“Programming is about trying to make the future less painful. It’s about making things easier for our teammates.”

The Pragmatic Programmer
Andy Hunt & Dave Thomas

CeMM

SCIENCE IS OUR MEDICINE

CENTER FOR MEDICAL DATA SCIENCE
MEDICAL UNIVERSITY OF VIENNA
Institute of Artificial Intelligence

BOCK LAB

MR.PARETO
More Time for Science!

Stephan Reichl
Predoctoral Fellow @ Bock Lab
September 20th, 2024

Research Center for Molecular Medicine
of the Austrian Academy of Sciences

www.cemm.at

Motivation – Three Observations at the End of 2021

Increased demand,
but limited resources.

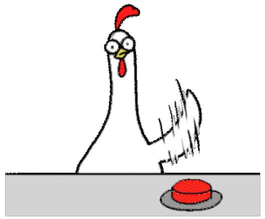
Increased fields of relevance,
but not more time.

Reproducibility crisis and
technological developments.

Over time it became clear that I created MR.PARETO
because I need it to handle my many multi-omics projects!

CeMM

Scope of MR.PARETO



What it facilitates (in scope):	What it doesn’t facilitate (out of scope):
• Accelerated arbitrarily complex end-to-end best practice analyses.	• Liberate you of thinking, the opposite is the case. It provides you with more time to think.
• Exploration of the computational option space and comparison of different approaches/hypotheses.	• Tell you how to use the supported methods e.g., parameters.
• Reproducible, transparent, documented, scalable, portable data analysis.	• Critically assess and interpret your results.

CeMM

It’s not a panacea i.e., #BigRedButton you just push.

Metaprogramming | Separation of Concerns

Explained Using The World’s Simplest Program

"Out with the details!" Get them out of the code. While we're at it, we can make our code highly configurable and "soft"—that is, easily adaptable to changes.

Not reusable,
single-purpose code.

```
print("Hello World!")
```

↓

```
Hello World!
```

Bad practice,
but most first analyses.

Reusable by copy-paste-edit,
but duplication of code base.

```
message = "Hello World!"
...
print(message)
```

↓

```
Hello World!
```

Common practice,
most analyses in papers.

Reusable.

config.yaml

```
message: “Hello Mars”
```

↓

```
message = config["message"]
...
print(message)
```

↓

```
Hello Mars!
```

Best practice,
but rare in papers.

Metaprogramming | Separation of Concerns

Advantages

Reusable.

```
message: “Hello Mars”
```

config.yaml

```
message = config["message"]
...
print(message)
```

```
Hello Mars!
```

Best practice,
but rare in papers.

Advantages

- “Do not repeat yourself” (DRY-)principle of coding
- Enables continuous improvement and compounding effects (i.e., solve it once for everyone → YAY Science).
- Reusable for different data, do not re-invent the wheel.
- Less error prone (i.e., change parameter, variable, path).
- Consistency, stability, robustness, …
- Scalable, reproducible, portable, …

Everyone

But Stephan, this looks like SO MUCH more work.

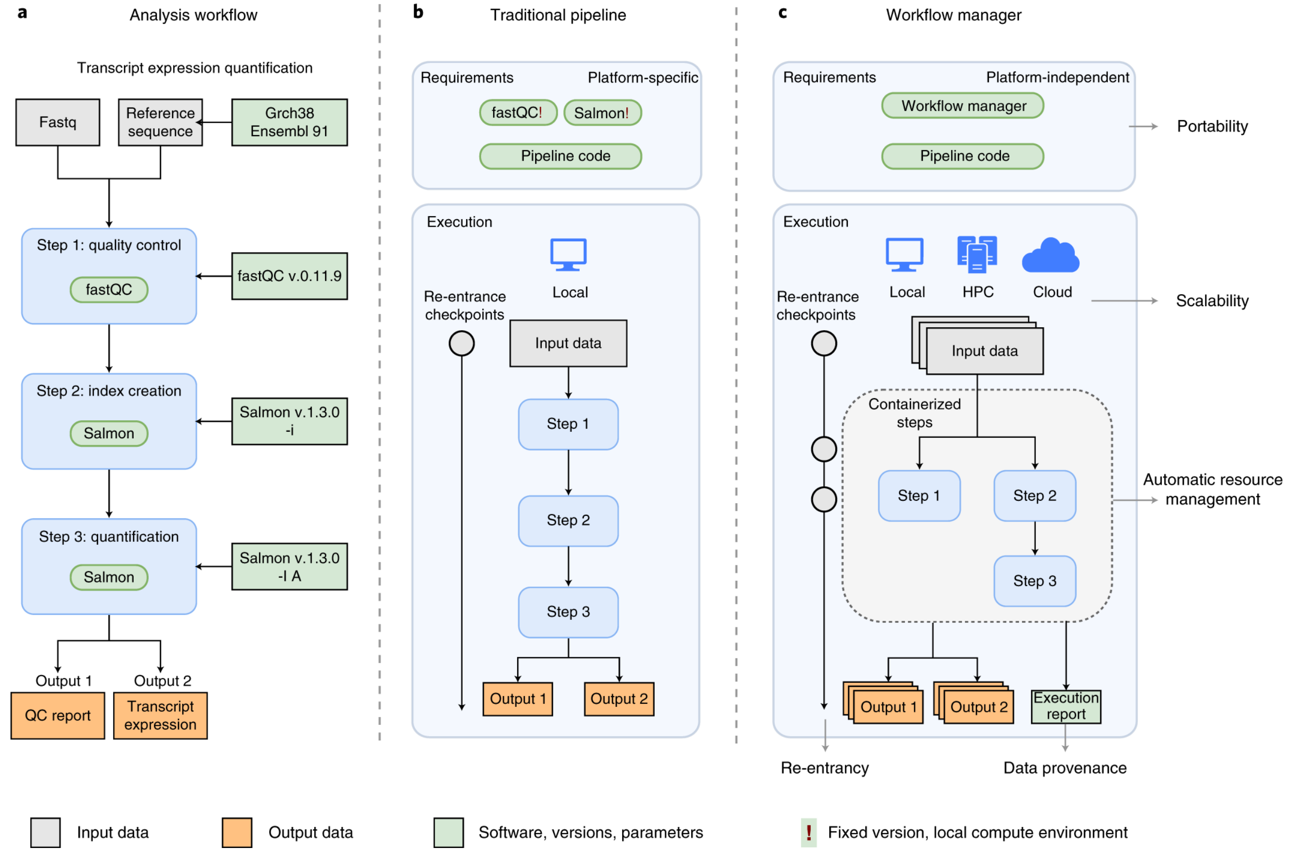
Okay, convinced, but it looks difficult…

