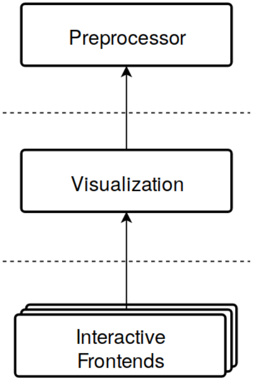
SiScLab Project 8

Katta, Partmann, Wasmer

January 24, 2019

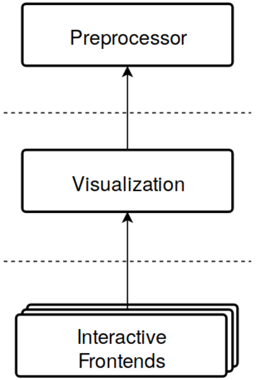
Module Design Goals



Multifunctionality:

- automated workflows like in AiiDA
- manual data analysis with Python

Module Design Goals

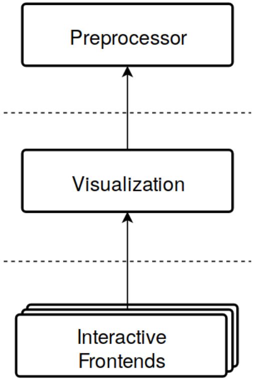


Multifunctionality:

- automated workflows like in AiiDA
- manual data analysis with Python

- no boilerplate code!

Module Design Goals



Multifunctionality:

- automated workflows like in AiiDA
- manual data analysis with Python

- no boilerplate code!

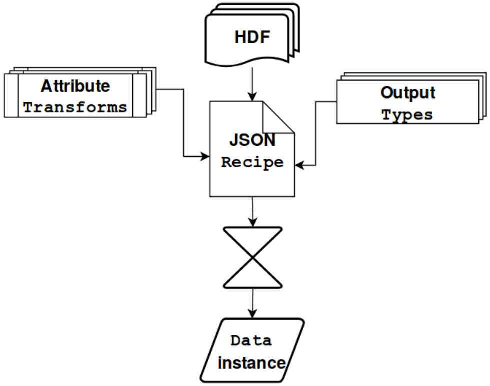
- Desktop
- Web → like in AiiDAlab

Preprocessor Module

Input: Fleur calculation results stored in Hierarchical Data Format (HDF).

- modular output types for application domain (e.g. viz)
- dependency resolution

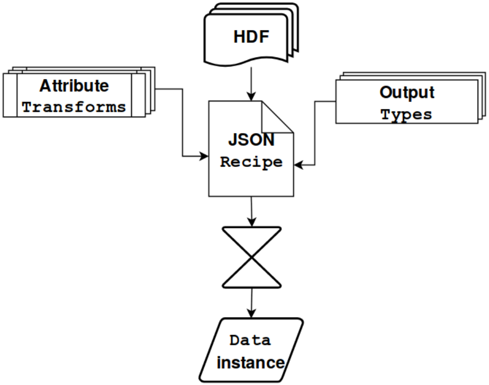
Preprocessor Module



Input: Fleur calculation results stored in Hierarchical Data Format (HDF).

- modular output types for application domain (e.g. viz)
- dependency resolution

Preprocessor Module

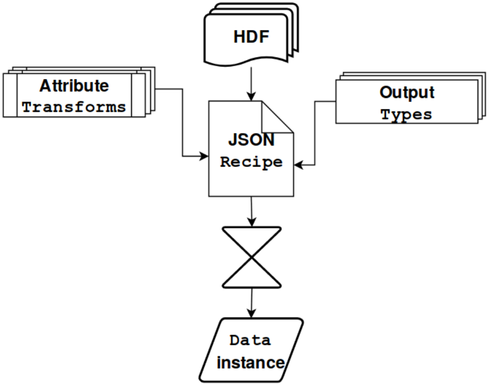


Input: Fleur calculation results stored in Hierarchical Data Format (HDF).

