EUROPE DISCOVERY SUMMIT

Mastering the Art of Scripting Seamless JMP-Like User Interfaces with Structured JSL

Jarmo Hirvonen (jthi)
Murata Electronics

Only JMP's $SAMPLE_DATA or simulated data is used in this poster

muRata INNOVATOR IN ELECTRONICS

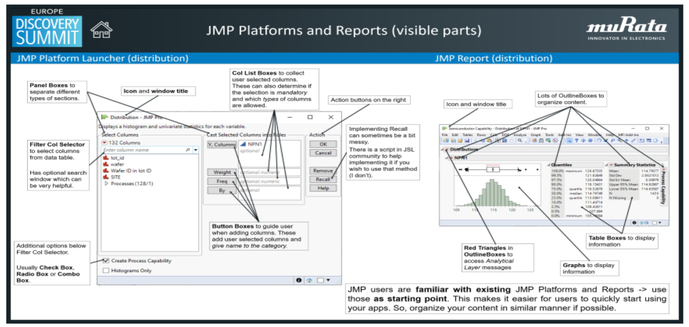
Abstract

I create a lot of user-interfaces using JSL, some simple and some much more complicated, but most do follow similar principles: the scripts use similar templates, they have similar layout on the launcher, reports look similar, etc.

In this presentation, I show how I use JSL to script JMP-like user interfaces, which type of templates I tend to use, and the decisions I make while developing such platforms. I also demonstrate different display boxes, how to create a user interface with JSL, script templates, the decision process I use when scripting user interfaces, and some tips and best practices for JMP scripting.

Mimic JMP Platforms

JMP users are **familiar with existing** JMP Platforms and Reports -> use those **as starting point**. This makes it easier for users to quickly start using your apps. So, organize your content in similar manner if possible.



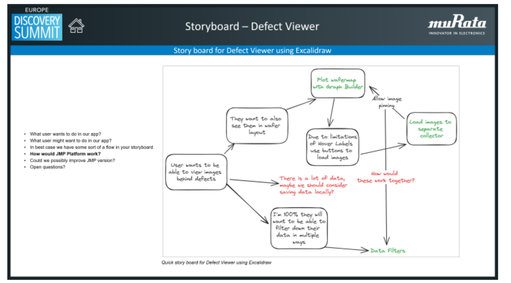
JMP Platforms and Reports (click to zoom)

Create storyboards and mockups

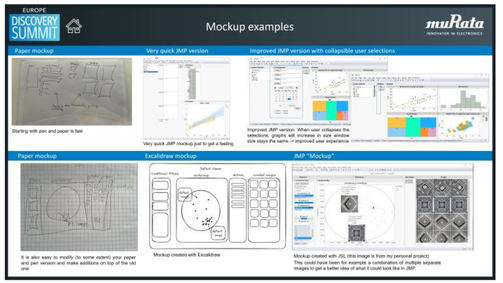
Think how user will be using your app (think about how JMP platform would work).

Create some quick mockups of your upcoming apps and show them to users in hopes to get some early feedback. In best cases you can also ask some initial mockups from your users.

I create my mockups using varying tools: pen and paper, PowerPoint, Paint, JSL, Excalidraw, drawio,…

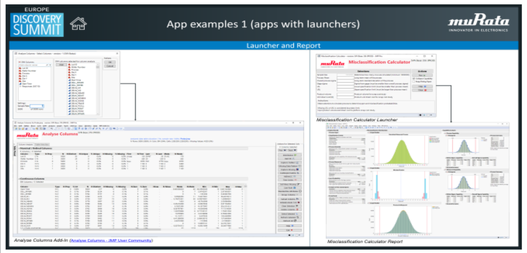


Storyboard (click to zoom)

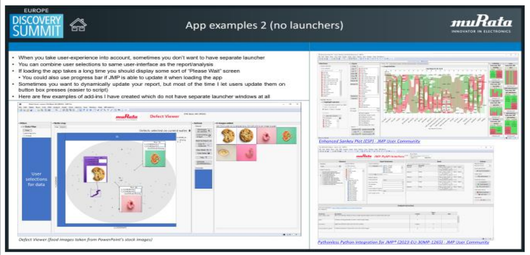


Mockup examples (click to zoom)

Consider templates and user experience



app Examples 1 (click to zoom)



app Examples 2 (click to zoom)

Have a template and scripting style. Makes creating apps faster, more robust and they look more professional.

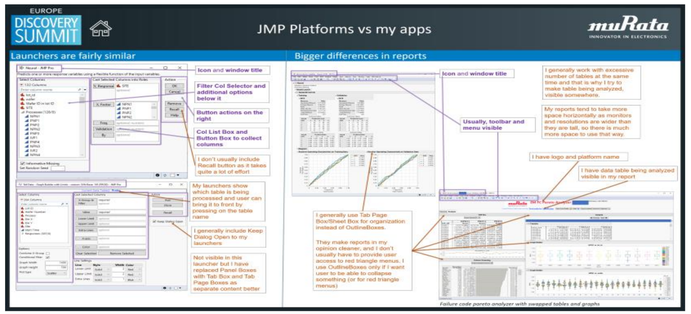
Take also the **user experience** (UX) into account. Few examples of **bad user experience** are platforms which uses **modal windows** when they shouldn't OR if the **window resizes** when user performs actions in the window.

Try to keep **Toolbars and Menus visible on reports**. Makes it much easier for user to launch different platforms directly from your app

Don't design your UI for larger resolutions than 1920x1080 with 100% scaling, as most of the users are most likely still using that.

JMP Platforms vs my apps

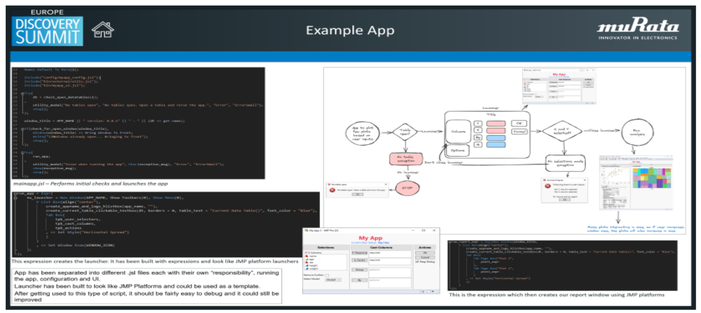
Generally, my applications do look like JMP platforms (especially the launchers). Column selectors on the left, options below them, columns are cast to middle and action buttons on the right. I do have few extra things (table name and title) which JMP platforms do not. Reports usually are very different as I don't use OutlineBoxes unless I want user to be able to collapse something.



