



---

# **BUILDING ROBUST AND COLLABORATIVE CODED ENGINEERING MODELS**

Colloquium on occasion of Professor Sotiris Pratsinis retirement

Particle Technology Laboratory. Department of Mechanical &  
Process Engineering, ETH Zürich, Switzerland

February 26, 2025

J.D. LANDGREBE, DATA-DELVE LLC

[jdlandgrebe@data-delve.com](mailto:jdlandgrebe@data-delve.com)

FORMER PRINCIPAL ENGINEER,  
PROCTER AND GAMBLE CO. INC.



# INTRO

- 30+ years Procter and Gamble technologist/Principal Engineer – Consumer Packaged Goods R&D
- BS ChE (Purdue University USA 1984)
- MS ChE (U. of Cincinnati USA/Pratsinis 1989)  
Thesis: Gas-phase Particulate Manufacture: The Interplay of Chemical Reaction and Aerosol Coagulation
- Data science and technical modeling consulting practice ([datadelveengineer.com](http://datadelveengineer.com))
- Adjunct Professor, University of Delaware USA Chemical Engineering



**Data Delve** 



# SOME THINGS I LEARNED FROM “DR. P”\*

\* My then newlywed wife's name for Sotiris circa 1987

- Penetrate technically to the full extent of the topic

230 documents have cited:

A discrete-sectional model for particulate production by gas-phase chemical reaction and aerosol coagulation in the free-molecular regime  
Landgrebe J.D., Pratsinis S.E.  
(1990) *Journal of Colloid And Interface Science*, 139 (1), pp. 63-86.

- Research is a team sport. Be collaborative and even friends with your colleagues wherever they may be
- Great visuals rule the day (Corollary: “Say it with dimensionless numbers”)
- Mentoring means giving your mentees/those under you a “seat at the table”
  - Conferences
  - Consulting
- Lead a balanced life that pays full attention to our creator and to those around us

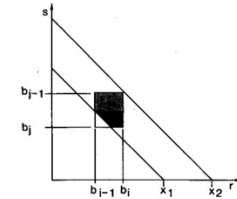
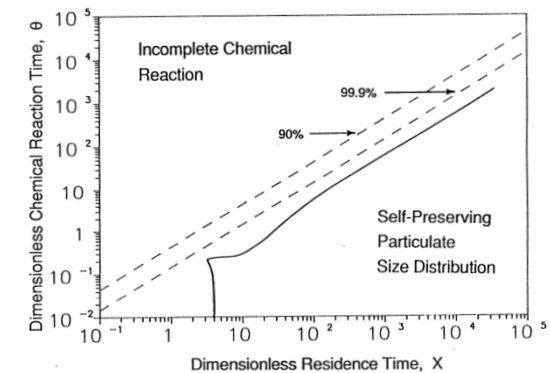


FIG. A1. Collision integral integration limits. The discrete sizes  $b_{i-1}$ ,  $b_i$ ,  $b_{j-1}$ , and  $b_j$  represent the discrete size upper and lower boundaries of two sections,  $i$  and  $j$ ;  $x_1$  and  $x_2$  represent the boundaries of a third section. The shaded area shows combinations of  $i$ - and  $j$ -section particles resulting in a particle in the third section.



With congratulations, gratitude, and warmest best wishes to Sotiris and Eleni!!

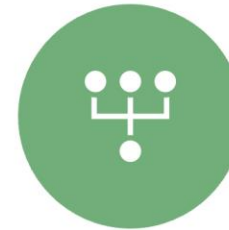
Sue and J.D. Landgrebe

# WHY BEST PRACTICES FOR CODED MODELS?

- Have you ever:
  - Picked up a colleague's, former student's (or your own 🤪😞) model/code and spent hours trying to understand it?
  - Had doubts or uncertainty about the validity of someone else's model?
  - Had difficulty extending or scaling a previous model (your own or someone else's?)
- HUGE opportunity exists to set best practices individually, in research groups and corporate teams –it's a leadership job!!!



