DiffPy-CMI - a software toolbox for real-space structure analysis

Simon J. L. Billinge

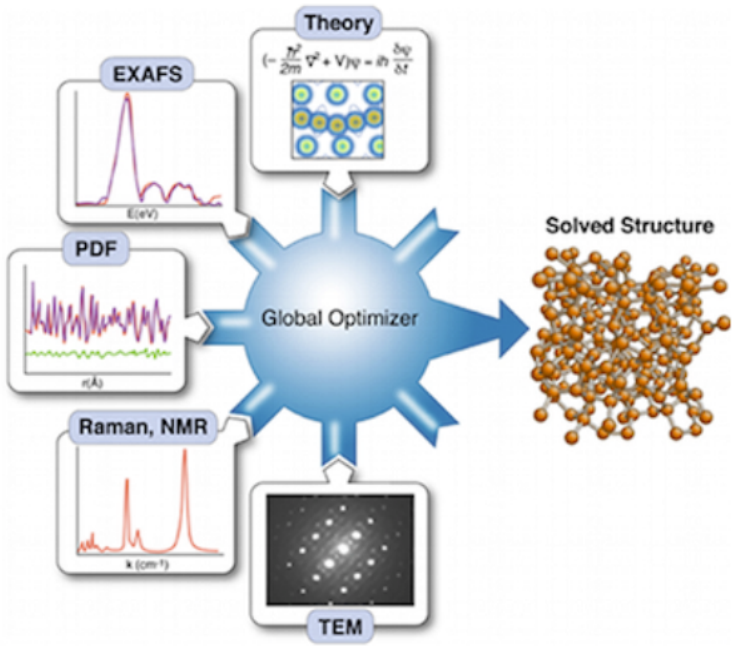
[1]Materials Department,
University of California, Santa Barbara

January 13, 2026

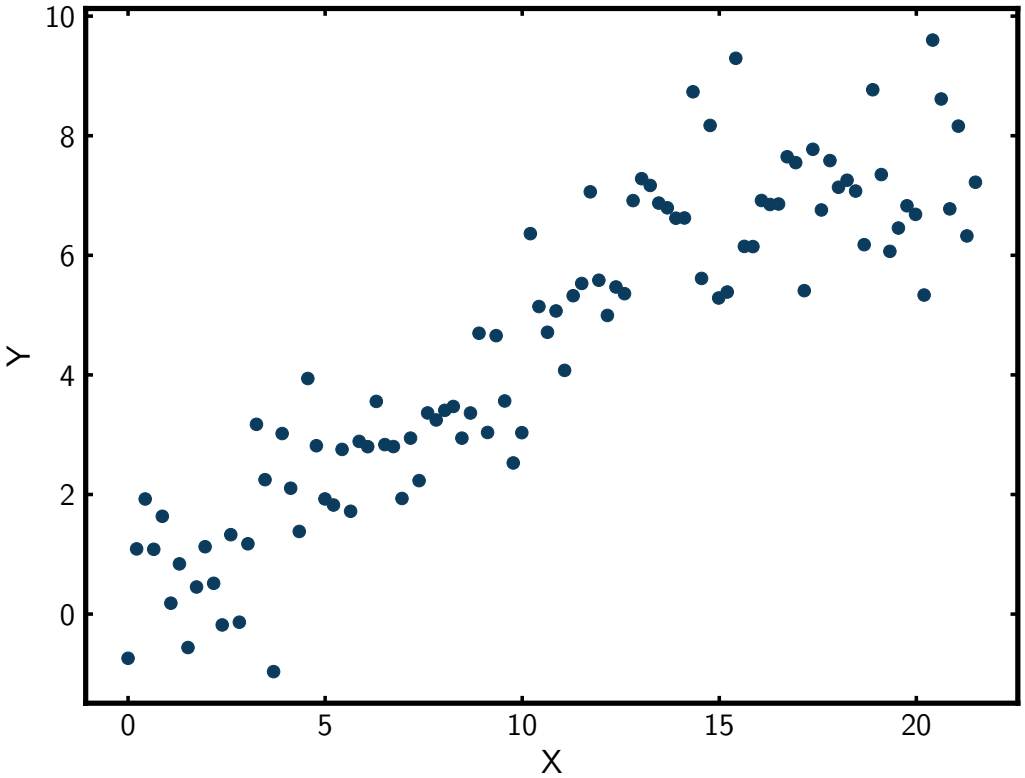
DIFFPY.CMI

diffpy.cmi

- Who is it for?
- What does it help them do?
- How does it work?
- How do I get it?



Anatomy of a regression

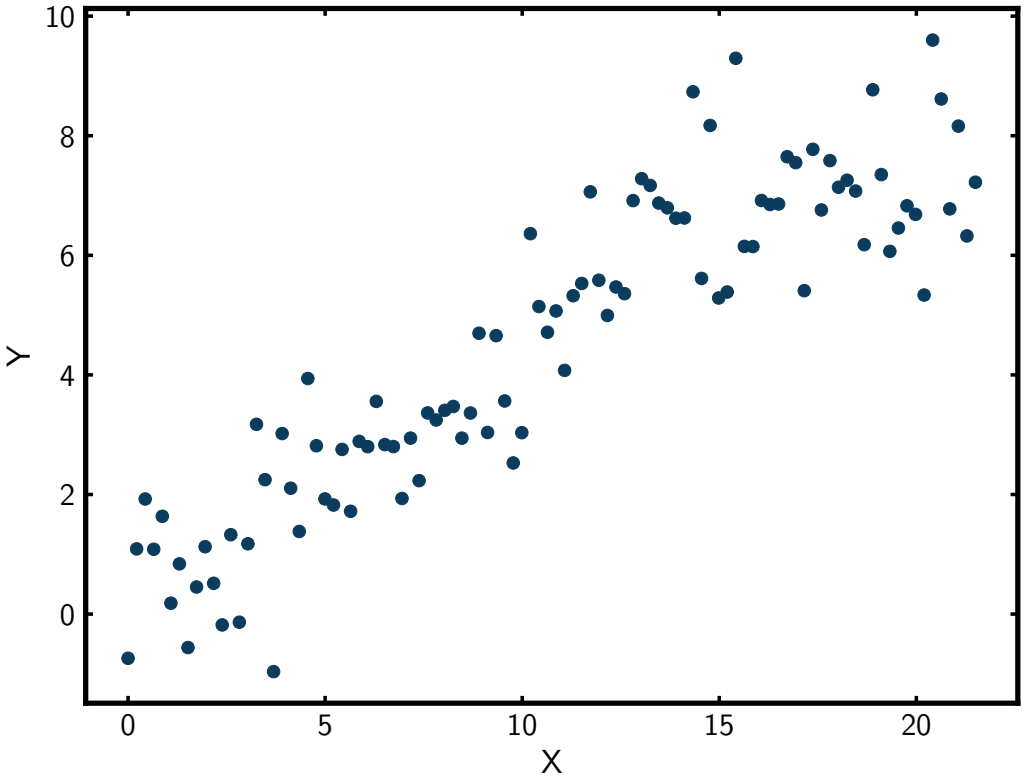


What is the task?

- I have measured some data
- It contains a signal and some noise
- I would like to know the signal

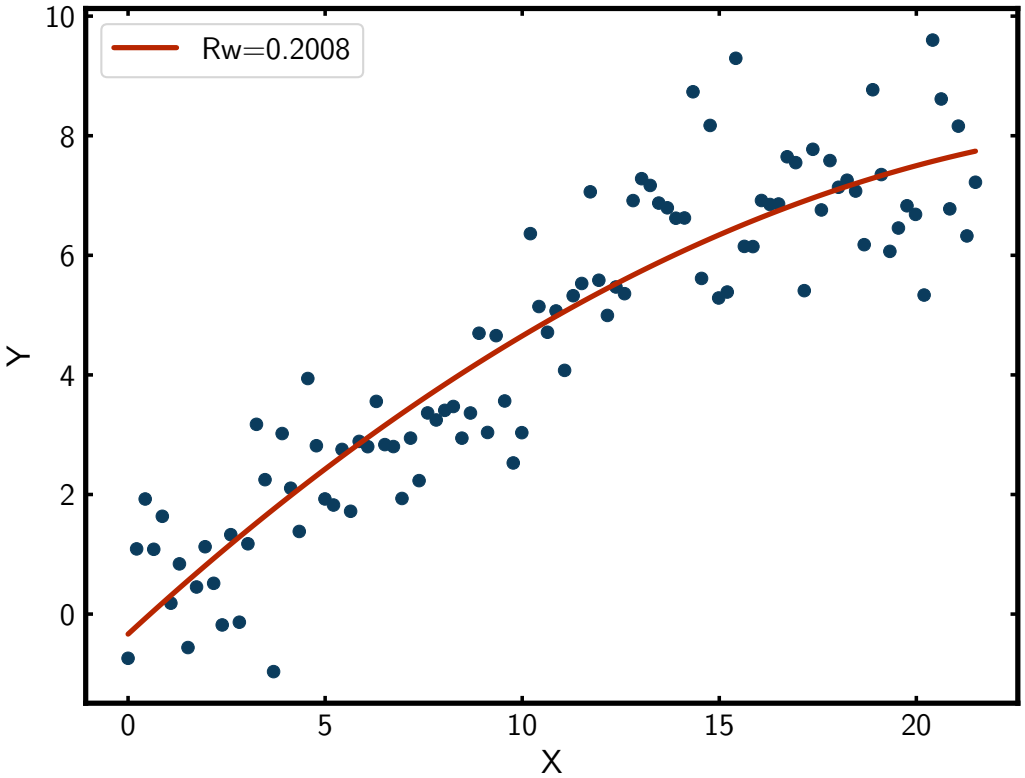
The signal is "the physics" of the problem

How do I get the signal?