JMP Platforms vs my apps (click to zoom)

Example App

I usually separate my apps into separate script files. Each script file tries to have its own responsibilities. This (in my opinion) makes it easier to manage bigger apps. I also have general utility functions which I have in most of my applications such as custom modal windows, checking if there are any tables, if platform is already running and so on…



Example App (click to zoom)

JMP Platforms and Reports (visible parts)

JMP Platform Launcher (distribution)

Panel Boxes to separate different types of sections.

Icon and **window title**

Col List Boxes to collect user selected columns. These can also determine if the selection is *mandatory* and which *types* of columns are allowed.

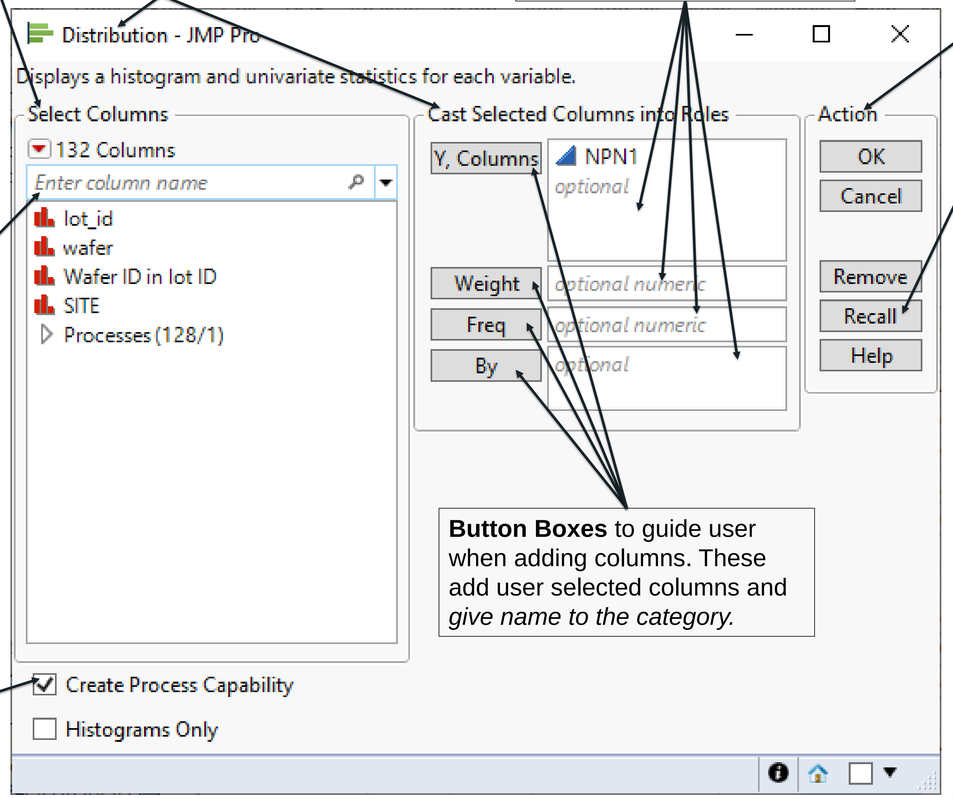
Action buttons on the right

Filter Col Selector to select columns from data table.

Has optional search window which can be very helpful.

Implementing Recall can sometimes be a bit messy.

There is a script in JSL community to help implementing it if you wish to use that method (I don't).



Button Boxes to guide user when adding columns. These add user selected columns and *give name to the category.*

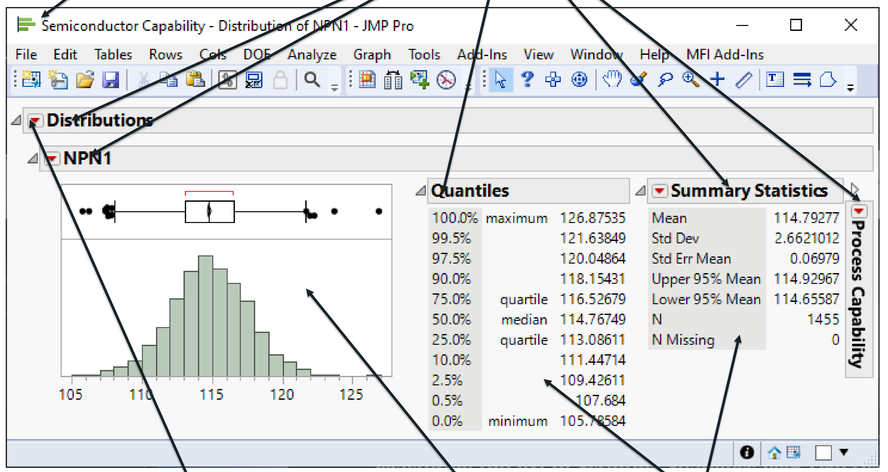
Additional options below Filter Col Selector.

Usually **Check Box**, **Radio Box** or **Combo Box**.

JMP Report (distribution)

Icon and window title

Lots of OutlineBoxes to organize content.