The right amount of code planning cuts down on rework and allows you to work with more complex topics



Picking a model back up from someone else (or from the one-year-ago you) can be challenging



Most models don't have an inspectable validation trail that builds confidence in them as you work with them and seek to extend them

## Target audience:

- Leaders of groups who want set culture for how things are done under their watch
- Individual modelers –graduate students, practicing engineers and scientists. Anyone who builds coded models but is not full-time software engineer or member of software development team

jlandgre

Engineering\_Model\_Best\_Practices

Type 7 to search

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

Files

main

Go to file

Roll\_Length\_Model\_Tutorial

images

libs

tests

RollModel\_Chat\_Background\_P...

RollModel\_Chat\_CodeWriting\_P...

Roll\_Length.ipynb

cushiony\_tp\_length\_vs\_diam.csv

tutorial.md

.gitignore

Best\_Practices\_Coded\_Models.p...

LICENSE

Engineering\_Model\_Best\_Practices / Roll\_Length\_Model\_Tutorial / tutorial.md

aa7b03d · 5 minutes ago

History

Preview

Code

Blame

117 lines (85 loc) · 13.1 KB

Raw

Tutorial for Using LLM Model Chat to Code An Engineering Model

J.D. Landgrebe, Data Delve LLC  
February, 2025


This tutorial walks through creating Python code and validation for a useful industrial model. We demonstrate using an LLM AI tool, Github Copilot Chat to code the model and Pytest tests in a separate file to validate the model's calculations. We teach use of a code architecture that does a good job "curating" the model –making it transparent to work with and expand later. We also view it as critical to have rerunnable model validation in the form of coded tests. The Chat tool makes this efficient --doing the bulk of the coding work and requiring just minor, human cleanup of its output.

Contents

- [Modeling Case Study Description](#)
- [Overview of Coding the Model](#)
- [Getting Needed Software](#)
- [Architecting the Code](#)
- [Using LLM Chat To Generate the Python Code](#)
- [Style Suggestions to Curate the Model](#)

Modeling Case Study Description

As a case study to work with, we will code a model that can calculate the caliper or z-direction thickness of a substrate wound on a roll. Some example substrates are nonwoven fabrics and film raw materials such as packaging film. Toilet paper and paper towels are consumer product examples. We will use test our model on data measured for toilet paper purchased from a grocery store.



# POSTED TUTORIAL

■ See full story with tutorial at:  
[https://github.com/jlandgre/Engineering\\_Model\\_Best\\_Practices](https://github.com/jlandgre/Engineering_Model_Best_Practices)

■ If unfamiliar with Github, click on  
**Readme.md** doc for download  
instructions