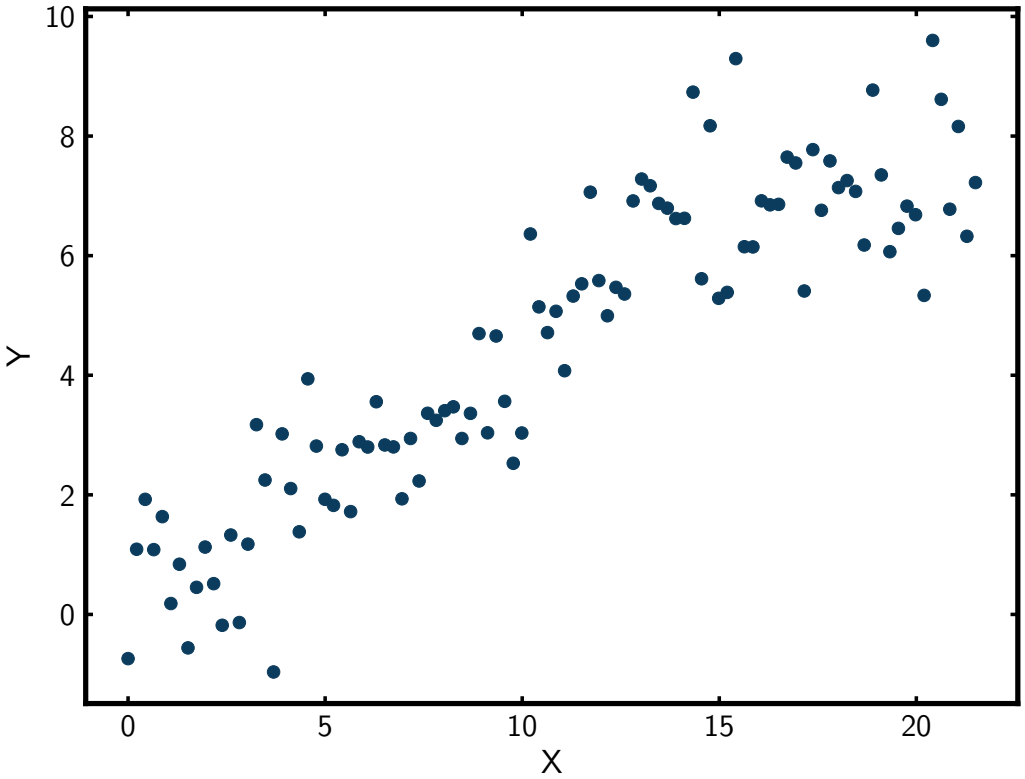
Simple, just draw a smooth line through the noisy data!
That's it, the smooth line you draw is the signal. Humans are good at this!

How do I get the signal?



Simple, just draw a smooth line through the noisy data!
That's it, the smooth line you draw is the signal. Humans are good at this!

What if I want to design a machine to do this instead?

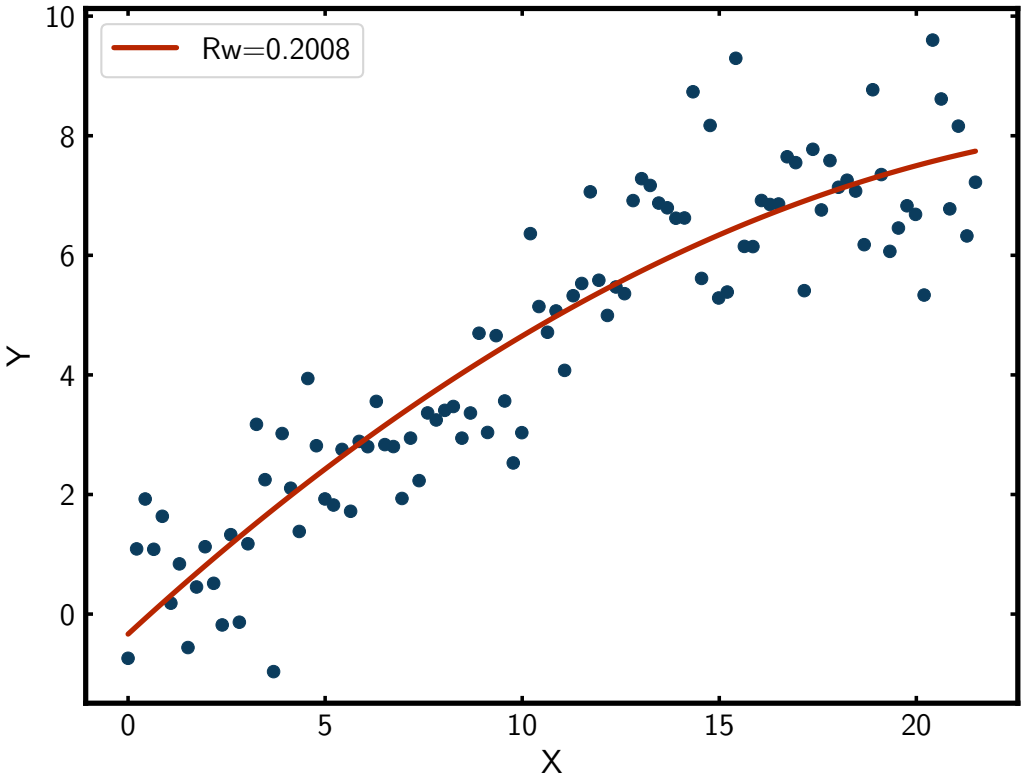


Have the machine mimic the human?

- How did I do it?
- Used my feelings and my inate sense or aesthetics

but machines don't have feelings or inate senses of aesthetics.

We have to be a bit clever...



Assumption:

- there is a function, $f(x)$, that maps the point at each value of x to a value of y
- i.e., $y = f(x)$
- We can call $f(x)$ the "the physics function" because it is the "physics" of the problem.

Task is now to discover $f(x)$.

Task is to discover $f(x)$

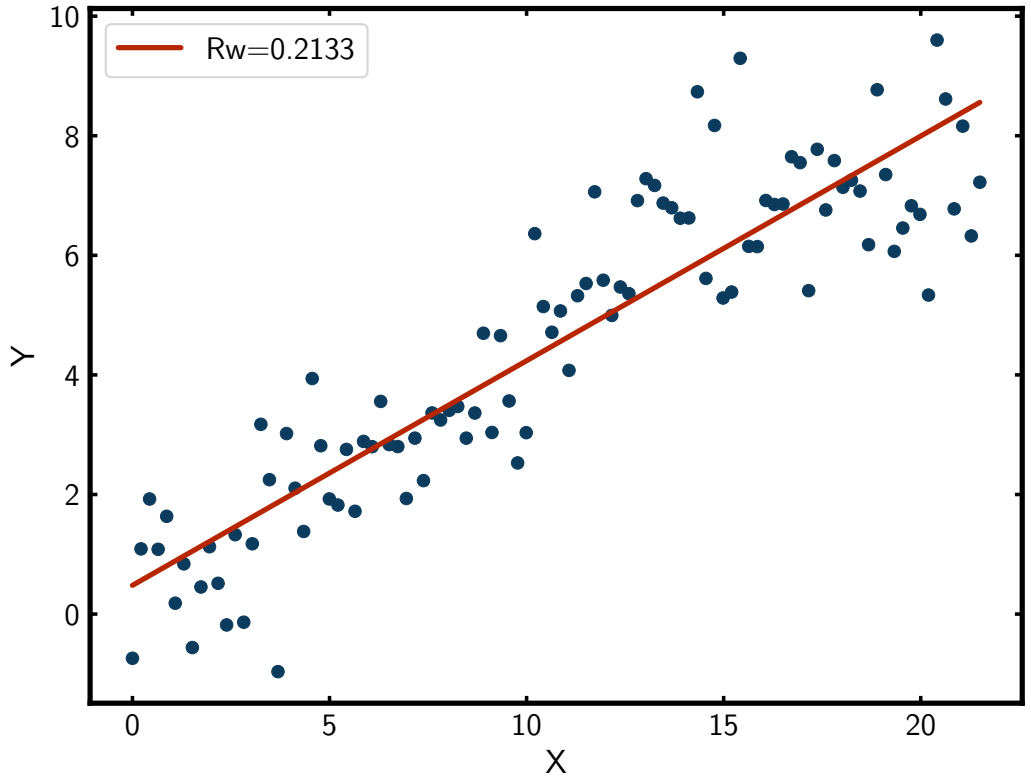
We often have a functional form for $f(x)$, e.g., from some known physics,

- let's say $f'(x, \mathbf{p}')$, that depends on a bunch of parameters, $\mathbf{p}' = \{p'_1, p'_2, \ldots, p'_n\}$
- e.g. f might be diffraction or PDF equations
- New Task: to estimate the "best values", for the parameters in $\mathbf{p}'$
- Note, another possible task is deciding if the function $f'(x, \mathbf{p}')$ is best, or maybe some other slightly different physics functions, $f''(x, \mathbf{p}'')$, $f'''(x, \mathbf{p}''')$ and so on, are better.

The former task is called "parameter estimation" and the latter is called "model selection" (more on this later)

