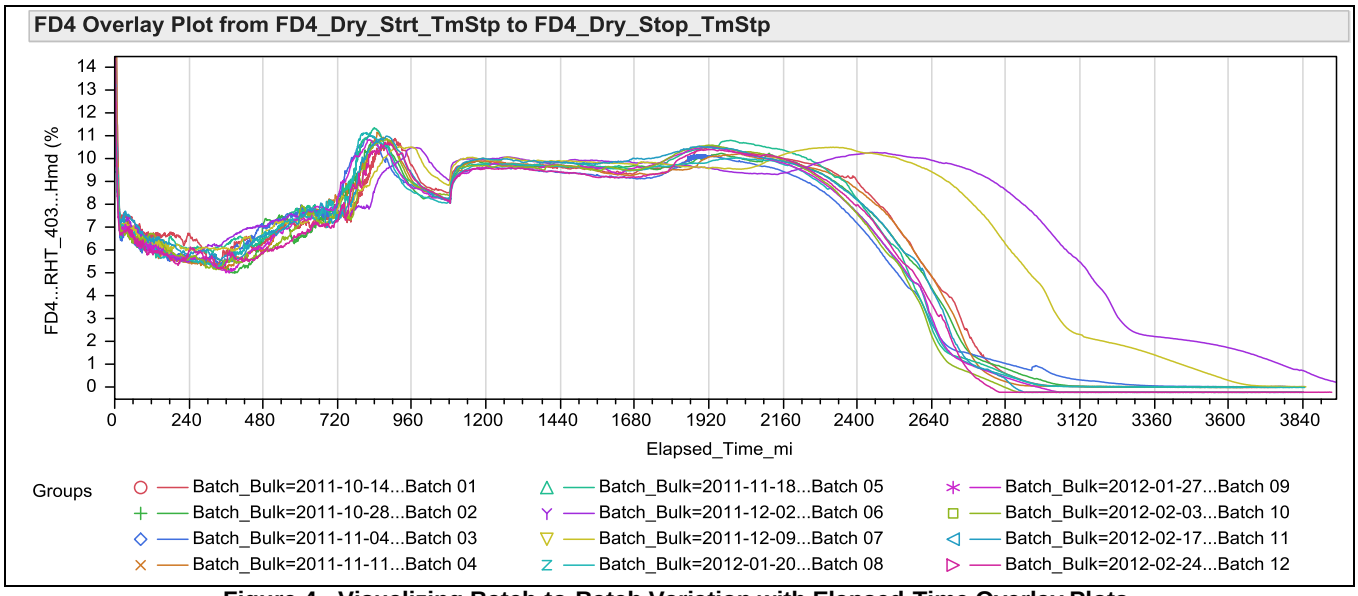

Figure 4: Visualizing Batch-to-Batch Variation with Elapsed-Time Overlay Plots

BATCH MANUFACTURING OVERLAY PLOT BUILDER APPLICATION

Performing elapsed-time data manipulations manually is cumbersome and time-consuming when working with large data sets. Additionally, it is necessary to know precisely when each batch operation started/stopped to retrieve the desired information from the data historian. Fortunately, the JMP Scripting Language has enabled us to develop a custom application that directly interfaces with the data historian and automatically generates configurable multi-batch overlay plots. The user interface for the "Batch Manufacturing Overlay Plot Builder" tool is shown in Figure 5.