Yes and no. Only use it for reusable code that works. Copy-paste-edit doubles code .

True, but luckily someone else already did the heavy-lifting.

CeMM

Workflow Management Systems & Snakemake



What is Snakemake?
A framework for reproducible and scalable data analysis

- Readability (python based)
- Portability (conda & container)
- Modularization (script, notebooks, wrapper),
- Transparency (reports)
- Scalability (local, cluster, cloud)

Highly popular

- >11 new citations per week
- >1,000,000 downloads
- Open source (MIT licensed)

CeMM

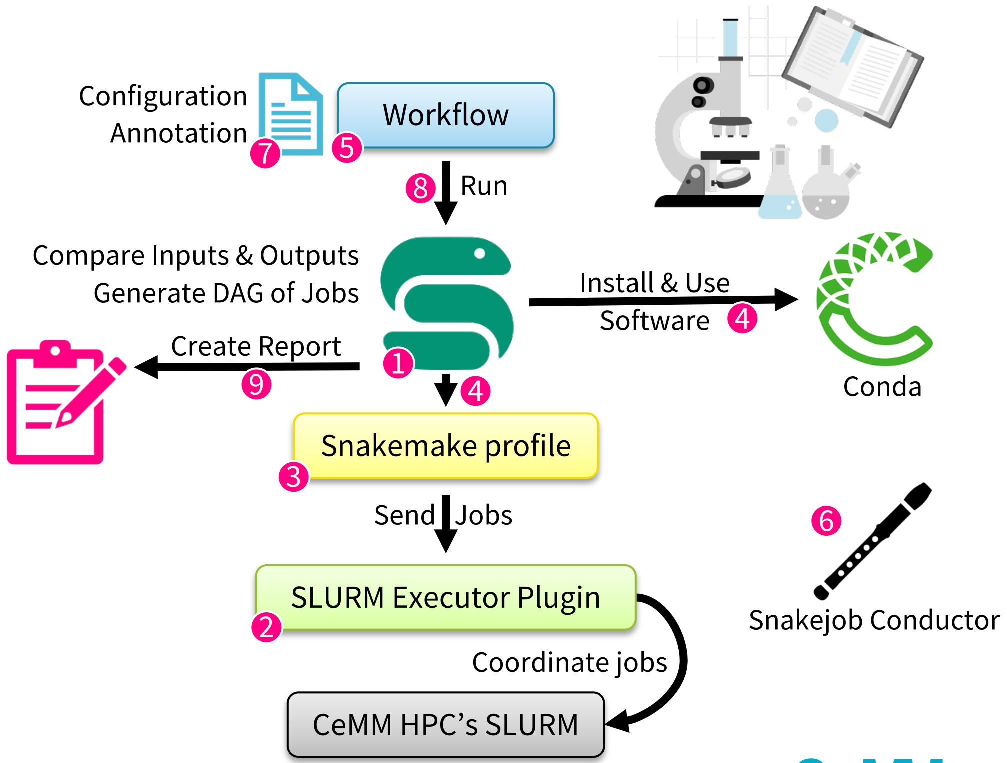
Wratten et al 2021 Nature Methods; https://snakemake.github.io/; https://snakemake.readthedocs.io/

Workflow Management by Snakemake

How to run a Snakemake Workflow (at CeMM)

1. Install Snakemake(1)*
2. Install SLURM executor plugin*.
3. Clone CeMM's global Snakemake profile(2)*
4. Set environment variables*/**
   a. Conda environments
   b. Snakemake profile
5. Clone/Deploy workflow
6. Setup Snakejob Conductor(2)**
7. Configure workflow for analysis
8. Run workflow within Snakemake
9. Generate Snakemake report

