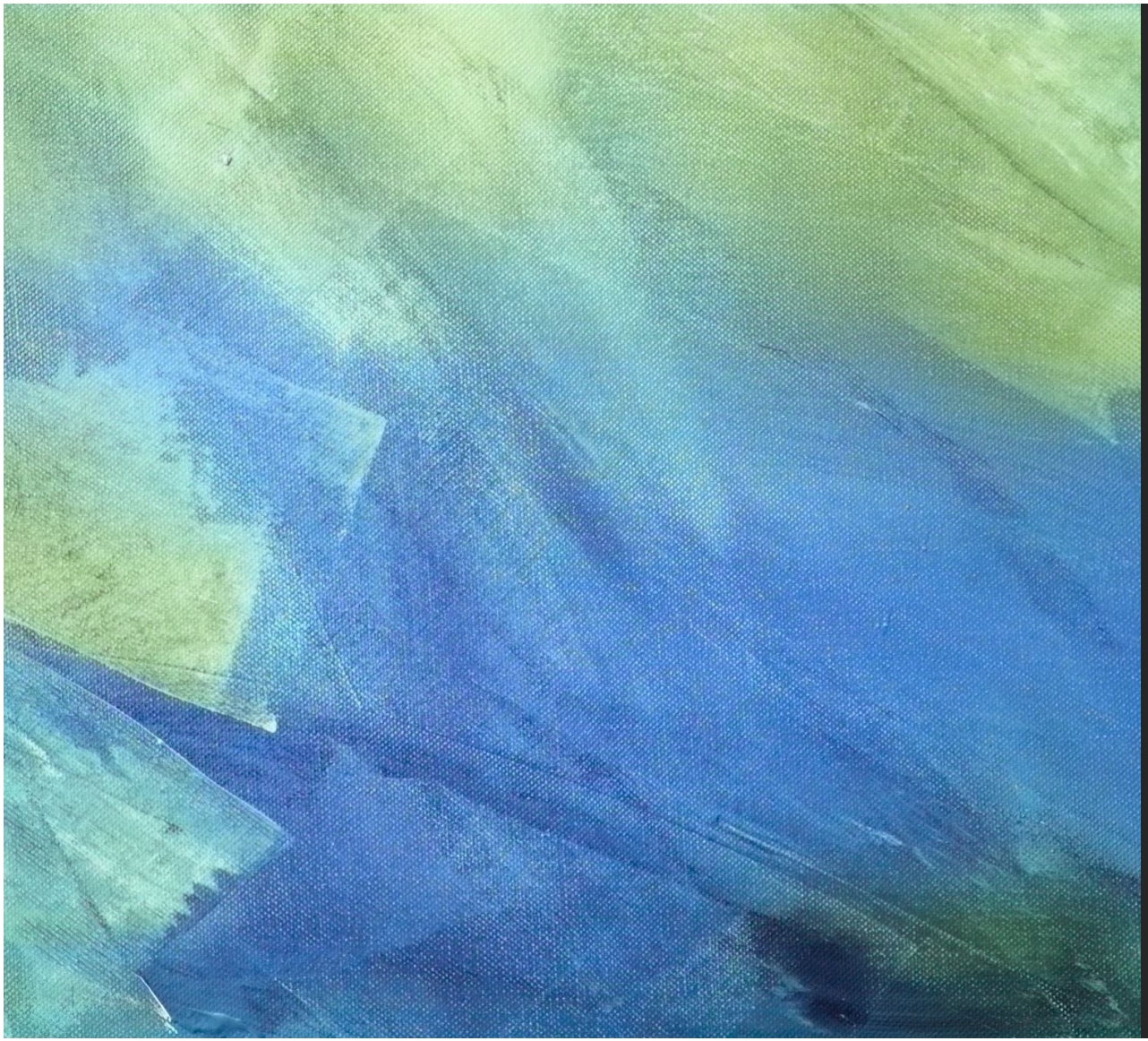


DATA TECHNIQUES FOR ENGINEERS AND SCIENTISTS

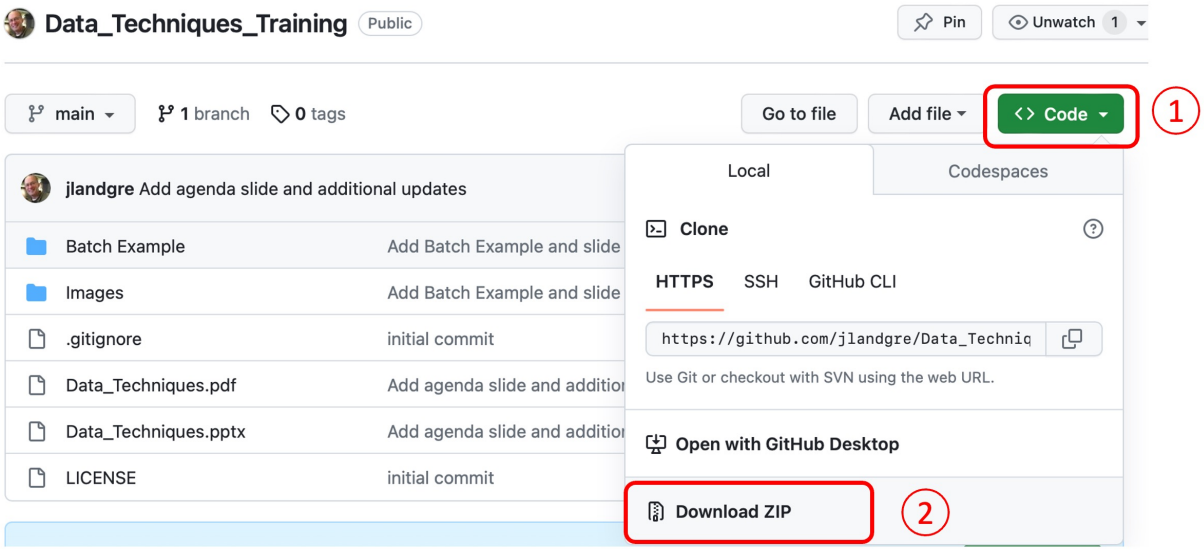
J.D. LANDGREBE

DEPT. OF CHEMICAL ENGINEERING,
UNIVERSITY OF DELAWARE

DATA DELVE ENGINEER LLC

AGENDA: DATA TECHNIQUES FOR ANALYSIS AND MODELING

- What's a healthy basis for selecting and using software as an engineer or scientist?
  - "Generalist" engineer/scientist software recommendations
  - Using "Use Case" concept to create clarity talking about software
- Intro to JMP® software for exploratory data analysis, stats and visualization
- Using Python and Python libraries in a practical way
  - Anaconda for Python library management
  - "Tuhdoop" (aka TDD/OOP) approach for organizing bigger, Python projects
- Parsing instrument-generated data – Example, reapplicable Python script
- A roadmap for experimental data of all kinds
  - A good nomenclature and block diagram for data transformation
  - Using JMP® and Python/Pandas to transform data for analysis
- [Bonus Topic 1] Using engineering (and common sense) principles to linearize experimental data
- [Bonus Topic 2 Fitting models to consumer rating data

