A close-up photograph of a hand reaching towards a mushroom growing on a mossy surface. The hand is positioned at the top right, with fingers slightly curled. The mushroom is in the foreground, and the moss is a vibrant green. The background is blurred, showing more of the mossy ground.

Y'All

**An automated JMP AddIn
toolkit for sharing, updating,
and maintaining scripts**

Michael Akerman and Johan Olsen

October 5, 2021

Agenda

- Who we are
- Why we built Y'All
- Key features in Y'All
- The user experience
- Behind the scenes
- Summary and recap
- Adopting Y'All in your organization



Who We Are



Johan Pelck Olsen

Data Scientist

Novozymes

Agricultural and Industrial Biosolutions Application Research

jpko@novozymes.com / JPKO @ JMP Community







Michael Akerman

Sr. Data Scientist

Novozymes

Agricultural and Industrial Biosolutions Application Research

mxak@novozymes.com / Michael_MXAK @ JMP Community

T  **GETHER** **WE**
BIOLOGICAL  **ICAL** **FIND**
ANSWERS FOR
BETTER  **LIVES** **IN A**
WORLD  **WORLD** **GROWING**
LET'S RETHINK TOMORROW

About Novozymes

- Global biotechnology company building biological solutions for sustainability, improved performance, and reduced costs
- Supporting many industries, including agricultural, biofuels, household care, food and beverages, animal health, and more
- > 6000 employees globally
- Hundreds of JMP users!

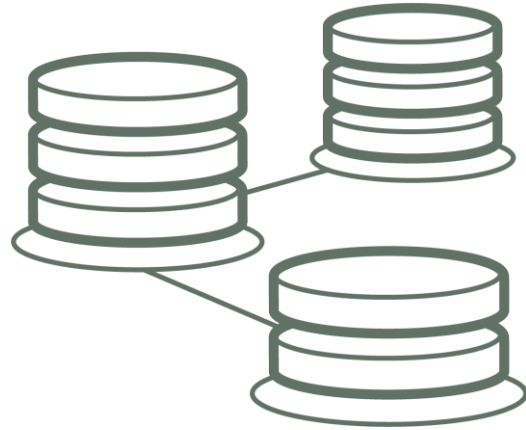
Motivation: Why we built Y'All



Make Updating Easy

No more emailing scripts, no more relying on personal network drives.

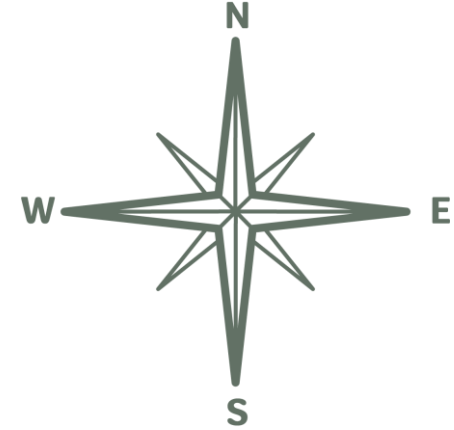
Make distributing new or updated JMP scripts effortless.



Archive and Safeguard Code

Provide a central repository where valuable code is stored and archived.

Automatically track versions of script over time.



Discover and Reuse Scripts

Make it easy to find relevant JMP scripts written by other developers.

Make it convenient to share, import, and reuse functions in JSL.

Y'All's Key Components



Python Build Script

Automatically parse and structure JSL files into Add-In menus.

Write Add-In definition files, generate imported Custom Functions, and ZIP into a .jmpaddin



GitLab Interface and CI/CD

Easy point-and-click upload of scripts to the central repository.

Automate execution of Python script and distribution of the add-in through Release and Beta version channels



JSL Load and Update

Check for new versions on selected channel on JMP load and offer to download and update

Import custom functions into the Scripting Index

ies Filter

yallutis:AddColumn
yallutis:appendTableToPostgresDatabaseRecord
yallutis:createRecordTable
yallutis:errorModal
yallutis:ExtractCol
yallutis:FilterQuery
yallutis:findReplace
yallutis:GenerateAlphaList
yallutis:GenerateNumList
yallutis:generateArray
yallutis:ListColumn
yallutis:makeList
yallutis:Summarize
yallutis:Recodelf
yallutis:reconnectDatabase

yallutis:appendTableToPostgresDatabaseRecord

```
appendTableToPostgresDatabaseRecord({ schema
```

Append the contents of the JMP datatable as new rows on a database table acting as a record. Similar to the built-in Save Database function, except that this Inserts rows, leaving existing rows in the table intact. Will also create the required record table if it does not exist.

Arguments:

- schema = The Postgres database schema to contain the row state record
- table_name = The Postgres database table to contain the row state record. Table will be created if it doesn't exist.
- datatable = The JMP table to write. Column names MUST match the Postgres database table.
- background = Boolean indicator of whether the query should run in the background or not

Background operations should be used when JMP has a few seconds to complete the table creation before data will need to be written. For instance, if the user is going to interact with a UI before adding records to the table.