Module uses *type introspection* to enable features:

- modular output types for application domain (e.g. viz)
- dependency resolution

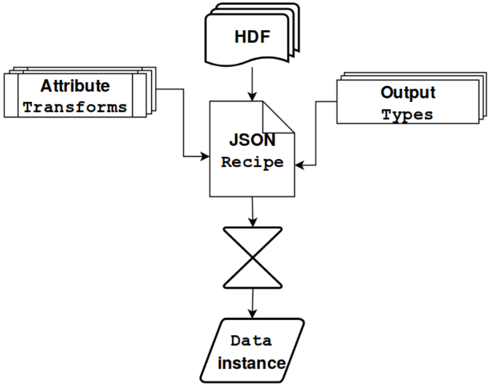
Preprocessor Module



Input: Fleur calculation results stored in Hierarchical Data Format (HDF).
Module uses *type introspection* to enable features:

- modular output types for application domain (e.g. viz)
- dependency resolution

Preprocessor Module



Input: Fleur calculation results stored in Hierarchical Data Format (HDF).
Module uses *type introspection* to enable features:

- modular output types for application domain (e.g. viz)
- dependency resolution

Data Selection for Viz

The main compute-intensive routine:

$$W^{\text{eff}}_{s,\mathbf{k},\nu} = \left(\frac{\sum\limits_{\substack{g \in \text{groups} \\ c \in \text{characters}}} n_{s,\mathbf{k},\nu,g,l} N_g}{\sum\limits_{\substack{g \in \text{all groups} \\ c \in \text{all characters}}} n_{s,\mathbf{k},\nu,g,l} N_g} \right) \left(W^{\text{unf}}_{s,\mathbf{k},\nu} \right)^{\alpha}$$

- $W^{\text{eff}}_{s,\mathbf{k},\nu}$: effective weight
- $n_{s,\mathbf{k},\nu,g,l}$: State-specific l-like charge
- N_g: no. of atoms in group
- $W^{\text{unf}}_{s,\mathbf{k},\nu}$: unfolding weight; $\alpha = 0 \implies$ no unfolding

Data Selection for Viz

The main compute-intensive routine:

$$W^{\text{eff}}_{s,\mathbf{k},\nu} = \left(\frac{\sum\limits_{\substack{g\in\text{groups}\\ c\in\text{characters}}} n_{s,\mathbf{k},\nu,g,l} N_g}{\sum\limits_{\substack{g\in\text{all groups}\\ c\in\text{all characters}}} n_{s,\mathbf{k},\nu,g,l} N_g} \right) \left(W^{\text{unf}}_{s,\mathbf{k},\nu} \right)^{\alpha}$$

- $W^{\text{eff}}_{s,\mathbf{k},\nu}$: effective weight
- $n_{s,\mathbf{k},\nu,g,l}$: State-specific l-like charge
- N_g: no. of atoms in group
- $W^{\text{unf}}_{s,\mathbf{k},\nu}$: unfolding weight; $\alpha = 0 \implies$ no unfolding

Data Selection for Viz

The main compute-intensive routine:

$$W^{\text{eff}}_{s,\mathbf{k},\nu} = \left(\frac{\sum\limits_{\substack{g \in \text{groups} \\ c \in \text{characters}}} n_{s,\mathbf{k},\nu,g,l} N_g}{\sum\limits_{\substack{g \in \text{all groups} \\ c \in \text{all characters}}} n_{s,\mathbf{k},\nu,g,l} N_g} \right) \left(W^{\text{unf}}_{s,\mathbf{k},\nu} \right)^{\alpha}$$

- $W^{\text{eff}}_{s,\mathbf{k},\nu}$: effective weight
- $n_{s,\mathbf{k},\nu,g,l}$: State-specific l-like charge
- N_g: no. of atoms in group
- $W^{\text{unf}}_{s,\mathbf{k},\nu}$: unfolding weight; $\alpha = 0 \implies$ no unfolding

Data Selection for Viz

The main compute-intensive routine:

$$W^{\text{eff}}_{s,\mathbf{k},\nu} = \left(\frac{\sum\limits_{\substack{g \in \text{groups} \\ c \in \text{characters}}} n_{s,\mathbf{k},\nu,g,l} N_g}{\sum\limits_{\substack{g \in \text{all groups} \\ c \in \text{all characters}}} n_{s,\mathbf{k},\nu,g,l} N_g} \right) \left(W^{\text{unf}}_{s,\mathbf{k},\nu} \right)^{\alpha}$$

- $W^{\text{eff}}_{s,\mathbf{k},\nu}$: effective weight
- $n_{s,\mathbf{k},\nu,g,l}$: State-specific l-like charge
- N_g: no. of atoms in group
- $W^{\text{unf}}_{s,\mathbf{k},\nu}$: unfolding weight; $\alpha = 0 \implies$ no unfolding

Data Selection for Viz

The main compute-intensive routine:

$$W^{\text{eff}}_{s,\mathbf{k},\nu} = \left(\frac{\sum\limits_{\substack{g \in \text{groups} \\ c \in \text{characters}}} n_{s,\mathbf{k},\nu,g,l} N_g}{\sum\limits_{\substack{g \in \text{all groups} \\ c \in \text{all characters}}} n_{s,\mathbf{k},\nu,g,l} N_g} \right) \left(W^{\text{unf}}_{s,\mathbf{k},\nu} \right)^{\alpha}$$

- $W^{\text{eff}}_{s,\mathbf{k},\nu}$: effective weight
- $n_{s,\mathbf{k},\nu,g,l}$: State-specific l-like charge
- N_g: no. of atoms in group
- $W^{\text{unf}}_{s,\mathbf{k},\nu}$: unfolding weight; $\alpha = 0 \implies$ no unfolding

Data Selection for Viz

Typically, $\sim 10^7$ data points are accessed.