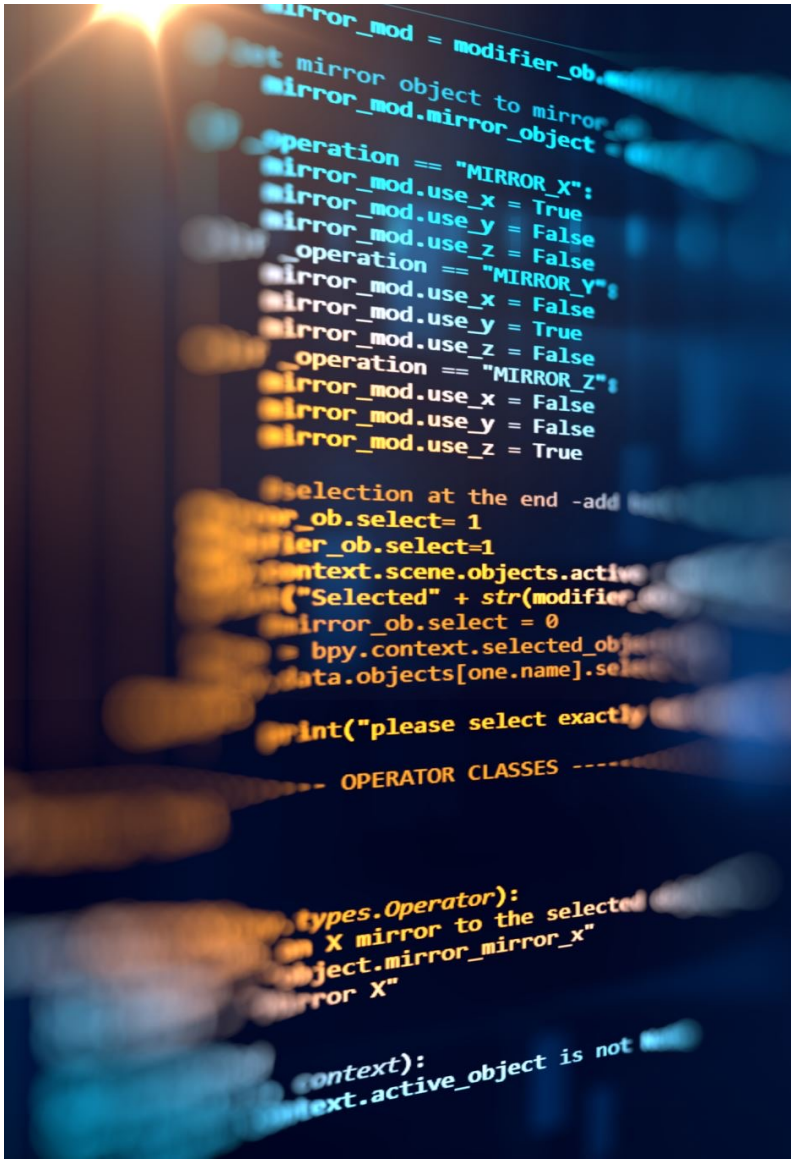


DOWNLOADING TUTORIAL FILES

- Download these notes and case study files from Github repository: https://github.com/jlandgre/Data_Techniques_Training
- Github: Good for engineers to know! It's common for sharing open-source materials like this tutorial
- Github repositories can be open (like this one) or private

SOFTWARE FOR ENGINEERS

- Pick software by “use case” and use each software in its “lane” where it is best tool for the job
- “Use cases” is a useful way of compartmentalizing tasks or activities in software
- Use cases start with “I need to…” (or “my customer needs to…”)
  - My supervisor needs to be able to scroll through our raw data to understand the results
  - I need to create a correlation matrix for my raw material’s quality data
  - I need to build a validated calculation model for reaction kinetics
- Use case focus
  - Avoids getting (yourself and others) distracted by things that software X is not good at – there is no perfect software tool
  - Puts a premium on making data portable across applications



GENERALIST ENGINEER/SCIENTIST SOFTWARE RECOS BY USE CASE

- **JMP® software** - design of experiments (DOE), exploratory analysis and data visualization with accompanying statistical analysis
- **Microsoft Excel®** - make data usable by others, generate “end of the pipeline” reports and create spreadsheet models with calculations for use by yourself or non-coders
- **Python scripts** for data reshaping and for developing coded models and data pipelines (possibly mixing in a little SQL)

Blog

The Right Modeling And Analysis Tools For the Job

Software tools in the data and modeling arena often lead individuals and teams into counterproductive patterns. By being informed and intentional, you can choose the best tool for a particular job. It's good to recognize that software providers, meaning companies and open-source communities, keep...

READ MORE

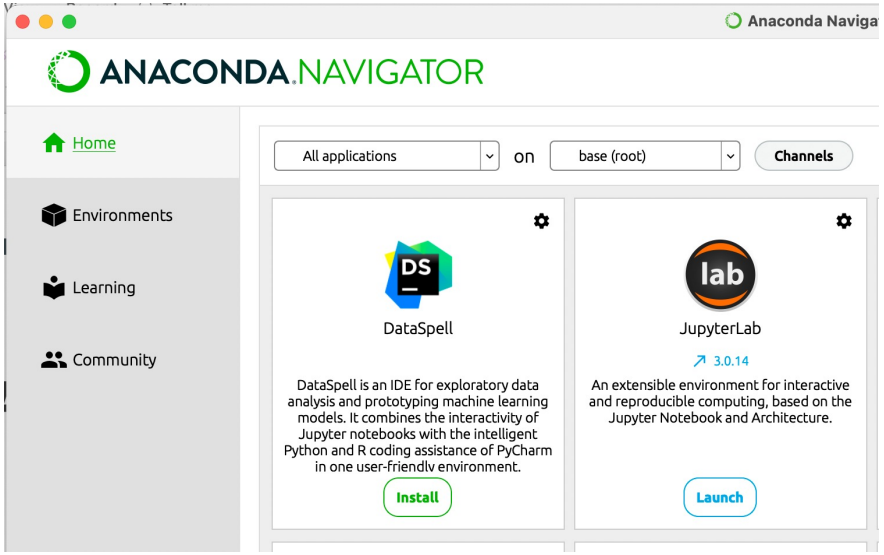
https://datadelveengineer.com/the-right-modeling-and-analysis-tools-for-the-job/

ADDITIONAL SOFTWARE RECOMMENDATIONS

- Python scripting tools - build model and data cleaning scripts
  - JupyterLab (via installation of Anaconda distribution)
  - VS Code script and text editor (https://code.visualstudio.com)
  - Github Copilot and Chat extensions VS Code ($$)
- VBA Scripting Language – Excel automation
- Advanced visualizations – Python Matplotlib
- Github - Open sharing of projects, trainings etc.

ANACONDA AND JUPYTERLAB

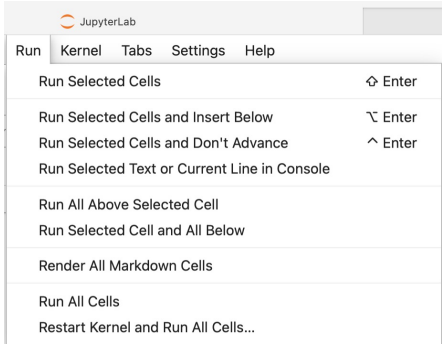
- The “Anaconda Distribution” from Anaconda.org is a good way to get Python and useful Python libraries
  - It installs the Anaconda-Navigator app on your computer
  - Can also use command line to launch Python tools
- Anaconda-Navigator
  - Home tab has tile for launching JupyterLab
  - Environments tab lists and maintains installed libraries

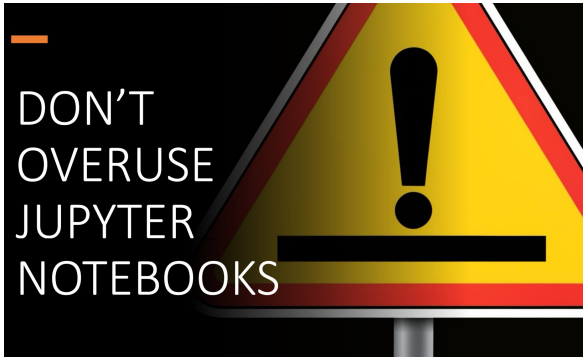


JUPYTER NOTEBOOKS AND BEST PRACTICES

(This slide is a placeholder for a longer-term journey being comfortable with using Python tools routinely!)

- Notebooks use *.ipynb file extension
- Notebook cells can be Code or markdown text
- Launches in browser but not internet-connected
- Has its own menus
  - Shift-Enter to run current cell
  - Run menu / Run All Cells to run all from start to end
  - Kernel / Restart Kernel to reset variable values
- Watchout: Running cells out of order can create unexpected results or errors!
- Exercise:
  - Write a function in Jupyter notebook
  - Move the function to a separate *.py file and run from there
- A robust development process involves a “Tuhdoop” approach with *.py files and testing.
  - Tuhdoop = TDD/OOP = Test Driven Development + Object Oriented Programming
  - See tutorial.md at https://github.com/jlandgre/TDD_OOP_Tutorial





TDD_OOP_Tutorial / tutorial.md

J.D. Landgrebe Fix tutorial typos 394e3ae · last ye

Preview Code Blame 166 lines (125 loc) · 12.9 KB Raw

This shares a case study using the combination of test-driven design (TDD) and object-oriented programming (OOP) in Python coding projects. While both are individually great, in my opinion, there is insufficient discussion online and in tutorials about how to pragmatically use them together. There is tremendous synergy from this.