# WHY BEST PRACTICES FOR CODED MODELS?

Best  
Practices  
for Models



Culture:  
“This is how we  
do things  
around here”  
aka TIHWDTAH

## Objectives of Implementing best practices

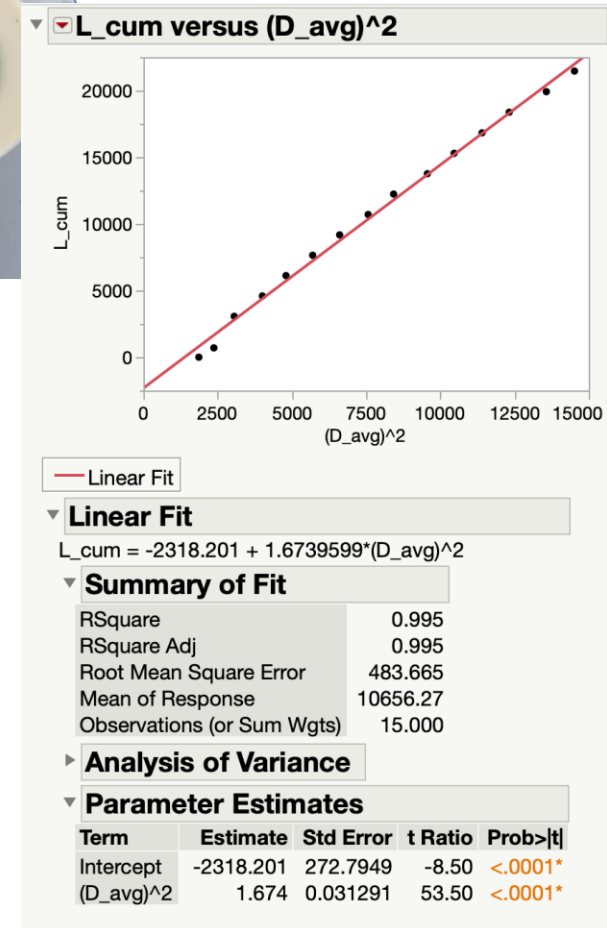
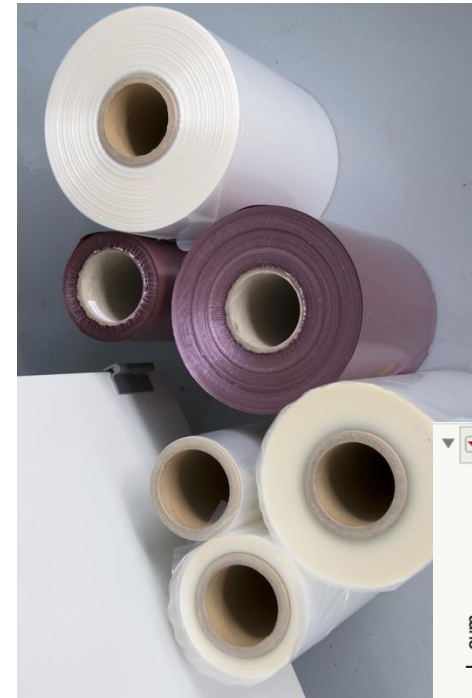
1. Gain ability to create Curated coded models personally and in your team  
(see [The Fourth Paradigm: Data-Intensive Scientific Discovery](#), Hey et al., Microsoft Research, 2009)
2. Validate with rerunnable tests – guard against future broken models  
(see [Test-Driven Design](#) and [here](#) inspiration but that’s a deep and debated topic. Keep it practical!)
3. Learn to develop efficiently using AI tools to write the code and tests (≈90%)

# BEST PRACTICES – PYTHON MODEL

- Github has tutorial to build, curate and validate an industrially useful model for calculating substrate roll properties (Length, L vs. Diam., D)
- A simple, technical example to teach software practices
- Tutorial includes how to use AI to write code for the model and for rerunnable Python (Pytest) tests in separate files

$$L = \frac{\pi(D_{roll}^2 - D_{core}^2)}{4C} \quad \longrightarrow \quad L = \left(\frac{\pi}{4C}\right) D_{roll}^2 + \left(\frac{-\pi D_{core}^2}{4C}\right) \quad \longrightarrow \quad Y = mX + b$$