Red Triangles in **OutlineBoxes** to access *Analytical Layer* messages

Table Boxes to display information

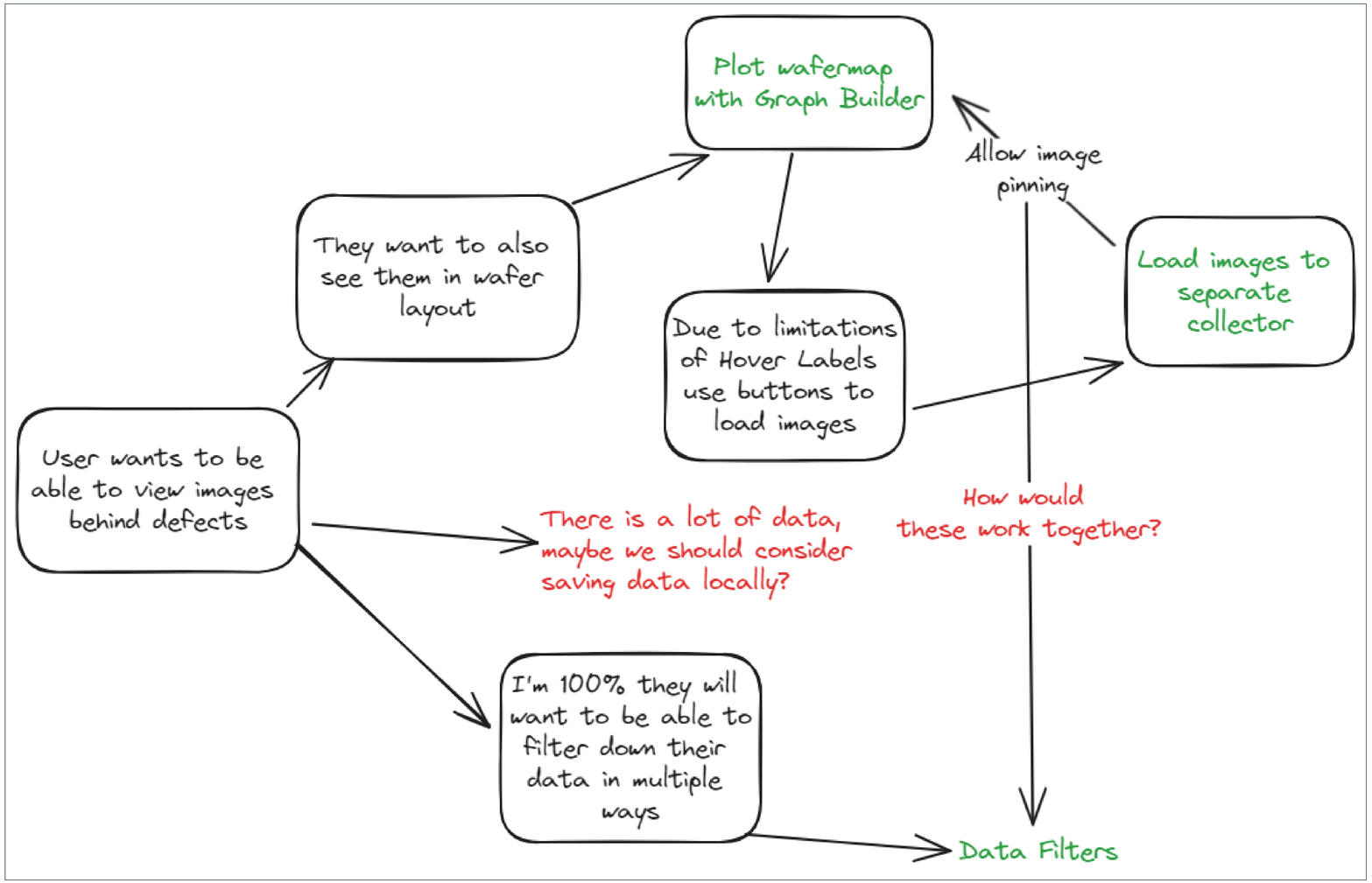
Graphs to display information

JMP users are **familiar with existing** JMP Platforms and Reports -> use those **as starting point**. This makes it easier for users to quickly start using your apps. So, organize your content in similar manner if possible.

Storyboard – Defect Viewer

Story board for Defect Viewer using Excalidraw

- What user wants to do in our app?
- What user might want to do in our app?
- In best case we have some sort of a flow in your storyboard
- **How would JMP Platform work?**
- Could we possibly improve JMP version?
- Open questions?



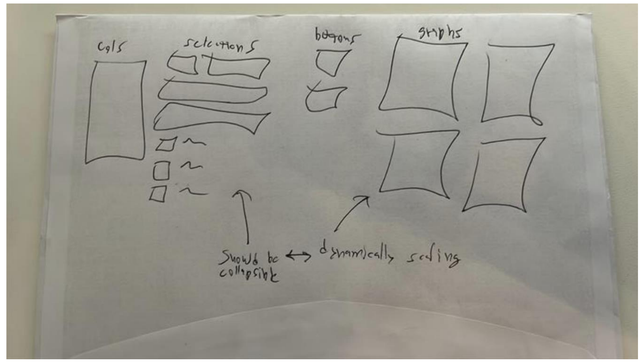
Quick story board for Defect Viewer using Excalidraw

EUROPE DISCOVERY SUMMIT

Mockup examples

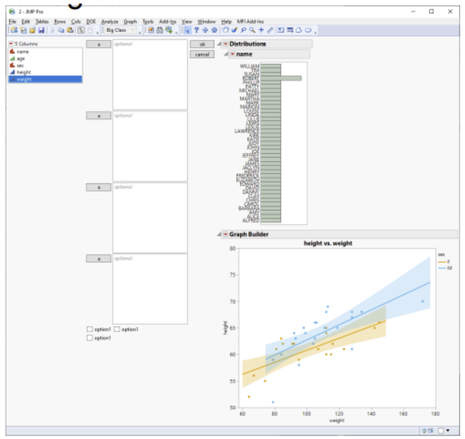
muRata INNOVATOR IN ELECTRONICS

Paper mockup



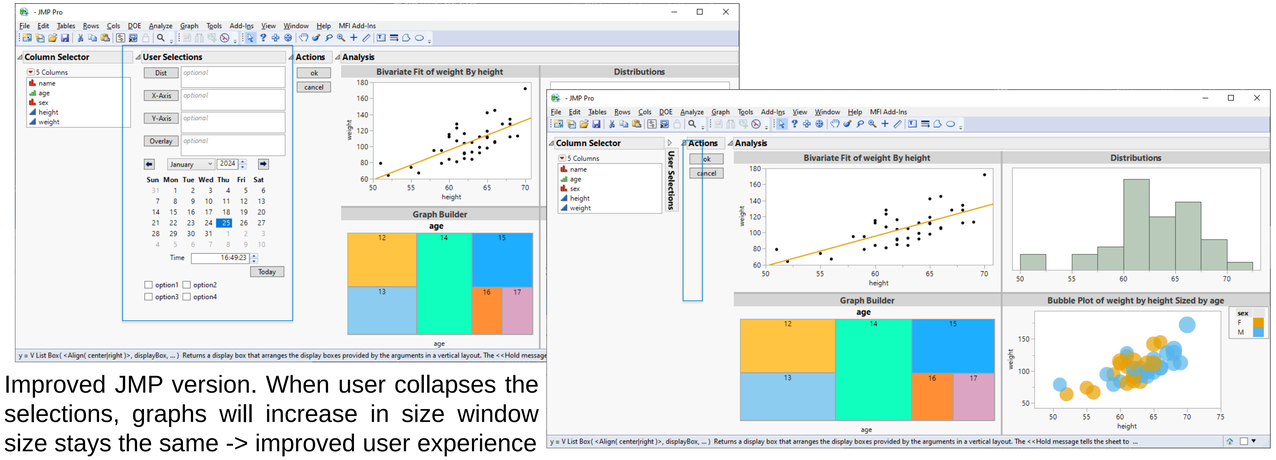
Starting with pen and paper is fast

Very quick JMP version



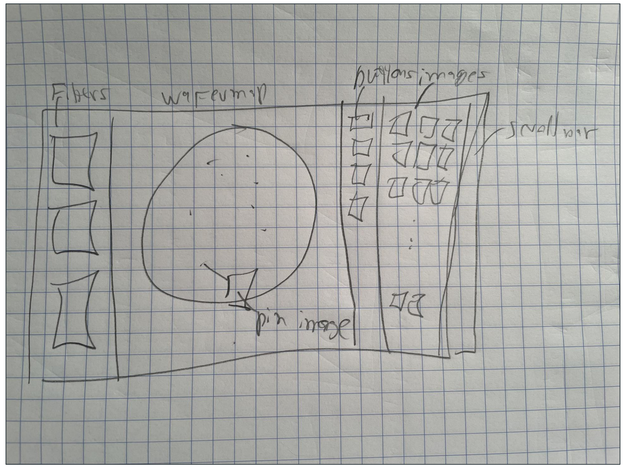
Very quick JMP mockup just to get a feeling