Introduction

It is one thing to know how to code classes aka objects in Python. It is another to know the TDD approach of writing a pytest test first and then writing the project code to pass the test. From experience though, there is magic in putting the two together –having each test validate a single-action class method and where simply calling the methods in order completes the procedural job-to-be-done of transforming some data or running a model. In consulting work, this has been wonderful for efficiently creating Python (and VBA) code that is easy to revisit and update later.

Using OOP gathers a topic’s methods and attributes together in a way that purely procedural code does not. Using TDD removes any future question about the program getting broken by changes. The TDD tests serve as part of the project's documentation and await being re-run at any time. Personally, I wish I had discovered this combo 20 years ago during my career as an R&D engineer. Certainly, I have always validated my models, but too often not in a way where it’s easy for myself or others to later extend the model or to re-ascertain that it is giving correct results. The TDD/OOP combo (Let’s call this “Tuhdoop” I guess?) has recently taken my working efficiency to a new level.

PYTHON LIBRARIES TO EXTEND FUNCTIONALITY

Useful Python libraries for Engineers and Scientists

You can get these with Anaconda installation!!

- Pandas – Data reshaping and analysis
- Numpy – Numerical calculations
- Pytest – Test/Validate your scripts
- OpenPyXL – Format outputs in Excel
- Re (Regular Expressions aka Regex) – Parse text strings

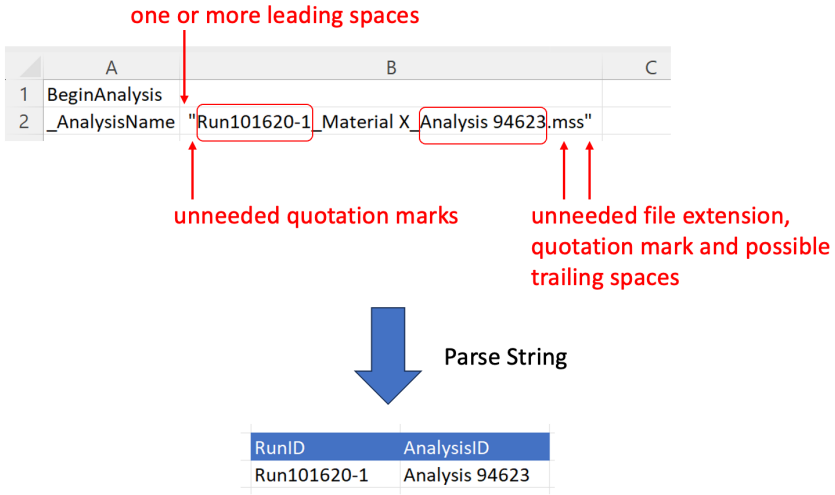
EXAMPLE: PYTHON REGEX PACKAGE TO EXTRACT STRINGS

regex 2023.10.3

Lab instrument software outputs IDs in string needing cleanup

regex Python package can strip out unneeded quotation marks, spaces and (not shown) file extension

(Not shown) Either regex or Python split() command can break the string into pieces using underscore characters as delimiters



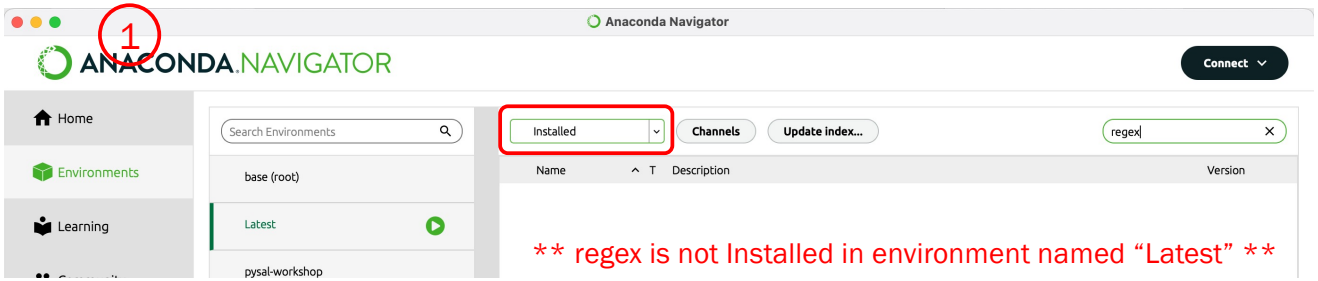
libs > curve_parse.py > ...

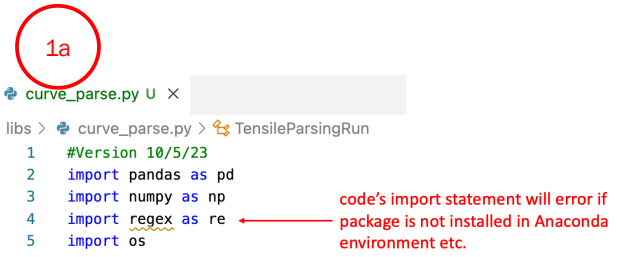
```
1   #Version 10/5/23
2   import pandas as pd
3   import numpy as np
4   import regex as re
5   import os
```

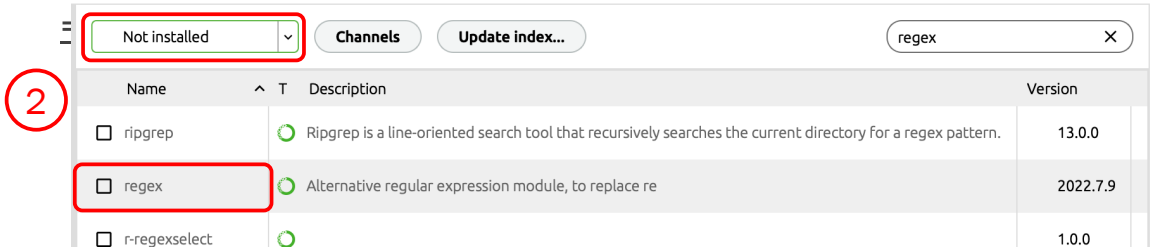
```
174     def id_string_cleanup(self, id_string):
175         """
176         Strip leading/trailing spaces and quotes from an ID string
177         JDL 10/5/23
178         """
179         # Define regular expression patterns
180         leading_space_pattern = r'^\s+'
181         trailing_space_pattern = r'\s+$'
182         quote_pattern = r'^"|"$'
183
184         # Use regular expression to strip leading/trailing spaces and quotes
185         id_string = re.sub(leading_space_pattern, '', id_string)
186         id_string = re.sub(trailing_space_pattern, '', id_string)
187         id_string = re.sub(quote_pattern, '', id_string)
188         return id_string
```

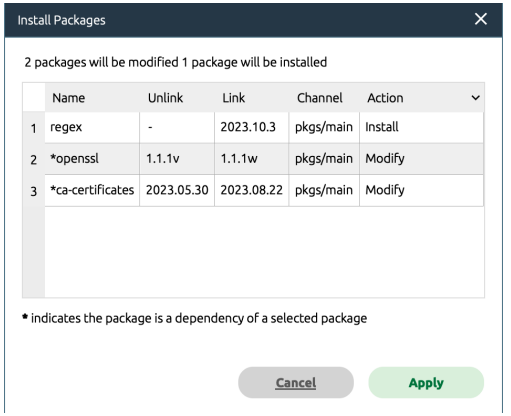
Regex code for "zero or more spaces at beginning of string" ("ChatGPT pls help!!")

INSTALLING PACKAGES (AKA LIBRARIES) IN ANACONDA









Anaconda selects related (aka “dependent”) packages needed by the selected package

EXERCISE WITH USING PYTHON + PANDAS

Exercise:

- Clone: https://github.com/jlandgre/Pandas_Intro_For_Noncoders
- Open Pandas_Fundamentals.ipynb and clear the outputs (Kernel / Restart Kernel and Clear All Outputs)
- Run the cells one at a time by clicking in the first one and hitting Shift+Enter
- Look up documentation for a Pandas method such as **read_excel** or **set_index**

OBJECTIVES
WORKING WITH EXPERIMENT DATA

Experiments* have a common data structure (we will call it “architecture”).

