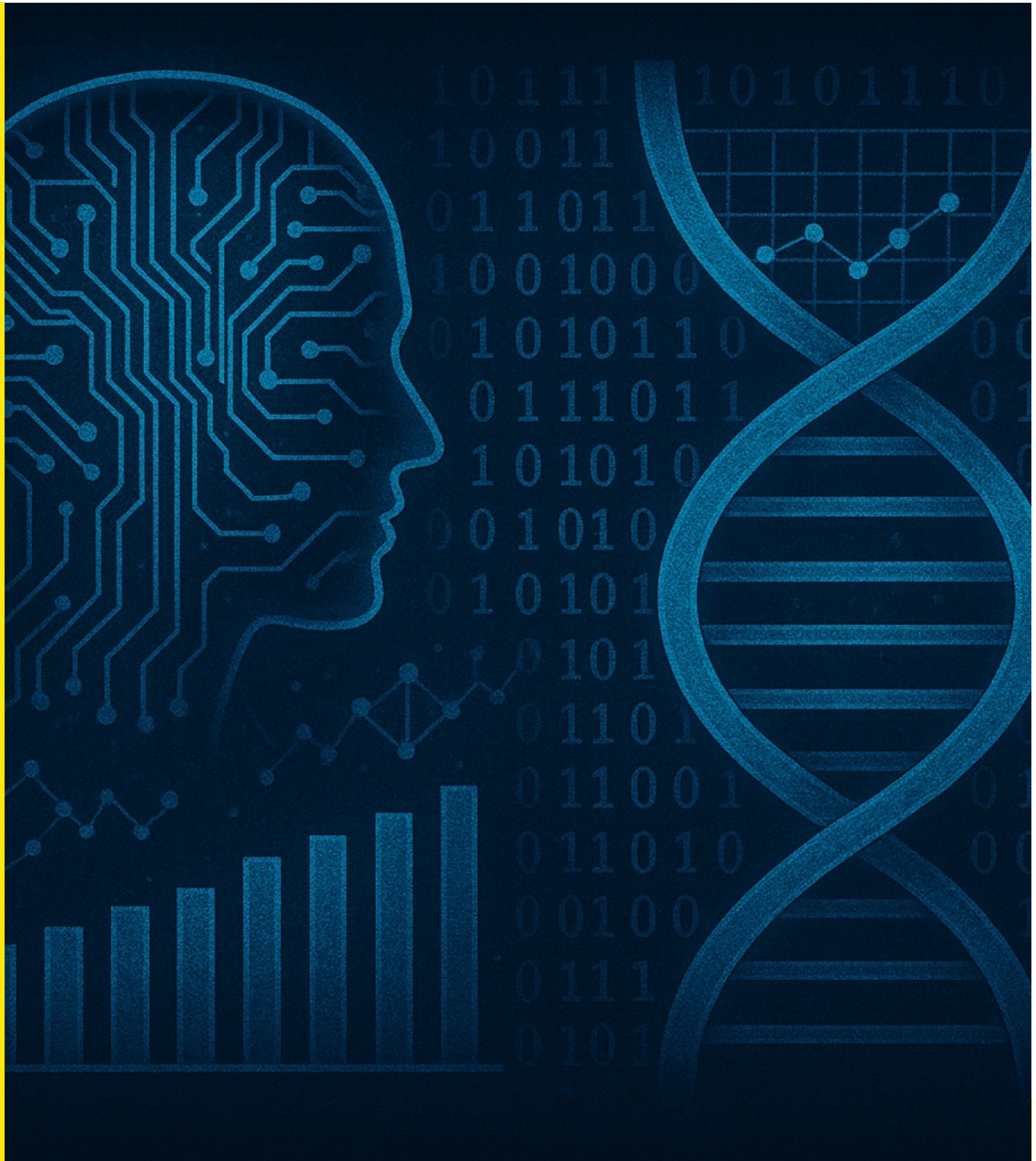
scRNA-seq analysis pipeline using Snakemake

