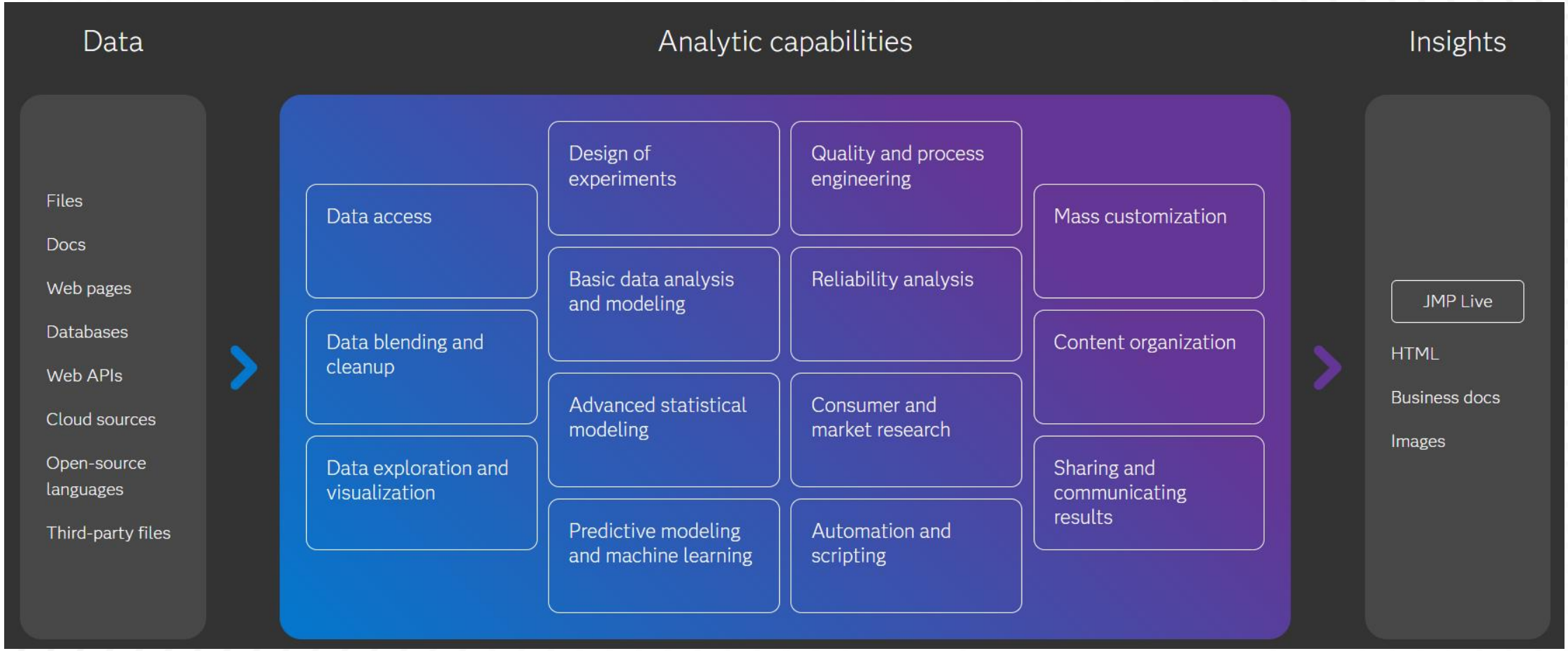
ALTA
Performance Materials

Connections make all the difference

Connecting JMP to an Aspen IP.21 historian

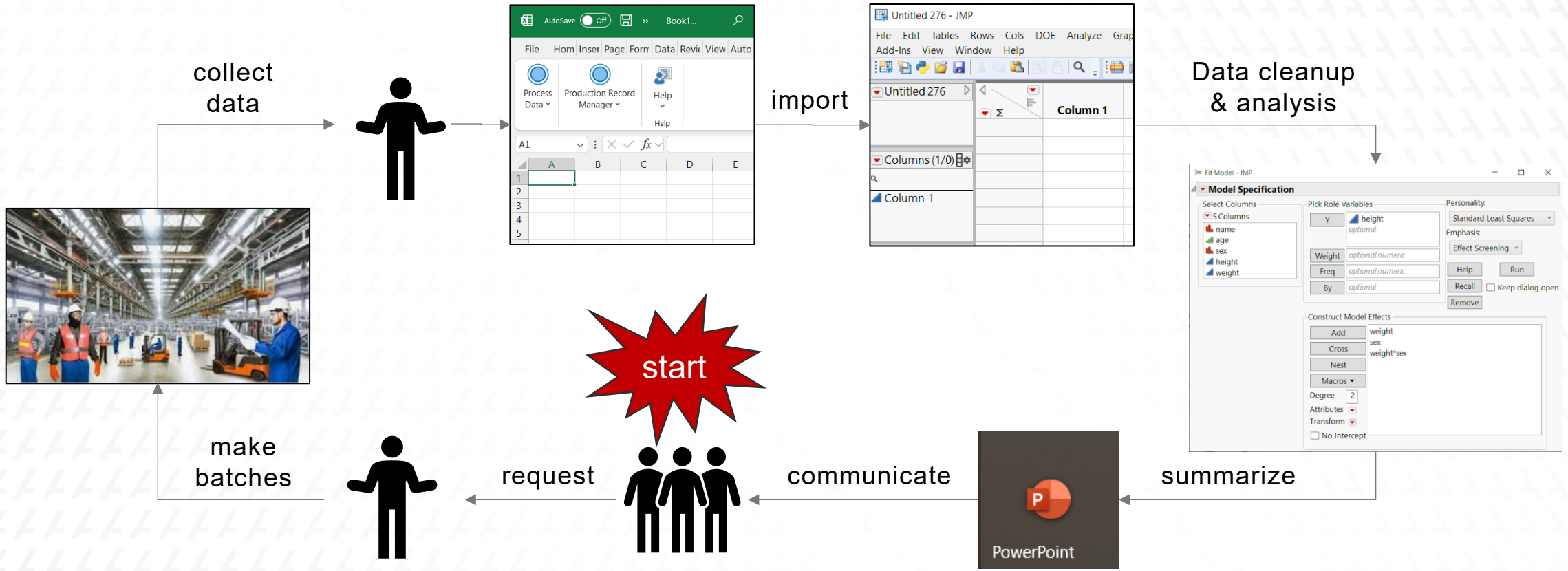
Tom Bidwell
23 July 2025

The JMP Analytic Workflow



What do you do when you can't easily connect to your data to use the workflow shown above?

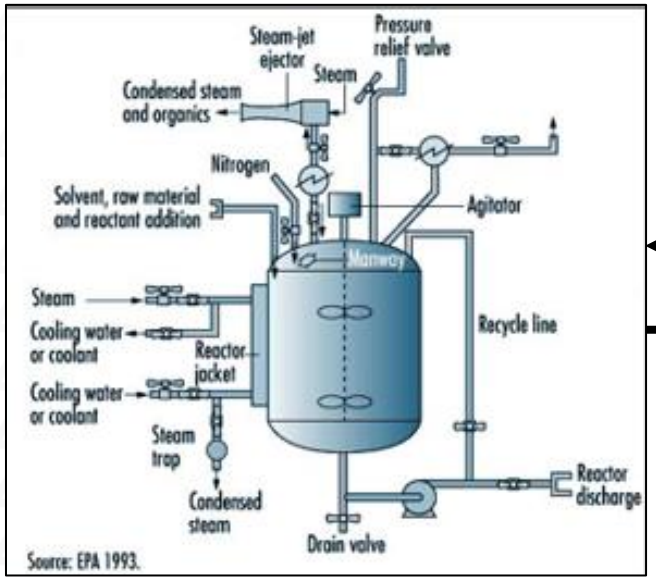
The Business problem



- **Current State:** The time to request, make, collect, import, clean, analyze, summarize and communicate the results took a month or more.
  - **The problem:** We wanted a direct connection to the data but had ***no idea*** how to make that happen.