Recognizing this lets us master common data techniques to analyze efficiently and have data in good formats for graphing

Objectives

- Learn reusable terms for the data elements from experiments
- Learn how use software to do needed transformations to go from raw data to analyzed summary
- Learn how to “curate” experimental data to make it easy to find and share

* Data don’t know whether they were generated in a lab or virtual experiment, so this applies to computer modeling data in addition to physical experiments

WHAT ARE EXPERIMENTS?

Experiments Cover An amazing range for engineers and scientists!

Make a production or lab-scale batch of several product formulations

Run a packing line using pre-set conditions for a set amount of time

Crash a sensor-equipped car into a wall

Diaper a baby with several product designs

Drive a F1 car around a track with several various fuel mixtures

Ask a man to shave his face with 3 different types of razors on consecutive days

Make batches of cookies with different types of chocolate

Wash test, fabric swatches with different detergent formulations

Use different catalysts to run a chemical conversion with the same starting materials

Use PCR to replicate and sequence the DNA from multiple virus samples

Mop floors with mops using different cleaning solutions

Surgically implant artificial joints manufactured with different polymeric coatings

WHAT DO THESE HAVE IN COMMON?

Experiments (or ”**studies**”…we will use synonymously) are a building block for designing and confirming products.

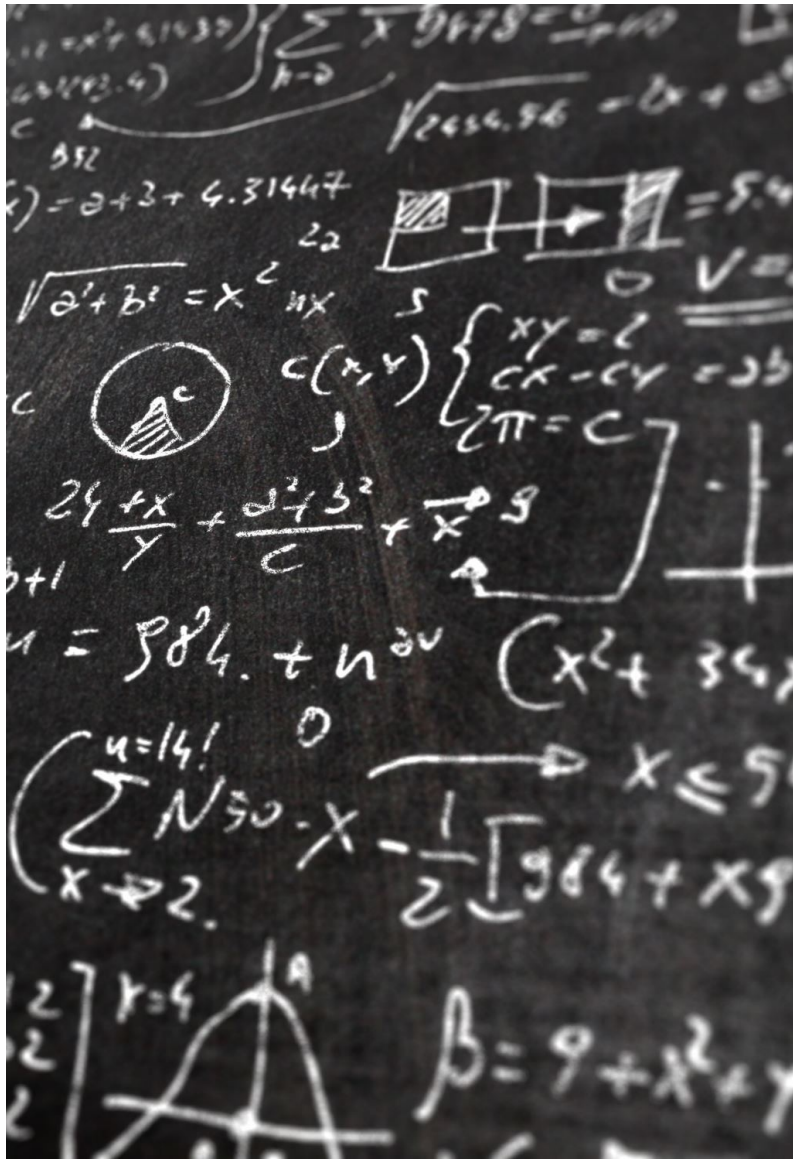
Experiments have “**runs**” consisting of one or more conditions having controlled inputs such as the starting materials and test conditions

A run can be described by a list of the “**run variable**” values which are the pre-set inputs and important ambient conditions recorded during the run.

Experiments generate “**raw data**” consisting of the original measurements to assess what happened (detailed speed versus time data for F1 car circuit around a track)

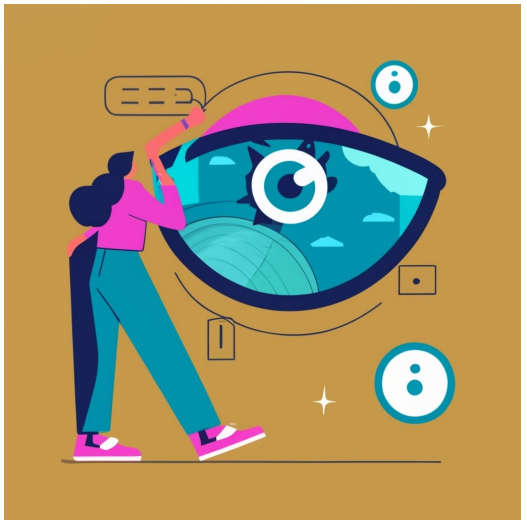
The raw data gets converted to “**parameterized data**” which is quantities reflecting the outcome (average speed for a complete track circuit)

Run variable values can be assembled into a “**run matrix**” with a single row of values for each run

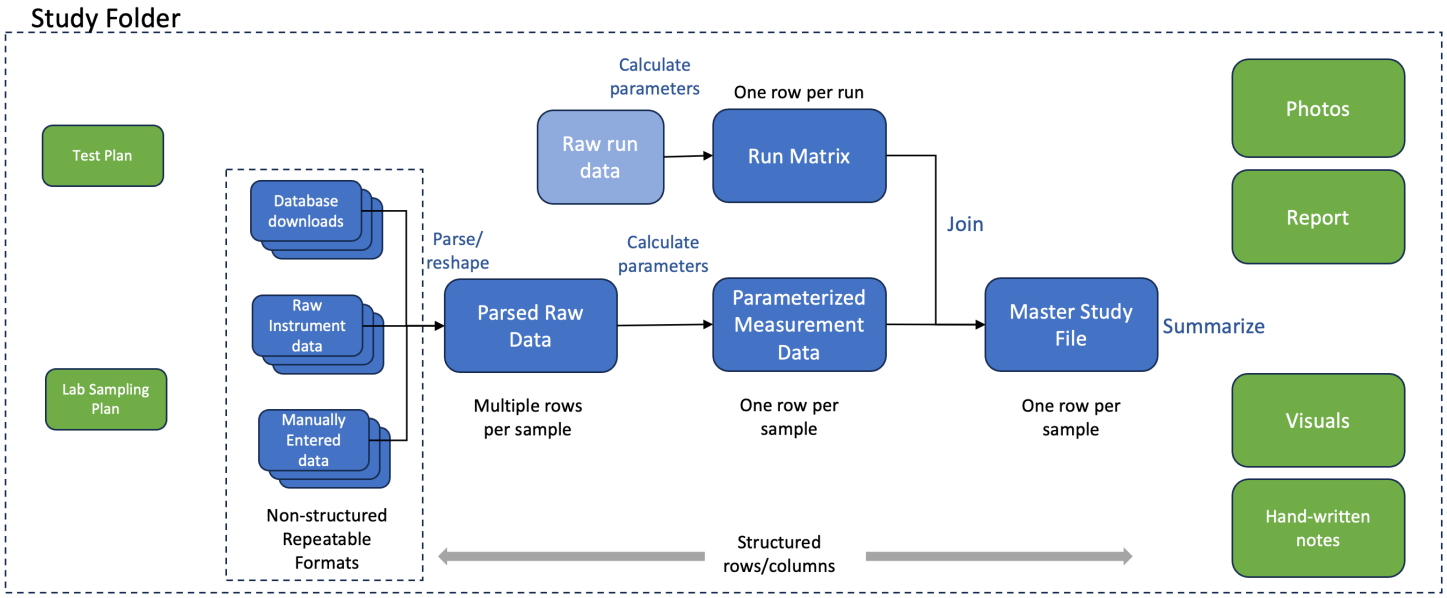


WHY DOES STANDARD STUDY NOMENCLATURE MATTER?