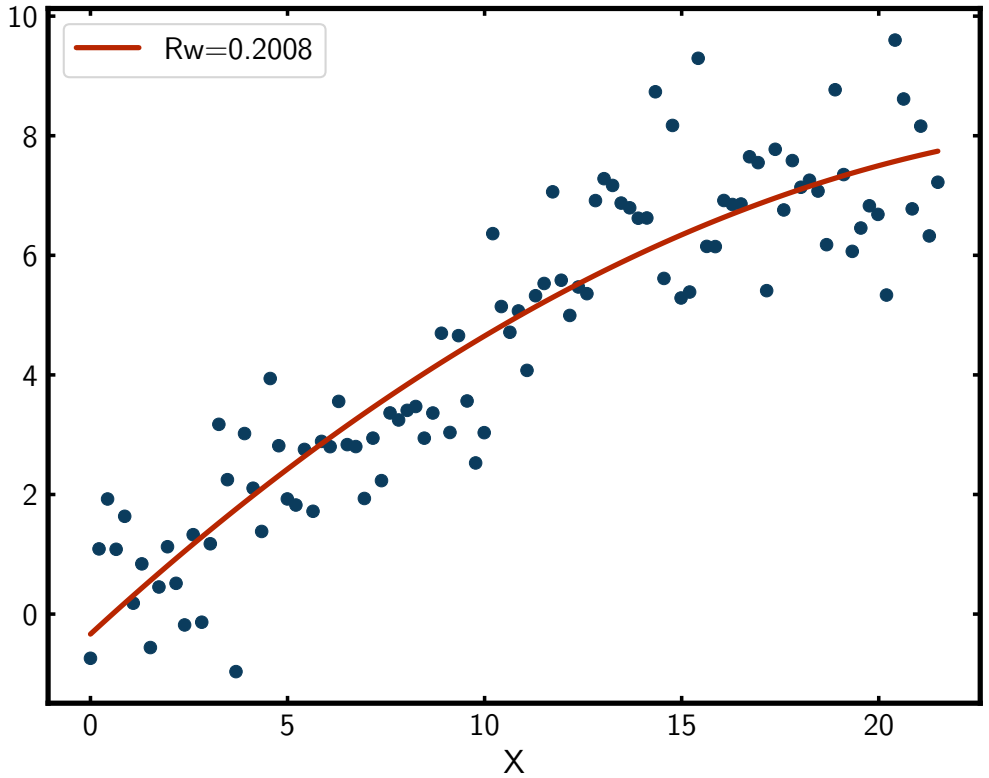
Task is to discover $f(x)$

- $f'(x, \mathbf{p}') = mx + c$



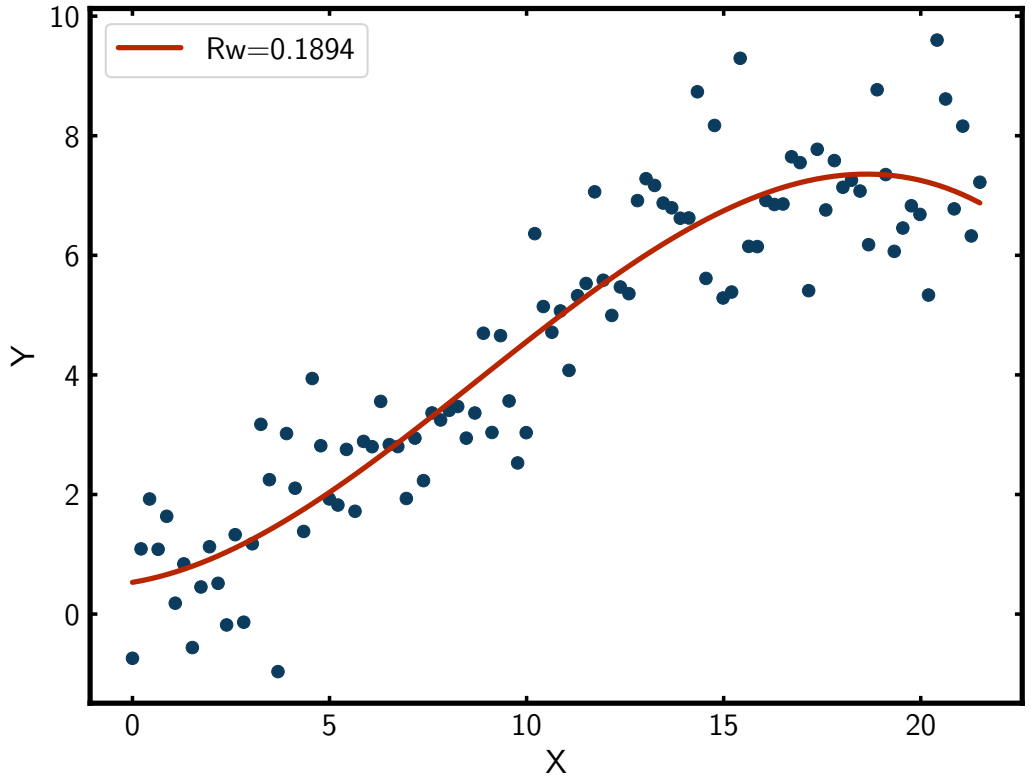
Task is to discover $f(x)$

- $f''(x, \mathbf{p}'') = p_2 x^2 + p_1 x + p_0$



Task is to discover $f(x)$

- $f'''(x, \mathbf{p}''') = p_3 x^3 + p_2 x^2 + p_1 x + p_0$



But how do we persuade a machine to estimate the parameters?

Machines are kind of dumb.

- We really only know how to make machines do a few things...
- find extrema (maxima and minima) of functions
- (we also know how to have machines apply logic and do mathematical operations)

But how do we persuade a machine to estimate the parameters?

We have to make every problem look like a function extremization problem!

- We don't want to find the maximum of our physics function,
- so we need another function that has the property "the function is extremized when the values of the parameters are "the best they can be"."

We call this function, the Objective function.

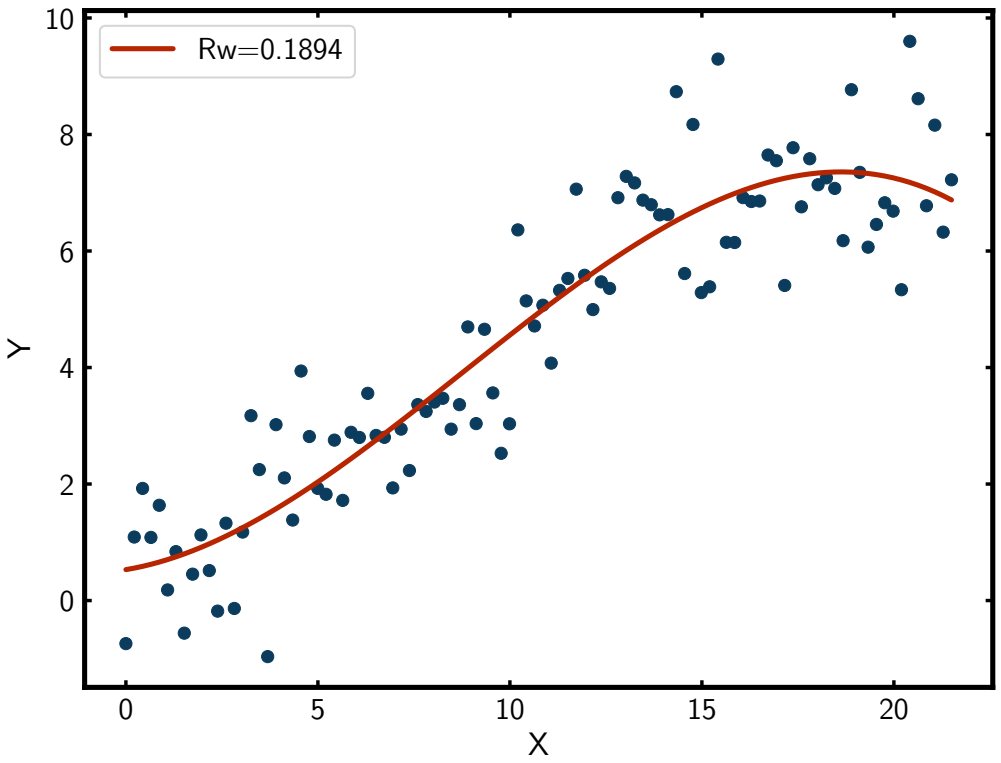
Objective function

An Objective function is any function that is at an extremum when we achieve our objective.

- The Objective function is not unique. We can choose it.
- Common ones are "mean square error (L2 norm)", "absolute error (L1 norm)", χ^2, R_{wp}, but there are many many variants. Adding constraints and restraints modifies the Objective function.
- The combination of the "physics function" and the "objective function" we can call the "math(s) model" or just "the model".

Important: every time you change the objective function, even if you don't change the physics function, you are changing the problem that you are solving. This is not well understood by people.

How does the machine find the extrema of the objective function?



The machine finds the extrema of the objective function using a regression engine

- The regression engine, or regressor for short, is just an algorithm (and code) the machine uses for finding the optimum of the objective function
- examples are Newton's method, Levenberg-Marquart, BFGS, Metropolis, ...
- Why is there more than one regressor?

Why is there more than one regressor?

- different regressors work better in different situations with different models, for example, depending on the shape of the objective surface as a function of $\mathbf{p}$, the number of parameters, the amount of data, the nature of any constraints, and so on.
- Newton's method (if you have analytic derivatives)
- Levenberg-Marquart, BFGS (better), conjugate gradient if you don't
- Nelder-Mead/Powell if data are noisy,
- Metropolis (a variant is used in RMC), stochastic gradient descent, if you have many parameters

Why is there more than one regressor?

Important: Regressors can

- Converge to the right solution
- Converge to the wrong solution (e.g., a metastable minimum),
- Or not converge.
- But **if they converge to the right solution, they will all give the same answer** to the same model. The model is where you are going, the regressor is the vehicle you are travelling in.

Simon says: Describe your **model** in detail not your regressor. Mention your regressor in the context of performance and convergence.

Last Slide: don't mix up parameters and variables

We tend to use the words parameters and variables interchangeably but keep them separate in your mind.

- parameters are the $\mathbf{p}' = \{p'_1, p'_2, \ldots, p'_n\}$ that appear in your physics function
- variables are things that the regressor varies
- but wait, doesn't the regressor vary the parameters to find the optimal values?
- yes and no. Variables are mapped on to parameters through constraint equations
- e.g., $V_1 = p_1$ would be the most simple such relation
- but if we have $V_1 = a$, $V_1 = c$, $V_2 = b$, (and $\alpha = \beta = \gamma = 90$ and not varying) this implies a tetragonal symmetry. Here we have six parameters but only two variables in our model.

Summary

- Define your model carefully, including the physics description **and** the objective function including all constraints and restraints
- mention which regressor you are using. If you like, compare a few different regressors to get a better feeling about their performance and convergence characteristics (this is done very rarely but should be done more often)
- Remember, when you change anything in your model (objective or physics) you are solving a different problem

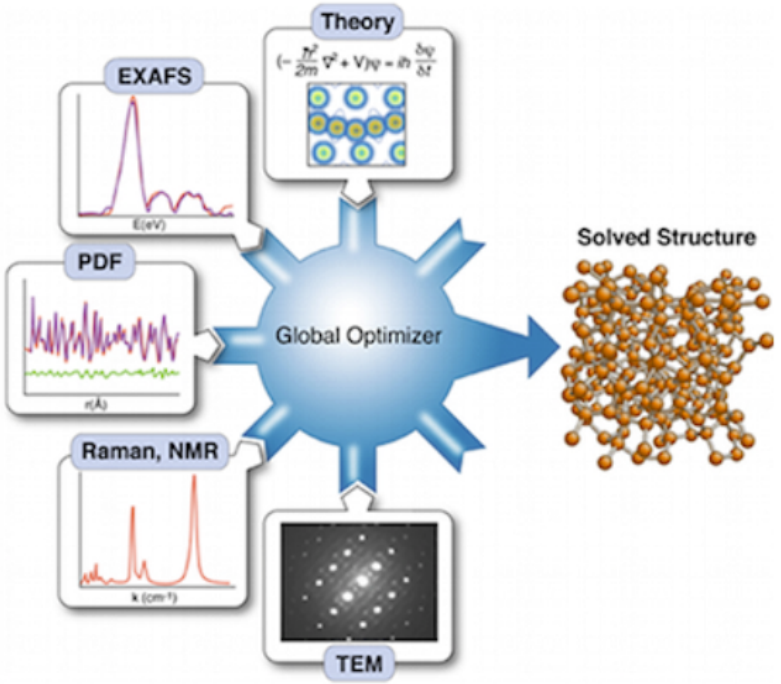
All models are wrong, some models are useful (George Box). Get into this mindset and use your models to learn something you want to know (i.e., make them useful).

DIFFPY.CMI

Who is it for?