Data flow from Reactors to JMP

Reactors:



The focus of today's talk

Emerson Delta V Distributed Control System

control

data

Process data

Batch data

Aspen IP.21* Historian

Aspen Batch Historian

Use Query Builder to access this SQL db

Excel

Access db

Other data

Process data

Batch data

Other data

jmp

* Aspen InfoPlus.21 (IP.21) is an industrial process historian collecting and storing large volumes of process data for analysis and reporting

Types of Connections required…

PHYSICAL

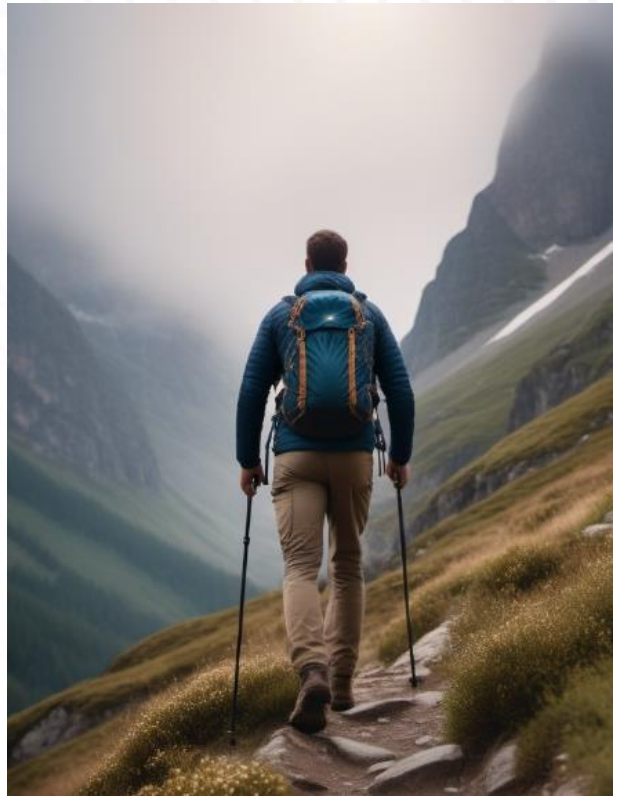
- Data connectors
  - How do we connect to Aspen?
    - We didn't know much about ODBC
    - We knew nothing about REST API

RELATIONAL

- Team of experts
  - No one else in ALTA did this kind of work
  - IT didn't know how to help…and they installed the wrong ODBC driver (32-bit)!

LANGUAGE

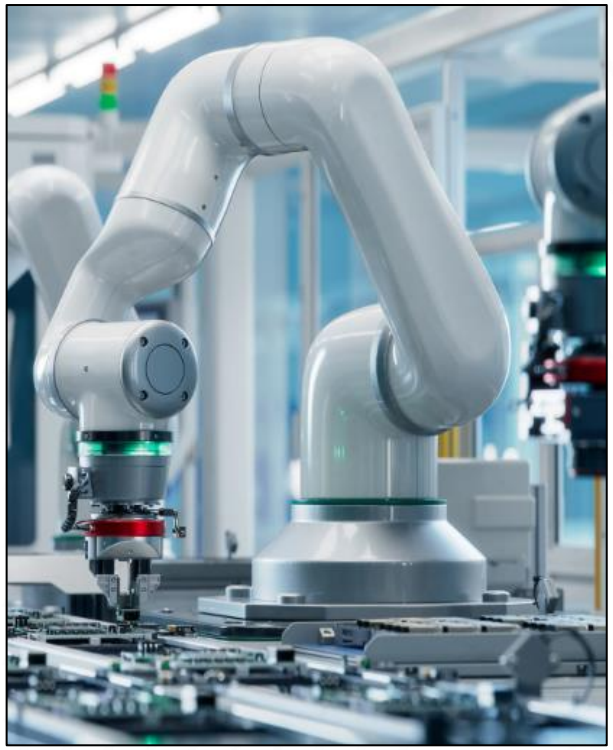
- Scripting
  - At the start, we didn't know much about scripting
  - How do we get JSL to use Aspen's query language, SQL+?



We needed a few guides (or sherpas) to join our team

The Vision & the Journey

Make the connection (ODBC)

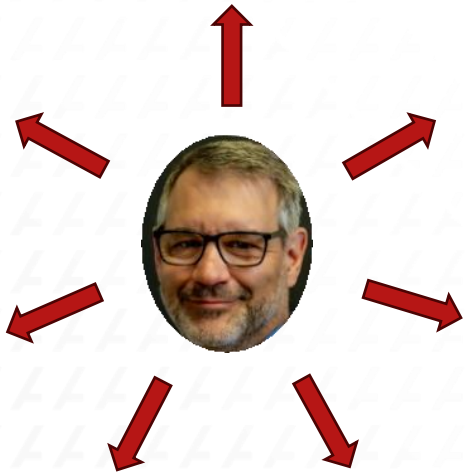


Automate & expand (JMP Add-in)

Accelerate & share (REST API + updated Add-in)

2017 → 2018-2023 → 2023-2025

Connections…the beginning



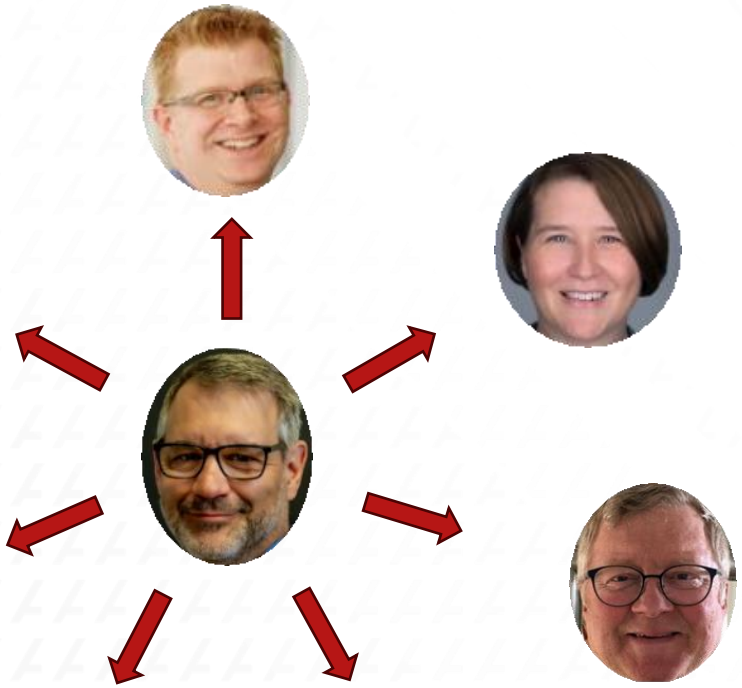
To climb the mountain, we realized we needed help. Over time, we received help from a variety of folks when we hit a wall.