- Reusable ”lens” that you can use in any context
- Knowing nomenclature and architecture gives you a “seeing eye” for how to plan and organize experiments
- Identifies efficient, generic ways of converting the Run Matrix and Raw Data into analyzed data and conclusions
  - Parsers to convert raw data to structured tables
  - “Join” and “Concatenate” to bring data together into a “Master Study File”



EXPERIMENT DATA "ARCHITECTURE"



CREATING A "MASTER STUDY FILE" TO SUMMARIZE AN EXPERIMENTAL STUDY

- Experiment has two Runs that vary **Activity_Tgt**
- Once raw data are parsed into rows/columns, we can join data and the run matrix to analyze the study
- Example study data has two measurement files and Run Matrix
  - ELN_19_AB1234 Measurements_Activity.csv
  - ELN_19_AB1234 Measurements_pH and Viscosity.csv
  - ELN_19_AB1234 Run Matrix.csv
- Goal is analyze by Run Matrix variables
- Need to also check within-run variability

Measurement Data 1

RunID	SampleID	Position	Replicate	Viscosity	pH
ELN_19_AB1234-1	Begin_1	Begin	1	1031	5.6
ELN_19_AB1234-1	Begin_2	Begin	2	1026	5.9
ELN_19_AB1234-1	Begin_3	Begin	3	1010	5.6
ELN_19_AB1234-1	Middle_1	Middle	1	1001	5.6
ELN_19_AB1234-1	Middle_2	Middle	2	991	5.6
ELN_19_AB1234-1	Middle_3	Middle	3	1022	5.7
ELN_19_AB1234-1	End_1	End	1	1008	5.6
ELN_19_AB1234-1	End_2	End	2	985	5.7
ELN_19_AB1234-1	End_3	End	3	982	5.5
ELN_19_AB1234-2	Begin_1	Begin	1	1214	6.7
ELN_19_AB1234-2	Begin_2	Begin	2	1232	6.5
ELN_19_AB1234-2	Begin_3	Begin	3	1223	6.5
ELN_19_AB1234-2	Middle_1	Middle	1	1189	6.5
ELN_19_AB1234-2	Middle_2	Middle	2	1204	6.3
ELN_19_AB1234-2	Middle_3	Middle	3	1202	6.5
ELN_19_AB1234-2	End_1	End	1	1189	6.6
ELN_19_AB1234-2	End_2	End	2	1182	6.6
ELN_19_AB1234-2	End_3	End	3	1194	6.5

Measurement Data 2

RunID	SampleID	Activity
ELN_19_AB1234-1	1	25
ELN_19_AB1234-1	2	24.4
ELN_19_AB1234-1	3	24.7
ELN_19_AB1234-1	4	24.7
ELN_19_AB1234-1	5	24.1
ELN_19_AB1234-1	6	24.8
ELN_19_AB1234-1	7	24.4
ELN_19_AB1234-1	8	24.3
ELN_19_AB1234-1	9	24.5
ELN_19_AB1234-1	10	25.1
ELN_19_AB1234-2	1	25.6
ELN_19_AB1234-2	2	24.8
ELN_19_AB1234-2	3	25.8
ELN_19_AB1234-2	4	26
ELN_19_AB1234-2	5	26.1
ELN_19_AB1234-2	6	25.9
ELN_19_AB1234-2	7	25.5
ELN_19_AB1234-2	8	25.6
ELN_19_AB1234-2	9	25.2
ELN_19_AB1234-2	10	26

Run Matrix

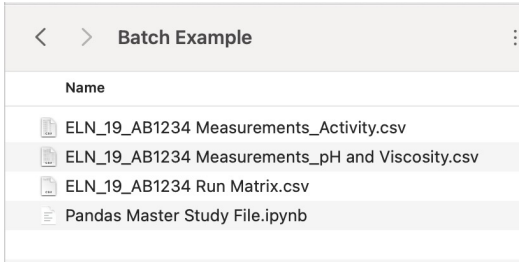
RunID	Date Made	Activity_Tgt	BatchID	Operator
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB

Concatenate + Join (in JMP or Pandas)

Master Study File

RunID	Date Made	Activity_Tgt	BatchID	Operator	SampleID	Position	Replicate	Viscosity	pH	Activity
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	Begin_1	Begin	1	1031	5.6	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	Begin_2	Begin	2	1026	5.9	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	Begin_3	Begin	3	1010	5.6	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	Middle_1	Middle	1	1001	5.6	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	Middle_2	Middle	2	991	5.6	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	Middle_3	Middle	3	1022	5.7	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	End_1	End	1	1008	5.6	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	End_2	End	2	985	5.7	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	End_3	End	3	982	5.5	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	Begin_1	Begin	1	1214	6.7	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	Begin_2	Begin	2	1232	6.5	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	Begin_3	Begin	3	1223	6.5	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	Middle_1	Middle	1	1189	6.5	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	Middle_2	Middle	2	1204	6.3	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	Middle_3	Middle	3	1202	6.5	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	End_1	End	1	1189	6.6	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	End_2	End	2	1182	6.6	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	End_3	End	3	1194	6.5	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	1					25
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	2					24.4
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	3					24.7
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	4					24.7
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	5					24.1
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	6					24.8
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	7					24.4
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	8					24.3
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	9					24.5
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	10					25.1
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	1					25.6
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	2					24.8
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	3					25.8
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	4					26
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	5					26.1
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	6					25.9
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	7					25.5
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	8					25.6
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	9					25.2
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	10					26

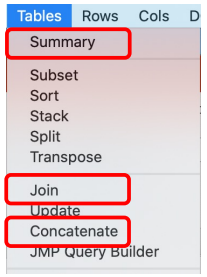
EXERCISE: ASSEMBLE MASTER STUDY FILE



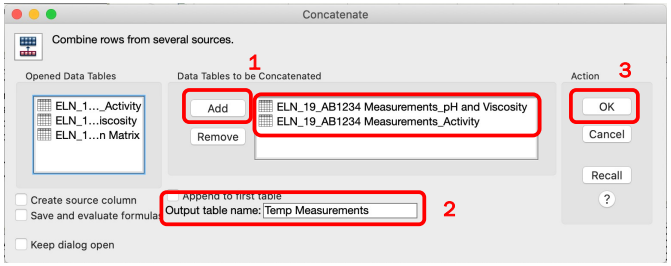
1. Open the Batch Example files and manually copy/paste to arrange like the Master Study File on previous slide
   - Copy/paste to put the two sets of measurement data one on top of the other –inserting blank columns as needed for Activity data
   - Copy paste the appropriate Run Matrix row in blank columns to left of the data
   - Clean up by deleting one of the two RunID columns
2. Use JMP to do the same thing (See instructions on following slides)
3. Open Jupyter notebook Pandas Master Study File.ipynb and use it to create the Master Study File

AGGREGATING AND SUMMARIZING DATA (JMP)