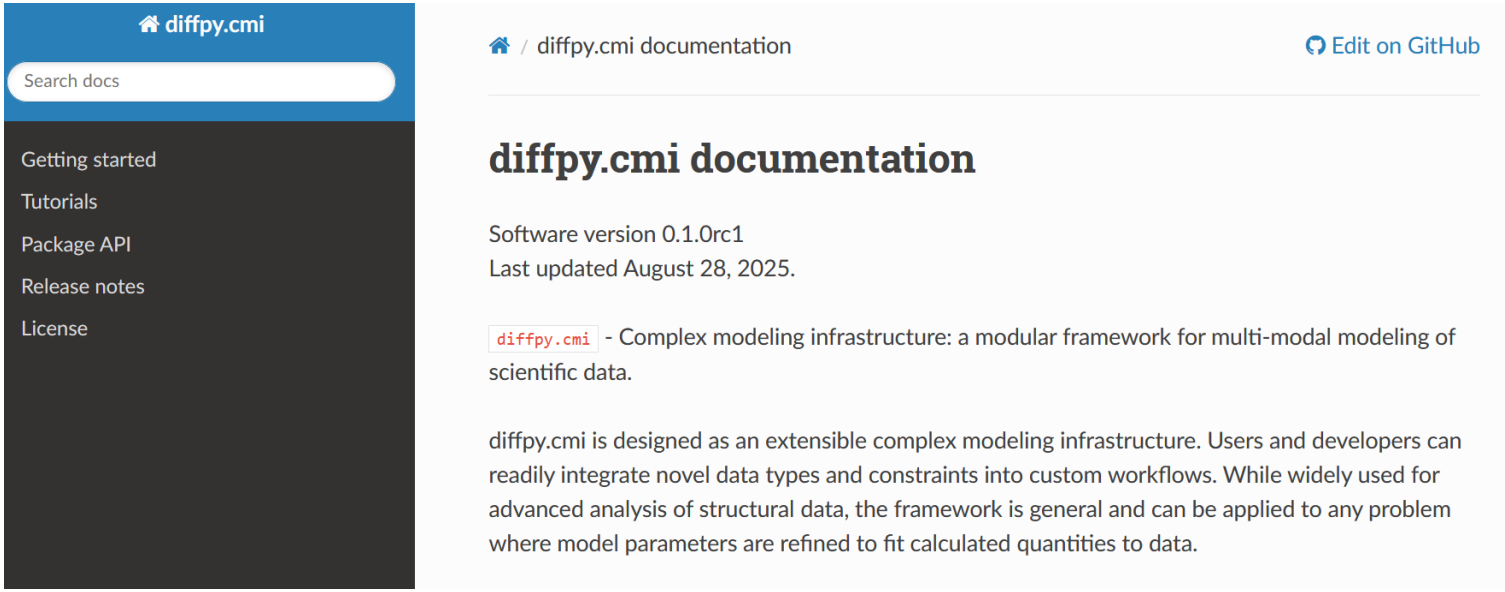
DIFFPY.CMI

What does it help them do?



DIFFPY.CMI

How do I get it?



1. `conda install diffpy.cmi`
2. diffpy.org/diffpy.cmi
3. (https://github.com/diffpy/diffpy.cmi)

DIFFPY.CMI

diffpy.cmi

Search docs

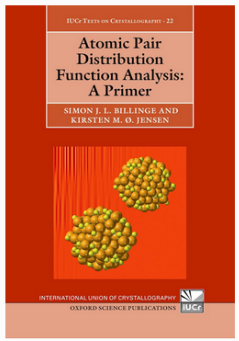
- Getting started
  - Installation
  - Pack and Profile Installation
  - Data and Examples
- Tutorials
- Package API
- Release notes
- License

Getting started / Edit on GitHub

Getting started

For detailed instructions and in-depth examples of modeling with `diffpy.cmi`, we highly recommend the book,

Atomic Pair Distribution Function Analysis: A Primer by Simon J. L. Billinge and Kirsten M. Ø. Jensen (Oxford University Press, 2023).



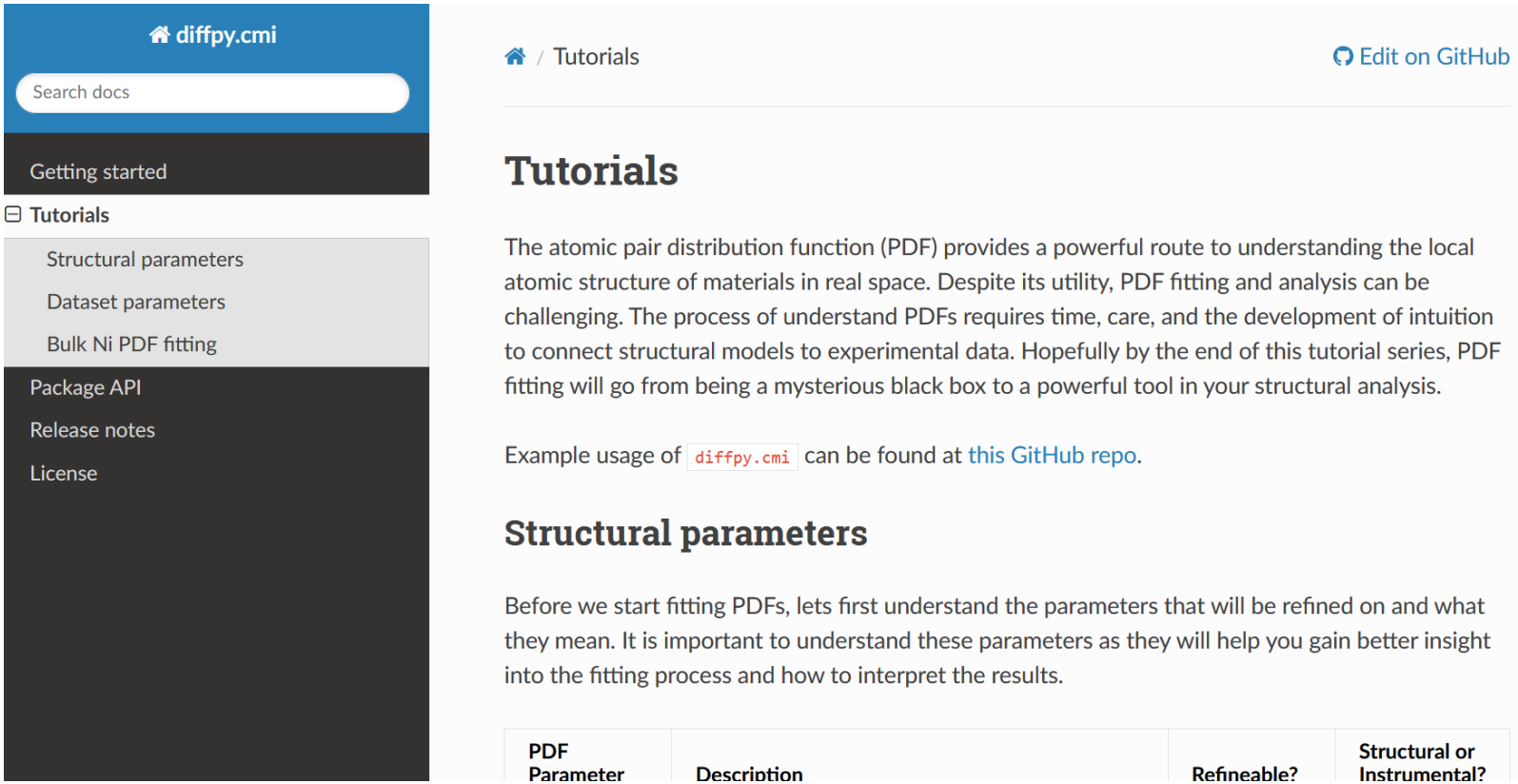
To purchase this book, please visit this link.

Installation

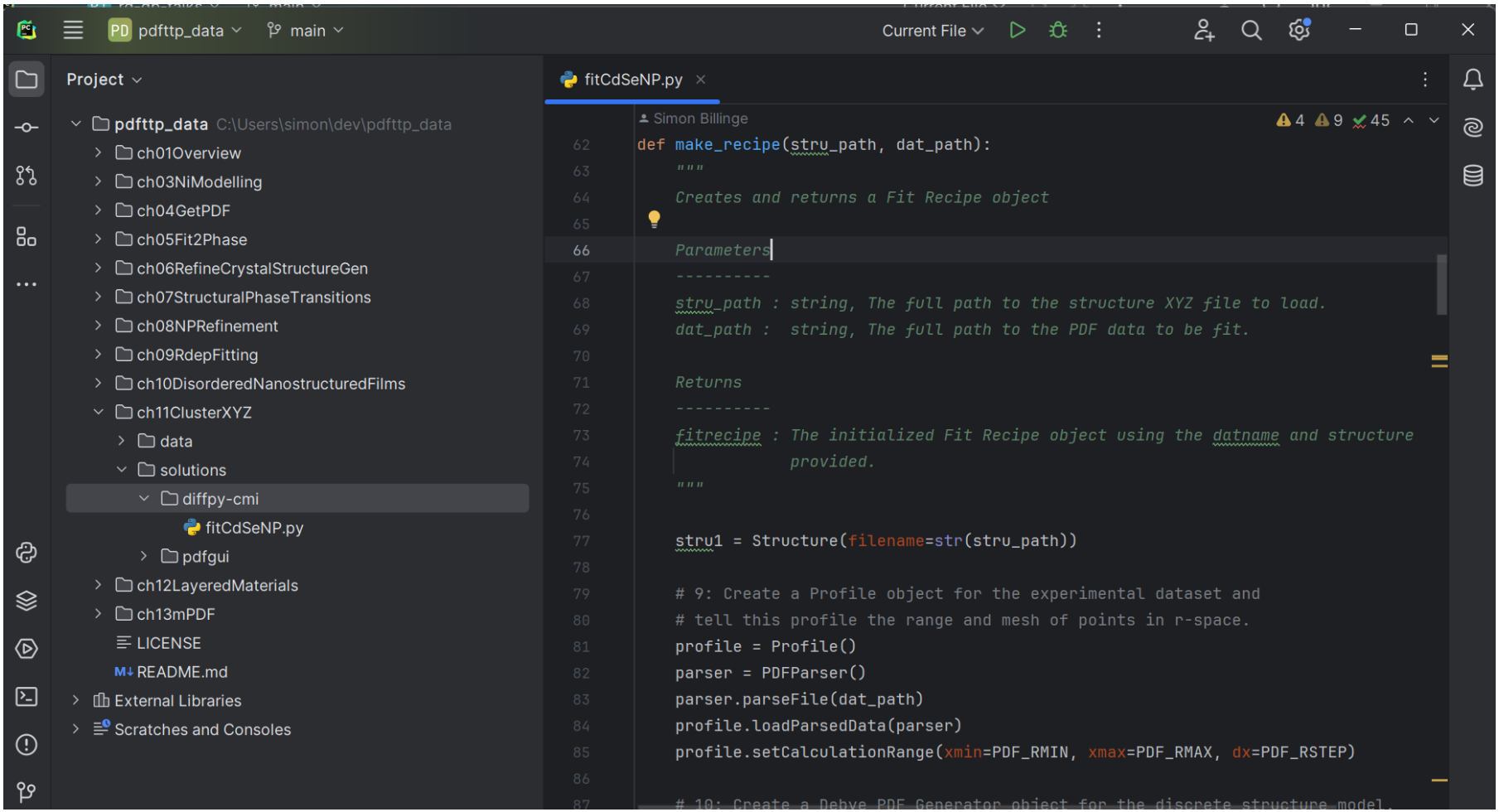
To install `diffpy.cmi`, create a new conda environment or activate an existing environment and install the package from the conda-forge channel.

```
conda create -n diffpy.cmi-env
conda install -c conda-forge diffpy.cmi
conda activate diffpy.cmi-env
```