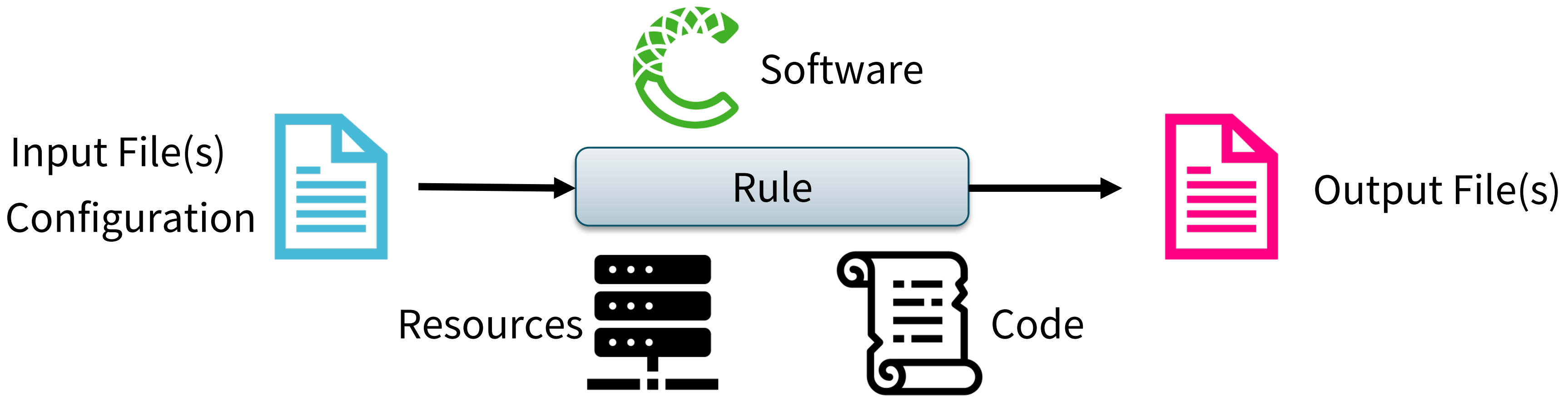
→ Do impactful science and have fun!



(1) https://snakemake.readthedocs.io/; (2) https://github.com/epigen/cemm.slurm.sm; *only once; **optional, but recommended

Rules - Definition

Rules are specific computational tasks.
They can be bash commands, scripts, notebooks, or plain (Python) code.
Input, output, software, configuration and computational resources are pre-defined.

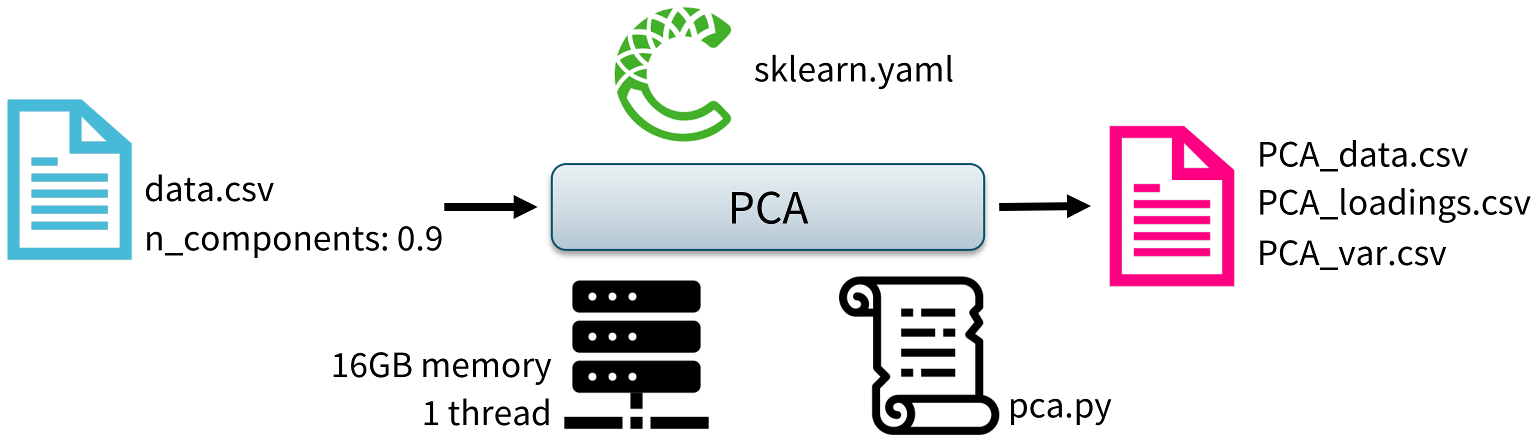


CeMM

Rules – Example: PCA

Rule

Perform a Principal Component Analysis (PCA), using software provided in `sklearn.yaml` and code in `pca.py`, on `data.csv` with configuration `n_components: 0.9`. The job will get `16GB` of memory and `1` thread.