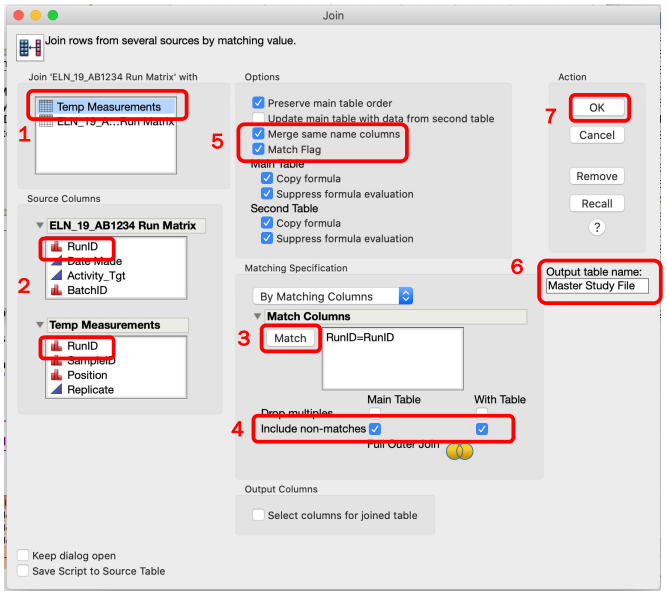
- Open JMP and three *.csv files in Batch Example sub-folder
- To get Master Study File and Summary
  1. Tables/Concatenate: Measurements_Activity.csv and Measurements_pH and Viscosity.csv (Makes Temp Measurements.csv)
  2. Tables/Join: Run Matrix.csv and Temp Measurements.csv
  3. Tables/Summary
- Use Master Study File for statistical analysis



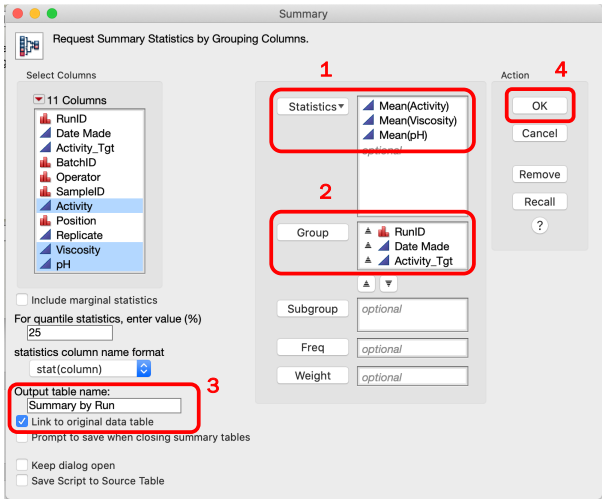
Concatenate measurements



Join Run Matrix with measurements



Summarize by Run

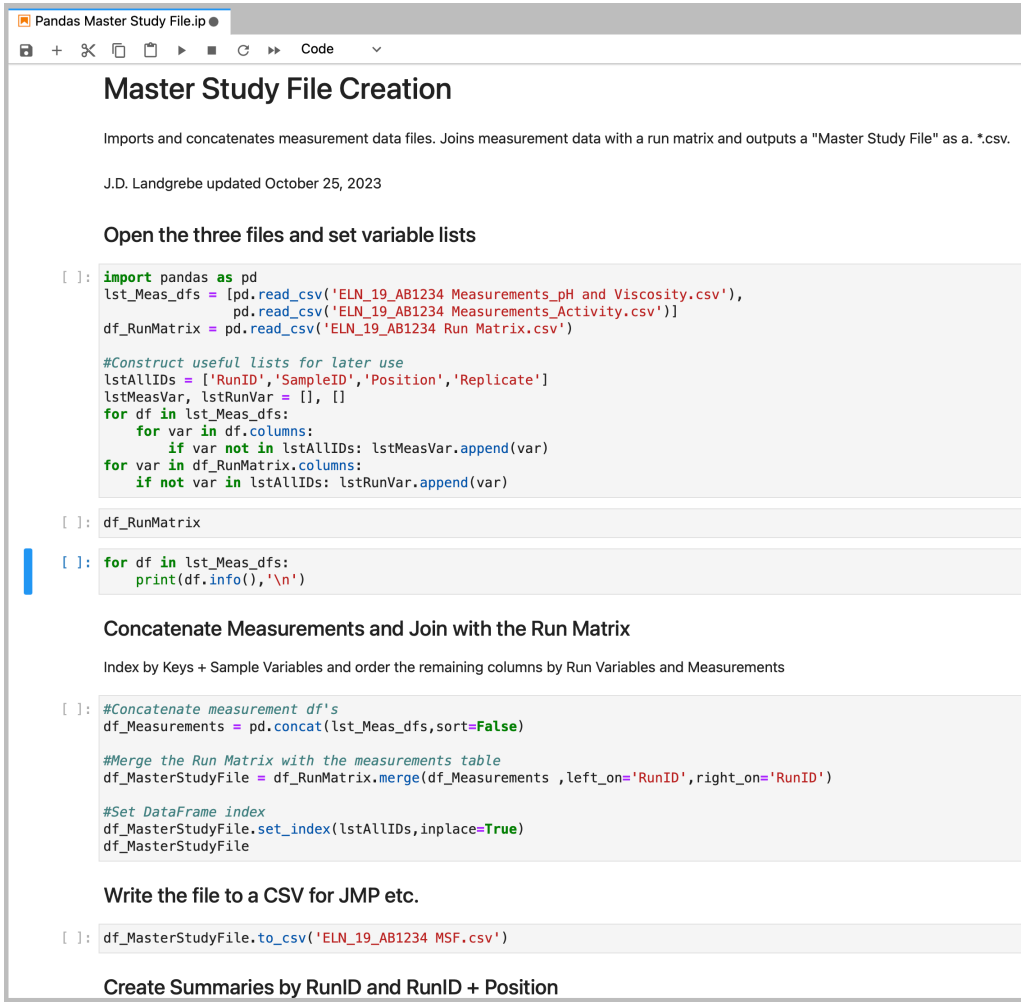


AGGREGATING DATA (PANDAS)

See Jupyter Notebook:

Pandas Master Study File Demo.ipynb

This performs the same concatenate and join transformations as the previous JMP example



EXPERIMENTAL DATA "KEYS"

- Keys [aka data nametags] make data easy to summarize and traceable
- A measurement data point needs up to 6 keys to fully identify the measurement and its source
- Avoid generic Run ID's (e.g. "Run 1") because makes difficult to combine studies
- Check: Is each row identified by unique keys?

Other Keys Needed
AnalysisID – Uniquely ID's an instance of analyzing Samples from a Run (Lab software often assigns this)
PositionID – Identifies order of data points in curve data

Sample ID – Uniquely identifies "when/where" tested material came from

Run ID – Uniquely ID's composition and conditions aka Batch ID, Laundry Load ID, Panelist ID, Modeling Run ID

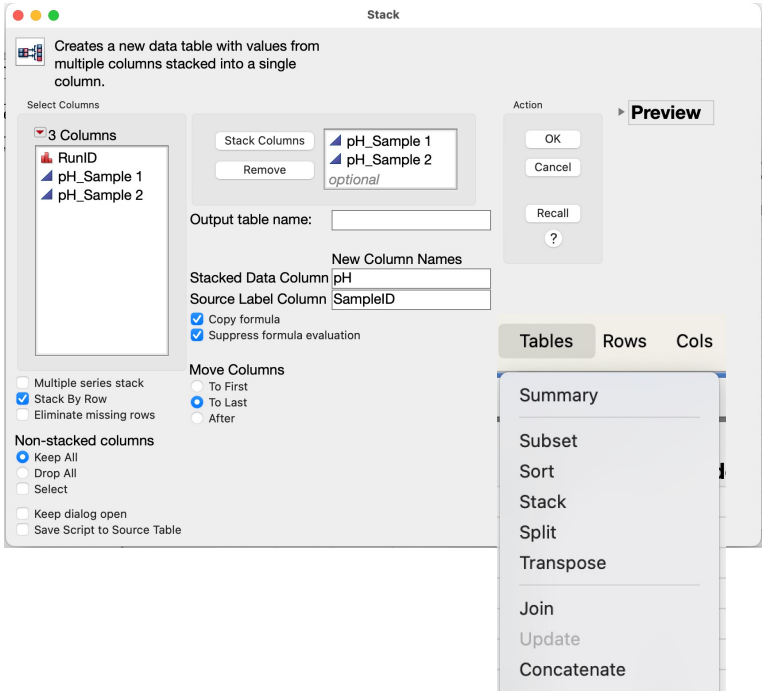
Analyte Name

Study ID – Uniquely ID's collection of Runs; Use ELN Experiment number or created date string as StudyID