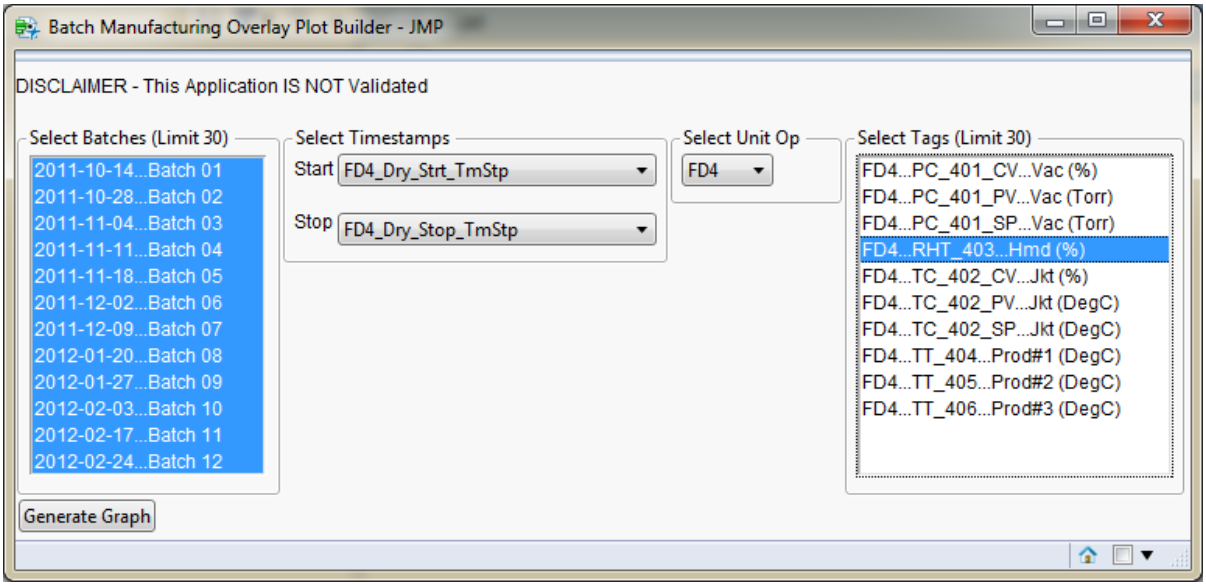

Figure 5: User Interface for Batch Manufacturing Overlay Plot Builder

The first step in automating the overlay plot generation process involves establishing a connection between JMP and the data historian. JMP software utilizes Open Database Connectivity (ODBC) to access external Structured Query Language (SQL) databases. Once the appropriate ODBC drivers have been obtained, a Machine Data Source (User or System DSN) can be configured on the local JMP computer to connect with the historical database. If direct access to the data historian is not achievable or advisable, it may still be possible to connect through a "bridge" database with read-only linked server access to the SCADA system. Refer to the *JMP® 10 Scripting Guide* for additional information on connecting to external databases.

Once the link between JMP and the data historian has been achieved, the next task involves identifying useful reference time points in the batch manufacturing recipe. For the specified example batch process, timestamps for raw material additions, transfers between vessels, reactions, and dry times are of particular interest. For many automated processes, unique batch identification numbers and relevant timestamps are also stored in a historical database. However, if no electronic batch record information is available, it will be necessary to manually populate a table with the relevant timestamps for each batch. An excerpt from the timestamp table for the FD4 unit operation is provided in Table 2.

BatchID	FD4_Dry_Strt_TmStp	FD4_Dry_18hrMrk_TmStp	FD4_Dry_SS_Hmd_TmStp	FD4_Dry_Stop_TmStp	FD4_XfrOut_Strt_TmStp	FD4_XfrOut_Stop_TmStp
2011-10-14...Batch 0	10/14/2011 6:42:08 PM	10/15/2011 12:42:29 PM	10/16/2011 8:43:17 PM	10/17/2011 10:43:37 AM	10/17/2011 10:58:30 AM	10/17/2011 11:11:18 AM
2011-10-28...Batch 0	10/28/2011 6:06:07 PM	10/29/2011 12:06:30 PM	10/30/2011 9:09:02 PM	10/31/2011 10:07:26 AM	10/31/2011 10:22:08 AM	10/31/2011 10:36:02 AM
2011-11-04...Batch 0	11/04/2011 6:08:18 PM	11/05/2011 12:08:41 PM	11/06/2011 10:03:03 PM	11/07/2011 9:10:24 AM	11/07/2011 9:25:15 AM	11/07/2011 9:37:44 AM
2011-11-11...Batch 0	11/11/2011 6:29:30 PM	11/12/2011 12:29:52 PM	11/13/2011 7:25:02 PM	11/14/2011 10:36:50 AM	11/14/2011 10:51:38 AM	11/14/2011 11:10:58 AM
2011-11-18...Batch 0	11/18/2011 6:13:16 PM	11/19/2011 12:13:38 PM	11/20/2011 8:05:36 PM	11/21/2011 10:15:12 AM	11/21/2011 10:29:59 AM	11/21/2011 10:42:16 AM
2011-12-02...Batch 0	12/02/2011 6:24:58 PM	12/03/2011 12:25:20 PM	12/05/2011 12:12:53 PM	12/05/2011 12:12:53 PM	12/05/2011 12:27:40 PM	12/05/2011 12:45:00 PM
2011-12-09...Batch 0	12/09/2011 6:15:10 PM	12/10/2011 12:15:32 PM	12/12/2011 7:25:37 AM	12/12/2011 10:24:55 AM	12/12/2011 10:39:40 AM	12/12/2011 10:53:09 AM
2012-01-20...Batch 0	01/20/2012 6:20:52 PM	01/21/2012 12:21:12 PM	01/22/2012 8:30:54 PM	01/23/2012 10:26:35 AM	01/23/2012 10:41:21 AM	01/23/2012 10:55:32 AM
2012-01-27...Batch 0	01/27/2012 6:22:03 PM	01/28/2012 12:22:23 PM	01/29/2012 9:22:06 PM	01/30/2012 10:23:45 AM	01/30/2012 10:38:32 AM	01/30/2012 10:51:56 AM
2012-02-03...Batch 1	02/03/2012 6:17:18 PM	02/04/2012 12:17:39 PM	02/05/2012 7:04:54 PM	02/06/2012 10:19:27 AM	02/06/2012 10:34:08 AM	02/06/2012 10:57:52 AM
2012-02-17...Batch 1	02/17/2012 6:20:05 PM	02/18/2012 12:20:28 PM	02/19/2012 7:29:55 PM	02/20/2012 10:23:14 AM	02/20/2012 10:37:57 AM	02/20/2012 10:52:37 AM
2012-02-24...Batch 1	02/24/2012 6:27:58 PM	02/25/2012 12:28:21 PM	02/26/2012 6:15:17 PM	02/27/2012 12:01:10 PM	02/27/2012 12:20:38 PM	02/27/2012 12:33:50 PM