CeMM

Rules – Example: PCA - Code

Rule

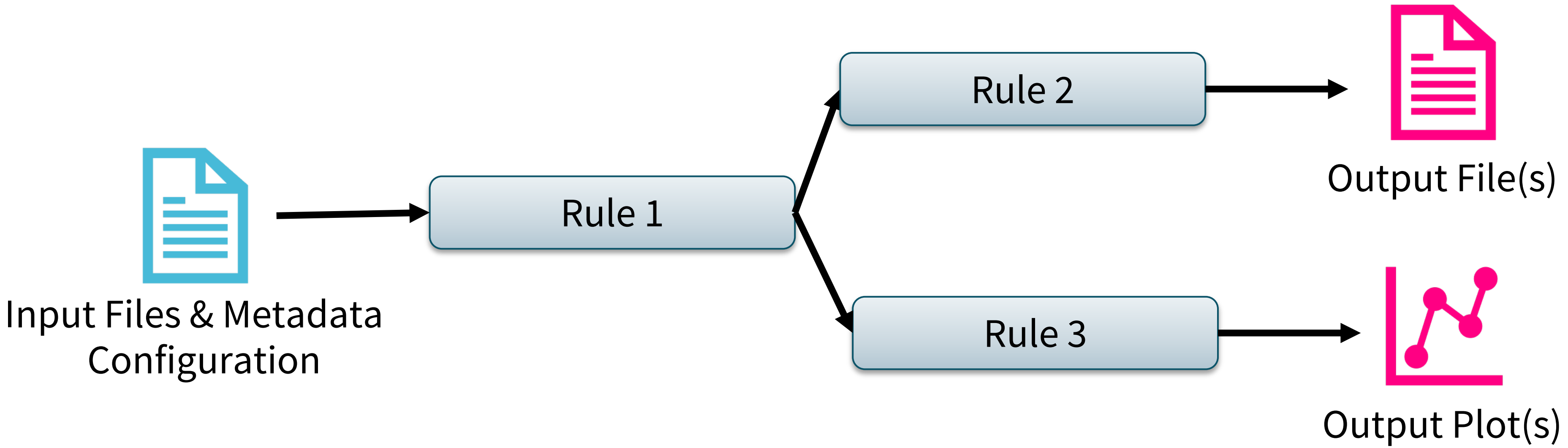
```
####### perform Principal Component Analysis (PCA) #######
rule pca:
    input:
        unpack(get_sample_paths),
    output:
        …
        result_data = os.path.join(result_path,'{sample}','PCA','PCA_{parameters}_data.csv’),
        …
    resources:
        mem_mb=config.get("mem", "16000"),
    threads: config.get("threads", 1)
    conda:
        "../envs/sklearn.yaml"
    log:
        os.path.join("logs","rules","PCA_{sample}_{parameters}.log"),
    params:
        samples_by_features = get_data_orientation,
    script:
        "../scripts/pca.py"
```

CeMM

https://github.com/epigen/unsupervised_analysis/blob/b52cb55ef2499bb8b9080e443d66f75d5a5b454c/workflow/rules/dimred.smk#L1-L23

Modules - Definition

Modules are Snakemake workflows, consisting of **Rules** for multi-step analyses.
They can be general-purpose (e.g., Unsupervised Analysis) or modality-specific (e.g., RNA-seq).



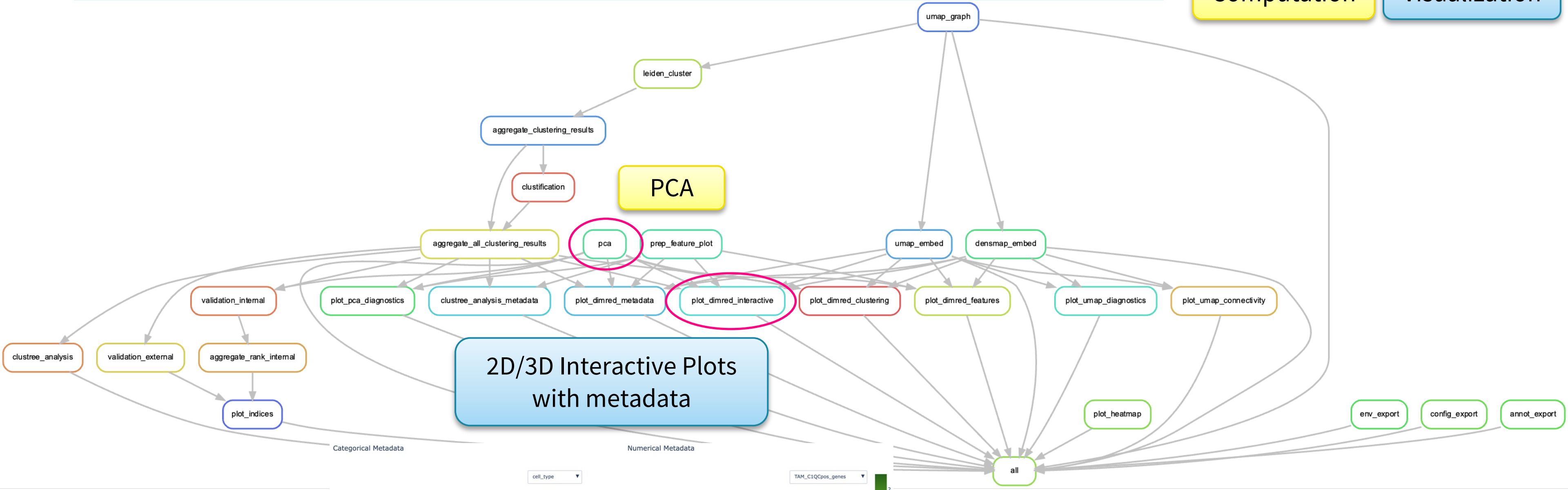
CeMM

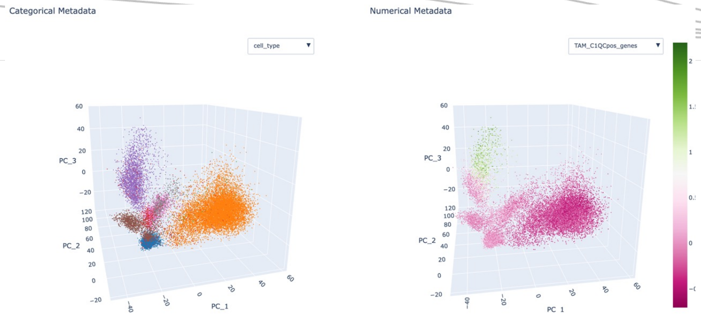
Modules – Example: Unsupervised Analysis

Rule

Computation

Visualization





CeMM

*https://github.com/epigen/unsupervised_analysis

Sustainable & Reproducible via Documentation & Reports

Rapid changes and updates to projects are great, but there's one aspect of development that has long suffered from quick iteration: **project documentation**. – Techrepublic*