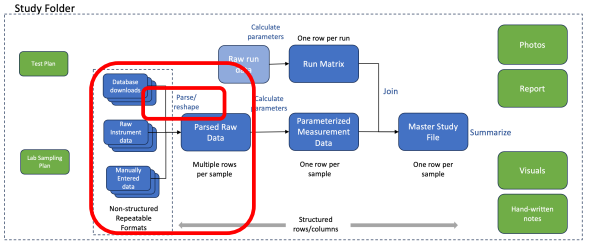
RunID	Date Made	Activity_Tgt	BatchID	Operator	SampleID	Position	Replicate	Viscosity	pH	Activity
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	Begin_1	Begin	1	1031	5.6	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	Begin_2	Begin	2	1026	5.9	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	Begin_3	Begin	3	1010	5.6	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	Middle_1	Middle	1	1001	5.6	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	Middle_2	Middle	2	991	5.6	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	Middle_3	Middle	3	1022	5.7	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	End_1	End	1	1008	5.6	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	End_2	End	2	985	5.7	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	End_3	End	3	982	5.5	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	Begin_1	Begin	1	1214	6.7	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	Begin_2	Begin	2	1232	6.5	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	Begin_3	Begin	3	1223	6.5	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	Middle_1	Middle	1	1189	6.5	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	Middle_2	Middle	2	1204	6.3	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	Middle_3	Middle	3	1202	6.5	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	End_1	End	1	1189	6.6	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	End_2	End	2	1182	6.6	
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	End_3	End	3	1194	6.5	
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	1					25
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	2					24.4
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	3					24.7
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	4					24.7
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	5					24.1
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	6					24.8
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	7					24.4
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	8					24.3
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	9					24.5
ELN_19_AB1234-1	2/1/2020	24.5	XM-2456	OG	10					25.1
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	1					25.6
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	2					24.8
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	3					25.8
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	4					26
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	5					26.1
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	6					25.9
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	7					25.5
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	8					25.6
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	9					25.2
ELN_19_AB1234-2	2/5/2020	25.75	XM-2633	AB	10					26

EXERCISE: JMP TABLES MENU

1. The folder **JMP_Tables**_Menu contains exercises for Tables menu items – see **JMP_Tables_Menu_Exercises.pdf**
2. Each menu command has sample data to use for exploring how the menu option works



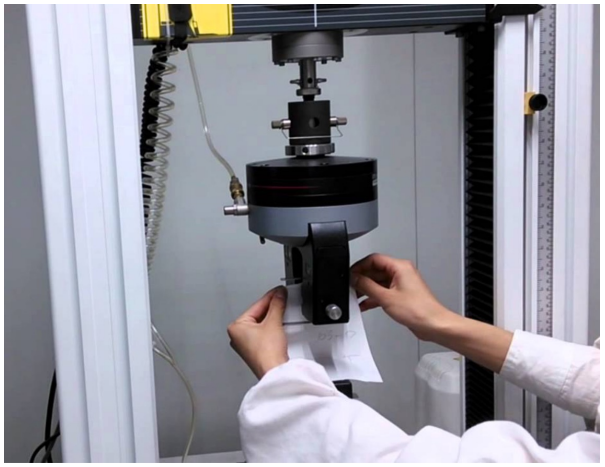
PARSING RAW DATA

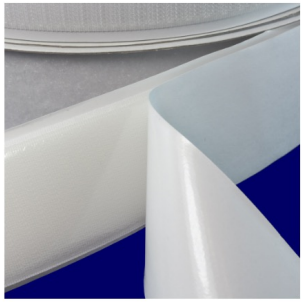


- "Parsing" refers to going from a repeatable non-structured format to structured rows/columns
- Parsing needs to identify samples by its RunID, SampleID (e.g. sampling time/location) and AnalysisID metadata for the result to be considered "structured" (e.g. each sample's data uniquely identified and traceable back to its source)
- **Study owner needs to pre-plan/intervene with lab to ensure metadata are available/parseable post-measurement!!!**
- Parsing requirements vary [wildly!] depending on data source
- Manually-entered data are a special case because humans are very creative about how they choose to enter data
- Lab instruments typically have specialized software that outputs raw and/or parameterized data
- Parsing example: tensile tester data https://github.com/jlandgre/Python_Curve_Parser

RAW DATA PARSING EXAMPLE
DATA FROM MTS TENSILE TESTER SOFTWARE

- Testing involves pulling apart substrate(s) such as hook and loop fastener or adhesive-coated strip + landing surface
- Measurement of force versus displacement while separating the materials –peak load and average load are key results
- Requires multiple samples to get good data: Testing is noisy due to sample prep of joining substrates and general noise from failure testing