Optimizations:

- reshaping $(\mathbf{k}, \nu) \rightarrow (\mathbf{k} \cdot \nu)$
- weight filter t: $W^{\text{eff}}_{s,\mathbf{k},\nu} > t$
- using optimized `numpy` functions for tensor product
- buffering on selection change

➔ Speedup $\sim 10^2$

Data Selection for Viz

Typically, $\sim 10^7$ data points are accessed.

Optimizations:

- reshaping $(\mathbf{k}, \nu) \to (\mathbf{k} \cdot \nu)$
- weight filter t: $W^{\text{eff}}_{s,\mathbf{k},\nu} > t$
- using optimized `numpy` functions for tensor product
- buffering on selection change

➔ Speedup $\sim 10^2$

Data Selection for Viz

Typically, $\sim 10^7$ data points are accessed.

Optimizations:

- reshaping $(\mathbf{k}, \nu) \rightarrow (\mathbf{k} \cdot \nu)$
- weight filter t: $W^{\text{eff}}_{s,\mathbf{k},\nu} > t$
- using optimized `numpy` functions for tensor product
- buffering on selection change

➔ Speedup $\sim 10^2$

Data Selection for Viz

Typically, $\sim 10^7$ data points are accessed.

Optimizations:

- reshaping $(\mathbf{k}, \nu) \rightarrow (\mathbf{k} \cdot \nu)$
- weight filter t: $W^{\text{eff}}_{s,\mathbf{k},\nu} > t$
- using optimized `numpy` functions for tensor product
- buffering on selection change

➔ Speedup $\sim 10^2$

Data Selection for Viz

Typically, $\sim 10^7$ data points are accessed.

Optimizations:

- reshaping $(\mathbf{k}, \nu) \rightarrow (\mathbf{k} \cdot \nu)$
- weight filter t: $W^{\text{eff}}_{s,\mathbf{k},\nu} > t$
- using optimized `numpy` functions for tensor product
- buffering on selection change

➔ Speedup $\sim 10^2$

Data Selection for Viz

Typically, $\sim 10^7$ data points are accessed.

Optimizations:

- reshaping $(\mathbf{k}, \nu) \rightarrow (\mathbf{k} \cdot \nu)$
- weight filter t: $W^{\text{eff}}_{s,\mathbf{k},\nu} > t$
- using optimized `numpy` functions for tensor product
- buffering on selection change

➔ Speedup $\sim 10^2$

Data Selection for Viz

Typically, $\sim 10^7$ data points are accessed.

Optimizations:

- reshaping $(\mathbf{k}, \nu) \rightarrow (\mathbf{k} \cdot \nu)$
- weight filter t: $W^{\text{eff}}_{s,\mathbf{k},\nu} > t$
- using optimized `numpy` functions for tensor product
- buffering on selection change

➔ Speedup $\sim 10^2$

Visualization Module

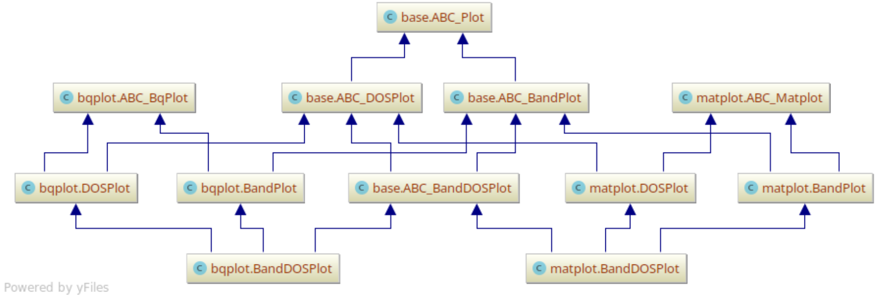
- Abstract interfaces for different viz. libs and applications
- `InteractiveControlDisplay` as frontend contracts