MSc Data Science, FHNW

Crispin M. Lang

24 November 2025



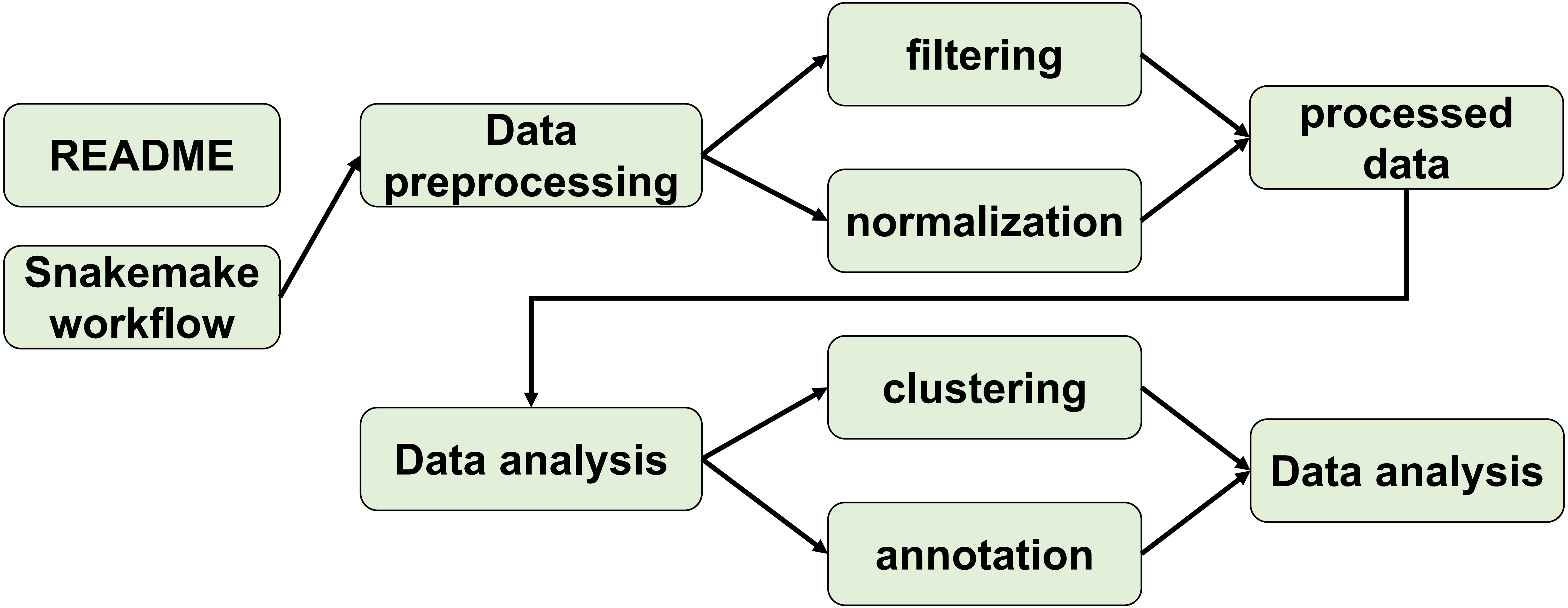
Content

- Project goals
- Introduction
- Dataset
- Analysis Pipeline Overview
- Results
- Version control

Project goals

- Snakemake workflow
  - Use wildcards
  - Add a configuration file
  - Use a mix of shell scripts and Python
  - Create a conda environment
  - Graphical output:
    - UMAP plot
    - DAG
- Version control with GitHub
  - Minimum of 2 commits

Project overview



README

Snakemake workflow

Data preprocessing

filtering

normalization

processed data

Data analysis

clustering

annotation

Data analysis

Introduction

scRNA sequencing:

- Transcriptomic technologies
- Exploration of cellular heterogenity
- high-dimensional dataset.

Significance:

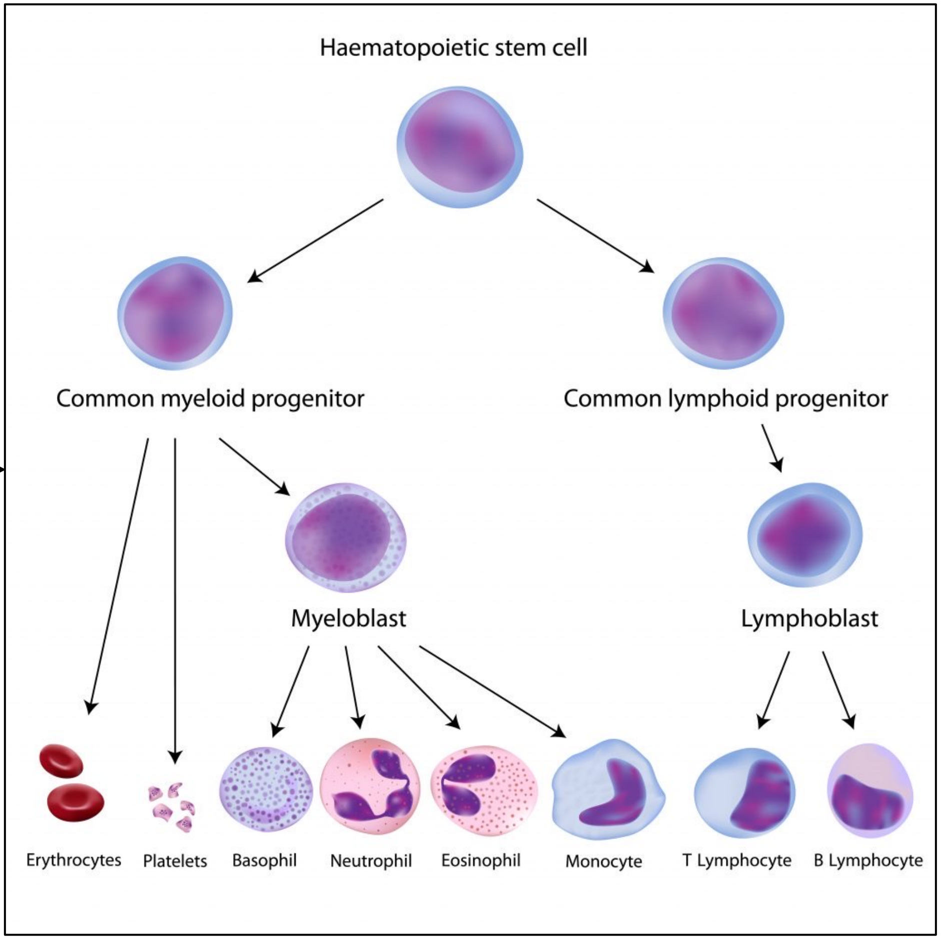
- uncover diversity in heterogeneous cell populations
- grouping cells with similar expression profiles.
- which genes are expressed and in what quantities

Source: https://www.genewiz.com, https://www.scdiscoveries.com

Dataset

Data:

- pbmc3k (peripheral blood mononuclear cells)
- ~2,700 cells, ~33,000 genes
- different cell types →



Preprocessing:

- min. 200 genes/cell
- min. 3 genes/cell
- 2000 final cells selected
- Final output: AnnData object containing cells (observations) × genes (variables).