GitHub Repository

GitHub Page

Authors

Software

Methods

Features

Usage

Configuration

Examples

Links

Hook for Releases →

Zenodo Repository for DOI

Automatic Curation →

Snakemake Workflow Catalog

Snakemake Reports

Self contained HTML

* https://www.techrepublic.com/article/state-documentation-why-failing-it-users/; https://github.com/epigen/mr.pareto/wiki/Sustainability-%26-Reproducibility

Projects using (multiple) Modules

You can (re-)use and combine pre-existing workflows within your projects by loading them as **Modules**. i.e., Workflows-of-Workflows.

```
# load local clones of a module
module MyData_other_workflow:
    snakefile: "path/to/other_workflow/Snakefile"
    config: config["MyData_other_workflow"]

use rule * from MyData_other_workflow as MyData_other_workflow_*

# load modules directly from GitHub
module MyData_other_workflow:
    snakefile: github(”epigen/unsupervised_analysis”, path="workflow/Snakefile", tag="v2.0.0")
    config: config["MyData_other_workflow"]

use rule * from MyData_other_workflow as MyData_other_workflow_*
```

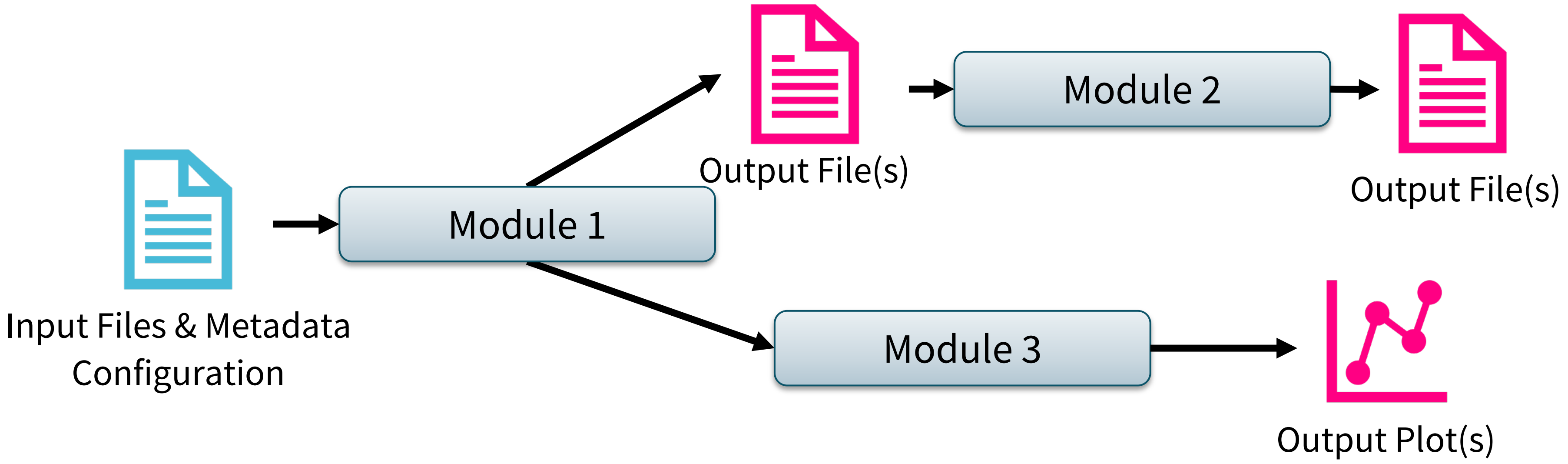
The combination of multiple modules into projects that analyze multiple datasets represents the overarching vision and superpower of MR.PARETO.

https://snakemake.readthedocs.io/en/stable/snakefiles/modularization.html#modules; https://github.com/epigen/mr.pareto/wiki/Module-Usage-in-Projects

