

Declarative Configs for Maintainable Reproducible Code

Jonathan Striebel

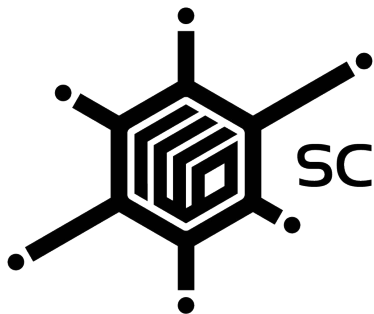


Hi, I'm **Jonathan Striebel**.

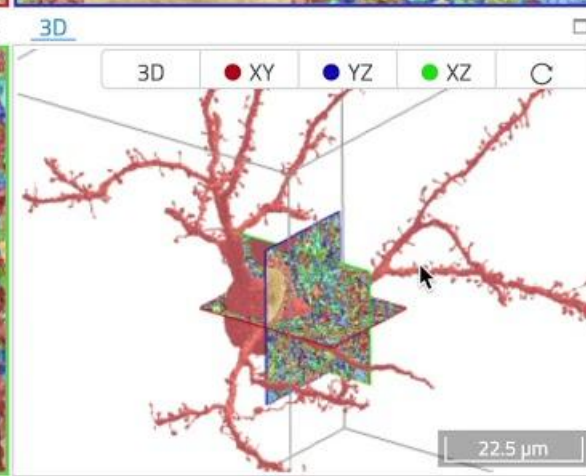
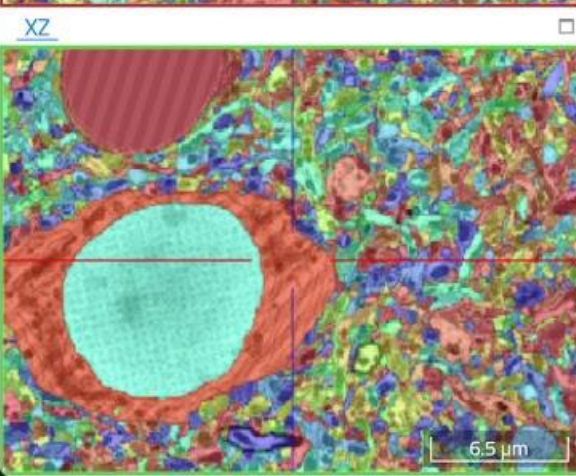
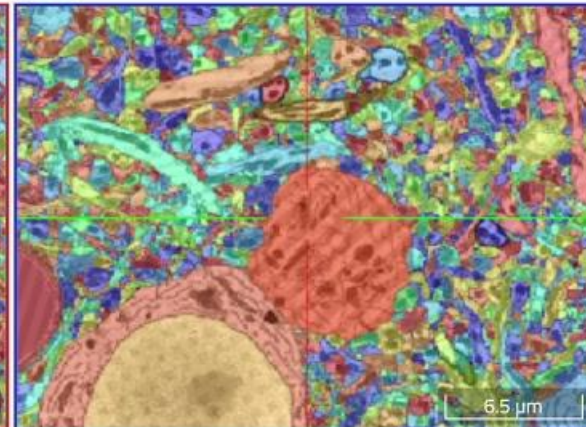
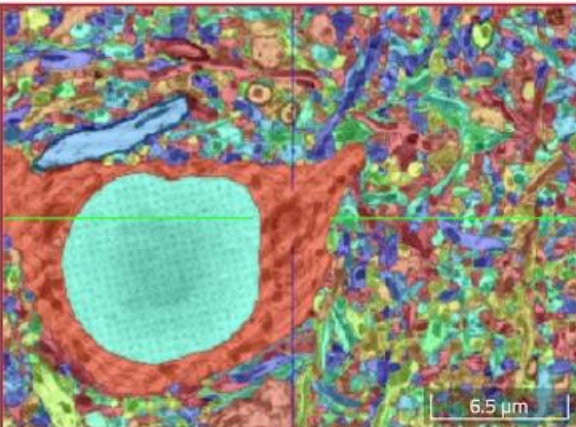