Source: https://www.10xgenomics.com/, https://cgt.global/

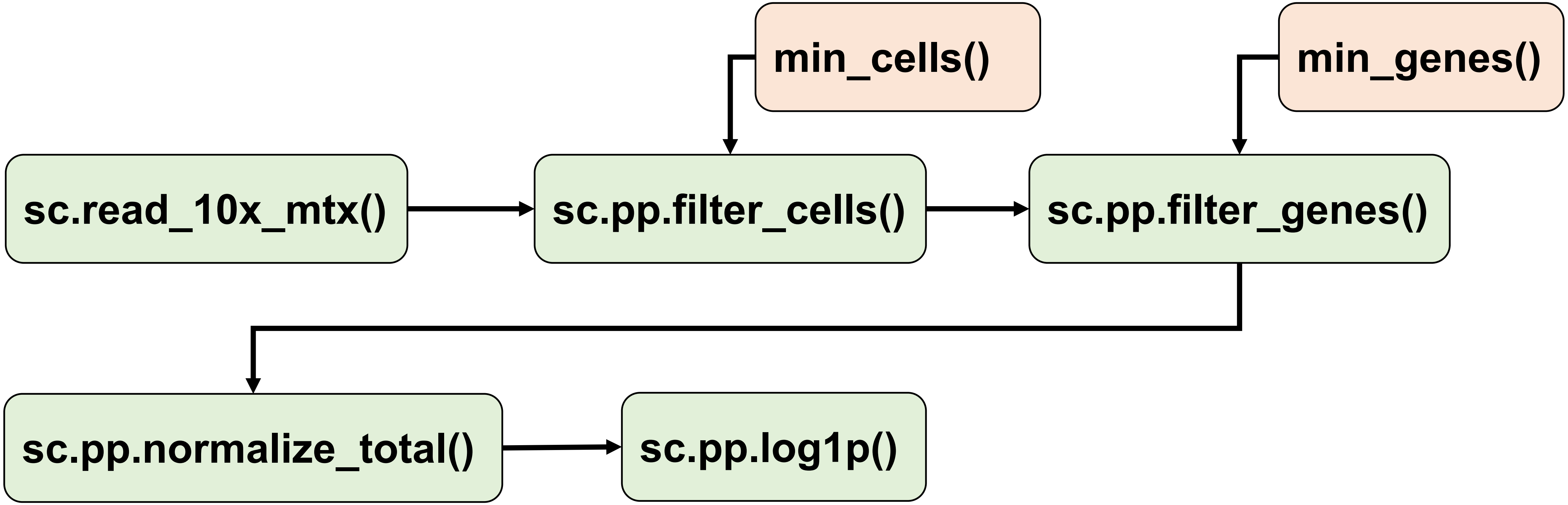
Analysis Pipeline Overview

```
snakemake_project/
├──── snakefile
├──── config.yaml
├──── envs/
│  └────── scanpy.yaml
├──── scripts/
│  ├────── preprocess.py
│  └────── analysis.py
├──── data
├──── results/
└──── README.md
```

Scanpy:

- streamlines the workflow
- ensures compatibility between steps (same framework)
- AnnData structure keeps data and metadata together
- reduces the complexity (no data moving needed)

Scanpy - preprocessing



sc.read_10x_mtx()

sc.pp.filter_cells()

min_cells()

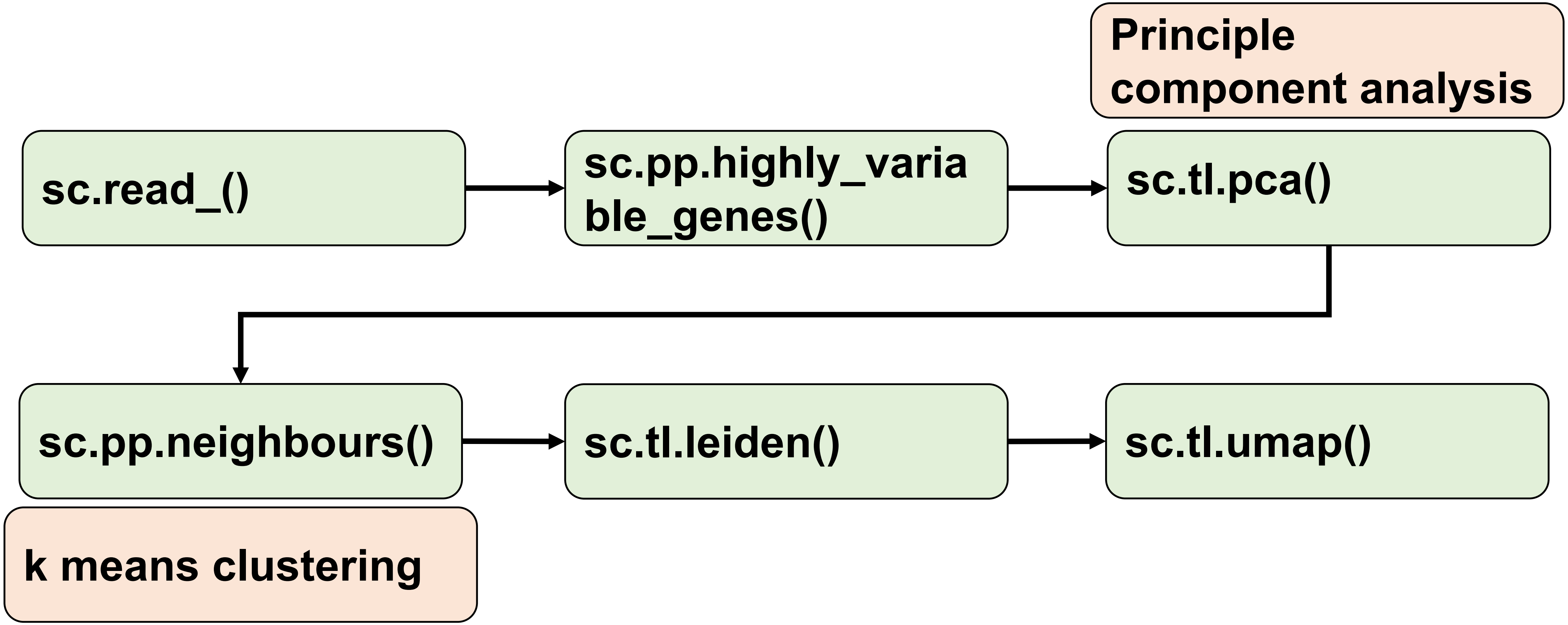
sc.pp.filter_genes()