Improved JMP version with collapsible user selections



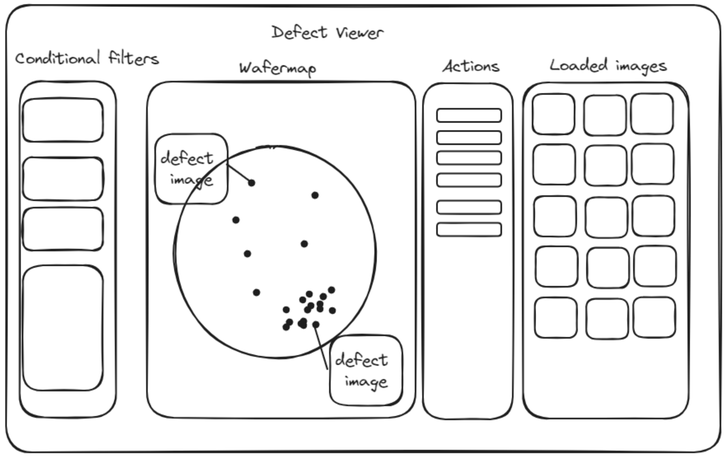
Improved JMP version. When user collapses the selections, graphs will increase in size window size stays the same -> improved user experience

Paper mockup



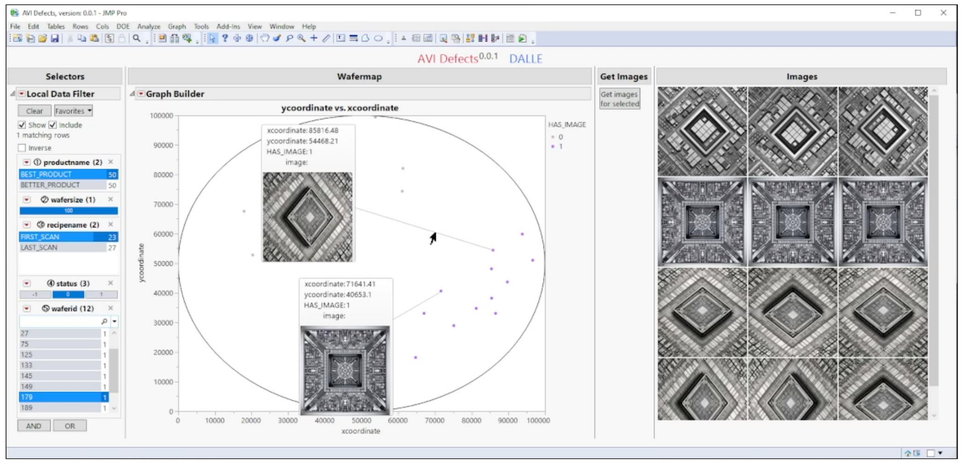
It is also easy to modify (to some extent) your paper and pen version and make additions on top of the old one

Excalidraw mockup



Mockup created with Excalidraw

JMP "Mockup"

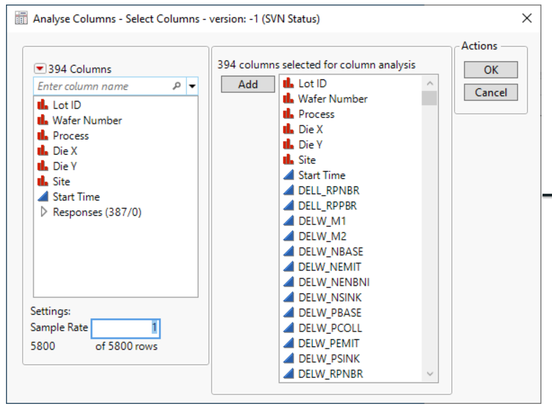


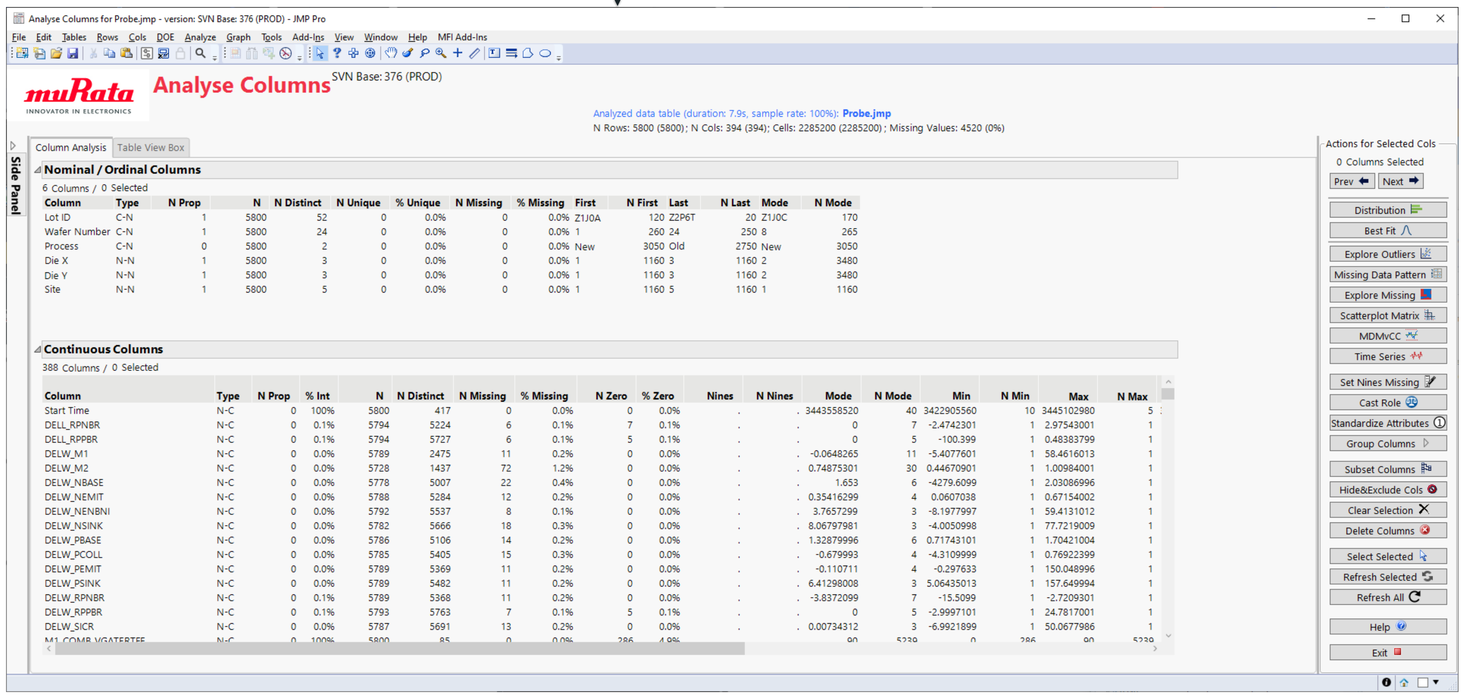
Mockup created with JSL (this image is from my personal project)

This could have been for example a combination of multiple separate images to get a better idea of what it could look like in JMP.