C = material's caliper or thickness

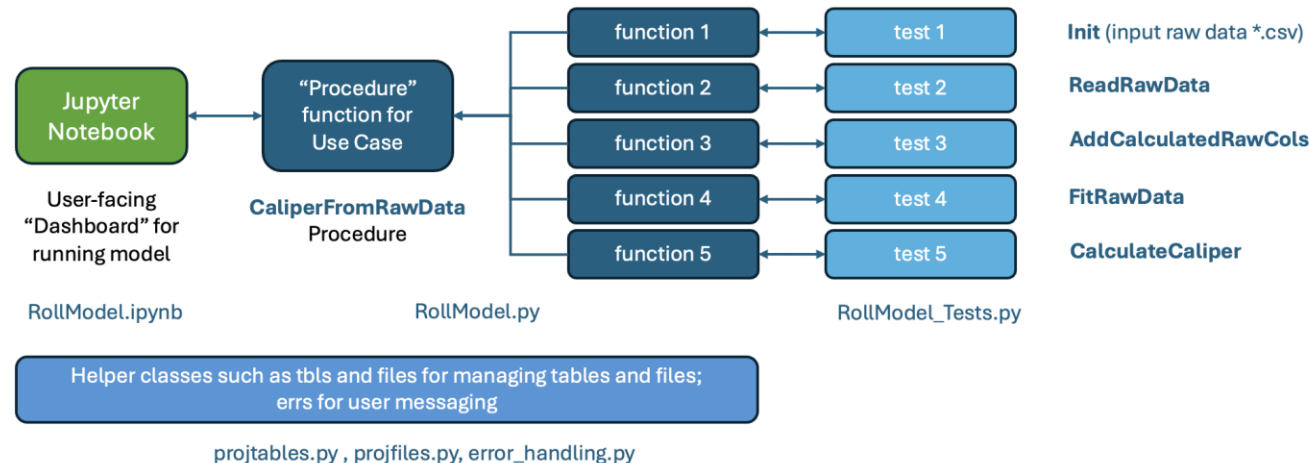


# CURATING – PYTHON MODEL EXAMPLE

Curating  $\neq$  long, multi-step code with obscure names

- Code should be easily inspectable by non-coding leader/manager
- Macroscopic, predictable architecture helps curate
- Single-action functions (Clean Code: A Handbook of Agile Software Craftsmanship”, R. Martin, 2008)
  - Easy to test
  - Easy for AI tool to write the code and write rerunnable test(s)
- Object-oriented “Class-based”
  - Easy to bring in trusted, helper libraries for working with data etc.
  - Model’s instanced code objects are easy to inspect and debug

A Good  
Architecture  
for Model  
Code



Meandering,  
“spaghetti”  
code

## Multistep “Procedure”

```
def CaliperFromRawDataProcedure(self):
    """
    Procedure to fit a line to transformed raw, length versus diam data
    and thereby enable calculation of an effective caliper for the
    material on a roll of substrate.

    This use case only uses the file_raw Class input --to read in raw
    data
    """
    self.ReadRawData()
    self.AddCalculatedRowCols()
    self.FitRawData()
    self.CalculateCaliper()
```

## Single-action Function

```
def FitRawData(self):
    """
    Fit line using diam_m^2 as x and length as y
    JDL Feb 18, 2025
    """
    X = self.df_raw[['diam_m^2']]
    y = self.df_raw['length']
    model = LinearRegression().fit(X, y)

    # Set model attributes based on the fit
    self.slope = model.coef_[0]
    self.intcpt = model.intercept_
    self.R_squared = model.score(X, y)
```



# TESTING/VALIDATION

Testing is under-utilized by engineers/scientists; yet easy to learn

- Many “validations” consist of feeding a model some inputs, looking at the results and pronouncing it validated
- Tests = rerunnable proof that future changes don’t break the model
- Tests are documentation that explains how each function works
- Standard Python Pytest library manages testing from \*.py file separate from model code’s \*.py
- Advise minimum one test per model function

```

(base) $ pytest test_roll_model.py -v -s
===== test session starts =====
platform darwin -- Python 3.12.7, pytest-7.4.4, pluggy-1.0.0 -- /opt/anaconda3/bin/python
cachedir: .pytest_cache
rootdir: /Users/j.d.landgrebe/Box Sync/Projects/Engineering_Model_Best_Practices/Roll_Length_Model_Tutorial/tests
plugins: anyio-4.2.0
collected 6 items

test_roll_model.py::test_rl_fixture PASSED
test_roll_model.py::test_init PASSED
test_roll_model.py::test_ReadRawData PASSED
test_roll_model.py::test_AddCalculatedRawCols PASSED
test_roll_model.py::test_FitRawData PASSED
test_roll_model.py::test_CalculateCaliper PASSED

===== 6 passed in 0.90s =====
(base) $

```



Write test\_FitRawData.