Table 2: FD4 Timestamp Table

The modular design of the Batch Manufacturing Overlay Plot Builder application relies heavily on forming SQL statements through string concatenation. Conforming to consistent naming conventions is critical for successful operation. It is helpful to utilize "views" in the historical database so that table and field names follow convenient, logical standards.

In order to generate the interactive selection lists and drop-down menus in the user-interface, a reference table must be created containing the tagnames for all historically trended I/O by unit operation, along with the associated engineering units and a short description. This information can then be combined to form a detailed description, improving the level of background information available for display on the y axis of the overlay plots. It is often possible to build this query from an information schema view in the historical database. An excerpt from the tagname table for the FD4 unit operation is provided in Table 3.

Table_Name	Tag_Name	Unit_Op	Eng_Units	Short_Desc	Long_Desc
SCADA_FD4	PC_401_C	FD4	%	Vac	FD4...PC_401_CV...Vac (%)
SCADA_FD4	PC_401_P	FD4	Torr	Vac	FD4...PC_401_PV...Vac (Torr
SCADA_FD4	PC_401_S	FD4	Torr	Vac	FD4...PC_401_SP...Vac (Torr
SCADA_FD4	RHT_403	FD4	%	Hmd	FD4...RHT_403...Hmd (%)
SCADA_FD4	TC_402_C	FD4	%	Jkt	FD4...TC_402_CV...Jkt (%)
SCADA_FD4	TC_402_P	FD4	DegC	Jkt	FD4...TC_402_PV...Jkt (DegC
SCADA_FD4	TC_402_S	FD4	DegC	Jkt	FD4...TC_402_SP...Jkt (DegC
SCADA_FD4	TT_404	FD4	DegC	Prod#1	FD4...TT_404...Prod#1 (DegC
SCADA_FD4	TT_405	FD4	DegC	Prod#2	FD4...TT_405...Prod#2 (DegC
SCADA_FD4	TT_406	FD4	DegC	Prod#3	FD4...TT_406...Prod#3 (DegC

Table 3: FD4 Tagname Table

After the timestamp and tagname tables have been assembled, the following JSL code is used to retrieve them from the database. Note the first line of script prevents the automatic generation of table variables containing the ODBC connection strings, as these can contain user ID and password information. Also, once the tables have been opened, the windows are minimized so they do not take up screen space unnecessarily.

```
//Preference prevents creation of table variable that can contain User ID and password info
pref(ODBC Hide Connection String(1));

//Open data table containing all available batches and timestamps
SQL_Statement = "SELECT * FROM V_TmStp ORDER BY BatchID";
dt1 = Open Database(
    "DSN=DemoDSN;APP=JMP;DATABASE=DemoDB;Trusted_Connection=Yes",
    SQL_Statement,
    "V_TmStp");

//Open data table containing list of SCADA tags with engineering units and descriptions
SQL_Statement = "SELECT * FROM V_SCADA_Tag_List ORDER BY Unit_Op, Tag_Name";
dt2 = Open Database(
    "DSN=DemoDSN;APP=JMP;DATABASE=DemoDB;Trusted_Connection=Yes",
    SQL_Statement,
    "V_SCADA_Tag_List");

//Minimize opened data tables
dt1 << minimize window( 1 );
dt2 << minimize window( 1 );
```