Recipes - Definition

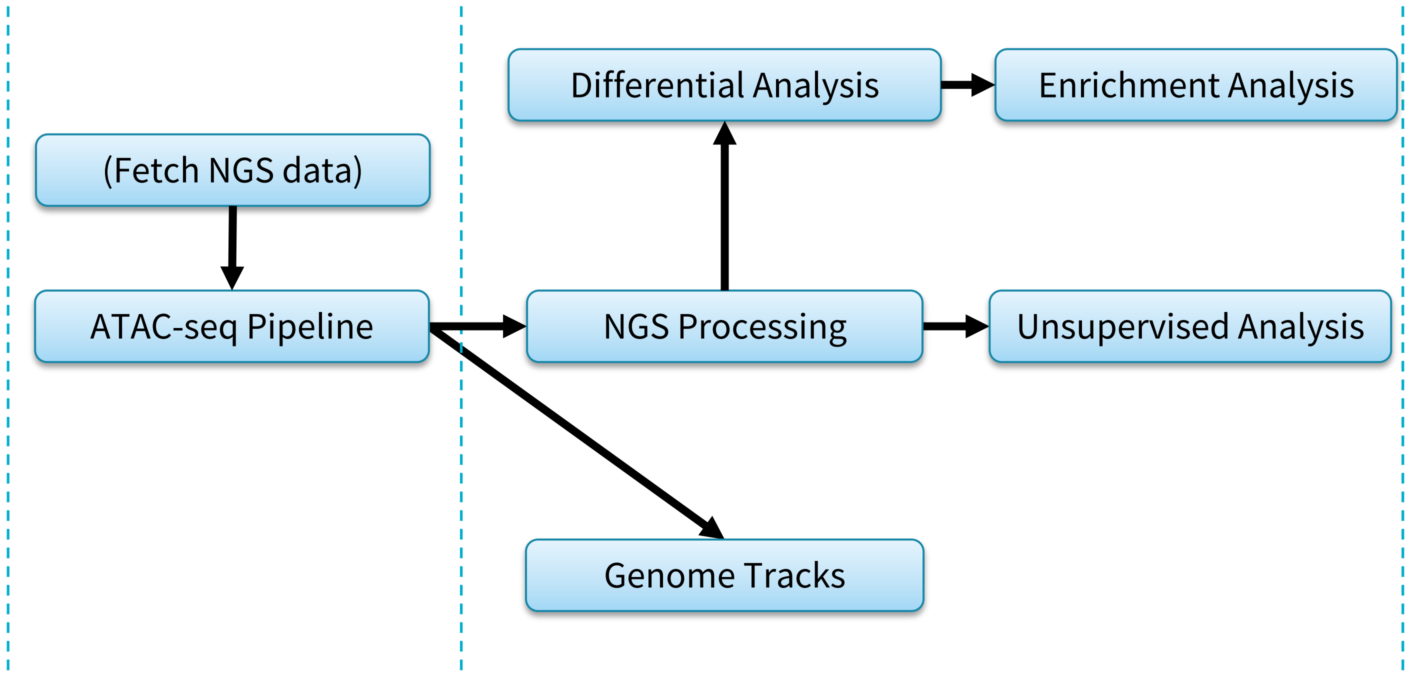
Computational Module

Recipes are combinations of existing modules into end-to-end best practice analyses.



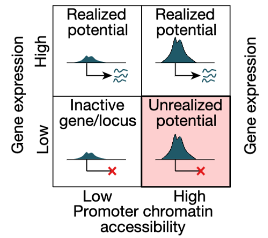
CeMM

Recipe for ATAC-seq Analysis

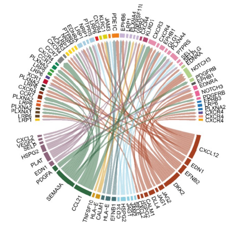


More time for downstream analyses!

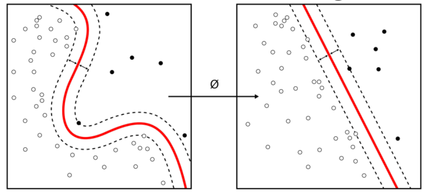
Epigenetic potential



Cell-cell communication



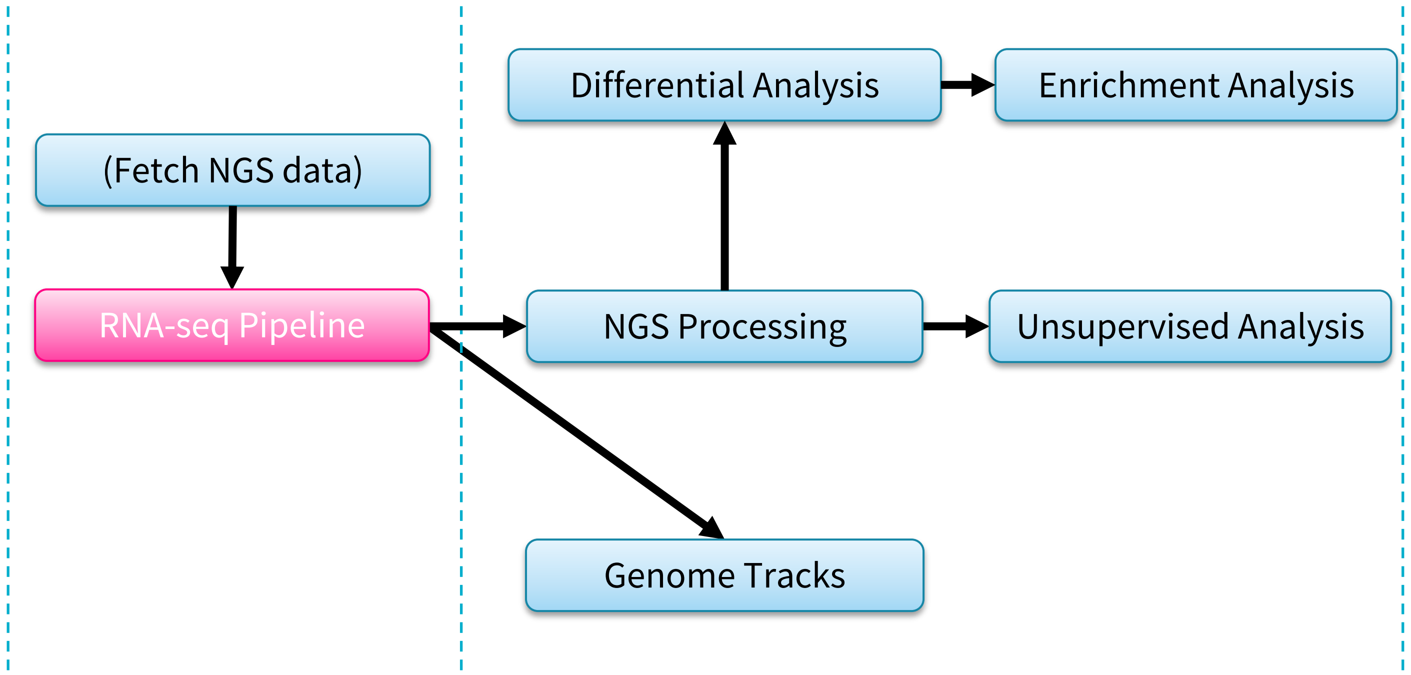
Machine learning



Data → Information → Knowledge

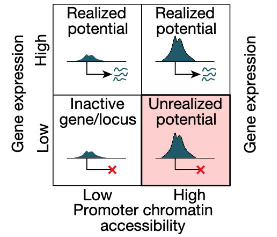
https://github.com/epigen/mr.pareto/wiki/ATACseq-Analysis-Recipe

Recipe for RNA-seq Analysis

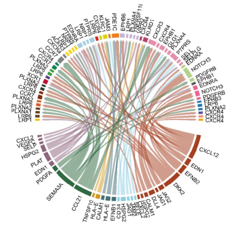


More time for downstream analyses!