min_genes()

sc.pp.normalize_total()

sc.pp.log1p()

Scanpy - analysis



Principle component analysis

sc.read_()

sc.pp.highly_variable_genes()

sc.tl.pca()

sc.pp.neighbours()

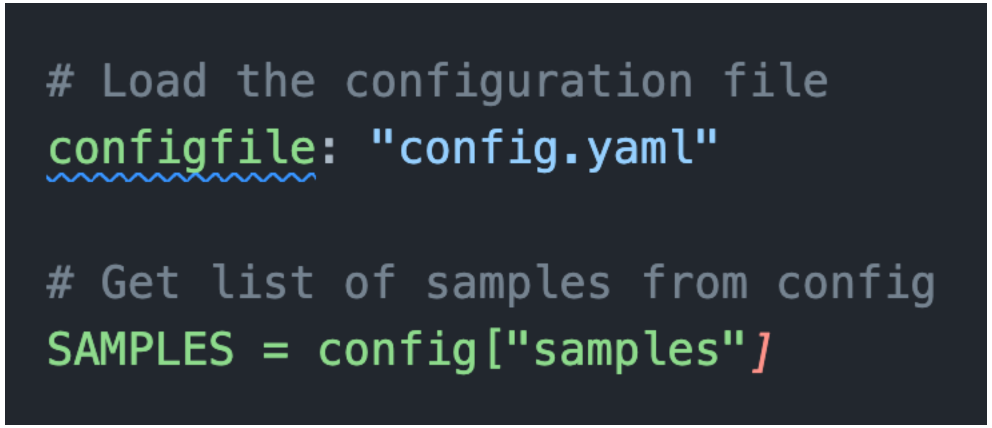
sc.tl.leiden()

sc.tl.umap()

k means clustering

Snakefile:

```
snakemake_project/
├──── snakefile
├──── config.yaml
├──── envs/
│  └───── scanpy.yaml
├──── scripts/
│  ├───── preprocess.py
│  └───── analysis.py
├──── data
├──── results/
└──── README.md
```