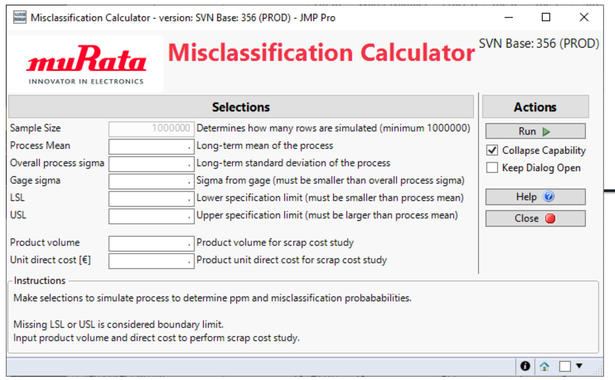
App examples 1 (apps with launchers)

Launcher and Report

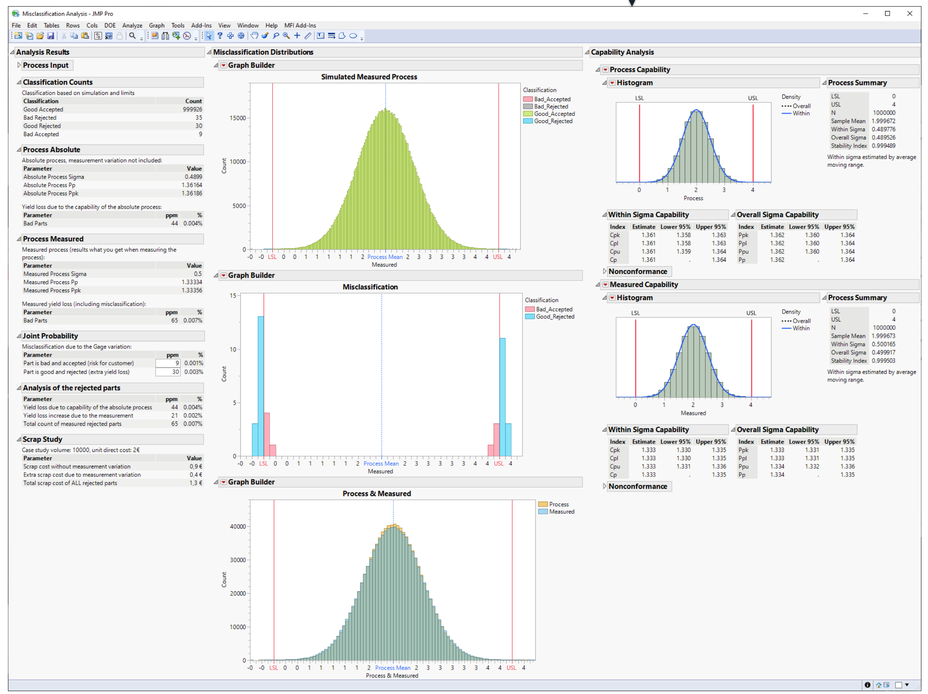




Analyse Columns Add-In (Analyse Columns - JMP User Community)