Our connections to team members came from:

- JMP Discovery conferences
  - Plenaries, Papers & Meet the Experts
- JMP User group meetings
  - Central Ohio JMP User Group (COJUG)
- JMP User community
  - Discussion board - Ask questions

➢ Additional help came from the JMP Scripting Index

Connections…making a connection



- **Step 1:** Connect to the Aspen IP.21 historian via ODBC, using Aspen's SQL+ language.
  - 64-bit ODBC drivers – Stan Kaprowski (2017 - St. Louis JMP Discovery conference)
  - Custom SQL – Scripting Index
- **Step 2:** Automate through scripting
  - Basic Scripting in JSL – Gwen Hallberg, Alkermes (2018 COJUG User's group meeting)
    - Gwen's JSL scripting presentation files from 11-29-2018 COJUG meeting - JMP User Community
  - User interfaces in JSL – Peter Mroz, Janssen Pharmaceutical (2018 Cary JMP Discovery conference)
    - Supercharge Your User Interfaces in JSL (US 2018 113) - JMP User Community

SQL+ to JSL via ODBC and Custom SQL

Aspen SQLplus V12.0 - aspenONE - [redacted] - [Continuous historian query for JMP Discovery.SQL]

File Edit Record View Debug Query Window Help

```
SELECT name as tagname,
       CAST(ts as CHAR FORMAT 'YYYY-MM-DD HH:MI') as datetime,
       value as tagvalue
FROM history
WHERE period = 00:01:00
       AND ts between '06-JAN-24 06:00:00' and '06-JAN-24 06:05:00'
       AND ( name = '[redacted]_AI1_PV' OR name = '[redacted]_AI1_PV')
```

tagname	datetime	tagvalue
[redacted]_AI1_PV	2024-01-06 06:00	49.9056
[redacted]_AI1_PV	2024-01-06 06:01	49.2638
[redacted]_AI1_PV	2024-01-06 06:02	50.6364
[redacted]_AI1_PV	2024-01-06 06:03	49.3382
[redacted]_AI1_PV	2024-01-06 06:04	49.9212
[redacted]_AI1_PV	2024-01-06 06:00	114.348
[redacted]_AI1_PV	2024-01-06 06:01	114.338
[redacted]_AI1_PV	2024-01-06 06:02	114.328
[redacted]_AI1_PV	2024-01-06 06:03	114.318
[redacted]_AI1_PV	2024-01-06 06:04	114.308

```
//Querying the data using ODBC
New SQL Query(
    //put your Data Source name after ODBC:DSN=
    Connection( "ODBC:DSN=My SQLplus data;" ),
    Custom SQL(
        "
            SELECT name as tagname,
                CAST(ts as CHAR FORMAT 'YYYY-MM-DD HH:MI') as datetime,
                value as tagvalue
            FROM history
            WHERE period = 00:01:00
                AND ts between '06-JAN-24 06:00:00'
                    and '06-JAN-24 06:05:00'
                AND ( name = '" || aa["AspenTag1"] ||"'
                    OR name = '" || aa["AspenTag2"] ||"')
        "
    )
) << run foreground();
dt = current data table();
```

Insert the DSN for your ODBC connection here

Replace these with your actual Aspen tag names

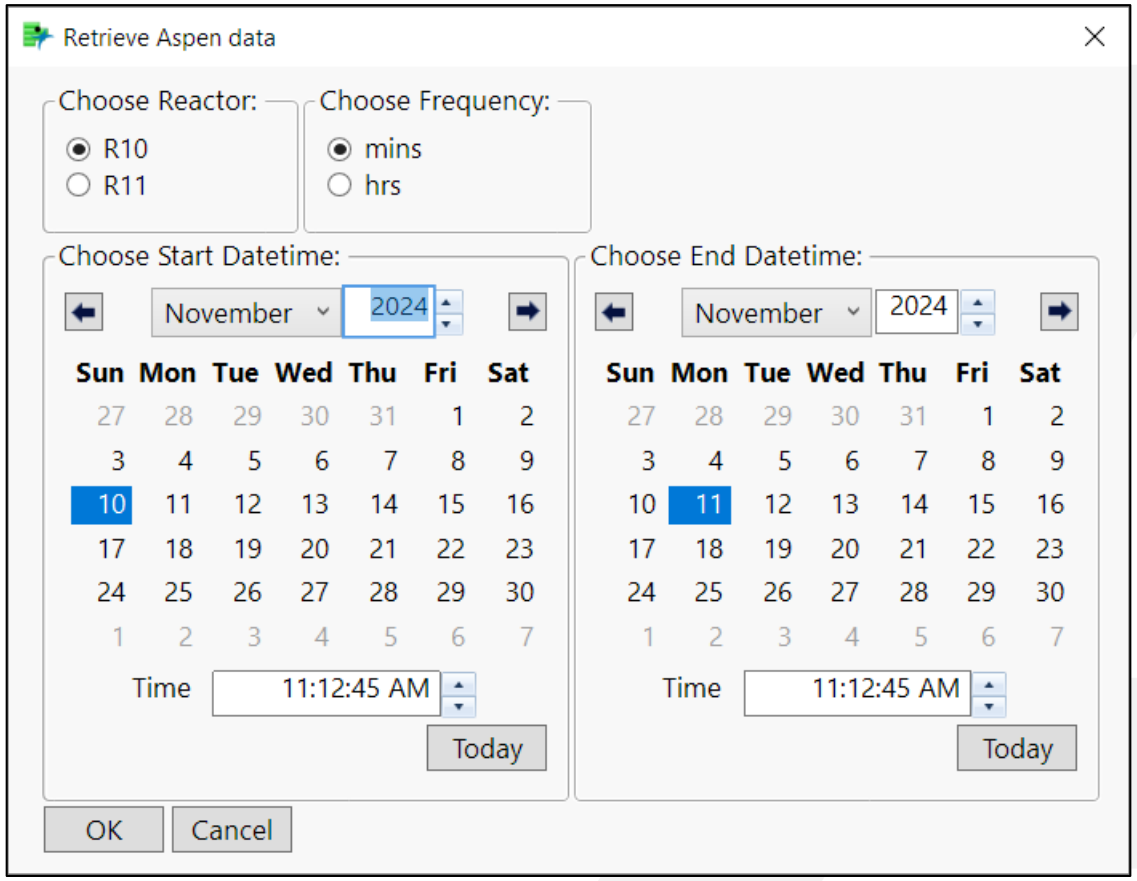
Aspen tags generally take the form of: PLT_ID-1234_AI1_PV

Paste your SQL+ query into the Custom SQL

Automation using a JMP Add-in for ODBC

- We needed an Add-in to automate the:
  - Reactors (and tags)
  - Sampling frequency
  - Start and Stop datetimes

- The script produces the user interface shown to the right which has:
  - One vertical list box with 3 elements:
    - #1: horizontal list box w/ 2 panel boxes for choosing the Reactor and Frequency
    - #2: horizontal list box w/ 2 panel boxes for choosing start and end datetimes
    - #3: horizontal list box w/ 2 button boxes for the “OK” & “Cancel” buttons



(see next slide for JSL to make this UI)

JSL for Automation (creating a user interface for the query)

```
//user interface to collect key data for the query
nw = new window("Retrieve Aspen data", <<modal,
    V List Box(
        H List Box(
            panel box("Choose Reactor:",
                rbRxr = Radio box( {"R10", "R11"} ),
            ),
            panel box("Choose Frequency:",
                rbFreq = Radio box( {"mins", "hrs"} ),
            ),
        ),
        H List Box(
            panel box("Choose Start Datetime:",
                cal1 = calendar box( date( today() - 24*60*60) ),
            ),
            panel box("Choose End Datetime:",
                cal2 = calendar box( date( today() ) ),
            )
        ),
        H List Box(
            Button box("OK",
                start   = format(cal1 << get date;,
                            "Format Pattern", "<DD>-<MMM>-<YY> <hh>:<mm>");
                end     = format(cal2 << get date;,
                            "Format Pattern", "<DD>-<MMM>-<YY> <hh>:<mm>");
                rxr     = rbRxr << get;
                freq    = rbFreq << get;
            ),
            Button Box("Cancel"),
        ),
    ),
);
//close window if Cancel is pressed
if( nw == {Button(-1)}, stop() );
```

```
//use the user choices to create the text for the query
//creating the query string by inserting the text
//into the "\[...]\" construction

//create text for sampling frequency
If(freq==1, freqtext = "WHERE period = 00:01:00 ",
            freqtext = "WHERE period = 01:00:00 ");

//create text for the datetimes
datetext = "\[AND ts between ']\" || start
            || "\[' and ']\" || end || "\[']\";

//create text for the tags
If( rxr==1,
        tagtext = "\[AND ( name = ']\" || aa["AspenTag1"]
            || "\[' OR name = ']\" || aa["AspenTag2"] || "\[' )]\";
    rxr==2,
        tagtext = "\[AND ( name = ']\" || aa["AspenTag5"]
            || "\[' OR name = ']\" || aa["AspenTag6"] || "\[' )]\";
);

//create text for Aspen SQL+ query
sqlplustext =
    "   SELECT name as tagname,
            CAST(ts as CHAR FORMAT 'YYYY-MM-DD HH:MI') as datetime,
            value as tagvalue
        FROM history " || freqtext || datetext || tagtext;

new sql query(
    Connection( "ODBC:DSN=My SQLplus data;" ),
    Custom SQL( sqlplustext ),
) << run foreground();
```