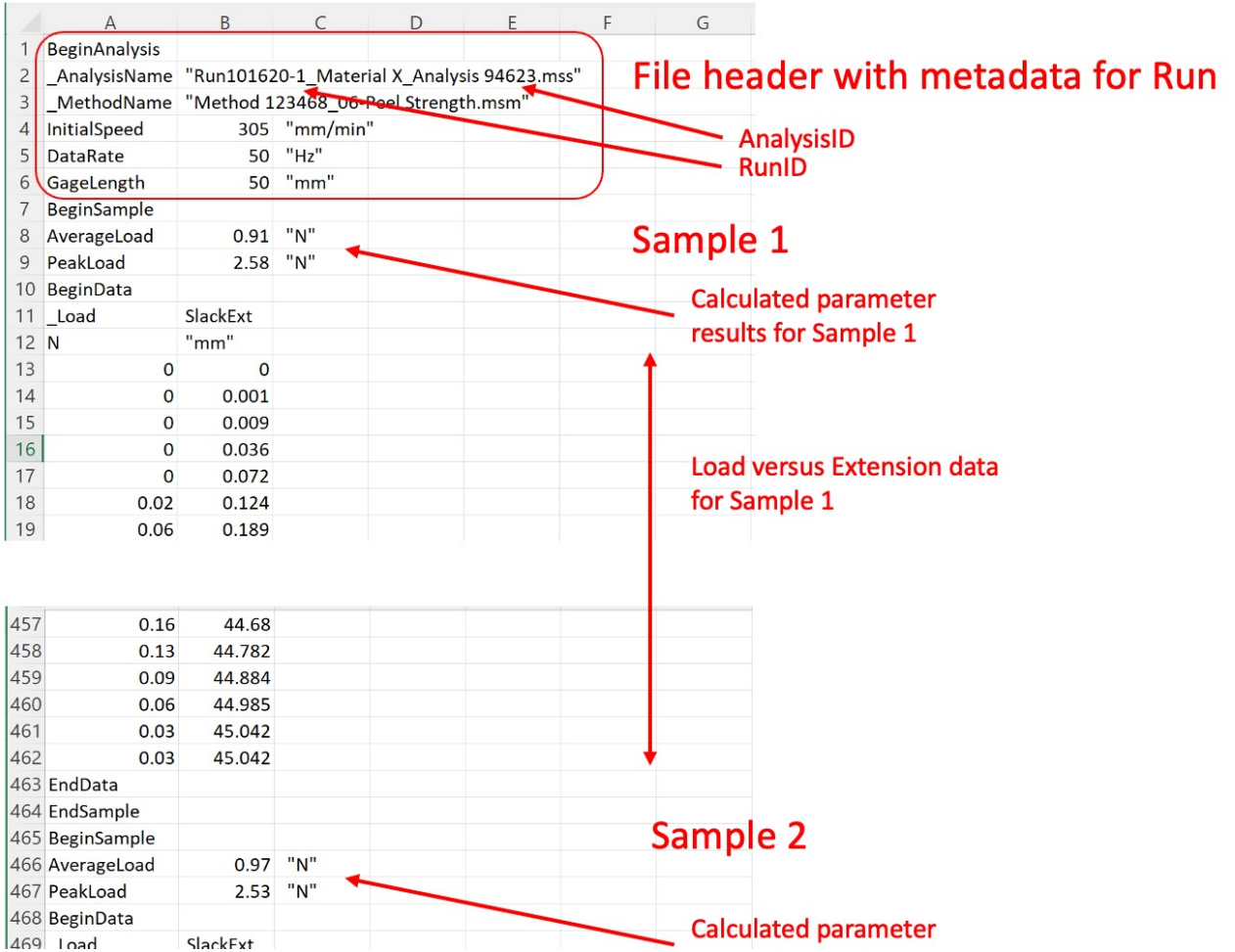


DuraGrip® Brand Adhesive
Backed Hook and Loop Fasteners

VELCRO® Brand Adhesive
Backed Hook and Loop Fasteners

RAW DATA PARSING EXAMPLE DATA FROM MTS TENSILE TESTER SOFTWARE

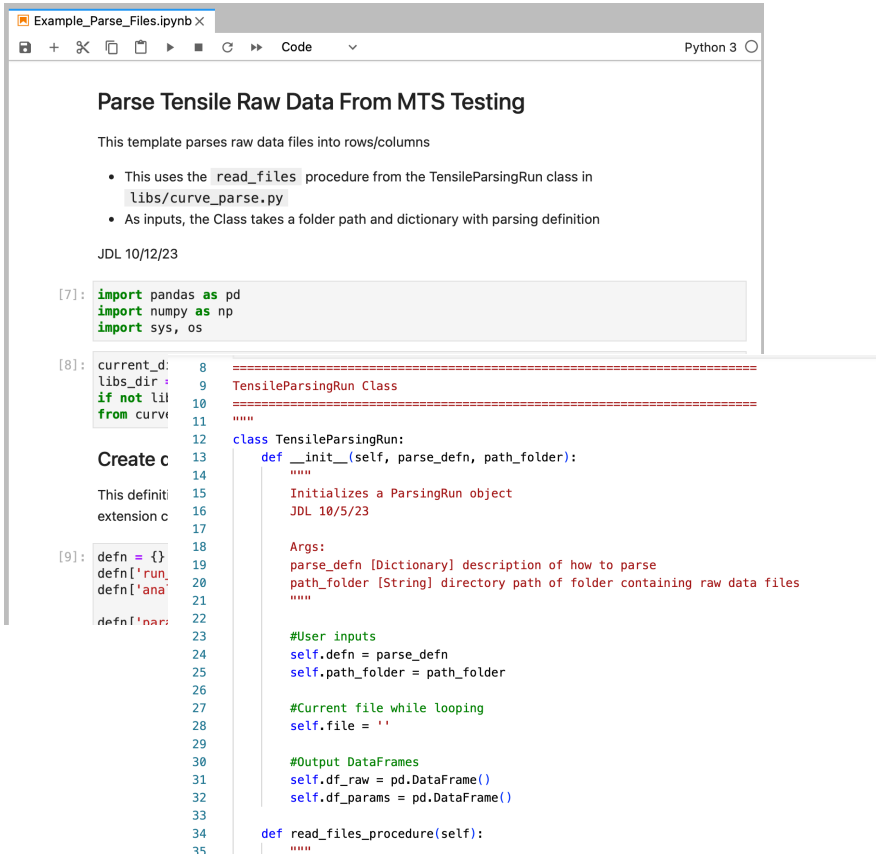
- Instrument software outputs files containing data for multiple samples from a run
- Raw files contain a combination of calculated parameters and raw data –we want to analyze both!!



	A	B	C	D	E	F	G
1	BeginAnalysis						
2	_AnalysisName	"Run101620-1_Material X_Analysis 94623.mss"					
3	_MethodName	"Method 123468_06-Peel Strength.msm"					
4	InitialSpeed	305	"mm/min"				
5	DataRate	50	"Hz"				
6	GageLength	50	"mm"				
7	BeginSample						
8	AverageLoad	0.91	"N"				
9	PeakLoad	2.58	"N"				
10	BeginData						
11	_Load	SlackExt					
12	N	"mm"					
13	0	0					
14	0	0.001					
15	0	0.009					
16	0	0.036					
17	0	0.072					
18	0.02	0.124					
19	0.06	0.189					
457	0.16	44.68					
458	0.13	44.782					
459	0.09	44.884					
460	0.06	44.985					
461	0.03	45.042					
462	0.03	45.042					
463	EndData						
464	EndSample						
465	BeginSample						
466	AverageLoad	0.97	"N"				
467	PeakLoad	2.53	"N"				
468	BeginData						
469	_Load	SlackExt					

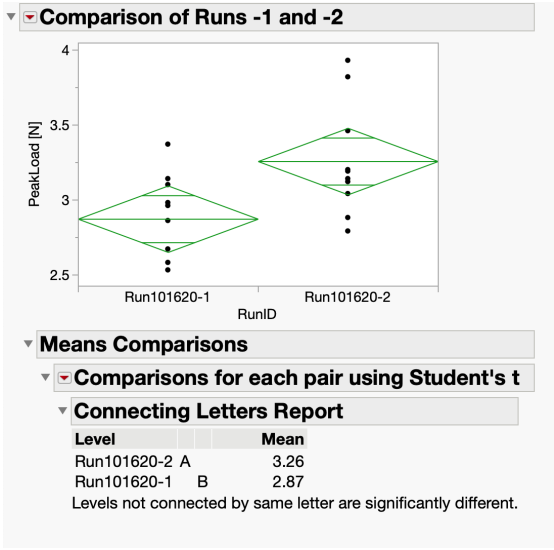
CUSTOM PYTHON PARSER FOR TENSILE DATA

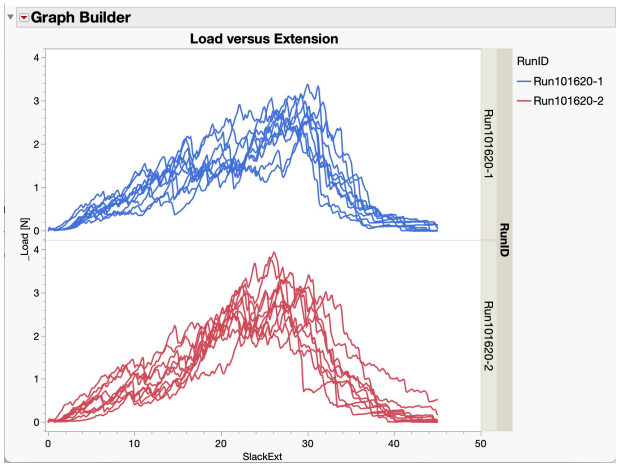
- To try it with example data, clone from: https://github.com/jlandgre/Python_Curve_Parser
- Parser consists of a Python Class in curve_parse.py (run with example data from Example_Parse_Files.ipynb)
- Creates output files: df_params.xlsx and df_raw.xlsx



JMP ANALYSIS OF PARSED DATA

- The whole point of the parser is to gain ability to visualize and study the data –without expending lots of “copy/paste” effort
- JMP Demo on quick analysis of parsed data
  - Analyze menu / Fit Y by X:
    - Run101620-2 has significantly higher Peak Load than -1 ($p<0.05$)
    - Standard deviations (spread of peak load data) look similar
  - Graphs of Load versus extension curves
    - Run -1 has [qualitatively] more consistent slope
    - Several -2 samples undergo rapid force increase at 20+ mm extension

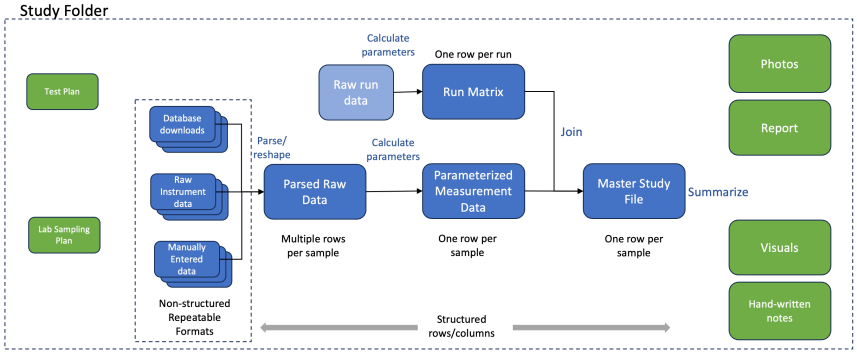




JMP CASE STUDY/ASSIGNMENT – SOLID COOLING

- See files in Canvas Assignment

WORKING WITH SOFTWARE AND EXPERIMENT DATA



- Software recos give you a toolbox for working with a broad range of engineering data
- JMP® and Python Pandas are important tools to always have around for working with data
- The Study Folder approach curates experiments to make them efficiently extensible and shareable
- The standard architecture and templates for experiments give you a reusable cookbook for designing experiments and working with the resulting data
- JMP gives you access to statistical analysis combined with visualizing data and building quantitative models