Once the timestamp and tagname tables are open in JMP, list variables must be created to populate the drop-down menus and list boxes in the user interface. Four separate variables store the lists of unique batch IDs, unit operations, SCADA tags, and timestamps. The order of the items within each list depends on the method by which the variable was created. Note the "BatchID" column name must be removed from the timestamp list.

```
//Create lists of batches, unit ops, SCADA tags, and timestamps for drop-down menus and list boxes
    //These lists are created in alphabetical order
    Summarize( BatchList = By( Column( dt1, "BatchID" ) ) );
    Summarize( UnitOpList = By( Column( dt2, "Unit_Op" ) ) );
    Summarize( TagList = By( Column( dt2, "Long_Desc" ) ) );

    //This list appears in the order the columns are sorted, as opposed to alphabetical
    TmStpList = dt1 << get column names( string ); //Useful to put timestamps in chronological order
    Remove From( TmStpList, 1, n = 1 ); //Remove "BatchID" from the list of available timestamps
```

The user interface consists of a series of panel, text, combo, and list boxes arranged within horizontal, vertical, and lineup boxes. For reference, several of these components are labeled in Figure 6. The corresponding code to generate the interactive user interface is provided below. Of particular interest is a dynamic H List Box where the tagnames displayed in the list box are dependent on the unit operation selected in the adjacent combo box. (For more information on this technique, there is an excellent example in Chapter 6 of the book *Jump into JMP® Scripting* involving the makes and models of cars.)