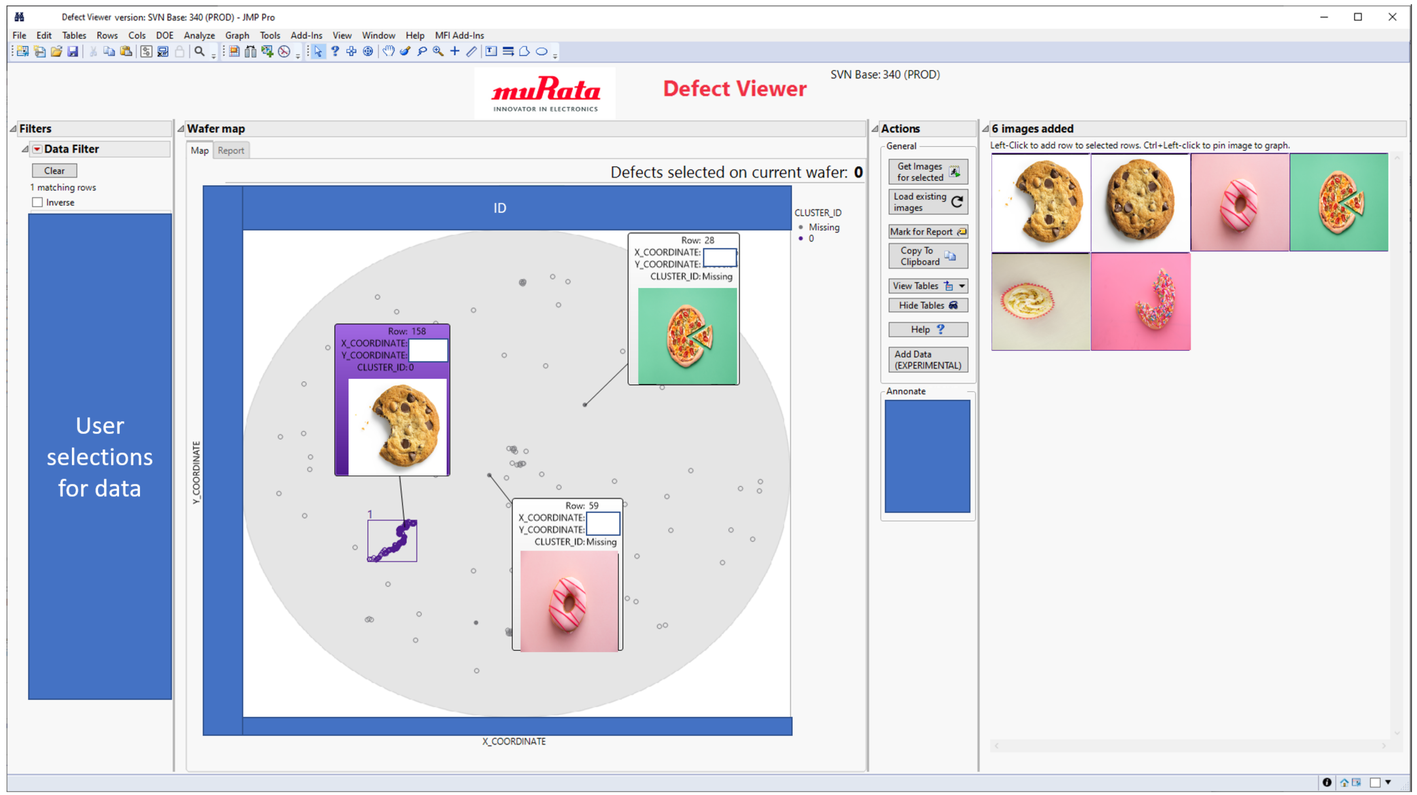
Misclassification Calculator Launcher



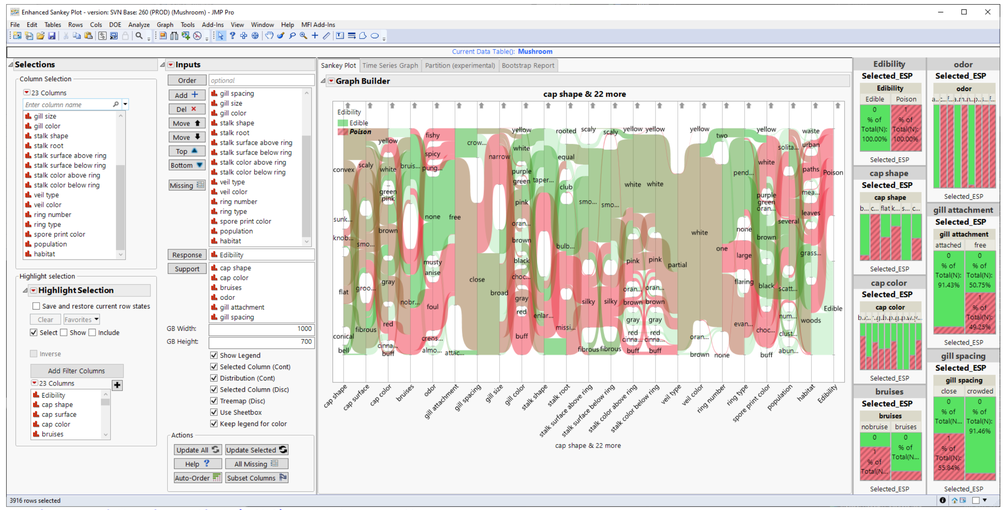
Misclassification Calculator Report

App examples 2 (no launchers)

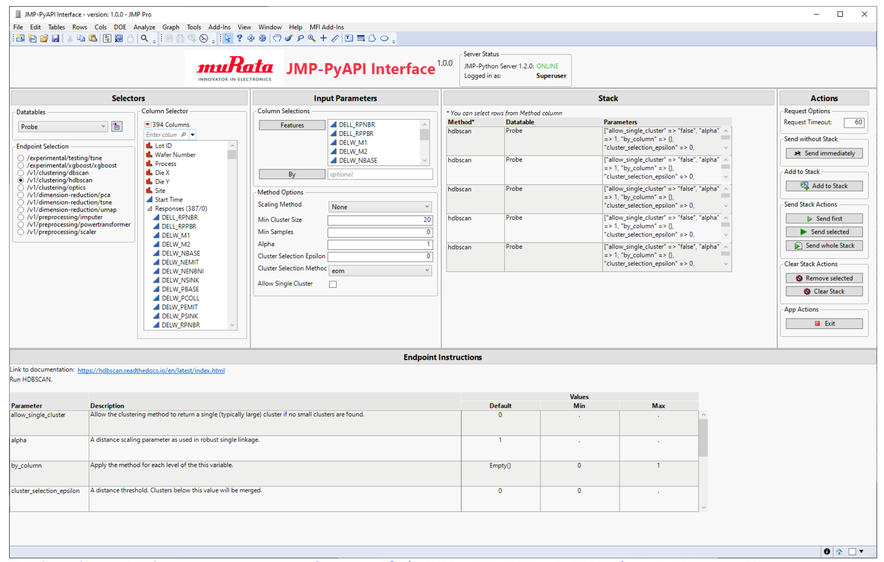
- When you take user-experience into account, sometimes you don't want to have separate launcher
- You can combine user selections to same user-interface as the report/analysis
- If loading the app takes a long time you should display some sort of "Please Wait" screen
  - You could also use progress bar if JMP is able to update it when loading the app
- Sometimes you want to dynamically update your report, but most of the time I let users update them on button box presses (easier to script)
- Here are few examples of add-ins I have created which do not have separate launcher windows at all



Defect Viewer (food images taken from PowerPoint's stock images)



[Enhanced Sankey Plot (ESP) - JMP User Community]



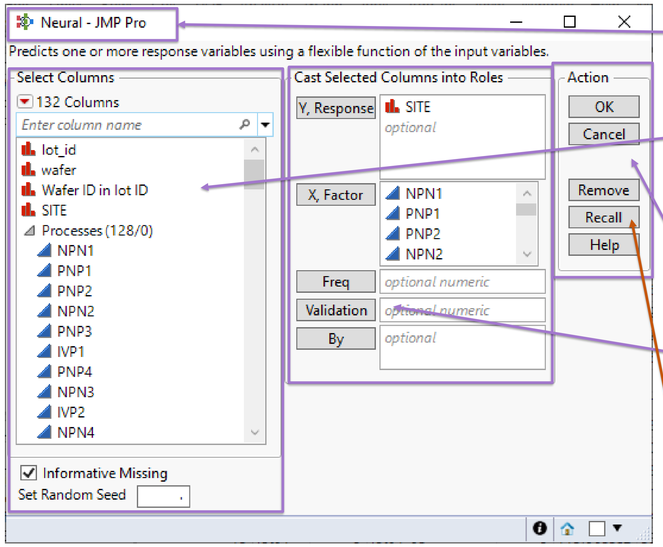
[Pythonless Python Integration for JMP® (2023-EU-30MP-1265) - JMP User Community]