@jostriebe1

jonathan@scalableminds.com



scalableminds



L4 Mouse Cortex Dense Segmentation
Segmentation created with Raw SBEM data and segmented by the Planck Institute for Brain Research. *Dense connectomic reconstruction of the somatosensory cortex* by Boergens, B Staffler, M B Weissler, M Helmstaedter. [10.1126/science.aay3133](https://doi.org/10.1126/science.aay3133)

11.24 \times 11.24 \times 28 nm^3/voxel
5445 \times 8380 \times 3285 voxel³
61.2 μm \times 94.2 μm \times 92.0 μm
8-8-4

[I] / [O] or [ALT] + [0] Zoom in/out
[0] or [D] / [F] Move Along 3rd Axis
[Mouse] Move
[Mouse] in 3D View Rotate 3D View

[More shortcuts...](#)



<https://webknossos.org>

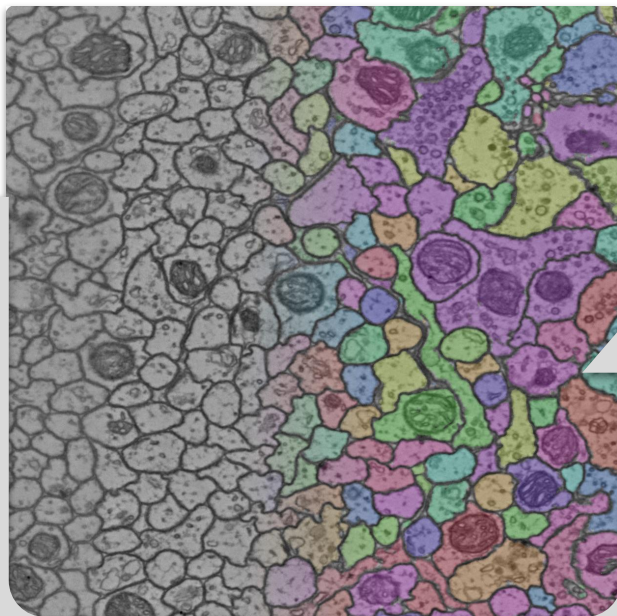
Large Scale Data-Analysis Experiments



50+ TB
image data

Machine Learning Systems

- Training Data
- Evaluation Data
- Week-long Training

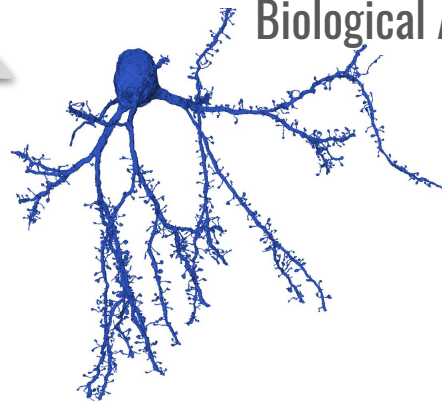


Segmentation & Agglomeration
Algorithms



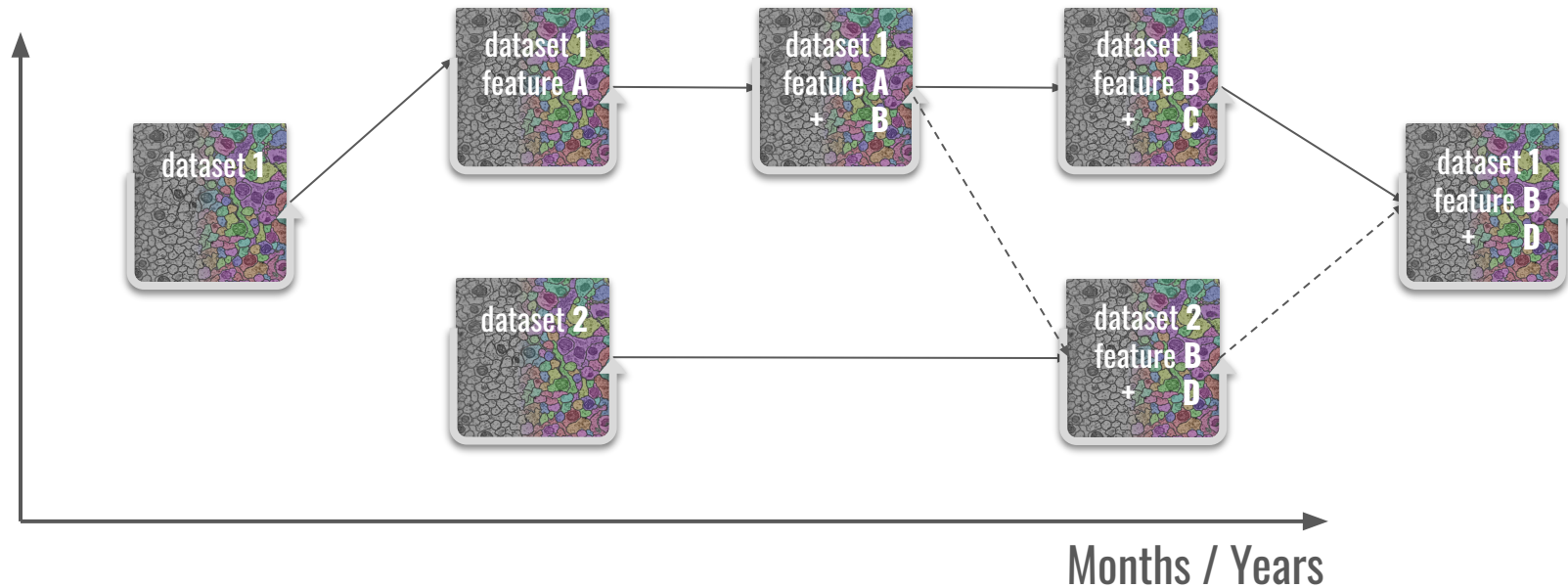
Scalability in HPC Clusters

Neuron
Reconstructions for
Biological Analysis



Experiments Hierarchy

Changing, adding & removing Features



Maintainability & Reproducibility

- Separation of Config & Code
- Config Verification
- Code Verification (Config Usage)
- Automated Migrations