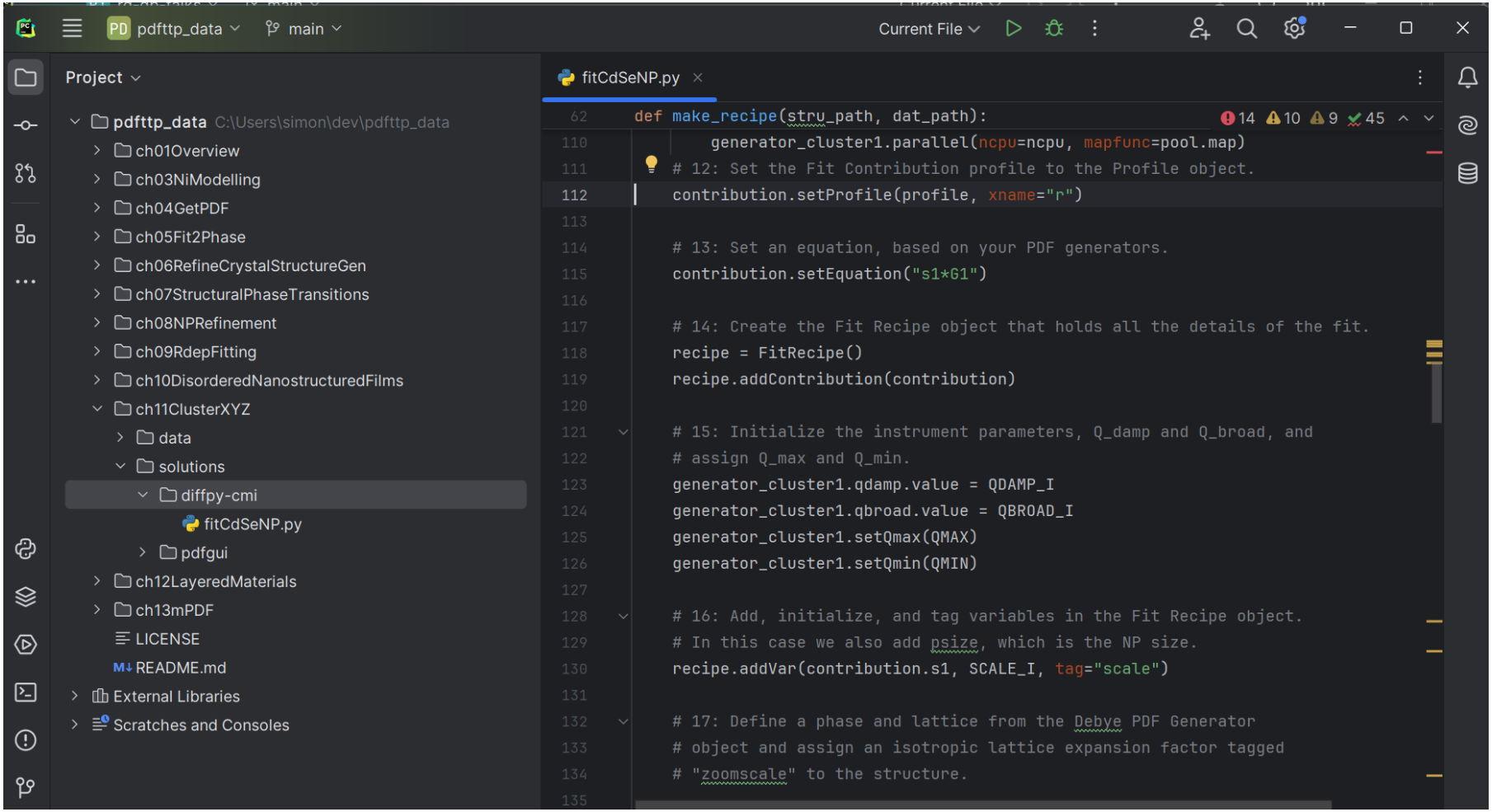
DIFFPY.CMI


diffpy.cmi
Search docs
Getting started
Tutorials
Structural parameters
Dataset parameters
Bulk Ni PDF fitting
Package API
Release notes
License
Tutorials
Edit on GitHub
Tutorials
The atomic pair distribution function (PDF) provides a powerful route to understanding the local atomic structure of materials in real space. Despite its utility, PDF fitting and analysis can be challenging. The process of understand PDFs requires time, care, and the development of intuition to connect structural models to experimental data. Hopefully by the end of this tutorial series, PDF fitting will go from being a mysterious black box to a powerful tool in your structural analysis.
Example usage of diffpy.cmi can be found at this GitHub repo.
Structural parameters
Before we start fitting PDFs, lets first understand the parameters that will be refined on and what they mean. It is important to understand these parameters as they will help you gain better insight into the fitting process and how to interpret the results.
PDF Parameter
Description
Refineable?
Structural or Instrumental?