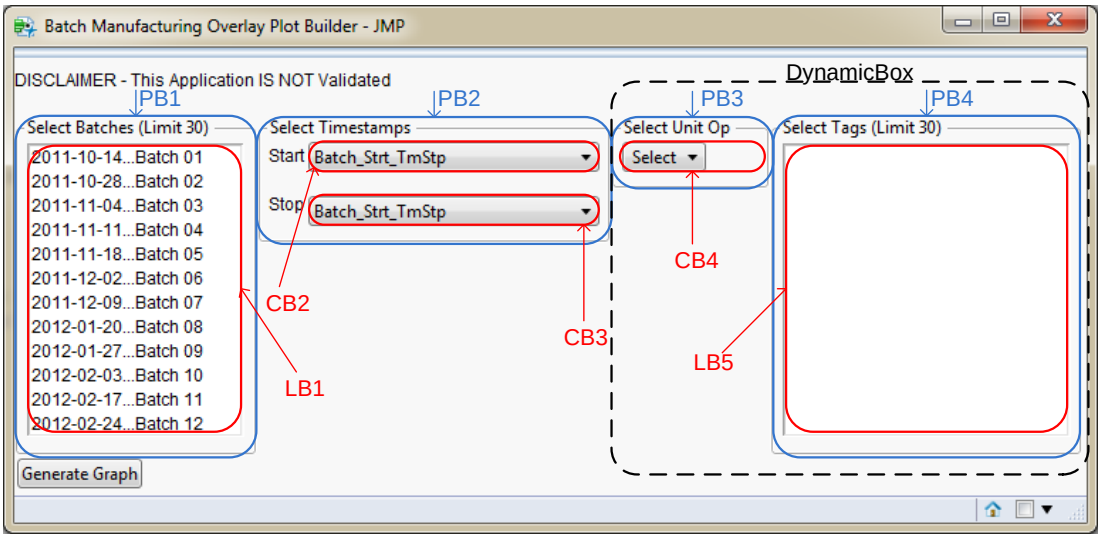

Figure 6: Individual Components of User Interface

```
nw = New Window( "Batch Manufacturing Overlay Plot Builder",

Text Box( "DISCLAIMER - This Application IS NOT Validated" ),
Text Box( "" ),

H List Box(
    //List Box to select batches
    PB1 = Panel Box( "Select Batches (Limit 30)",
        LB1 = List Box( BatchList, width( 150 ), maxSelected( 30 ), nlines( 12 ) )
    ),
    //Combo Boxes to select start and stop timestamps
    PB2 = Panel Box( "Select Timestamps",
        Lineup Box( N Col( 2 ), Spacing( 0 ),
            V List Box( Text Box( "Start " ), Text Box( "" ), Text Box( "Stop " ) ),
            V List Box( CB2 = Combo Box( TmStpList ), Text Box( "" ), CB3 = Combo Box( TmStpList ) )
        )
    ),
    //SCADA tag list is generated based on selected Unit Op
    DynamicBox = H List Box(
        PB3 = Panel Box( "Select Unit Op",
            CB4 = Combo Box(
                UnitOpList,
                dt2 << Select Where( :Unit_Op == UnitOpList[CB4 << get] );
                selRows = dt2 << Get Selected Rows;
                PB4 << delete;
                TagList = {};
                For( i = 1, i <= N Row( selRows ), i++,
                    Insert Into( TagList, Column( dt2, "Long_Desc" )[selRows[i]] )
                );
                DynamicBox << Append(
                    PB4 = Panel Box( "Select Tags (Limit 30)",
                        LB5 = List Box( TagList, width( 200 ), maxSelected( 30 ), nlines( 12 ) )
                    )
                )
            )
        ),
        PB4 = Panel Box( "Select Tags (Limit 30)",
            List Box( " ", width( 200 ), maxSelected( 30 ), nlines( 12 ) )
        )
    )
),
```

The button box is the final element of the user interface. Any script associated with the button box is executed once the button is clicked. Accordingly, the remainder of the JSL code to generate the elapsed-time overlay plots is contained within the button box display element. The first tasks involve closing the user interface window and retrieving the selections from the interactive elements of the user interface. Descriptive tagnames displayed on the user interface must also be matched with the simple field names used in the data historian.

```
Button Box( "Generate Graph",
    //Closes User Interface
    nw << close window;

    //Retrieves the selections from each of the user entry fields
    Batches = LB1 << get selected;
    StrtTmStpName = CB2 << get selected;
    StopTmStpName = CB3 << get selected;
    UnitOp = CB4 << get selected;
    LongTags = LB5 << get selected;

    //Generate List of associated "basic" SCADA tagnames from the selected "long" tagnames
    Tags = {};
    for(i=1, i<=N Items (LongTags),i++,
        dt2<<select where (:Unit_Op == UnitOp & :Long_Desc == LongTags[i]);
        selRow = dt2<<Get Selected Rows;
        Insert Into(Tags,:Tag_Name[selRow]);
    );
```