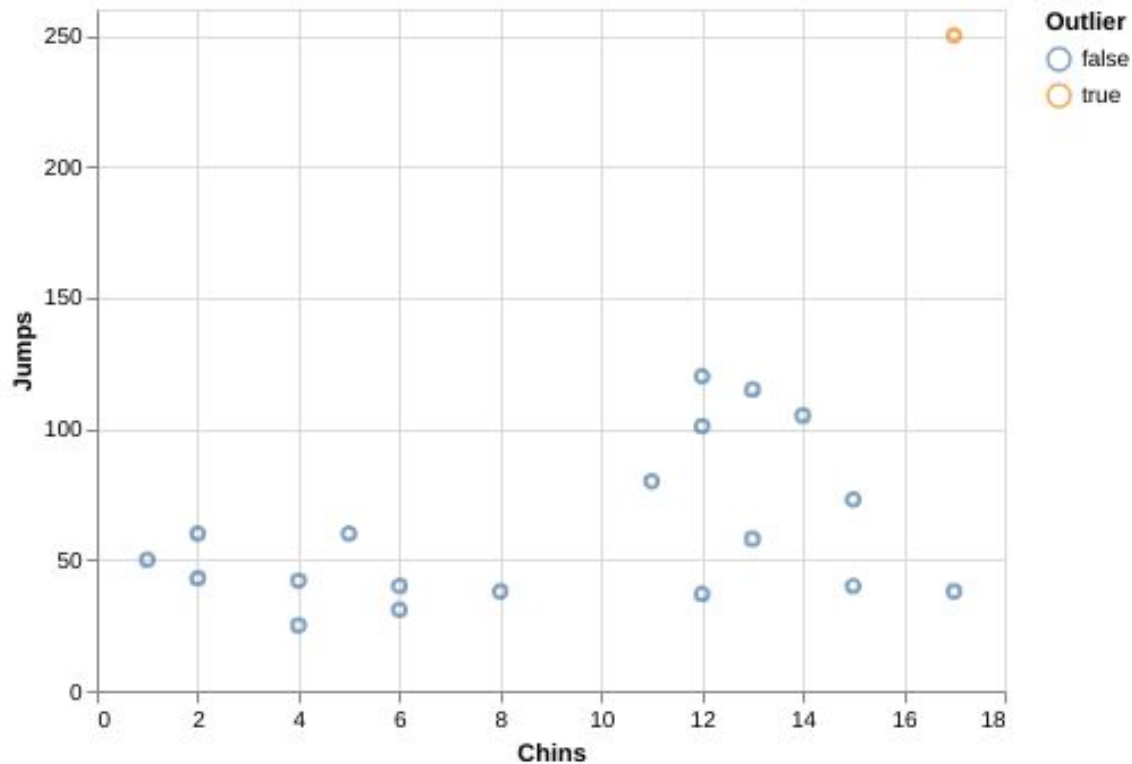


Toy Experiment

| | Chins | Situps | Jumps |
|---|-------|--------|-------|
| 0 | 5.0 | 162.0 | 60.0 |
| 1 | 2.0 | 110.0 | 60.0 |
| 2 | 12.0 | 101.0 | 101.0 |

...

Outlier Detection



config

path:

thresh:

plot_x:

plot_y:



experiment.py

data = load()

...

find_outliers(
 data,

thresh=)

...