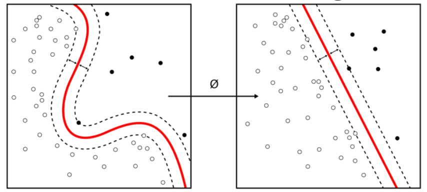
Epigenetic potential



Cell-cell communication



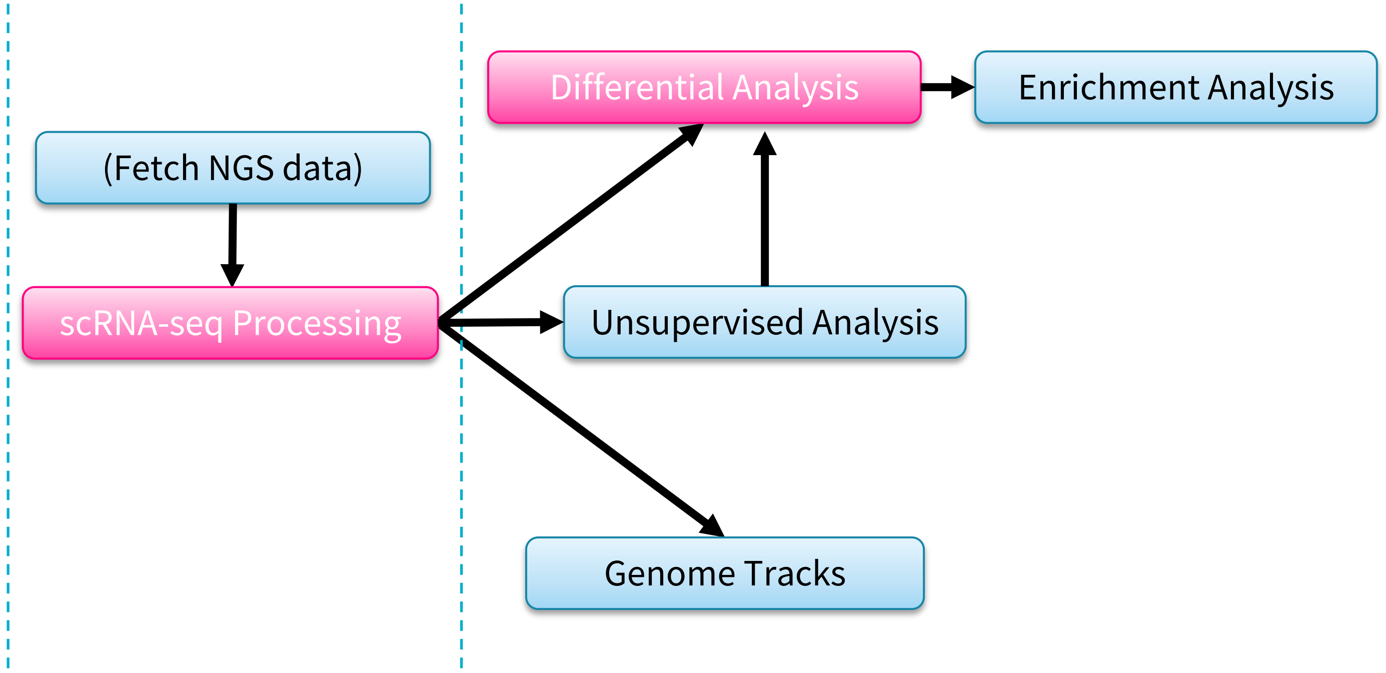
Machine learning



Data → Information → Knowledge

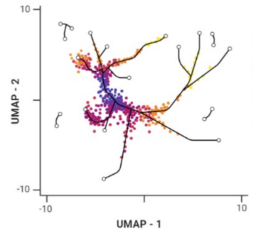
https://github.com/epigen/mr.pareto/wiki/RNAseq-Analysis-Recipe

Recipe for scRNA-seq Analysis

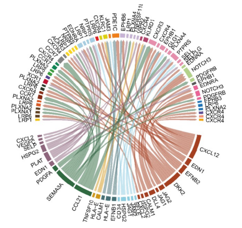


More time for downstream analyses!