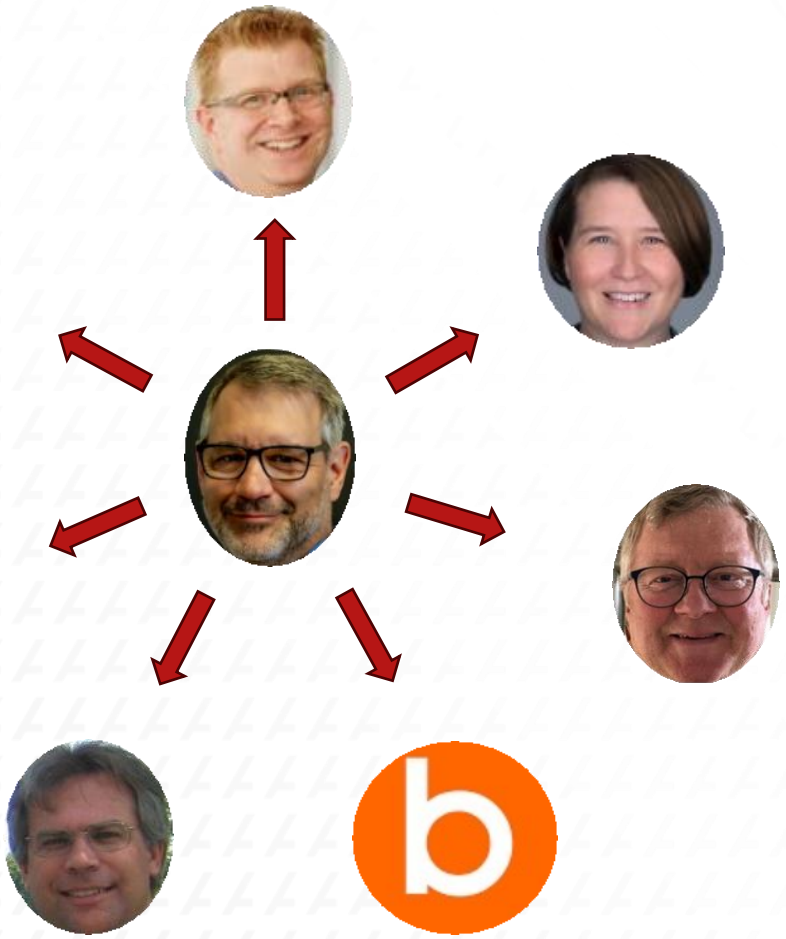
This JSL trick was so helpful

Insights from connecting JMP to Aspen w/ ODBC

- We've found opportunities to improve the DCS program to eliminate unknown programming issues causing cycle time losses.
- We've identified systemic equipment issues that were also unknown.
- We actively monitor new products as they are introduced into production.
- We now have a data source for modeling efforts to relate product properties to processing variables.

Connecting JMP to the Aspen data has been like a "Gemba Walk" through our process

Connections…accelerating w/ REST API

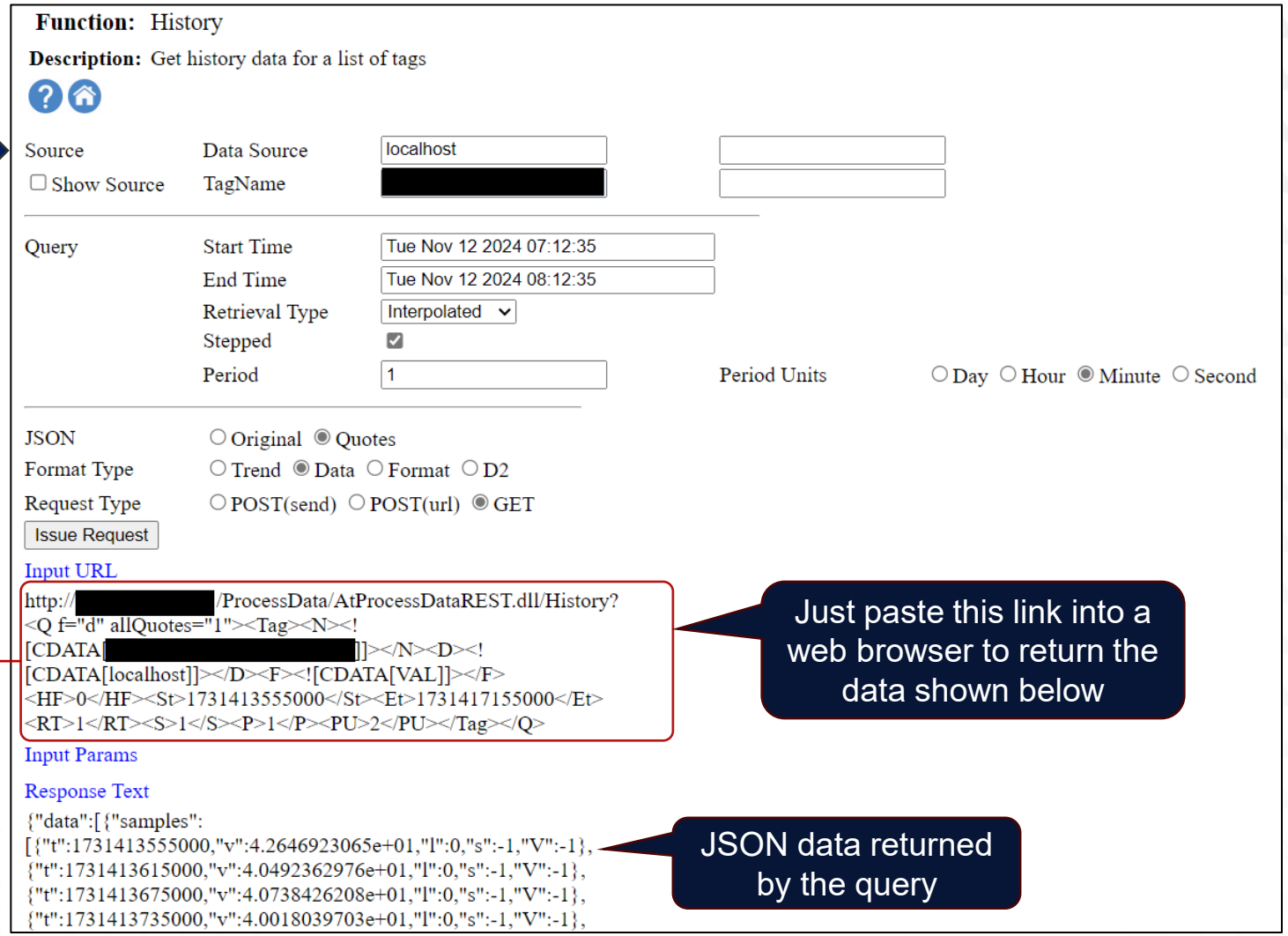


- REST started being talked about as early as 2018 by JMP associates. (see Bryan Boone's post below)
  - City Bikes Rest API - JMP User Community
  - In Nov 2022, Bill Worley was encouraging me to investigate REST API for querying Aspen.
- Bryan Boone & Michael Hecht, JMP, were added to the team in summer of 2023.
  - Multiple follow-up emails to sort through the details.
  - Final piece of the puzzle – 2023 Palm Springs Discovery Conference
  - *Converting from ODBC to REST API was a way to significantly decrease query times.*
  - **REST API also allowed for ALL JMP users to access the data**