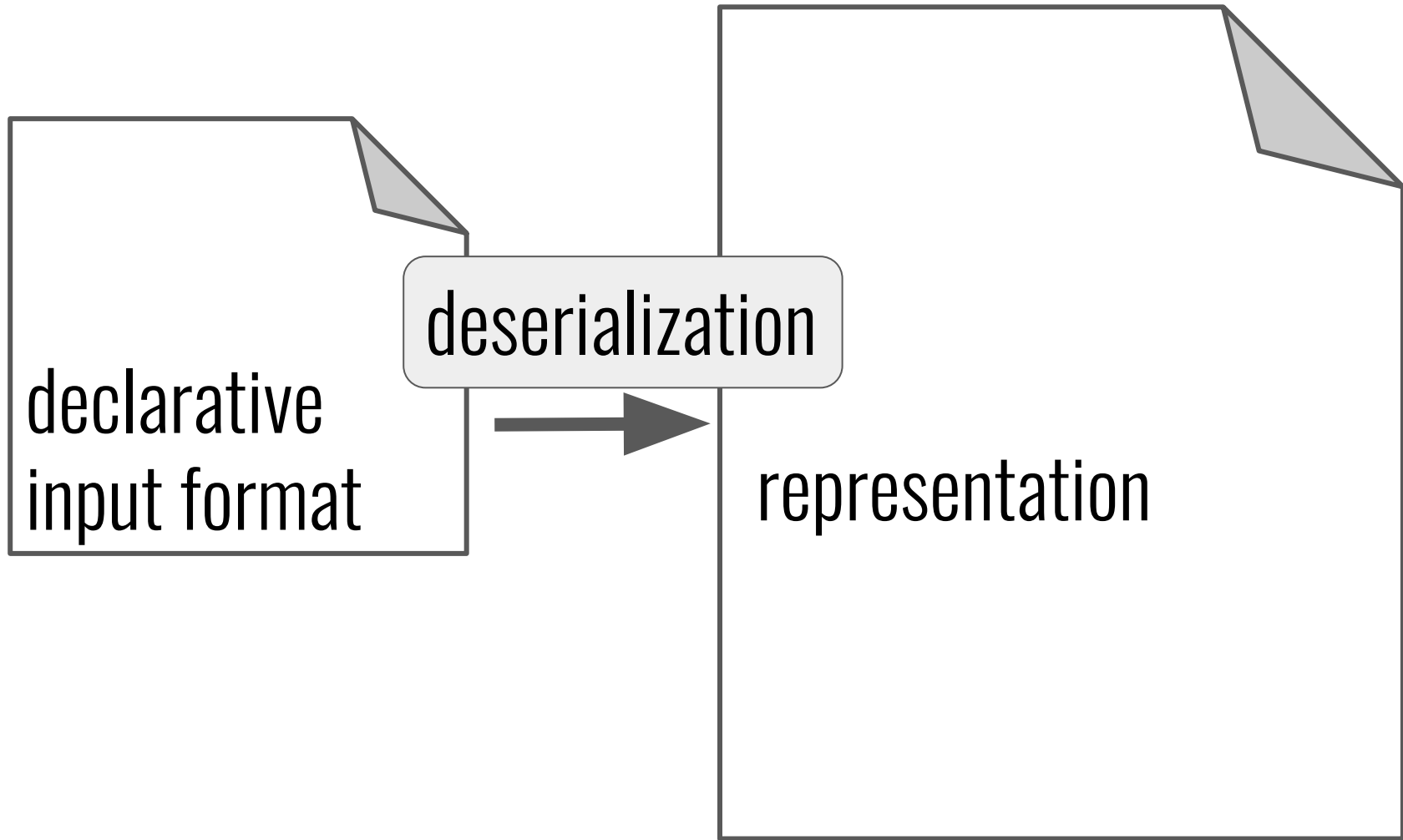
plot(
 data[,

data[

)

**Declarative
(data-only)**

Imperative



Input Formats

CLI Arguments:

```
$ python experiment.py \  
  --path somepath --thresh 10 --plot_x X ...
```

```
import typer
```

```
def main(path: str):  
    path
```

```
typer.run(main)
```

```
import argparse
```

```
parser = argparse.ArgumentParser()
```

```
parser.add_argument("path")
```

```
args = parser.parse_args()
```

```
args.path
```

```
args.wrong_key    Runtime Error
```

Input Formats

CLI Arguments:

```
$ python experiment.py \  
  --path somepath --thresh 10 --plot_x X ...
```

```
import typer
```

```
def main(path: str):  
    path
```

```
typer.run(main)
```

Environment Variables:

```
$ export PATH=somepath THRESH=10 ...  
$ python experiment.py
```

```
import os  
os.environ["path"]
```

Files:

```
$ python experiment.py config.yaml  
  # or .json, .toml, ...
```

```
import yaml  
with open(...) as f:  
    c = yaml.safe_load(f)  
    c["path"]
```

Representations

Basic Python Types:


dict, list, string, int, float, bool

```
config = {  
    "path": "somepath",  
    "thresh": 10,  
    "plot_x": "X",  
    "plot_y": "Y",  
}
```

```
config["wrong_key"]
```



Runtime Error



```
structuring: cattrs 🎉  
cattr.structure(  
    config,  
    ConfigSchema  
)
```

Objects (e.g. with attrs)

```
import attr  
@attr.s(auto_attribs=True)
```

class ConfigSchema:

```
    path: str  
    thresh: int  
    plot_x: str  
    plot_y: str
```

```
config.wrong_key
```



Static Type-Checks (e.g. with mypy)

- Input:**
- CLI-parameters
 - environment variables
 - json / **yaml** / toml / ini / ...