EUROPE DISCOVERY SUMMIT

JMP Platforms vs my apps

muRata INNOVATOR IN ELECTRONICS

Launchers are fairly similar



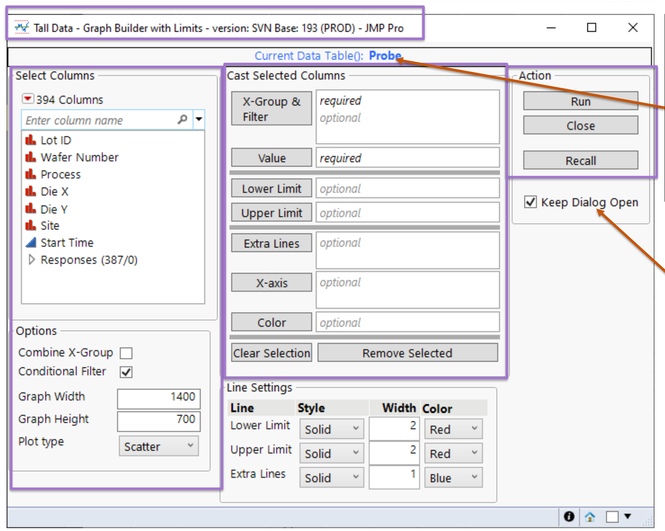
Icon and **window title**

Filter Col Selector and additional options below it

Button actions on the right

Col List Box and Button Box to collect columns

I don't usually include Recall button as it takes quite a lot of effort

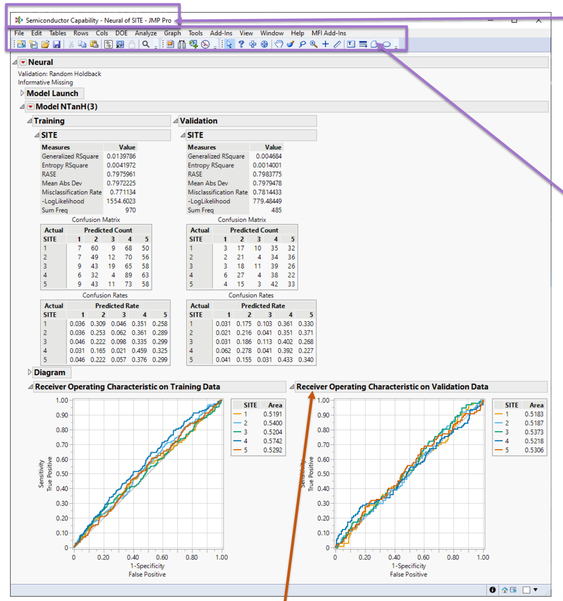


My launchers show which table is being processed and user can bring it to front by pressing on the table name

I generally include Keep Dialog Open to my launchers

Not visible in this launcher but I have replaced Panel Boxes with Tab Box and Tab Page Boxes as separate content better

Bigger differences in reports



Icon and **window title**

Usually, toolbar and menu visible

I generally work with excessive number of tables at the same time and that is why I try to make table being analyzed, visible somewhere.

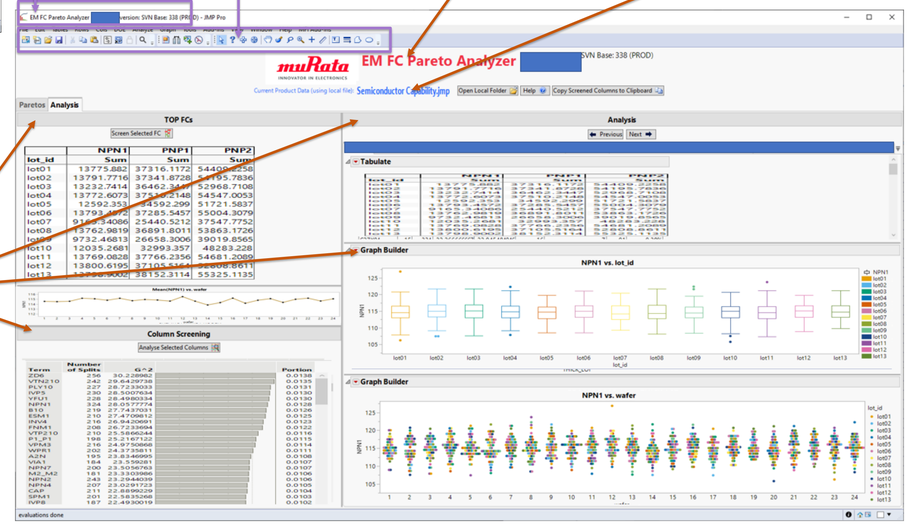
My reports tend to take more space horizontally as monitors and resolutions are wider than they are tall, so there is much more space to use that way.

I have logo and platform name

I have data table being analyzed visible in my report

I generally use Tab Page Box/Sheet Box for organization instead of OutlineBoxes.

They make reports in my opinion cleaner, and I don't usually have to provide user access to red triangle menus. I use OutlineBoxes only if I want user to be able to collapse something (or for red triangle menus)



Failure code pareto analyzer with swapped tables and graphs

Example App


```
Names Default To Here(1);

Include("config/myapp_config.jsl");
Include("bin/external/utils.jsl");
Include("bin/myapp_ui.jsl");

Try(
    dt = check_open_datatables(1);
,
    utility_modal("No tables open", "No tables open. Open a table and rerun the app.", "Error", "ErrorSmall");
    stop();
);

window_title = APP_NAME || " version: 0.0.1" || " - " || (dt << get name);

If(check_for_open_window(window_title),
    Window(window_title) << Bring Window To Front;
    Write("\!NWindow already open... Bringing to front");
    stop();
);

Try(
    run_app;
,
    utility_modal("Issue when running the app", Char(exception_msg), "Error", "ErrorSmall");
    show(exception_msg);
    stop();
);
```