REST API basics

- Thankfully, Aspen provides a webpage where you can create a query from a form, and it will generate the XML for you.
- Some key features include:
  - Queries made to the REST API must use datetimes that are calculated as a difference from the UNIX epoch date (01 Jan 1970 UTC) in **milliseconds**.
    - This implies having to deal with Daylight Savings Time when converting from local plant time to UTC.
  - Data returned in JSON format
  - The JSL command shown below which returns data in JSON format.

```
//REST API query using New HTTP Request
request = new http request(
    url( urltext ),
    method("GET"),
    Authentication Method("kerberos"),
    headers( {"Accept: application/json"}),
);
data = request << send("show progress both");
```



(http://localhost/ProcessData/samples/Sample_Home.html)

REST API basics – XML format

- Features of an XML query string:
  - You will need the URL for your aspenONE server.
    - Replace ‘localhostname’ with your host name
    - Replace ‘aspen_tagname1’ with your tag
  - Each line item or section is defined by an open <XXX> and a close </XXX>
  - Extra tags are simply appended to the query using the <Tag> … </Tag> syntax shown to the right.
  - Once you create the query string, place it into the url() argument of the New HTTP Request() function in JSL.

```
http://localhostname/ProcessData/AtProcessDataREST.dll/History?
    <Q f="d" allQuotes="1">
        <Tag>
            <N><![CDATA[aspen_tagname1]]></N>
            <D><![CDATA[localhostname]]></D>
            <F><![CDATA[VAL]]></F>
            <HF>0</HF>
            <St>1704528000000</St>
            <Et>1704614400000</Et>
            <RT>1</RT><S>1</S><P>1</P><PU>2</PU>
        </Tag>
        <Tag>
            <N><![CDATA[aspen_tagname2]]></N>
            <F><![CDATA[VAL]]></F>
            <HF>0</HF>
            <St>1704528000000</St>
            <Et>1704614400000</Et>
            <RT>1</RT><S>1</S><P>1</P><PU>2</PU>
        </Tag>
    </Q>
```

XML key:

<N>	= name
CDATA	= character data
<D>	= data source
<F>	= field
<HF>	= history format
<St>	= UTC start time
<Et>	= UTC end time
<RT>	= retrieval type
<S>	= stepped
<P>	= period size
<PU>	= period unit

Key JSL for REST API (new http request)

```
//creating the query string by inserting the text into the "\[...]\" construction
taglist         = Eval List({aa["AspenTag1"], aa["AspenTag2"], aa["AspenTag3"], aa["AspenTag4"]});
start_string    = "\[http://]\"
                        || aa["localhostname"]
                        || "\[/ProcessData/AtProcessDataREST.dll/History?<Q f="d" allQuotes="1">]\";
tag1            = "\[<Tag><N><![CDATA[]\" || taglist[1] || "\[]]></N>]\";
db_string       = "\[<D><![CDATA[]\"
                        || aa["localhostname"]
                        || "\[]]></D><F><![CDATA[VAL]]></F><HF>0</HF>]\";
time1           = "\[<St>]\" || char(aspen.start) || "\[</St>]\";
time2           = "\[<Et>]\" || char(aspen.end) || "\[</Et>]\";
freq_string     = "\[<RT>1</RT><S>1</S><P>1</P><PU>2</PU></Tag>]\";
end_string      = "\[</Q>]\";
tag2            = "\[<Tag><N><![CDATA[]\" || taglist[2] || "\[]]></N>]\";
tag3            = "\[<Tag><N><![CDATA[]\" || taglist[3] || "\[]]></N>]\";
tag4            = "\[<Tag><N><![CDATA[]\" || taglist[4] || "\[]]></N>]\";

querytext = start_string ||
                tag1 || db_string || time1 || time2 || freq_string ||
                tag2 || db_string || time1 || time2 || freq_string ||
                tag3 || db_string || time1 || time2 || freq_string ||
                tag4 || db_string || time1 || time2 || freq_string ||
            end_string;
urltext = substitute(querytext, " ", "%20");
//REST API query using New HTTP Request
request = new http request(
    url( urltext ),
    method("GET"),
    Authentication Method("kerberos"),
    headers( {"Accept: application/json"}),
);
data = request << send("show progress both");
//save text file("$Desktop\jsondata.json", data);
```

This JSL trick was so helpful

Save the JSON and use File > Open to get the JSON settings

```
//convert JSON data to a data table using the script
//from File > Open > Data (Using Preview)
    //data:     Col set as Expression
    //samples:  Row=check
    //t:        Col set as Numeric
    //v:        Col set as Numeric
dt = JSON To Data Table(
    data,
    JSON Settings(
        Stack( 0 ),
        Row( "/root/data/samples" ),
        Col(
            "/root/data",
            Column Name( "data" ),
            Fill( "Use Once" ),
            Type( "Expression" ),
            Format( {"Best"} ),
            Modeling Type( "Continuous" ),

        ),
        Col(
            "/root/data/samples/t",
            Column Name( "aspen time" ),
            Fill( "Use Once" ),
            Type( "Numeric" ),
            Format( {"Best"} ),
            Modeling Type( "Continuous" ),

        ),
        Col(
            "/root/data/samples/v",
            Column Name( "aspen value" ),
            Fill( "Use Once" ),
            Type( "Numeric" ),
            Format( {"Fixed Dec", 12, 1,
                use thousands separator( 1 )} ),
            Modeling Type( "Continuous" ),

        )
    )
);
```