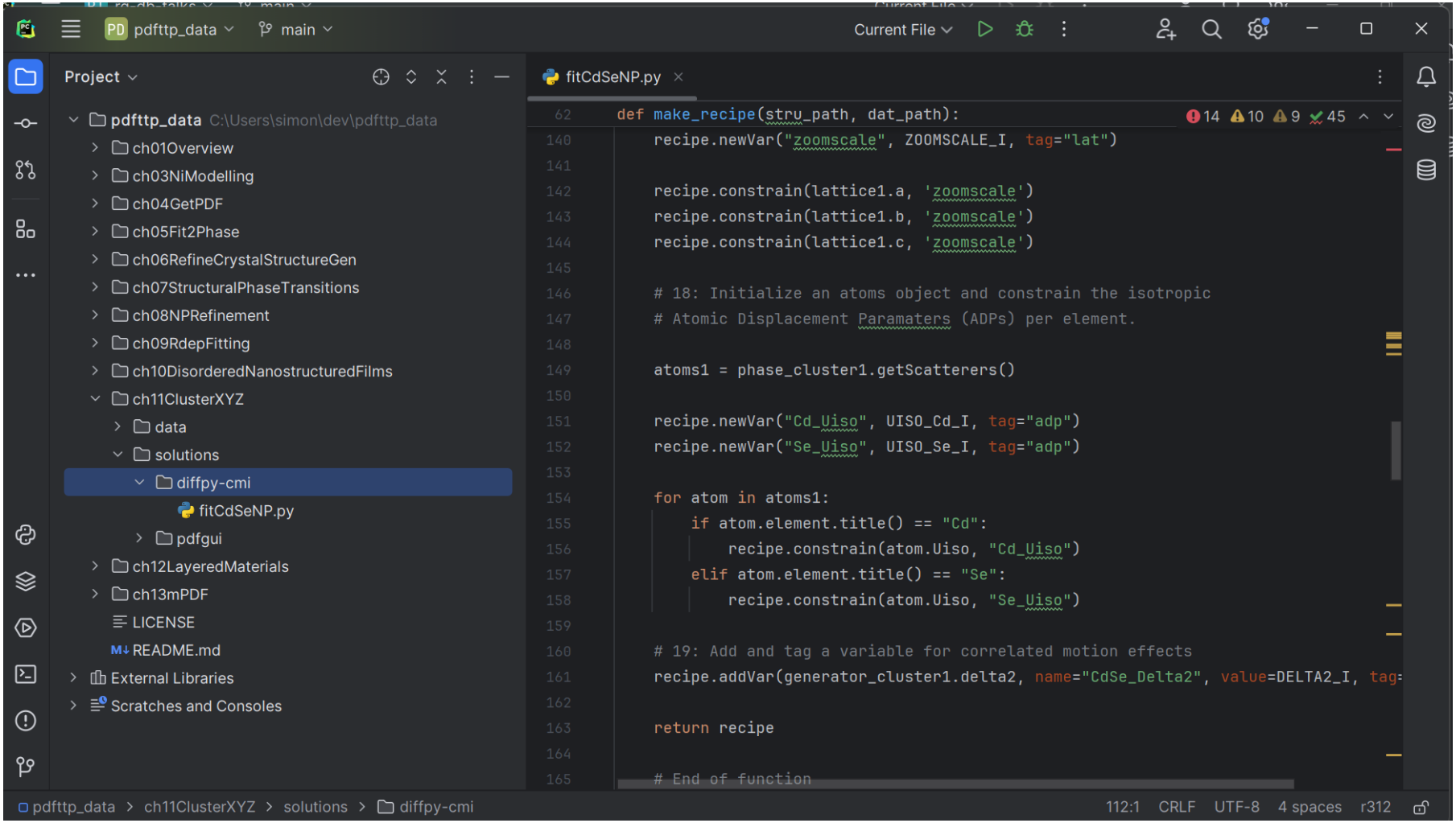
Diffpy-cmi script example



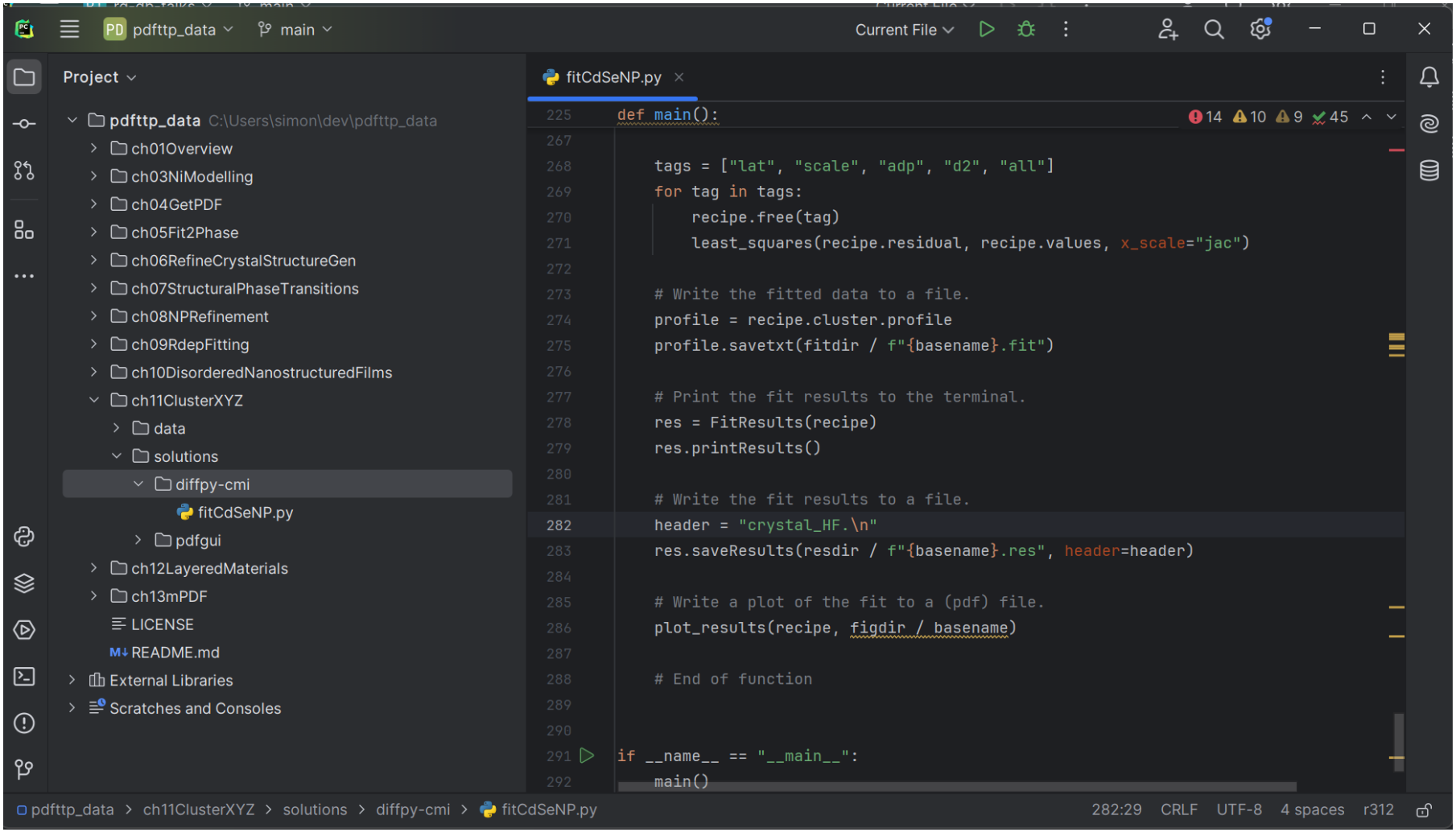
Diffpy-cmi script example



Diffpy-cmi script example



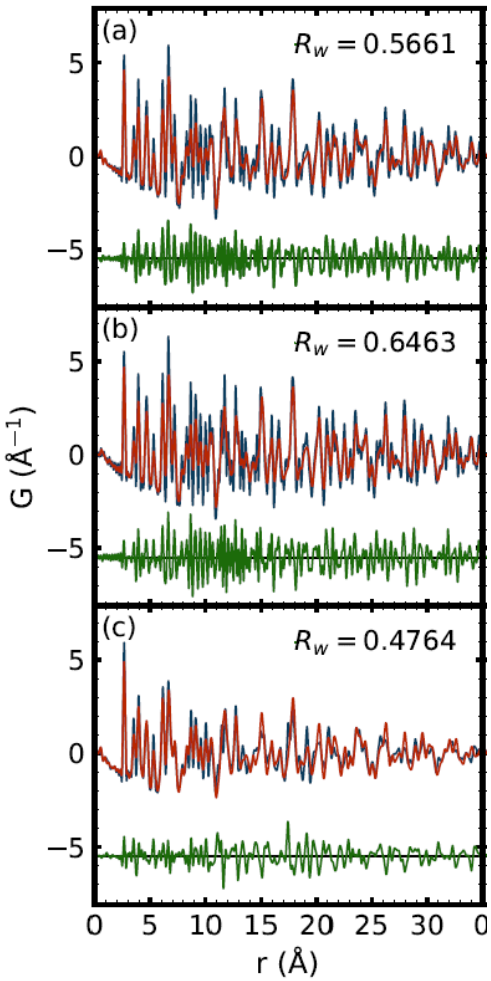
Diffpy-cmi script example



DIFFPY.MORPH

DIFFPY.MORPH

Three samples, which has a phase transition?

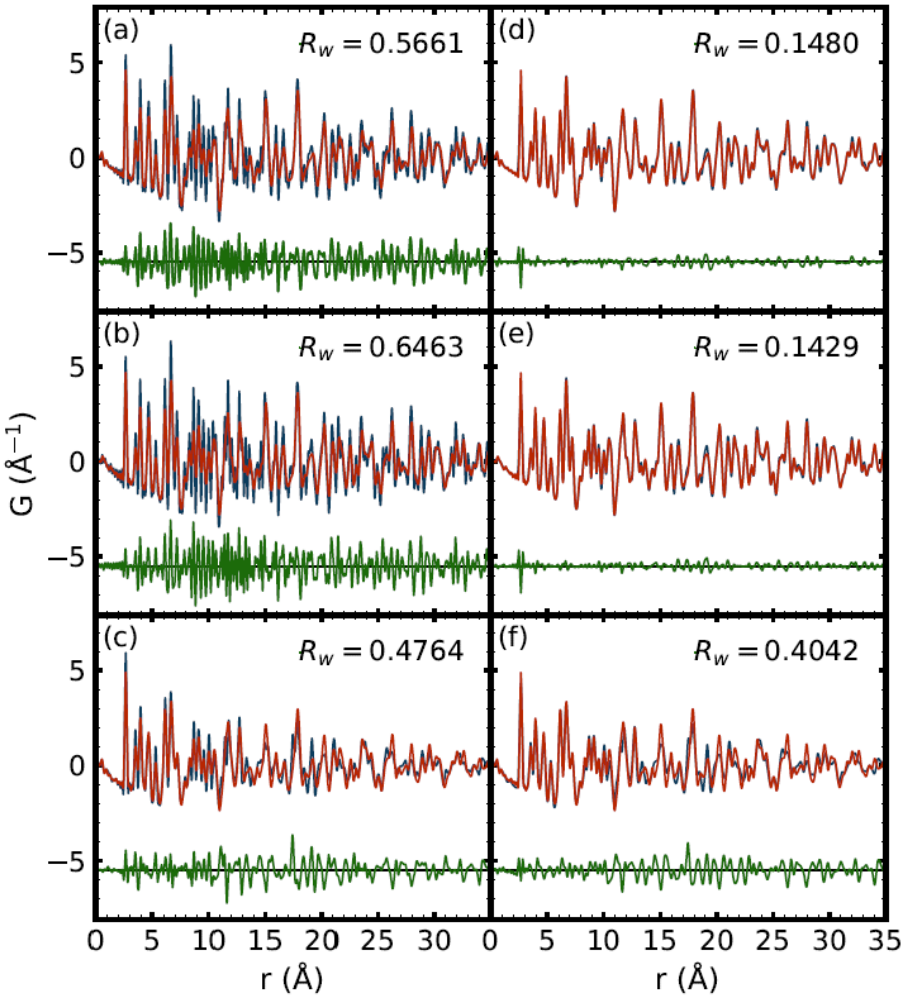


Three samples, comparing low (blue) and high (red) temperature PDFs:

- $Ir_{0.8}Rh_{0.2}Te_2$
- $Ir_{0.95}Pt_{0.05}Te_2$
- $IrTe_2$

which has a phase transition?'

Three samples, which has a phase transition?

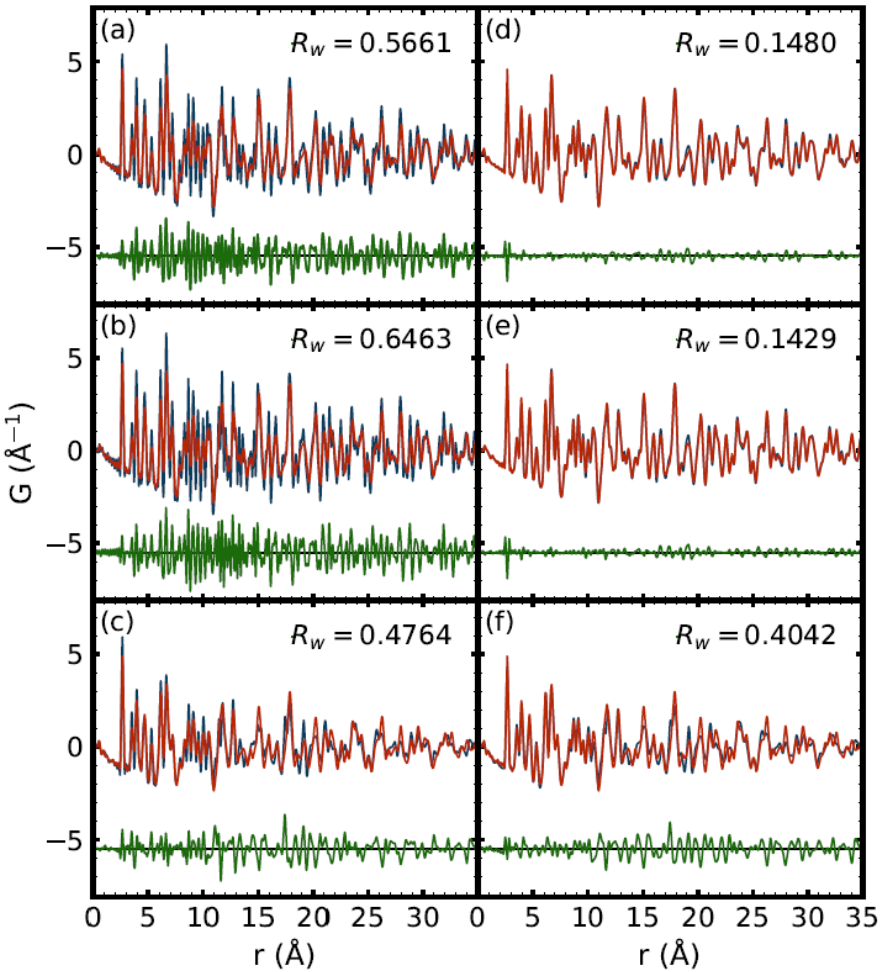


Three samples, comparing low (blue) and high (red) temperature PDFs:

- $Ir_{0.8}Rh_{0.2}Te_2$
- $Ir_{0.95}Pt_{0.05}Te_2$
- $IrTe_2$

$IrTe_2$ has a phase transition!

How does diffpy.morph work?