The next objective is to retrieve the requested information from the data historian. Utilizing a series of nested iterative loops, the historical data is requested from the SCADA system for one batch at a time. The application is designed to accommodate near real-time review of data for active batches, so if a particular timestamp is not yet available, it is replaced by the current time. By default, a minute is also subtracted from the start time and added to the stop time to account for the scan rate of the SCADA system. Once the data table for a particular batch is acquired, elapsed times are calculated in a separate formula column. Ultimately, all of the data obtained from the historian, including the calculated elapsed times, are concatenated into one common data table. The individual data tables are closed without saving.

```
//Initiate a loop that is executed once for each batch that is selected
For( i = 1, i <= N Items( Batches ), i++,
    //Create a variable that holds the string name to be used for each batch's data table
    dtname = Batches[i];

    //Construct the SCADA tag portion of the SQL string
    For( k = 1, k <= N Items( Tags ), k++,
        If( k == 1,
            SQLstring = "["||Tags[k]||"]",
            SQLstring1 = SQLString || ",["||Tags[k]||"]";
            SQLstring = SQLString1;
        )
    );

    //Lookup start and stop timestamps and format for use in SQL query
    dt1 << Select Where( dt1:BatchID == Batches[i] );
    BatchRowNum = dt1 << Get Selected Rows;
    If(Is Missing(Column(dt1,StrtTmStpName )[BatchRowNum]),
        StrtTmStp = Today(), StrtTmStp = Column( dt1, StrtTmStpName )[BatchRowNum]-60);
    If(Is Missing(Column(dt1,StopTmStpName )[BatchRowNum]),
        StopTmStp = Today(), StopTmStp = Column( dt1, StopTmStpName )[BatchRowNum]+60);
    StrtTmStpChar = Format( Eval( StrtTmStp ), "yyyy-mm-ddThh:mm" );
    StopTmStpChar = Format( Eval( StopTmStp ), "yyyy-mm-ddThh:mm" );
    StrtTmStpSQL = Munger( StrtTmStpChar, 1, "T", " " );
    StopTmStpSQL = Munger( StopTmStpChar, 1, "T", " " );

    //Concatenate SQL String to select the data table for each batch based on user input
    SQL_Statement = "SELECT Batch_Bulk = '"|| Batches[i] ||"',time_stamp,"||SQLstring||"
    FROM dbo.SCADA_"||UnitOp||"
    WHERE time_stamp > '"||StrtTmStpSQL||"' and time_stamp <= '"||StopTmStpSQL||"'
    ORDER BY time_stamp";

    //Open a JMP database table for each batch
    indiv_dt = Open Database(
        "DSN=DemoDSN;APP=JMP;DATABASE=DemoDB;Trusted_Connection=Yes",
        SQL_Statement,
        dtname
    );

    //Create a formula column to calculate elapsed times
    indiv_dt << new column("Elapsed_Time_min",Numeric,Continuous,
        formula(Date Difference(:time_stamp[1], :time_stamp, "Minute", "fractional")));

    //Concatenate the individual data tables
    If(i==1,
        indiv_dt<<Select All;
        overlay_dt=indiv_dt<<Subset(output table name("Overlay Plot Data Table"));
        overlay_dt << minimize window( 1 );,
        overlay_dt = overlay_dt<<Concatenate(indiv_dt, Append to first Table);
    );

    //Close each individual data table after the concatenation is done
    close(indiv_dt,NoSave);
);
```

After the historical data has been retrieved, the final task is to generate the elapsed-time overlay plots. The Batch Manufacturing Overlay Plot Builder application utilizes the “Overlay Groups” feature within the existing JMP

"Overlay Plot" platform for this purpose. Note if only one batch has been selected, timestamps will be displayed on the x axis, as opposed to elapsed times.

```
//Create Overlay Plots...
If( N Items( Batches ) == 1, //If only 1 batch selected, use timestamps on x axis
    xmin = Col Minimum( :time_stamp );
    xmax = Col Maximum( :time_stamp );
    op = Overlay Plot(
        X( :time_stamp ),
        Y( Eval( LongTags ) ),
        By( Eval( Batches ) ),
        Overlay( 0 ),
        Connect Points( 1 ),
        Show Points( 0 ),
        SendToReport(
            Dispatch( {}, "Overlay Plot", OutlineBox,
                {Set Title( UnitOp || " Overlay Plot from " || StrtTmStpName || " to "
                || StopTmStpName )} ),
            Dispatch( {}, "101", ScaleBox, {Min( xmin ), Max( xmax ),
                Interval( "Minute" ), Show Major Grid( 1 ), Rotated Labels( "Angled" )} )
        )
    );
, //Else if more than 1 batch is selected, use elapsed-times on x axis
    xmax = Col Maximum( :elapsed_time_min );
    op = Overlay Plot(
        X( :Elapsed_Time_min ),
        Y( Eval( LongTags ) ),
        Grouping( :Batch_Bulk ),
        Overlay Groups,
        Connect Points( 1 ),
        Show Points( 0 ),
        SendToReport(
            Dispatch( {}, "Overlay Plot", OutlineBox,
                {Set Title( UnitOp || " Overlay Plot from " || StrtTmStpName || " to "
                || StopTmStpName )} ),
            Dispatch( {}, "101", ScaleBox, {Min( 0 ), Max( xmax ), Show Major Grid( 1 )} )
        )
    );
    op << Send To Report( Dispatch( {}, "", LegendBox, {Set Wrap( 1 )} ) );//Set Legend Wrap at 1 Batch
);

//Adjust the frame size
If( (NItems(Batches) == 1 & NItems(Tags) == 1),
    op<<Send To Report(Dispatch( {}, "Overlay Plot Graph", FrameBox, {Frame Size( 800, 200 )} )),
    For( m = 1, m <= N Items( Tags ), m++,
        op << Send To Report( Dispatch( {}, "Overlay Plot Graph",
            FrameBox( N Items( Batches ) + m ), {Frame Size( 800, 200 )} ) )
    )
);
```

CASE STUDY

In the example batch process, recall the elapsed-time overlay plot of relative humidity in FD4 during drying operations. As highlighted in Figure 7, the relative humidity trends for Batch 06 and Batch 07 stray from the general population. The product temperature curves are similarly affected, ruling out an isolated malfunction with the humidity sensor.