mainapp.jsl – Performs initial checks and launches the app

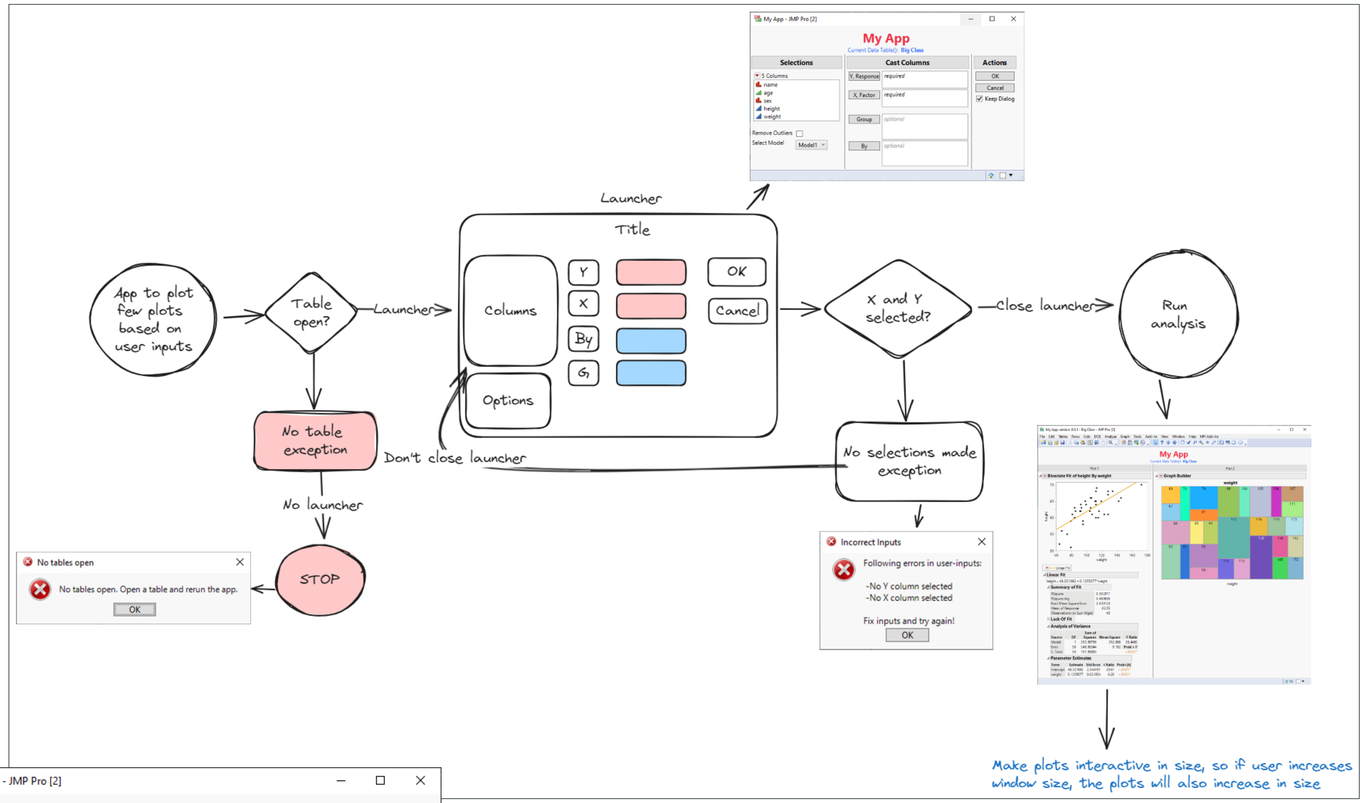
```
run_app = Expr(
    nw_launcher = New Window(APP_NAME, Show Toolbars(0), Show Menu(0),
        V List Box(align("center"),
            create_appname_and_logo_hlistbox(app_name, ""),
            create_current_table_clickable_textbox(dt, borders = 0, table_text = "Current Data Table()", font_color = "Blue"),
            Tab Box(
                tpb_user_selectors,
                tpb_cast_columns,
                tpb_actions
            , << Set Style("Horizontal Spread")
            )
        )
        , << Set Window Icon(WINDOW_ICON)
    );
);
```

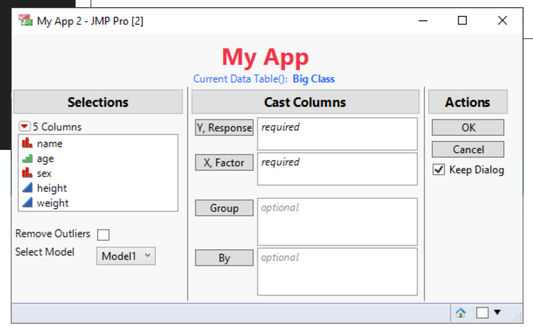
This expression creates the launcher. It has been built with expressions and look like JMP platform launchers

App has been separated into different .jsl files each with their own “responsibility”, running the app, configuration and UI.

Launcher has been built to look like JMP Platforms and could be used as a template.

After getting used to this type of script, it should be fairly easy to debug and it could still be improved






```
run_report_expr = Expr(New Window(window_title,
    V List Box(align("center"),
        create_appname_and_logo_hlistbox(app_name, ""),
        create_current_table_clickable_textbox(dt, borders = 0, table_text = "Current Data Table()", font_color = "Blue"),
        Tab Box(
            Tab Page Box("Plot 1",
                plot1_expr
            ),
            Tab Page Box("Plot 2",
                plot2_expr
            )
        , << Set Style("Horizontal Spread")
        );
    )
));
```

This is the expression which then creates our report window using JMP platforms

EUROPE DISCOVERY SUMMIT

Mastering the Art of Scripting Seamless JMP-Like User Interfaces with Structured JSL

Summary

muRata INNOVATOR IN ELECTRONICS

Summary for this poster

- **Mimic JMP** Platform launchers and reports as users are familiar with them
- Consider how user will use your app (**storyboard**)
- Create **mockups**
- Create a **template** to follow to make robust and professional looking apps
- Get familiar with common display boxes for content organization and for collecting user input
- JMP is very interactive tool, so do not forget about **user experience**
  - Avoid modal windows – let user move around different windows freely
  - Do not hide menus and toolbars from reports as user might want to launch other platforms directly from your report
  - Try to make your reports have static size after it has been loaded as report window which size changes constantly is annoying to work with (this is in my opinion a thing which is present in many JMP reports and it reduces usability)
- Think how user could easily rerun your platform
  - Implementing generic Recall can be very difficult
  - Saving settings is other option
  - Good initialization values also can help with this

General (scripting) tips

- Limit your variable scope
  - Usually starting scripts with **Names Default To Here(1);** is enough
  - User defined windows do have their own scope which you can use (window:), which can be very useful
- Don't leave Clear Symbols(); Delete Symbols(); Clear Globals(); Delete Globals() to your scripts as these can break other scripts in same JMP session
- Do not modify user preferences UNLESS it is the only purpose of your script
  - You might think that you can return user's settings when script execution ends, but I'm 100% sure you cannot do this always, so don't mess with them
  - If your script requires you to modify preferences (changing separator for csv for example), figure other way to handle the issue you are having
- Choose a style and do not change it in a project
  - Your style will develop and change, but try to keep it same for one project
- Do not forget the X in UI (UI and UX) – User Interface and User Experience
- Use version control
- Create code which can be reused (functions and sometimes even Custom Functions)
- If your script creates data tables, consider closing them when user closes your app
  - Other option could be using << Set Dirty(0) so user can close the window quicker

Additional reading

- A Structured Approach to Programming With JSL for Building Real World, Scalable ... - JMP User Community
- Recall Function Library - JMP User Community
- Progress Bar with Cancel Button - JMP User Community
- Solved: Re: Progress bars – my personal journey (also questions about manipulating windo... - JMP User Community
- Construct Custom Windows (jmp.com)
- JMP Scripters Club - JMP User Community
- Most Common JSL Mistakes and How to Avoid Them (2020-US-30MP-571) - JMP User Community

Some of my JMP Add-Ins you can find from JMP Community

- JMP-Tools Add-in overview - JMP addins written by jthi - JMP User Community
- Analyse Columns - JMP User Community
- Enhanced Sankey Plot (ESP) - JMP User Community
- Split and Set Spec Limits - JMP User Community