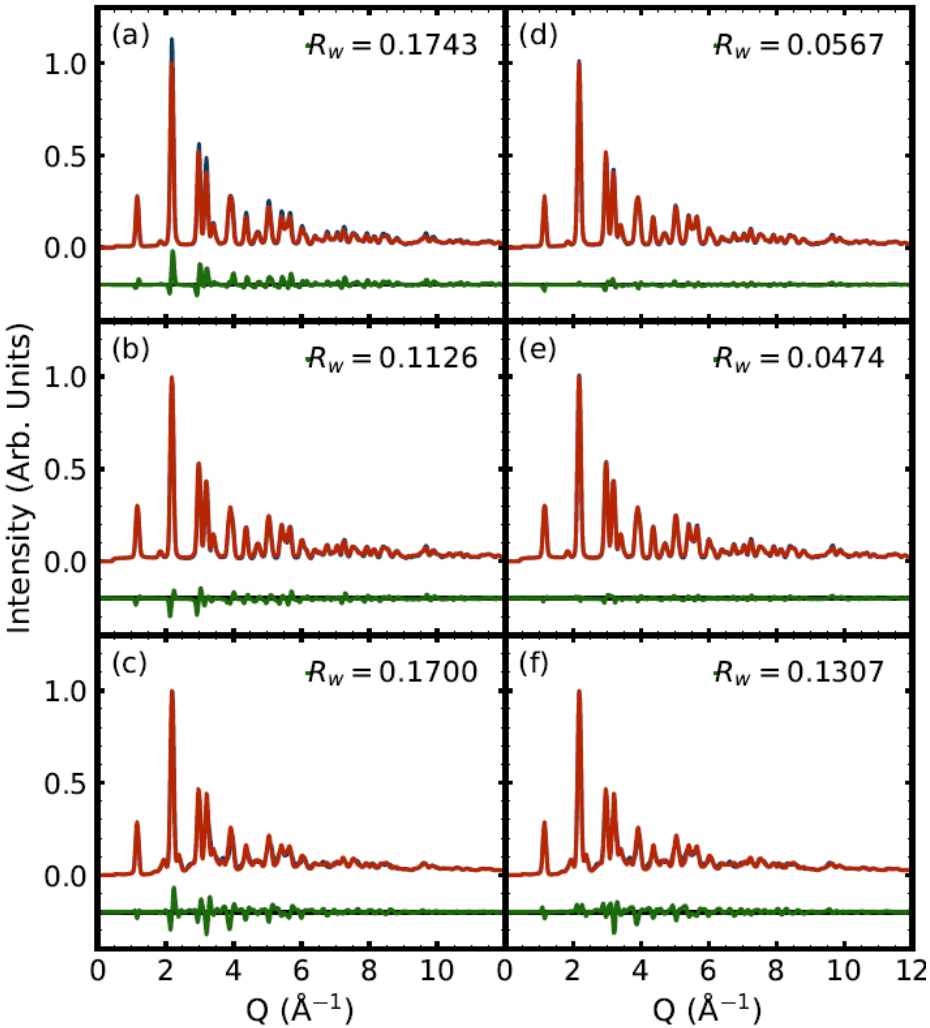


Apply "morphs" to account for "boring" effects such as thermal expansion

- `stretch` morph (linearly) stretches the x-axis
- `scale` morph scales the y-axis
- `smear` morph convolutes with a Gaussian to change peak width
- Vary the magnitude of the morphs until a "morphed" signal best matches a "target" signal.

In the PDF, `stretch` mimics thermal expansion, `smear` mimics increased thermal motion

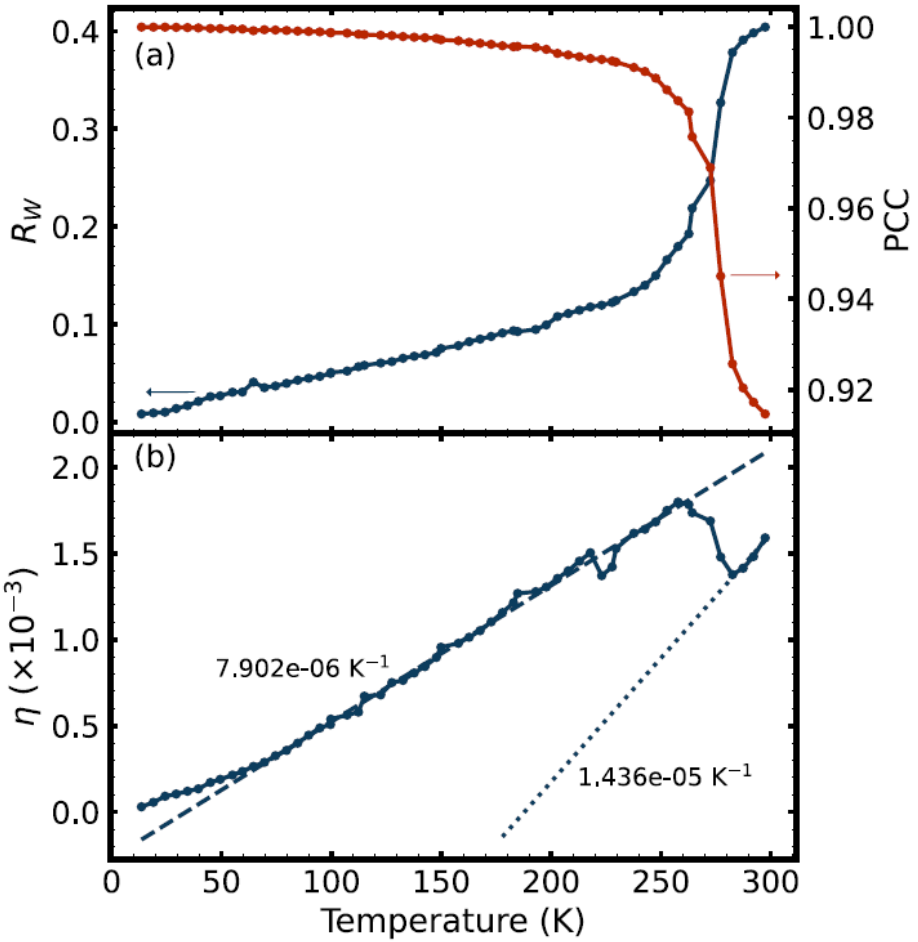
Does it just work on PDF?



No, it works on diffraction data too

in fact, it works on any signal in principle

I have a folder full of T-dep data

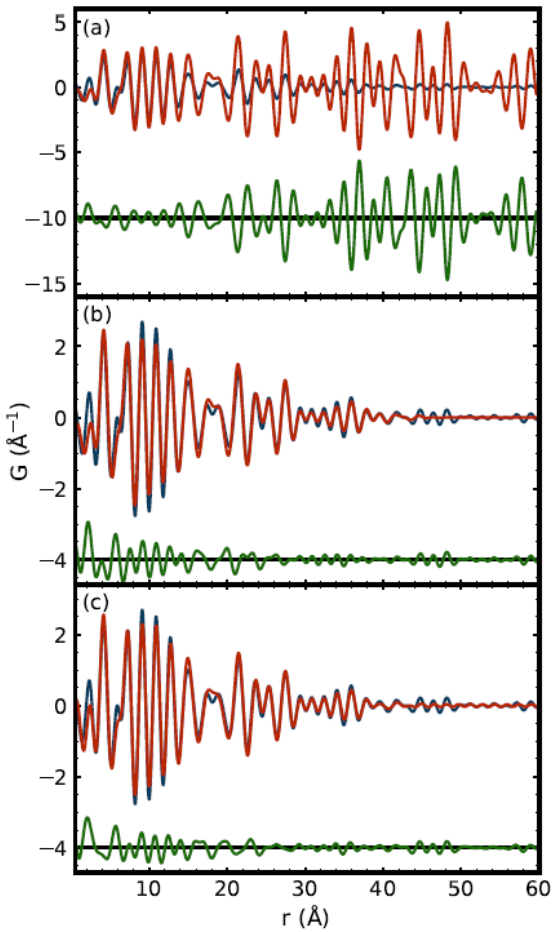


Use diffpy.morph to extract the thermal expansivity

- $IrTe_2$

in fact, find T_c and get the thermal expansivity of each phase

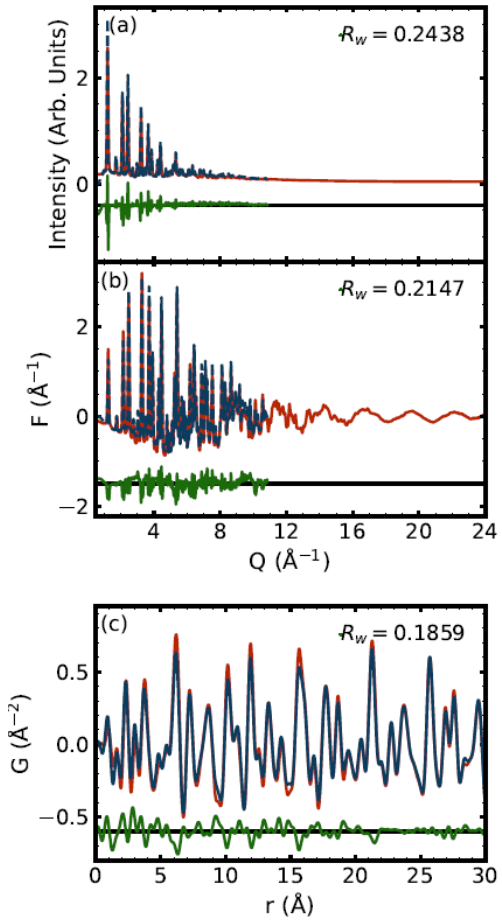
Let's get greedy and have diffpy.morph do more and more



Additional morphs

- particle shape function in the PDF, refine (major and minor) radius
- `shift` in `x` and `y`
- non-linear (polynomial) stretch (`squeeze`)
- `funcx`, `funcy`, `funcxy` - user-defined python function morphs
- multiple morphs (chain the morphs)

Let's get greedy and have diffpy.morph do more and more



Fancy use of `funcxy`

- let's use diffpy.morph to find optimal parameters in PDFgetX3!
- Give it a target $F(Q)$ (e.g., measured at a synchrotron or calculated)
- Give it `pdfgetter()` as the function in `funcxy`.
- Have it vary `rpoly`, `background scale`, etc.

Scikit Package

Scikit Package

- Who is it for?
- What does it help them do?
- How does it work?
- How do I get it?

Scikit Package

Who is it for?