See Also Topic Help

Key Features in Y'All

- Super easy script contribution from any JMP user
- Automatic version control and storage
- Automatic or manual JMP add-in updates
- Beta version channel for rapid build and testing
- Release channel for controlled deployment
- Discoverable and reusable JSL files
- Convenience utility functions added to Scripting Index and JSL editor

Demo Code Acknowledgements

In the JMP Community instance of Y'All, we've included some examples of scripts for demonstration purposes:

- [jsonparser.jsl](#) from Xan Gregg and Craige Hales
- [Data Cleaning Script Assistant](#) from Jordan Hiller and Mia Stephens, JDS 2020
- [ImageTable Tools](#) by Heidi Hardner and Serkay Olmez, JDS 2020
- [Color Row Selection.jsl](#) by Nick Holmes
- [Multiple Y-Axis Chart](#) by Byron Wingerd
- [Autovalidation Setup](#) by Mike Anderson
- [Importable Riffyn API functions](#) by Riffyn, Inc., from the RiffynTools for JMP Add-In



The image is a collage of three screenshots illustrating a workflow:

- Left Panel:** A terminal window showing shell scripts for building Yall. Key functions include `build_menu_hierarchy`, `check_command_names`, `sort_command_dict`, `addin_def_files`, `create_custom_functions_script`, and `build_addin`. The `build_addin` function is highlighted, showing it collects contents into a zip file.
- Center Panel:** A spreadsheet showing a 'Value' column with a git clone command: `rm -r -f /nfs/home/gitlab-runner/builds/yall; mkdir /nfs/home/gitlab-runner/builds/yall; cd /nfs/home/gitlab-runner/builds/yall; echo /nfs/home/gitlab-runner/builds/yall; if [! -e .git]; then git clone gitlab@gitlab.nzcorp.net:; git config fetch.recurseSubmodules false; rm -f .git/index.lock .git/shallow.lock; rm -f .git/HEAD.lock; git clean -ffdx; git reset --hard; git fetch origin --prune '+refs/heads/*:refs/remotes/ori; git checkout --force -b "ci_version_bump" "$CI_COMMIT_S`. Below the command, it specifies **Only policy:** branches, tags and **When:** on_success.
- Right Panel:** A terminal window showing a GUI update dialog. The dialog has two buttons: "Update" and "Ignore this version". The "Update" button triggers a sequence of actions: `win = Current Window(); write_addin_log(most_recent, 1); installRemoteAddin(addin_location); win << Close Window;`. The "Ignore this version" button triggers: `win = Current Window(); write_addin_log(most_recent, 0); saveIgnoreVersion(most_recent); win << Close Window;`. The dialog also includes a "Cancel" button and a conditional check: `If(!IsEmpty(here:installSuccess) & here:installSuccess == 1,`.

Live Demos:

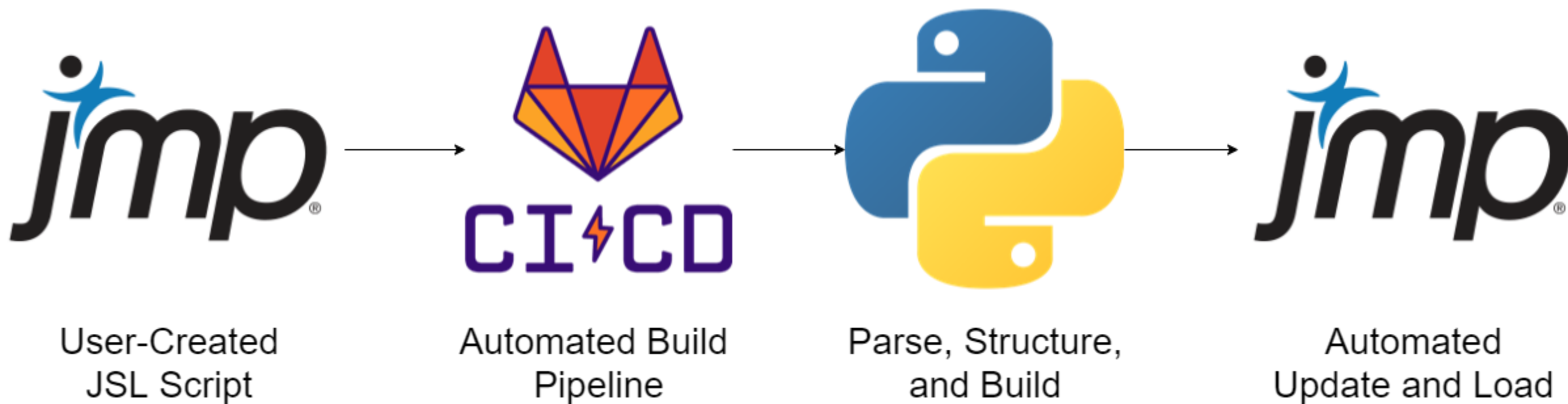
The user experience, and what's happening behind the scenes



Recap: Features

- Easy contribution
- Version control
- Automatic updates
- Beta and Release channels
- Discoverable code
- Utility functions

Recap: Technical Details



How to start if you want to use Y'All

- The ZIP file containing all necessary elements is available on the JMP Community
- The Python build script uses only native Python 3 libraries, and should run on any standard Python installation
- Central repository storage and automation can be achieved through GitLab, GitHub, Azure, AWS, etc. Implementation details are specific to your organization
- The CI/CD pipeline defined in *.gitlab-ci.yml* will also require modification to reflect your repository location and filesystem structure
- Any connection strings to our internal database have been replaced with “ODBC:my_connection_string” and will need to reflect your database

novozymes® 
Rethink Tomorrow