Key JSL for multi http request

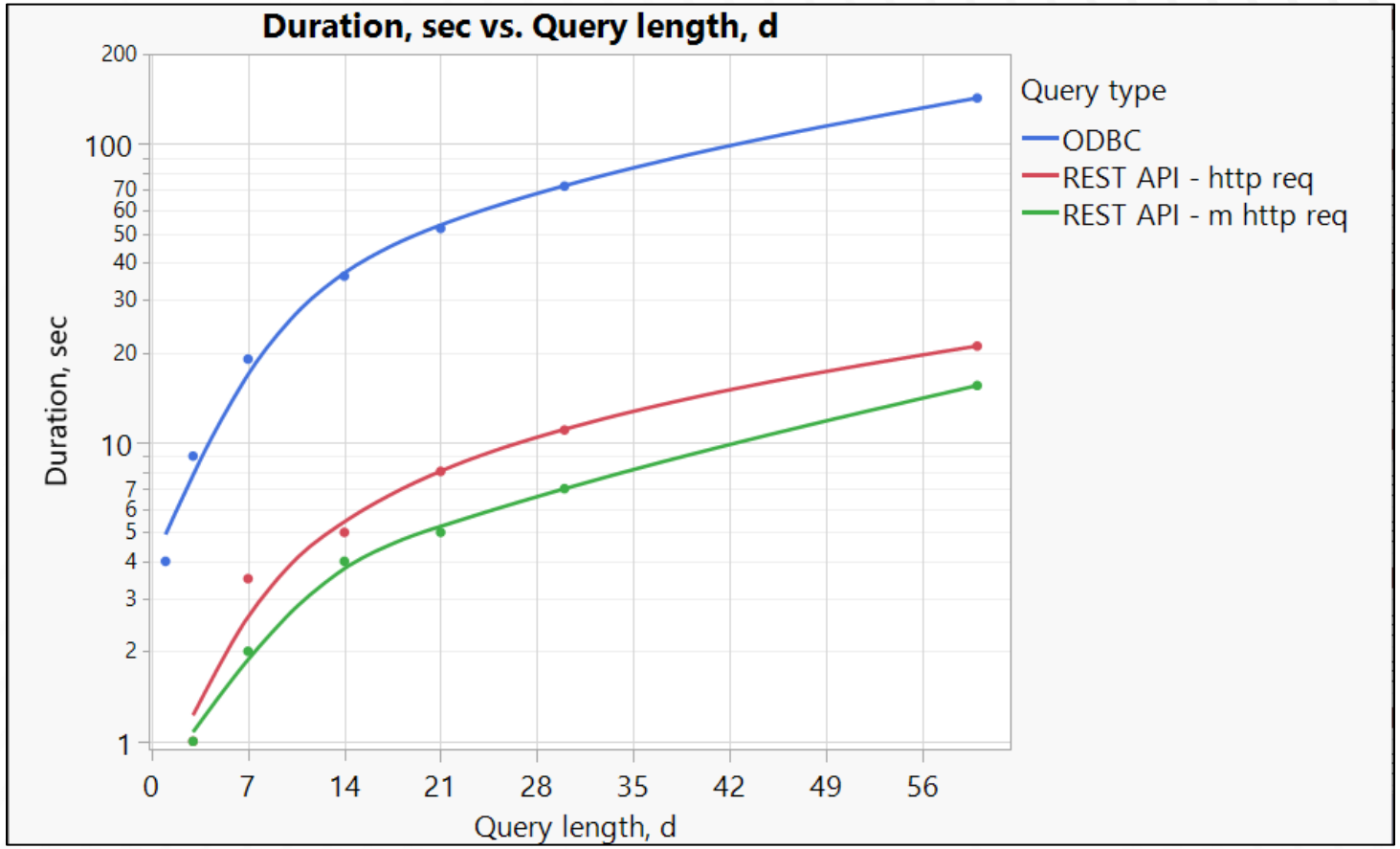
```
//creating the query string by inserting the text into the "\[...]\" construction
//for-loop to create a query and collect the requests
taglist         = Eval List({aa["AspenTag1"], aa["AspenTag2"]});
start_string    = "\[http://]\" || aa["localhostname"] ||
                        "\[/ProcessData/AtProcessDataREST.dll/History?<Q f="d" allQuotes="1">]\";
db_string       = "\[<D><![CDATA[]\" || aa["localhostname"] ||
                        "\[]]></D><F><![CDATA[VAL]]></F><HF>0</HF>]\";
time1           = "\[<St>]\" || char(aspen.start) || "\[</St>]\";
time2           = "\[<Et>]\" || char(aspen.end) || "\[</Et>]\";
freq_string     = "\[<RT>1</RT><S>1</S><P>1</P><PU>2</PU></Tag>]\";
end_string      = "\[</Q>]\";

n_tags = n items(taglist);
requests = new multi http request();
for(i=1, i<=n_tags, i++,
    tag         = "\[<Tag><N><![CDATA[]\" || taglist[i] || "\[]]></N>]\";
    querytext   = start_string || tag || db_string || time1 || time2 || freq_string || end_string;
    urltext     = substitute(querytext, " ", "%20");
    requests << Add(
        new http request(
            url( urltext ),
            Method( "GET" ),
            Authentication Method( "kerberos" ),
            headers( {"Accept: application/json"} ),
        );
    );
);
data = requests << send("show progress both");
```

Create a request for each tag

Multi-http request is needed for queries with > 10 tags because the max URL length is 2048 characters.

REST API vs. ODBC performance

- A query that pulls data for 7d at 1m intervals takes 3.5s using ‘new http request()’ vs. 20s for ODBC (~6x faster).
  - Note: All queries shown in the graph are for 2 aspen tags.
- Using ‘new multi http request()’ further improves performance.
  - ~10x faster than ODBC (e.g., a query of 30d takes 7s vs. 70s).



User improvement requests

After updating the ODBC add-in with the REST API functionality, **ALL** JMP users could now query the Aspen IP.21 historians.

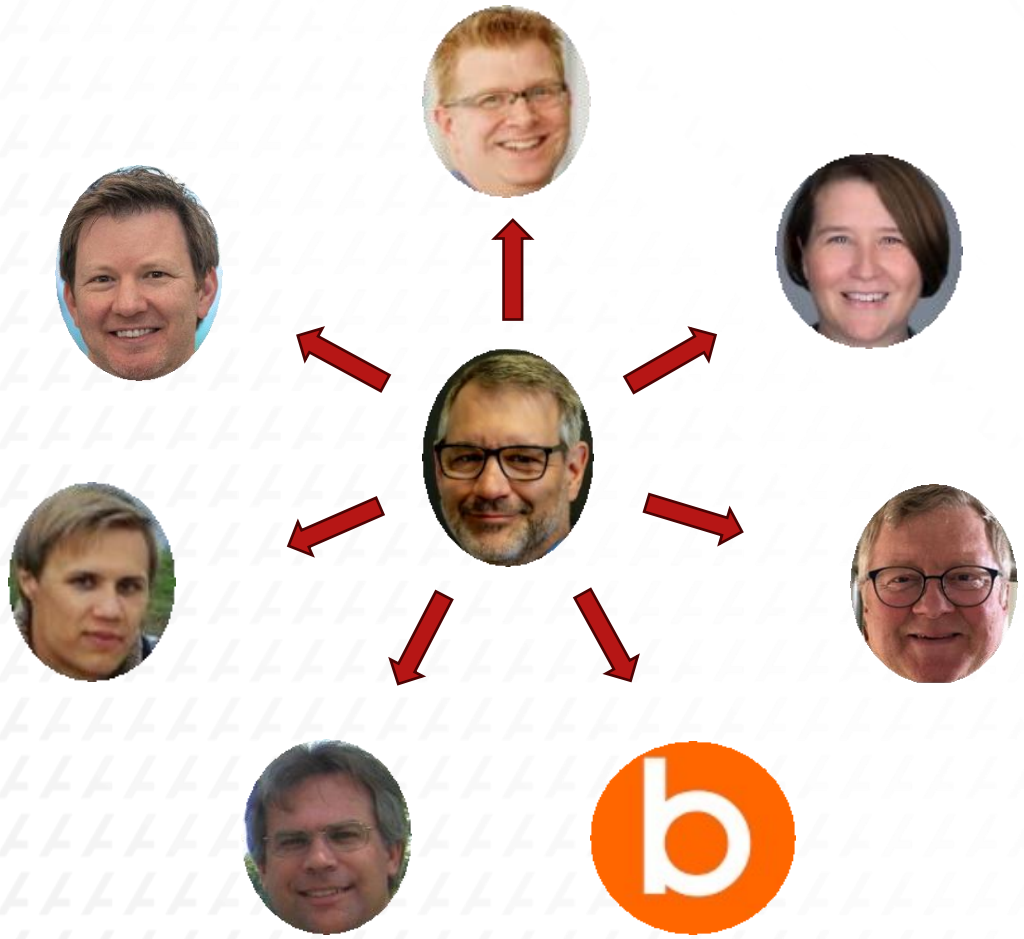
This generated multiple improvement requests. Some of the most important ones were:

1. Adding a method to directly ***query specific batches*** *without having to know how to use Aspen to collect start and end times*.
   - Development chemists and engineers wanted to ***query specific batches*** vs. process engineers that wanted to ***query specific time periods***.

2. Allowing the user to ***select multiple batches*** of the same product and return the stacked data table with important demographic data (Plant, Reactor, Batch number).

3. After selecting one batch or multiple batches, having the tool ***automatically choose the correct plant and reactor*** before selecting tags.