Representations

- TypedDict
- NamedTuple
- dataclasses
- pydantic
- **attrs**

Converters

- typedload
- dacite
- pydantic
- **cattr**

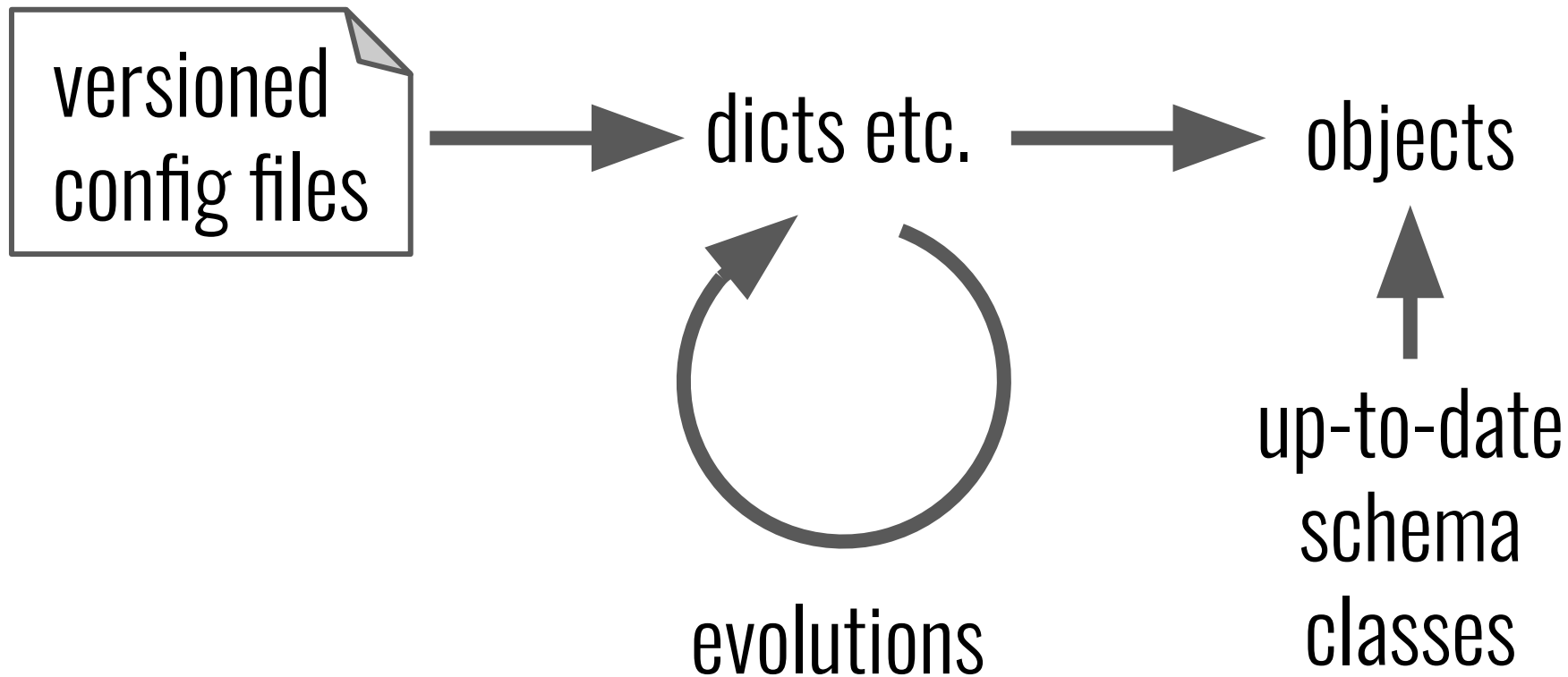
Type Checkers:

- mypy
- pytype
- ...