Visualization Module

- Abstract interfaces for different viz. libs and applications
- `InteractiveControlDisplay` as frontend contracts

Visualization Module

- Abstract interfaces for different viz. libs and applications
- `InteractiveControlDisplay` as frontend contracts



Desktop Frontend

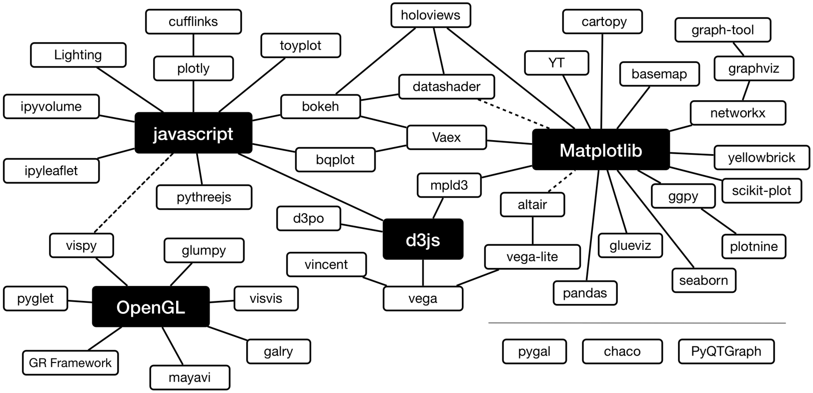
Choice of GUI Toolkit: **TKinter**, Kivy, PySide/PyQt, ...
Choice of Plotting tool: **matplotlib**

Web Frontend

The Python Visualization Landscape as of 2017...

Web Frontend

The Python Visualization Landscape as of 2017...



Python Visualization Landscape by rougier / BSD-2

Web Frontend

- Needed: an OSS **Tool Selection Process** for building a Web Dashboard using **only** .
- Decision Priority Order: *support...*
  - I. ... *interactive graphical control elements ('widgets')*
  - II. ... *easy deployment*
  - III. ... *some actual plotting libraries*

Web Frontend

- Needed: an OSS **Tool Selection Process** for building a Web Dashboard using **only** .
- Decision Priority Order: *support...*
    - I. ... *interactive graphical control elements ('widgets')*
    - II. ... *easy deployment*
    - III. ... *some actual plotting libraries*

Web Frontend

- Needed: an OSS **Tool Selection Process** for building a Web Dashboard using **only** .
- Decision Priority Order: *support...*
  - I. ... *interactive graphical control elements ('widgets')*
  - II. ... *easy deployment*
  - III. ... *some actual plotting libraries*

Web Frontend

- Needed: an OSS **Tool Selection Process** for building a Web Dashboard using **only** 🐍.
- Decision Priority Order: *support...*
  - I. ... *interactive graphical control elements ('widgets')*
  - II. ... *easy deployment*
  - III. ... *some actual plotting libraries*

Web Frontend

- Needed: an OSS **Tool Selection Process** for building a Web Dashboard using **only** .
- Decision Priority Order: *support...*
  - I. ... *interactive graphical control elements ('widgets')*
  - II. ... *easy deployment*
  - III. ... *some actual plotting libraries*

Web Frontend

I. **Widgets**	jupyter	pyviz panel	bokeh	dash
Languages	🐍	🐍	🐍 / JS	🐍 / JS

[1]Excluded: writing from scratch using Flask
[2]workaround. See also: appmode, voila, thebelab
[3]interactive only

Web Frontend

I. **Widgets**	jupyter	pyviz Panel panel	Bokeh bokeh	dash
Languages	🐍	🐍	🐍 / JS	🐍 / JS
II. **Deployment**				
- Jupyter	✔	✔	✖	✖
- Standalone[1]	(binder, 🐳)[2]	Bokeh	Bokeh	plotly plotly

[1] Excluded: writing from scratch using Flask

[2] workaround. See also: appmode, voila, thebelab

[3] interactive only

Web Frontend

I. **Widgets**	jupyter jupyter	pyviz Panel panel	Bokeh bokeh	dash
Languages	🐍	🐍	🐍 / JS	🐍 / JS
II. **Deployment**				
- Jupyter	✓	✓	✗	✗
- Standalone[1]	(binder, 🐳)[2]	Bokeh	Bokeh	plotly plotly
III. **Plots**[3]				
- 2D	mpl, bqplot, …	hvPlot hvplot, Bokeh	Bokeh	plotly
- 3D	ipyvolume	✗	Bokeh	plotly

[1] Excluded: writing from scratch using Flask

[2] workaround. See also: appmode, voila, thebelab

[3] interactive only