Connections…automation & sharing



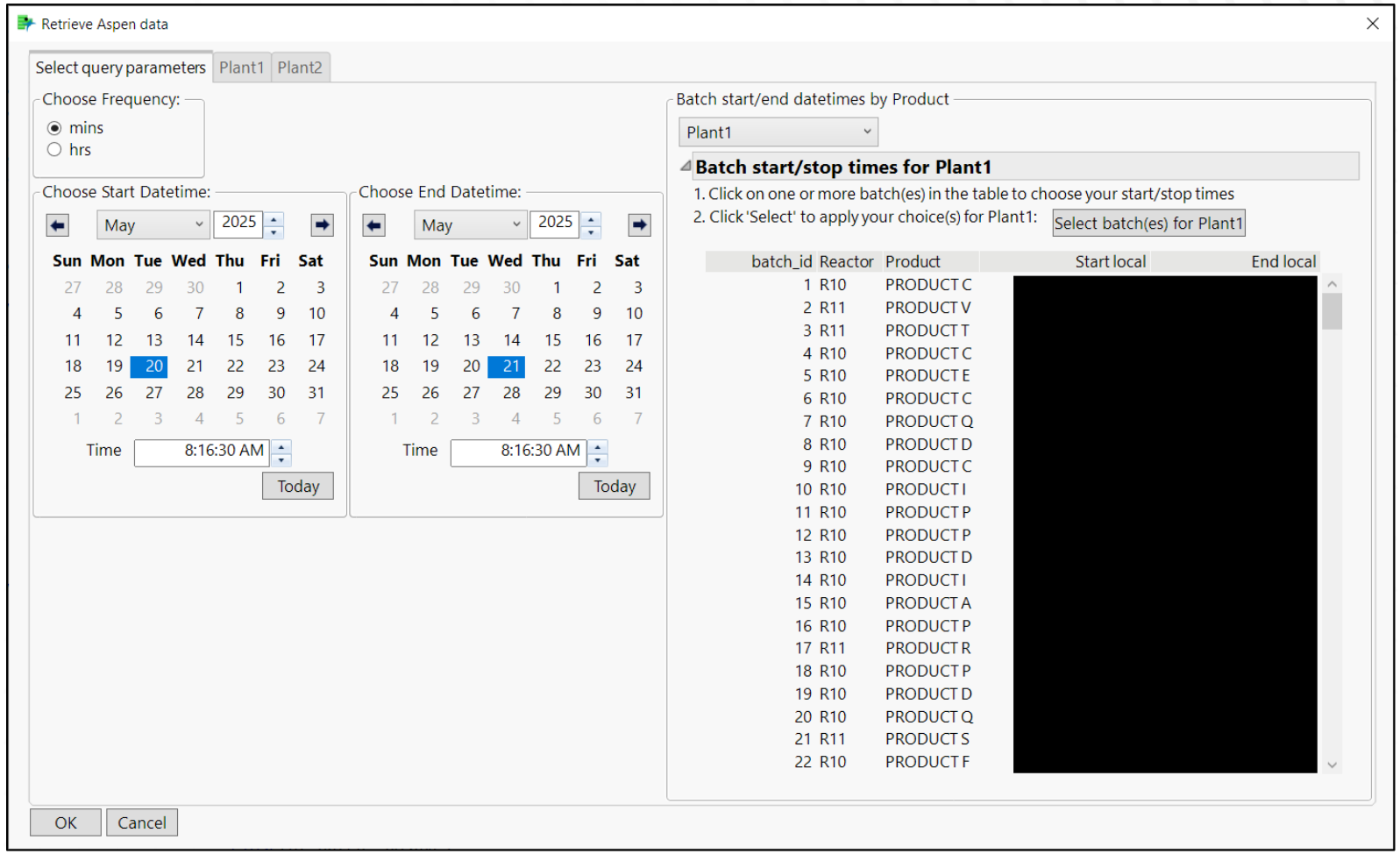
Creating and automating a user interface required multiple posts to the User Community to solve JSL questions to apply user improvement ideas.

- Jarmo Hirvonen, Murata Finland, helped create work-around because of a bug in JMP 17.1 that was fixed in JMP 18. (see Appendix for JSL using ‘Big Class.jmp’ as an example)
  - Solved: Re: How do I avoid an error when selecting a row from a filtered table box? - JMP User Community
- Scott Allen, JMP, helped automate the way the interface worked to make it easier for users to pull the correct data.
  - Solved: Use JSL to select a tab in a window - JMP User Community

Automation using REST API

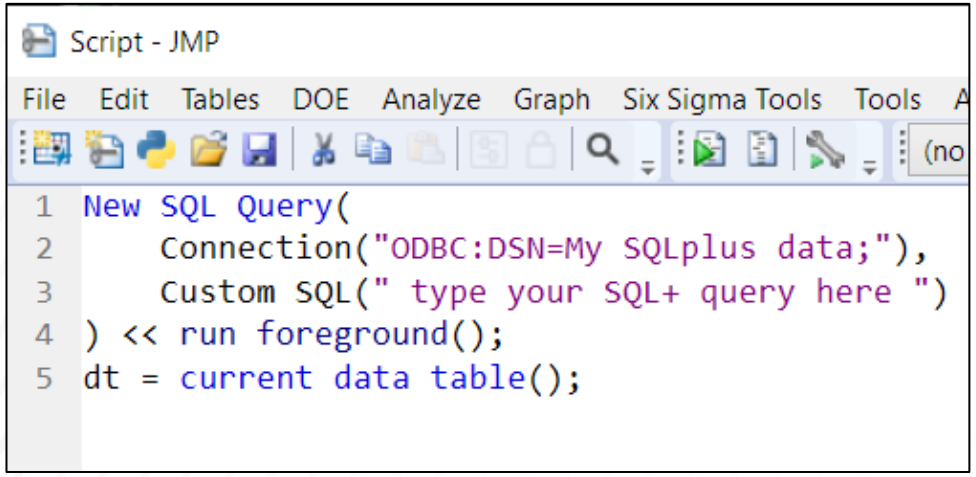
- You have the choice to run the add-in:
  - Manually (left side)
  - Selecting a product (right-side)

- The real-life add-in:
  - Incorporates all reactors and vessels at 4 different plants (~6,000 aspen tags).
  - Allows you to select your own tags at each plant
  - Automatically switches to the correct plant to select the tags



A summary of connecting to an Aspen IP.21 historian

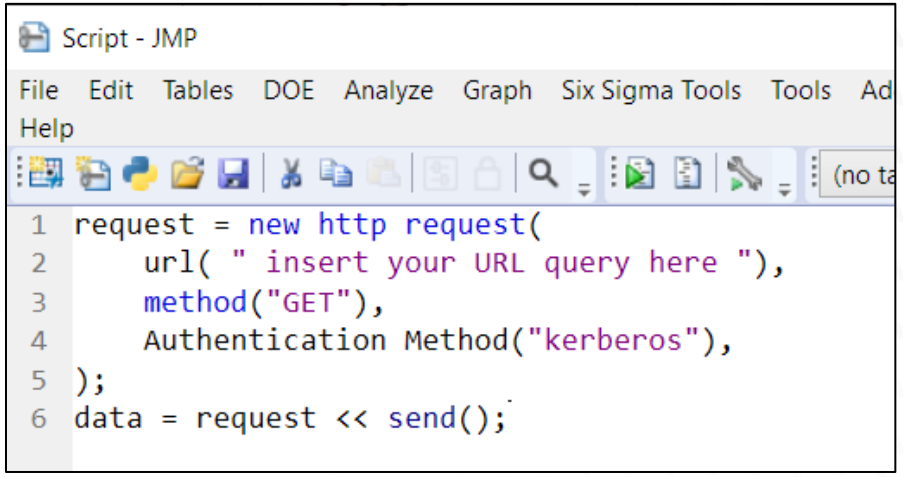
via ODBC:



You will need:

- Aspen installed on your computer.
- The 64-bit ODBC driver installed.
- To create the DSN for the ODBC connection.
- The names of your aspen tags.
- The structure of your SQL+ query.

via REST API:



You will need:

- The name of your aspenONE server
- The names of your aspen tags
- To create the URL query in JSL
  - Note: you can get the URL from your aspenONE sample query page (shown below) (http://localhost/ProcessData/samples/Sample_Home.html)

Final thoughts

- JMP provides multiple methods (ODBC, REST) to connect to the Aspen IP.21 historian.
  - In JMP 19, you will be able to query the IP.21 historian using JSL (New IP21 Client) or through File > Connect To > Historian > AspenTech IP.21 Server.

- Making this connection has reduced the total cycle time described in the “Business Problem” from many weeks, even months, to seconds.
  - Questions from chemists and engineers can now be answered in near real time BY the chemists and engineers themselves. It has become a revolutionary experience for them.
  - We have created a culture of “let’s find out by pulling some data…” (vs. the previous culture of “We think the process works this way…”)
  - REST API is significantly faster (10x) than ODBC **and** everyone with a JMP license can use it.