- set x and y numpy arrays from the appropriate .df\_raw column .values
- calculate delta\_y and delta\_x from the individual 2-element array values in x and y
- calculate the slope\_expected from delta\_y and delta\_x
- calculate the intercept\_expected from y[1], x[1] and slope\_expected
- use np.isclose to check that slope\_expected and intercept\_expected match hand calculated values of 1562.5 and -2.5 to within +/- 0.001
- use np.isclose to check that .slope, .intcpt and .R\_squared from the method match slope\_expected, intercept\_expected and 1.0 respectively

## Resulting Pytest Function

```

def test_FitRawData(rl):
    """
    Test - Fit line using diam_m^2 as x and length as y
    JDL 18/2/2025
    """
    rl.ReadRawData()
    rl.AddCalculatedRawCols()
    rl.FitRawData()

    #Numpy arrays for manual calculations
    x, y = rl.df_raw['diam_m^2'].values, rl.df_raw['length'].values

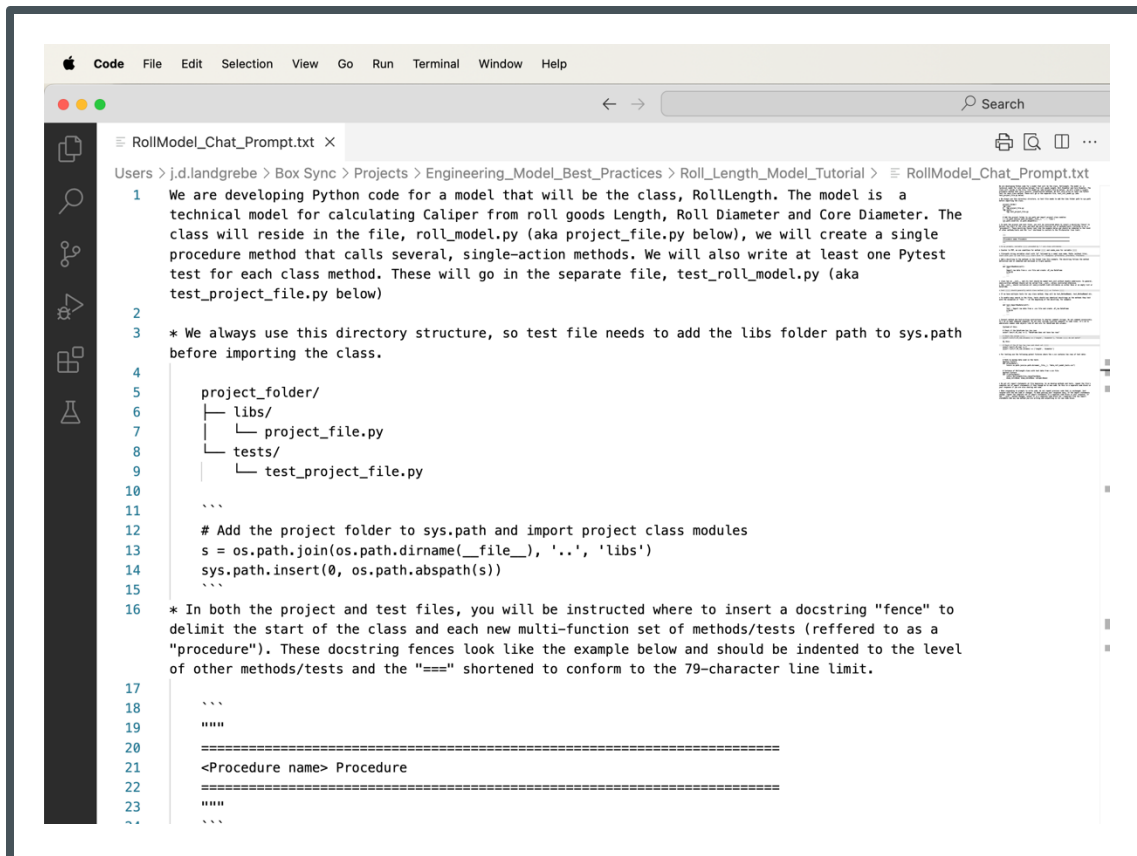
    slope_expected = (y[1] - y[0]) / (x[1] - x[0])
    intercept_expected = y[1] - slope_expected * x[1]

    # Check expected calcs versus hand calculations
    assert np.isclose(slope_expected, 1562.5, atol=0.001)
    assert np.isclose(intercept_expected, -2.5, atol=0.001)

    # Check attributes from FitRawData method
    assert np.isclose(rl.slope, slope_expected, atol=0.001)
    assert np.isclose(rl.intcpt, intercept_expected, atol=0.001)
    assert np.isclose(rl.R_squared, 1.0, atol=0.001)