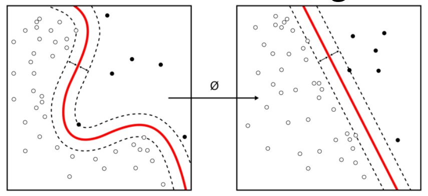
Trajectory analysis



Cell-cell communication



Machine learning

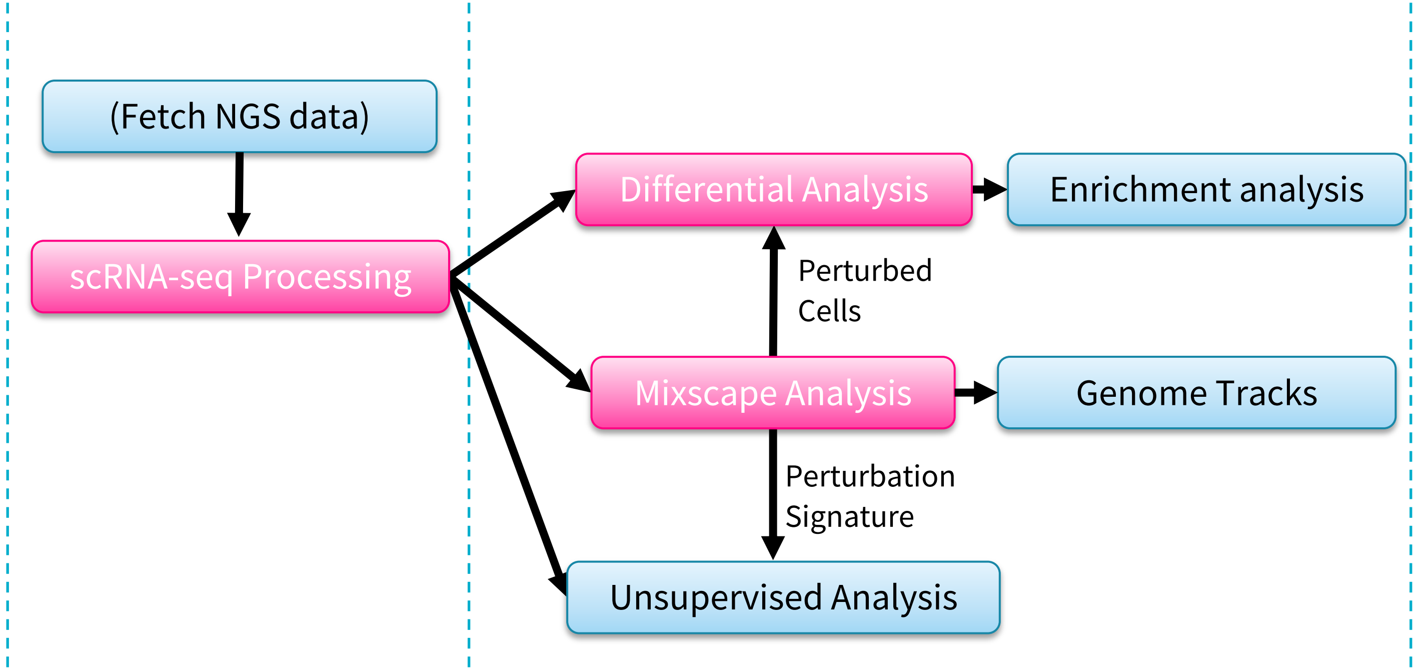


Data → Information → Knowledge

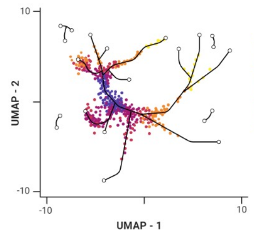
https://github.com/epigen/mr.pareto/wiki/scRNAseq-Analysis-Recipe

Recipe for scCRISPR-seq Analysis

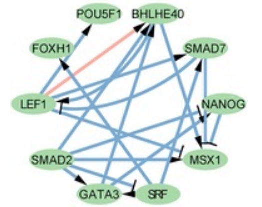
More time for downstream analyses!



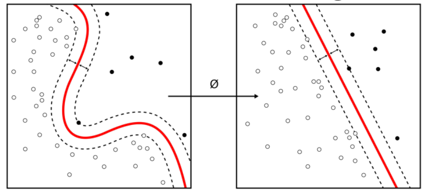
Trajectory analysis



Gene regulatory networks



Machine learning



Data → Information → Knowledge

MR. PARETO – More Time for Science!

Modules & **R**ecipes for **P**ragmatic **A**ugmentation of **R**esearch **E**fficiency **T**owards **O**ptimum

Achieve 80% of standard biomedical data science analyses semi-automatically with 20% effort by leveraging Snakemake's module functionality to use and combine pre-existing workflows into arbitrarily complex analyses.

ATAC-seq Analysis

RNA-seq Analysis

RNA/ATAC-seq Integrative Analysis

scRNA-seq Analysis

RNA-seq Processing

NGS Data Fetch

Genome Tracks

scCRISPR-seq Analysis

Split, Filter, Normalize & Integrate NGS data

scRNA-seq Processing using Seurat

Differential Analysis using Seurat

Enrichment Analysis (regions & genes)

ATAC-seq Processing

Perturbation Analysis using Seurat

Differential Analysis using limma

Unsupervised Analysis

Public | In Development | Recipes* | 👤 Contributors | ⭐ >150 GitHub stars across all 9 modules

Project repository: https://github.com/epigen/mr.pareto; GitHub list: https://github.com/stars/sreichl/lists/mr-pareto; * Recipes (WIP): https://github.com/epigen/mr.pareto/wiki

Resources

MR.PARETO

- List of current modules https://github.com/stars/sreichl/lists/mr-pareto
- Project repository: https://github.com/epigen/mr.pareto

External Snakemake Workflows:

- snakePipes https://snakepipes.readthedocs.io/en/latest/
- seq2science https://vanheeringen-lab.github.io/seq2science/index.html
- Snakemake Workflow catalog https://snakemake.github.io/snakemake-workflow-catalog/

Software:

- Snakemake Documentation https://snakemake.readthedocs.io/en/stable/
- Conda https://docs.conda.io/en/latest/

CeMM