The temperature and pressure control systems in FD4 both have the ability to impact product temperature and humidity. A quick review of the operating setpoints for both systems show they remain unchanged. The process values for the control loops also appear to be performing consistently for all 12 batches. Conversely, while the control values for the jacket temperature loop are repeatable, the control values for the pressure control loop appear to be lower than normal for Batch 06 and Batch 07.

Since lower vacuum levels would increase drying time and offset the product temperature profiles, troubleshooting efforts were quickly focused on the pressure control loop. The pressure transmitter PC_401_PV

was found to be in need of a calibration adjustment. Once the pressure sensor was serviced following the completion of Batch 07, the readings for all instruments returned to normal.

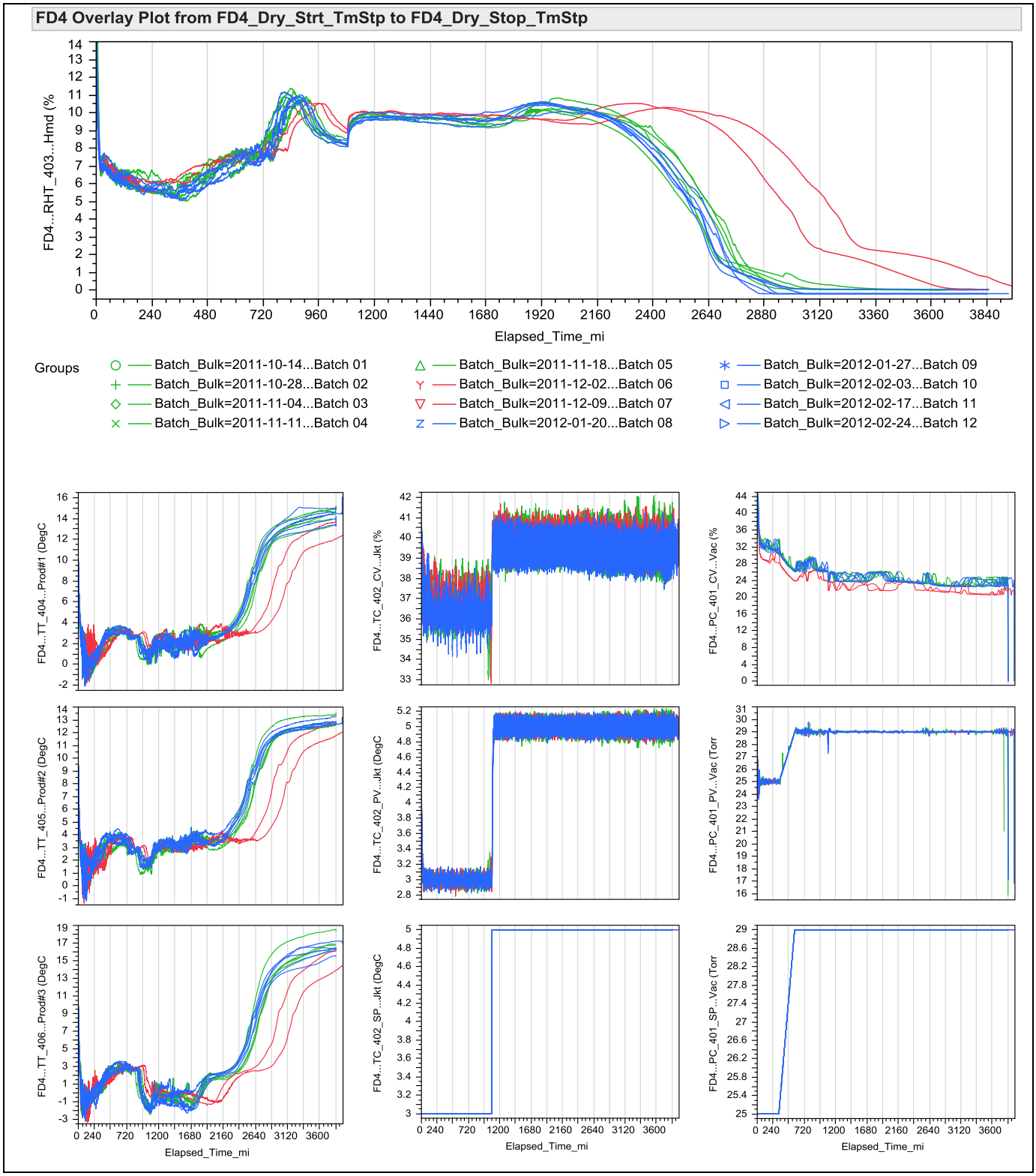

Figure 7: Case Study Batch 06 and Batch 07 Humidity and Product Temperature Irregularities