```

# AI TOOLS FOR CODED MODELS



The screenshot shows a VS Code editor window with a file named 'RollModel\_Chat\_Prompt.txt' open. The file contains a series of numbered instructions (1-24) for developing a Python model. The instructions describe the project structure, the class 'RollLength', the use of docstrings for procedures, and the setup of the sys.path to include a local 'libs' folder. The code is written in a clear, instructional style, likely generated by an AI tool.

```
1 We are developing Python code for a model that will be the class, RollLength. The model is a
2 technical model for calculating Caliper from roll goods Length, Roll Diameter and Core Diameter. The
3 class will reside in the file, roll_model.py (aka project_file.py below), we will create a single
4 procedure method that calls several, single-action methods. We will also write at least one Pytest
5 test for each class method. These will go in the separate file, test_roll_model.py (aka
6 test_project_file.py below)
7
8 * We always use this directory structure, so test file needs to add the libs folder path to sys.path
9 before importing the class.
10
11 project_folder/
12 |__ libs/
13 |   |__ project_file.py
14 |   |__ tests/
15 |       |__ test_project_file.py
16
17 ...
18
19 # Add the project folder to sys.path and import project class modules
20 s = os.path.join(os.path.dirname(__file__), '..', 'libs')
21 sys.path.insert(0, os.path.abspath(s))
22
23 ...
24
25 * In both the project and test files, you will be instructed where to insert a docstring "fence" to
26 delimit the start of the class and each new multi-function set of methods/tests (referred to as a
27 "procedure"). These docstring fences look like the example below and should be indented to the level
28 of other methods/tests and the "===" shortened to conform to the 79-character line limit.
29
30 ...
31
32 =====
33 <Procedure name> Procedure
34 =====
35
36 ...
```

- Organization should provide subscription and not-so-subtly encourage use (“Can you please also post your background prompt for Copilot?”)
- Github Copilot Chat + [Microsoft VS Code](#) is a good option
- [Typical] Use Case 1 = one-time queries like “How to write a custom, Python iterator for to iterate over a range of DataFrame cells?”
- [More valuable] Use Case 2 = “Write the **FitRawData** function and its Pytest **test\_FitRawData** that creates mockup data and checks the linear fit...”
  - Previous best practices of object-oriented and modular, single-action functions + tests are an ideal fit with using current AI chat tools
  - Need to supply tool with background to explain “how life works.” Cbe standard, pasteable text file (tutorial example)
  - Generally requires more pre-planning than historical coding practices
  - Does not eliminate need for knowing coding –AI tools make mistakes and/or can stray from background intent



## CONCLUSIONS

- People join organizations with a range of coding skills and practices
- Leveling up required with seminars, required trainings etc.
- Valuable to set cultural best practices for curation, validation and use of AI tools for coding models
- Heading down this path leads to high efficiency and trustworthiness of coded engineering and scientific models