➢ **JMP not only provided the software, but also the environments (i.e., Discovery, local User Groups, User Community) to recruit a team of experts to help solve problems that seemed impossible (at first).**

Thank you

ALTA
Performance Materials

Appendix of JSL snippets

JSL to check for Daylight Savings Time

```
//daylight savings check of the start date for aspen
//this method works for dates since 2007
dstyear = year(jmp.start);
    // find day of the month for the 2nd Sun of March
    // using Day of Week function:  1 = Sun, 2 = Mon, ... , 7 = Sat
    mar.dst.date = match( day of week( date mdy(3, 1, dstyear) ),
        1, 8,   //if March 1st is on a Sun then 2nd Sun = 8th
        2, 14,  //if March 1st is on a Mon then 2nd Sun = 14th
        3, 13,  //if March 1st is on a Tue then 2nd Sun = 13th
        4, 12,  //if March 1st is on a Wed then 2nd Sun = 12th
        5, 11,  //if March 1st is on a Thu then 2nd Sun = 11th
        6, 10,  //if March 1st is on a Fri then 2nd Sun = 10th
        7, 9    //if March 1st is on a Sat then 2nd Sun = 9th
    );
    // find day of the month for the 1st Sun of Nov
    // using Day of Week function:  1 = Sun, 2 = Mon, ... , 7 = Sat
    nov.dst.date = match( day of week( date mdy(11, 1, dstyear) ),
        1, 1,   //if Nov 1st is on a Sun then 1st Sun = 1st
        2, 7,   //if Nov 1st is on a Mon then 1st Sun = 7th
        3, 6,   //if Nov 1st is on a Tue then 1st Sun = 6th
        4, 5,   //if Nov 1st is on a Wed then 1st Sun = 5th
        5, 4,   //if Nov 1st is on a Thu then 1st Sun = 4th
        6, 3,   //if Nov 1st is on a Fri then 1st Sun = 3rd
        7, 2    //if Nov 1st is on a Sat then 1st Sun = 2nd
    );
//check to see if the Start date of the query is between
//2nd Sun of March at 2am and the 1st Sun of Nov at 2am
if( date mdy(3, mar.dst.date, dstyear) + 2*60*60 <= jmp.start <= date mdy(11, nov.dst.date, dstyear) + 2*60*60,
    dstoffset = 1*60*60;
    ,
    dstoffset = 0;
);
```

JSL to convert to Aspen datetimes

```
//UNIX epoch time = reference time for REST API in Aspen is 1/1/1970 UTC in MILLISECONDS
dateref     = date mdy(1,1,1970);
//get the timezone for each plant from the associative array
if( plant == "Plant1",
        timezone = num(aa["localtimezone"])*60*60;
    ,
    plant == "Plant2",
        timezone = num(aa["localtimezone2"])*60*60;
);

//convert JMP start time to Aspen time based upon milliseconds from UNIX epoch time
aspen.start = (jmp.start - dateref + timezone - dstoffset)*1000;
//convert JMP end time to Aspen time based upon milliseconds from UNIX epoch time
aspen.end   = (jmp.end - dateref + timezone - dstoffset)*1000;

start_string    = "\[http://]\" || localhost || "\[/ProcessData/AtProcessDataREST.dll/History?<Q f="d" allQuotes="1">]\";
db_string       = "\[<D><![CDATA[]\" || localhost || "\[]]></D><F><![CDATA[VAL]]></F><HF>0</HF>]\";
time1           = "\[<St>]\" || char(aspen.start) || "\[</St>]\";
time2           = "\[<Et>]\" || char(aspen.end) || "\[</Et>]\";
end_string      = "\[</Q>]\";
```

JSL for Multiple row selection in a table box (using Big Class)

```
Names Default To Here(1);

//open Big Class.jmp
dt = Open("$sample_data\big class.jmp", invisible);

//select the first 10 rows and make a report
rpt = dt << select where(Row() <= 10) << get as report;

//from Scripting Index for "get row change function"
tb = rpt[Table Box(1)];
tb << Set Selectable Rows(1);
tb << set row change function( Function( {this}, sel = this << get selected rows ) );
sel = tb << get row change function;

//create a user interface with the JMP table embedded in it
nw = New Window("test",
    V List Box(
        Outline Box("Big Class", rpt), //put the table box inside of an Outline box
        Text Box("Click 'OK' to choose height and weight"),
        Button Box("OK",
            hvals   = tb[Number Col Box("height")] << get;
            wvals   = tb[Number Col Box("weight")] << get;
            ht      = hvals[sel];   //get the height values of the selected rows
            wt      = wvals[sel];   //get the weight values of the selected rows
            Print(sel, ht, wt);     //print the selected rows, heights and weights in the log
        )
    )
);
close(dt, nosave);
```

JSL for selecting multiple batches in the Plant dropdown

```
V List Box(
    Panel Box("Batch start/end datetimes by Product",
        vlb.help = V List Box(
            cb.plant = Combo Box({"Click to choose a plant", "Plant1", "Plant2"},
                plant_selection = cb.plant << get selected();
                if(plant_selection == "Plant1",
                        try(remove from(vlb.help,2));       //remove previous selection, if it exists
                        try(close(dt.help, nosave));        //delete previous table, if it was opened
                        dt.batch = open(localpath||"Plant batch info.jmp", invisible); //open batch history table for Plant1
                        dt.help = dt.batch << subset(columns(:batch_id, :reactor, :product, :start local, :end local), invisible);
                        dt.help:start local << format("Format Pattern", "<MM>/<DD>/<YYYY> <hh24><:><mm>", 19);
                        dt.help:end local << format("Format Pattern", "<MM>/<DD>/<YYYY> <hh24><:><mm>", 19);
                        close(dt.batch, nosave);
                        rpt.plant = dt.help << get as report;
                        tb.plant = rpt.plant[Table Box(1)];
                        tb.plant << set selectable rows(1);
                        vlb.help << append(
                            Outline box("Batch start/stop times for Plant1",
                                vlb.batch = v list box(
                                    text box("1. Click on one or more batch(es) in the table to choose your start/stop times", set width(500)),
                                    v list box(
                                        h list box(
                                            text box("2. Click 'Select' to apply your choice(s) for Plant1:   "),
                                            button box("Select batch(es) for Plant1",
                                                all.start.plant = rpt.plant[NumberColBox("Start local")] << get;    //retrieves all of the start dates
                                                all.end.plant   = rpt.plant[NumberColBox("End local")] << get;      //retrieves all of the end dates
                                                all.prod.plant  = rpt.plant[StringColBox("Product")] << get;        //retrieves all of the products
                                                all.batch.plant = rpt.plant[StringColBox("batch_id")] << get;       //retrieves all batch numbers
                                                all.rxr.plant   = rpt.plant[StringColBox("Reactor")] << get;        //retrieves all reactor numbers
                                                sel.plant = tb.plant <<  get selected rows;                         //get selected row
                                                sel.start.plant = all.start.plant[sel.plant];                       //chooses the selected start date
                                                sel.end.plant   = all.end.plant[sel.plant];                         //chooses the selected end date
                                                sel.prod.plant  = all.prod.plant[sel.plant];                        //chooses the Products
                                                sel.batch.plant = all.batch.plant[sel.plant];                       //chooses the selected batch no
                                                sel.rxr.plant   = all.rxr.plant[sel.plant];                         //chooses the selected reactor
                                                if( sel.rxr.plant[1]=="R10",
                                                        tbox << set selected(2); tbox.plant << set selected(1); ,   //chooses Plant1 and R10 tabs
                                                    sel.rxr.plant[1]=="R11",
                                                        tbox << set selected(2); tbox.plant << set selected(2);     //chooses Plant1 and R11 tabs
                                                );
                                            ),
                                        ),
                                        rpt.plant
                                    ),
                                ),
                            ),
                        );
                    ,
```