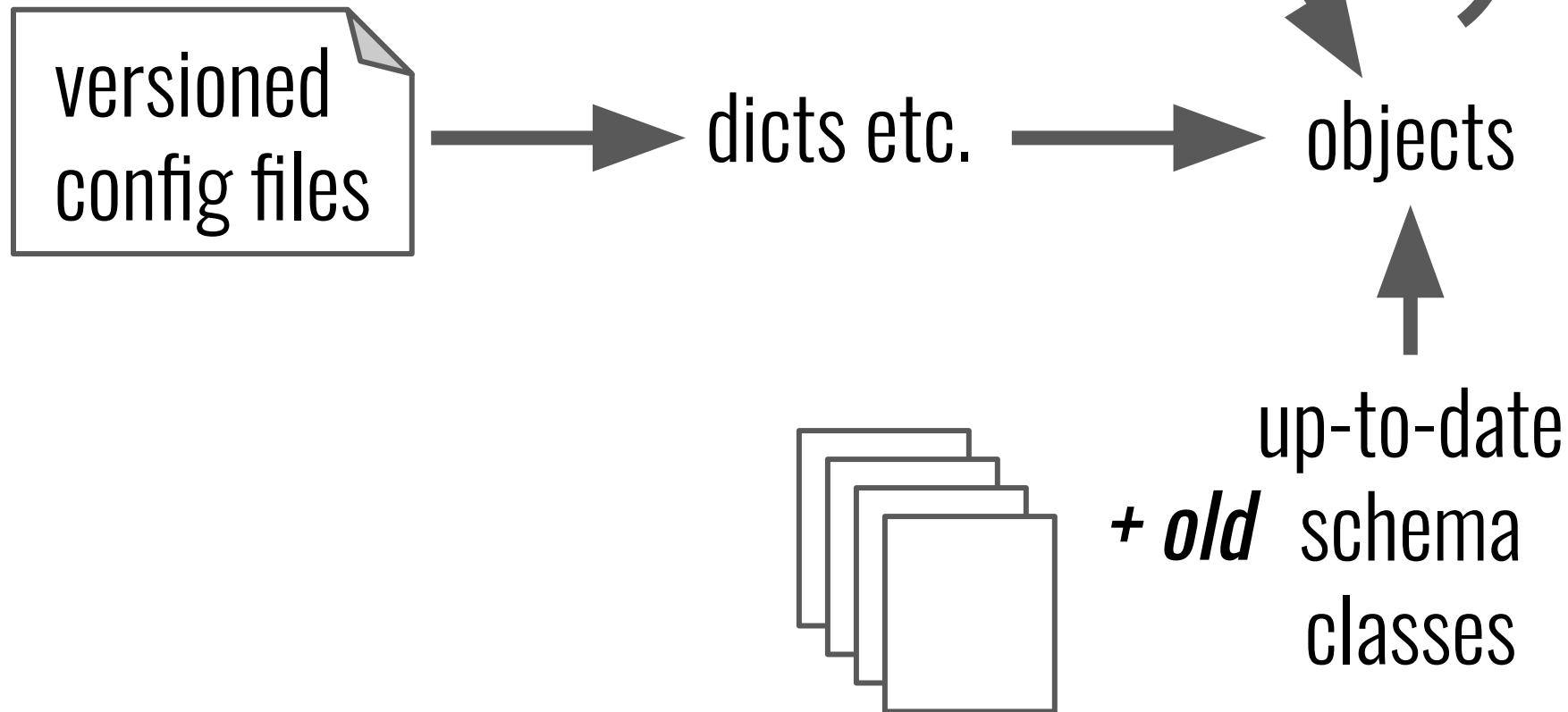
(Tracking)

sacred, MLflow, Guild

Schema Versions & Evolutions



Schema Versions & Evolutions



Maintainability & Reproducibility

- Separation of Config & Code
- Config Verification
- Code Verification (Config Usage)
- Automated Migrations

declarative
configs ✓

cattrs ✓

mypy ✓

evolutions ✓