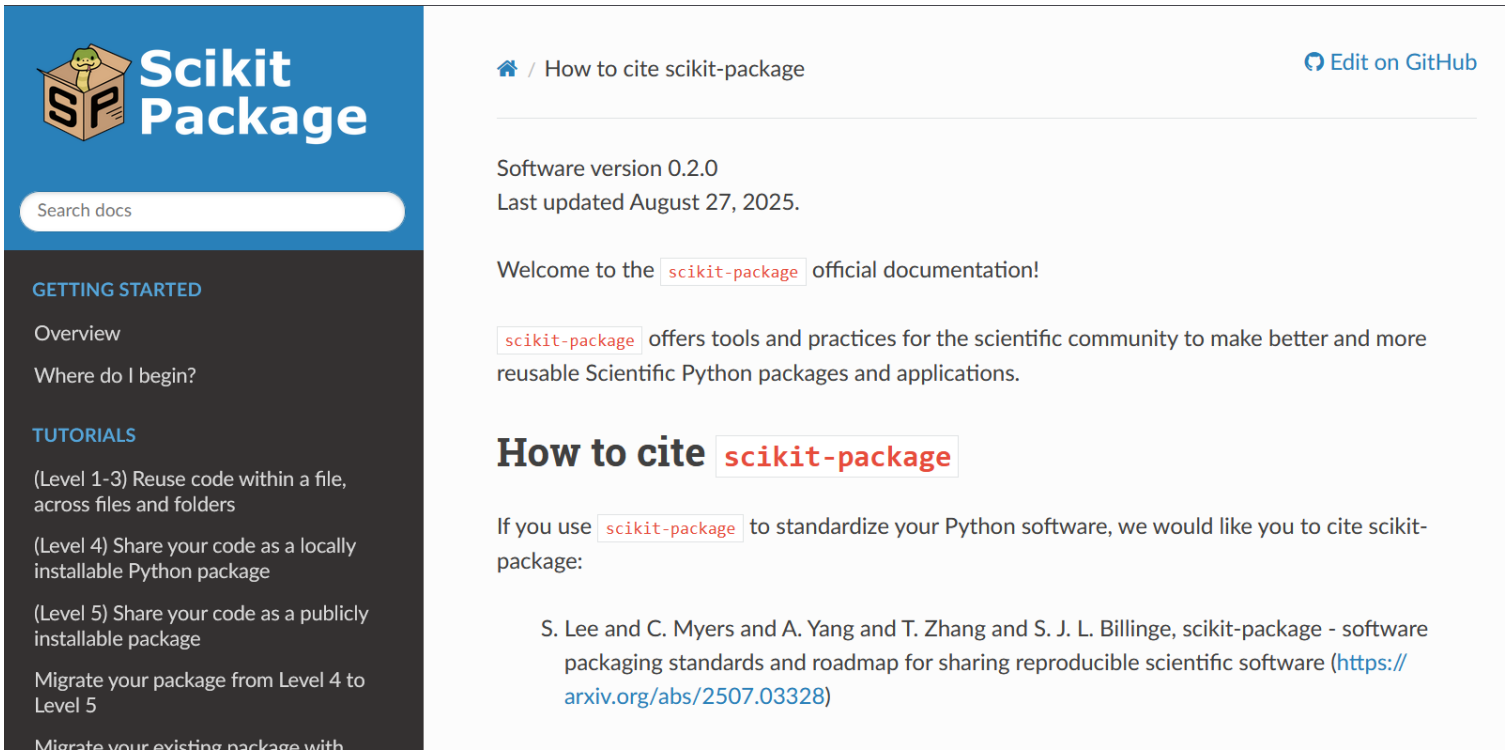
Scikit Package

What does it help them do?

"I wrote a software program and sold it for two million dollars."

Scikit Package

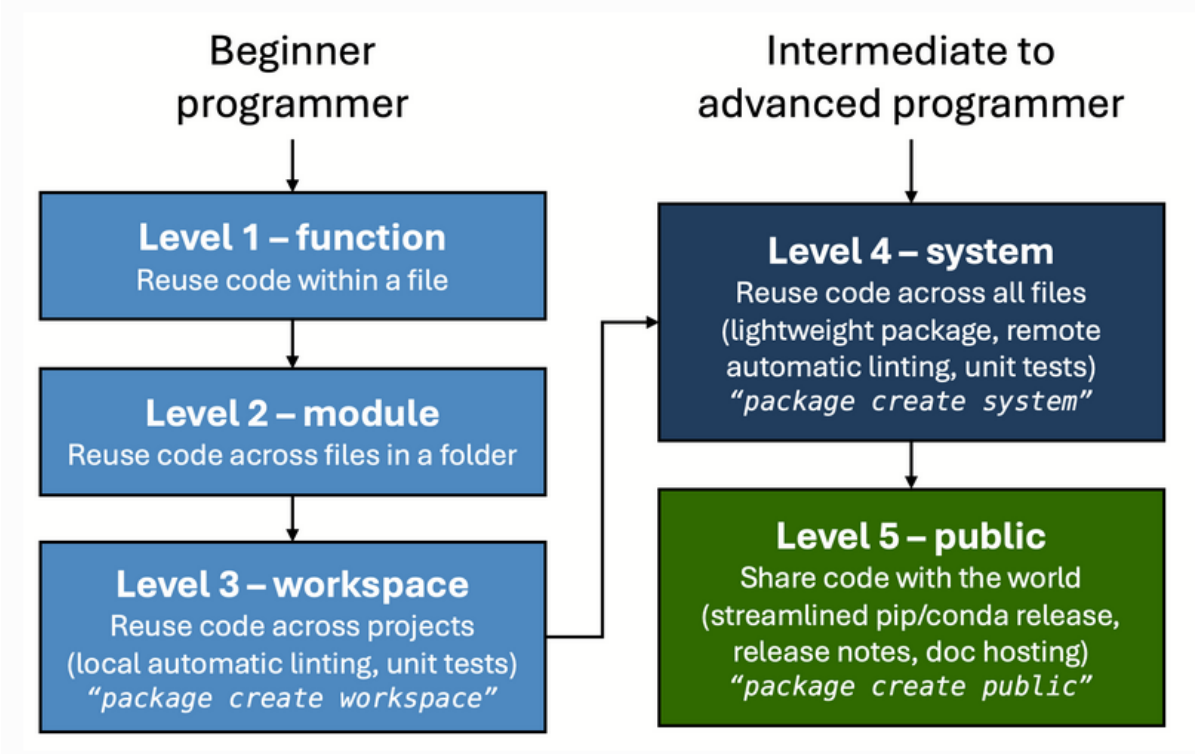
How do I get it?



1. `conda install scikit-package`
2. scikit-package.github.io/scikit-package
3. (https://github.com/scikit-package/scikit-package)

Scikit Package

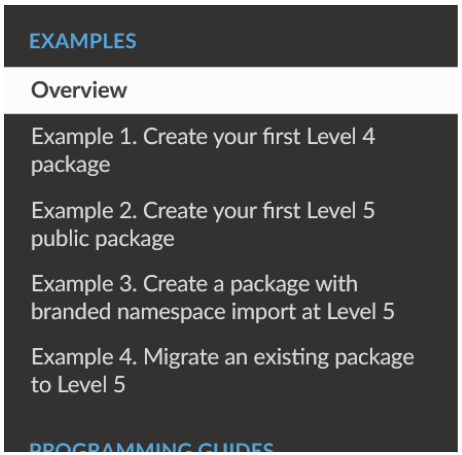
How does it work?



Scikit Package

- Level 1, `function`, which is already widely used, consists of simply defining functions within the same file or module.
- Level 2, `module`, expands on Level 1 by reusing functions across separate module files within the same directory.
- Level 3, `workspace`, restructures the organization so that a block of code can be reused across multiple projects.
- Level 4, `system`, enables users to create a lightweight package so that the code can be reused across all files locally.
- Level 5, `public`, is the final step, where the source code is uploaded online so that anyone in the world can install the package, sourced from PyPI or conda-forge.

Scikit Package



Overview

Here we provide 4 representative examples to illustrate how to use `scikit-package` for building shareable packages at Levels 4 and 5. We also include an example of how to use `scikit-package` to migrate your existing packages to the `scikit-package` standards.

- Example 1. Create your first Level 4 package
- Example 2. Create your first Level 5 public package
- Example 3. Create a package with branded namespace import at Level 5
- Example 4. Migrate an existing package to Level 5

Scikit Package

Example 1. Create your first Level 4 package

In this example, we assume **Mr Neutron** previously initiated a project called `diffraction-utils` using `scikit-package` Level 3 and developed a shared class called `DiffractionObject`. This `DiffractionObject` is used in code analyzing various diffraction data sourced from x-ray, neutron, and electron instruments. This class is written in a module called `diffraction_objects.py` which is reused in Python scripts in the `scattering` sub-project folder. **Mr Neutron**'s folder structure looks like the following:

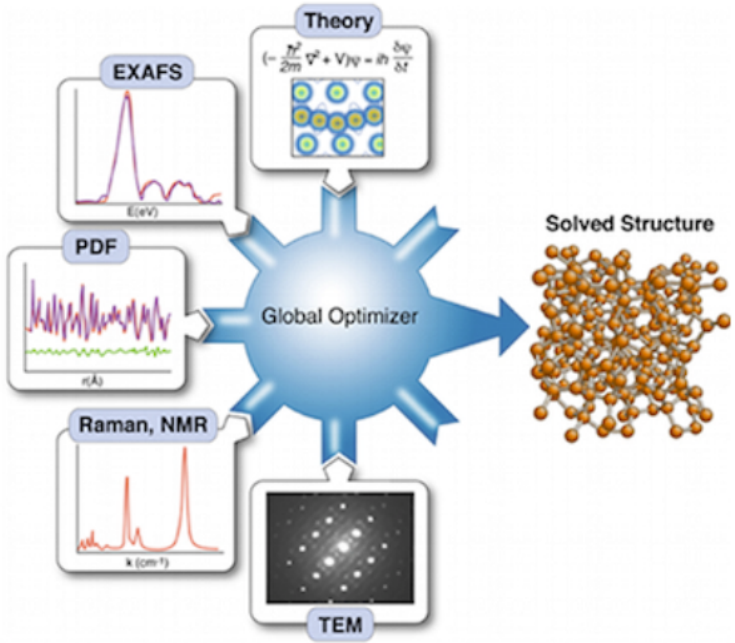
```
somewhere/
    |-- on/
        |-- my/
            |-- computer/
                |-- diffraction-utils/
                    |-- README.md
                    |-- diffraction_objects.py
                    |-- scattering/
                        |-- __init__.py
                        |-- neutron.py
                        |-- xray.py
                        |-- electron.py
```

Since the project is a data analysis project, **Mr Neutron** followed common practice to place this

DIFFPY.CMI TUTORIAL

Quick view of the workshop tomorrow/Wednesday

- Live demonstration by Caden Myers
- Installation
- Hands-on examples for you to code
- Simple fitting of a user-defined function
- Structural fitting to a PDF
- T-dep fitting



Temporary page!

LaTeX was unable to guess the total number of pages correctly. A
there was some unprocessed data that should have been added
the final page this extra page has been added to receive it.
If you rerun the document (without altering it) this surplus page
will go away, because LaTeX now knows how many pages to expe
for this document.