CONCLUSION

The Batch Manufacturing Overlay Plot Builder has become an invaluable tool for the manufacturing support staff at the Alkermes Wilmington facility. Prior to the introduction of this application, a 12-batch elapsed-time overlay plot would have taken multiple hours to generate. Now, it is possible to examine the historical data for every instrument on a particular unit operation within seconds. This custom application is currently being utilized to trend batch operations across 5 manufacturing lines containing 54 separate unit operations and 1,717 individual sensors. The ability to rapidly visualize this vast amount of information has transformed routine process monitoring and troubleshooting activities.

REFERENCES AND ACKNOWLEDGEMENTS

- SAS Institute Inc. 2012. *JMP® 10 Scripting Guide*. Cary, NC: SAS Institute Inc.
- Murphrey, Wendy, and Rosemary Lucas. *Jump into JMP® Scripting.* Cary, NC: SAS Institute Inc.

- I would like to thank the JMP Technical Support team who answered the numerous requests I submitted to support@jmp.com. Without their prompt and helpful solutions, this project would not have been possible.

APPENDIX A:
I/O List for Model Batch Process

MX1: Mix Tank 1		
Tag Name	**Units**	**Description**
ST_100	%	Agitator Speed
PC_101_CV	%	Tank Pressure Control Value
PC_101_PV	psig	Tank Pressure Process Value
PC_101_SP	psig	Tank Pressure Setpoint
TC_102_CV	%	Tank Temp Control Value
TC_102_PV	DegC	Tank Temp Process Value
TC_102_SP	DegC	Tank Temp Setpoint
LT_103	%	Tank Level
FC_104_CV	%	Flow Rate Control Value
FC_104_PV	kg/min	Flow Rate Process Value
FC_104_SP	kg/min	Flow Rate Setpoint

MX2: Mix Tank 2		
Tag Name	**Units**	**Description**
ST_200	%	Agitator Speed
PC_201_CV	%	Tank Pressure Control Value
PC_201_PV	psig	Tank Pressure Process Value
PC_201_SP	psig	Tank Pressure Setpoint
TC_202_CV	%	Tank Temp Control Value
TC_202_PV	DegC	Tank Temp Process Value
TC_202_SP	DegC	Tank Temp Setpoint
LT_203	%	Tank Level
FC_204_CV	%	Flow Rate Control Value
FC_204_PV	kg/min	Flow Rate Process Value
FC_204_SP	kg/min	Flow Rate Setpoint

RX3: Reactor 3		
Tag Name	**Units**	**Description**
ST_300	%	Agitator Speed
PC_301_CV	%	Tank Pressure Control Value
PC_301_PV	psig	Tank Pressure Process Value
PC_301_SP	psig	Tank Pressure Setpoint
TC_302_CV	%	Tank Temp Control Value
TC_302_PV	DegC	Tank Temp Process Value
TC_302_SP	DegC	Tank Temp Setpoint
LT_303	%	Tank Level
FC_304_CV	%	Flow Rate Control Value
FC_304_PV	kg/min	Flow Rate Process Value
FC_304_SP	kg/min	Flow Rate Setpoint

FD4: Filter Dryer 4		
Tag Name	**Units**	**Description**
PC_401_CV	%	Vac Pressure Control Value
PC_401_PV	Torr	Vac Pressure Process Value
PC_401_SP	Torr	Vac Pressure Setpoint
TC_402_CV	%	Jacket Temp Control Value
TC_402_PV	DegC	Jacket Temp Process Value
TC_402_SP	DegC	Jacket Temp Setpoint
RHT_403	%	Filter Dryer Relative Humidity
TT_404	DegC	Product Temp #1
TT_405	DegC	Product Temp #2
TT_406	DegC	Product Temp #3