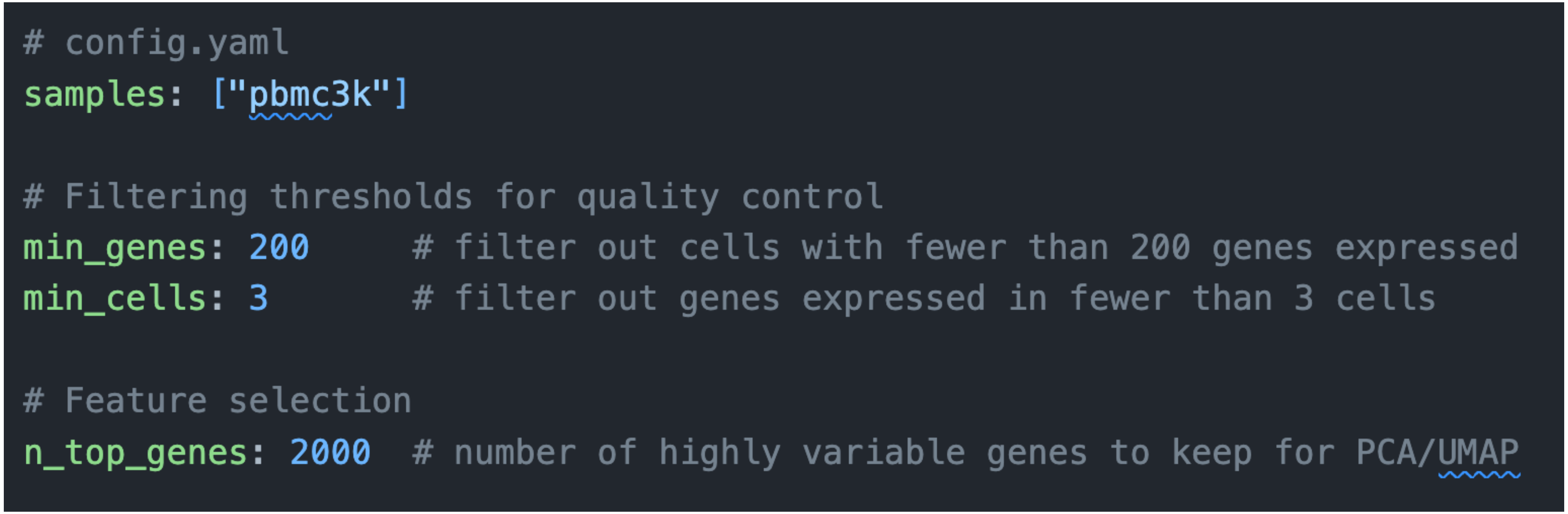


```
# Load the configuration file
configfile: "config.yaml"

# Get list of samples from config
SAMPLES = config["samples"]
```

```
snakemake_project/
├───── snakefile
├───── config.yaml
├───── envs/
│   └───── scanpy.yaml
├───── scripts/
│   ├───── preprocess.py
│   └───── analysis.py
├───── data
├───── results/
└───── README.md
```

Snakefile:



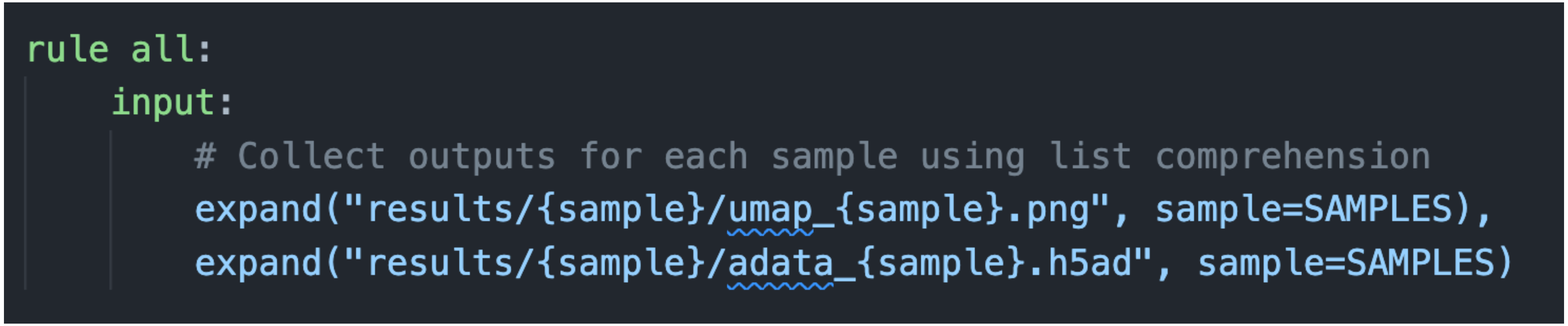
```yaml
# config.yaml
samples: ["pbmc3k"]

# Filtering thresholds for quality control
min_genes: 200     # filter out cells with fewer than 200 genes expressed
min_cells: 3       # filter out genes expressed in fewer than 3 cells

# Feature selection
n_top_genes: 2000  # number of highly variable genes to keep for PCA/UMAP
```

Snakefile:

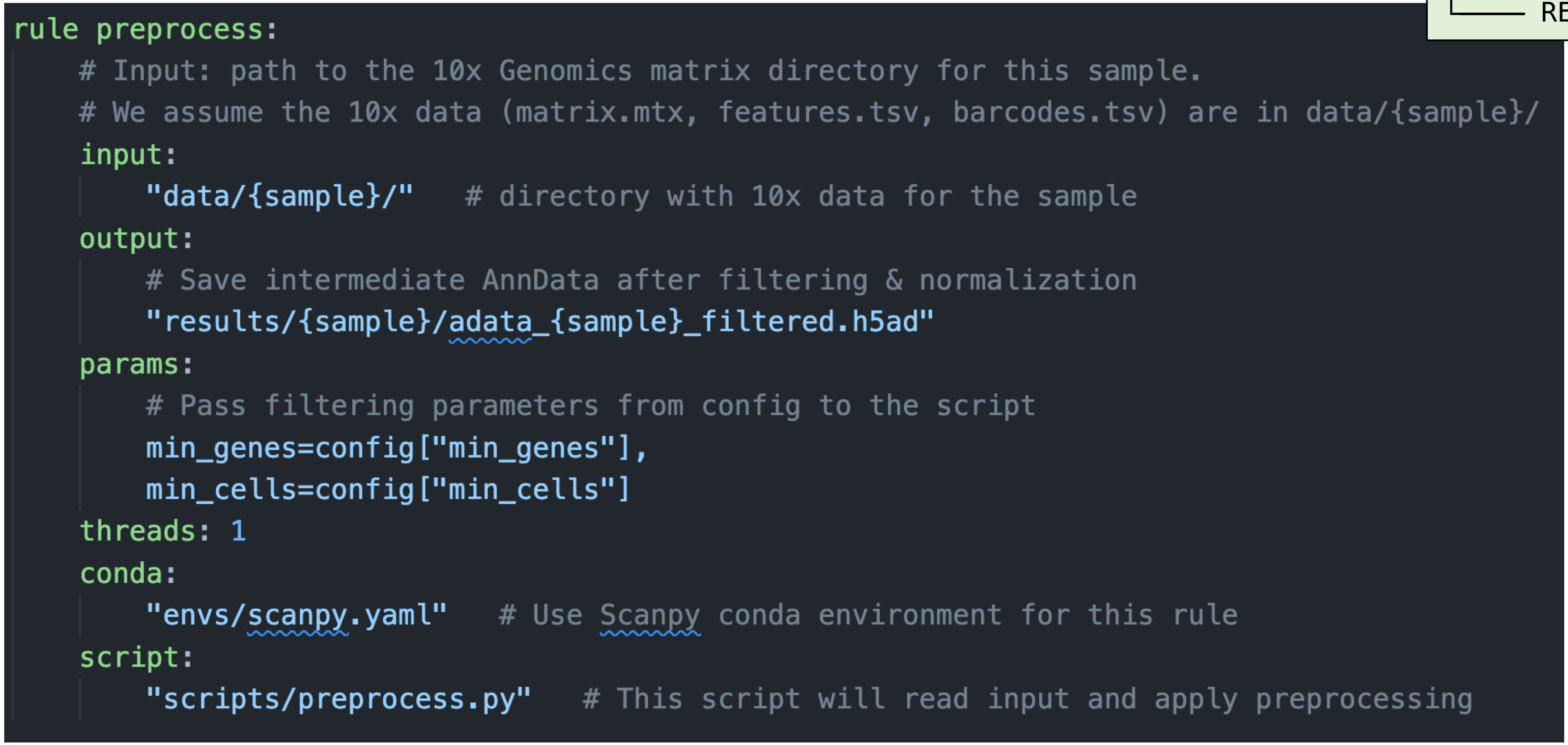
```
snakemake_project/
├──── snakefile
├──── config.yaml
├──── envs/
│  └──── scanpy.yaml
├──── scripts/
│  ├──── preprocess.py
│  └──── analysis.py
├──── data
├──── results/
└──── README.md
```



```
rule all:
    input:
        # Collect outputs for each sample using list comprehension
        expand("results/{sample}/umap_{sample}.png", sample=SAMPLES),
        expand("results/{sample}/adata_{sample}.h5ad", sample=SAMPLES)
```

```
snakemake_project/
├──── snakefile
├──── config.yaml
├──── envs/
│  └───── scanpy.yaml
├──── scripts/
│  ├───── preprocess.py
│  └───── analysis.py
├──── data
├──── results/
└──── README.md
```

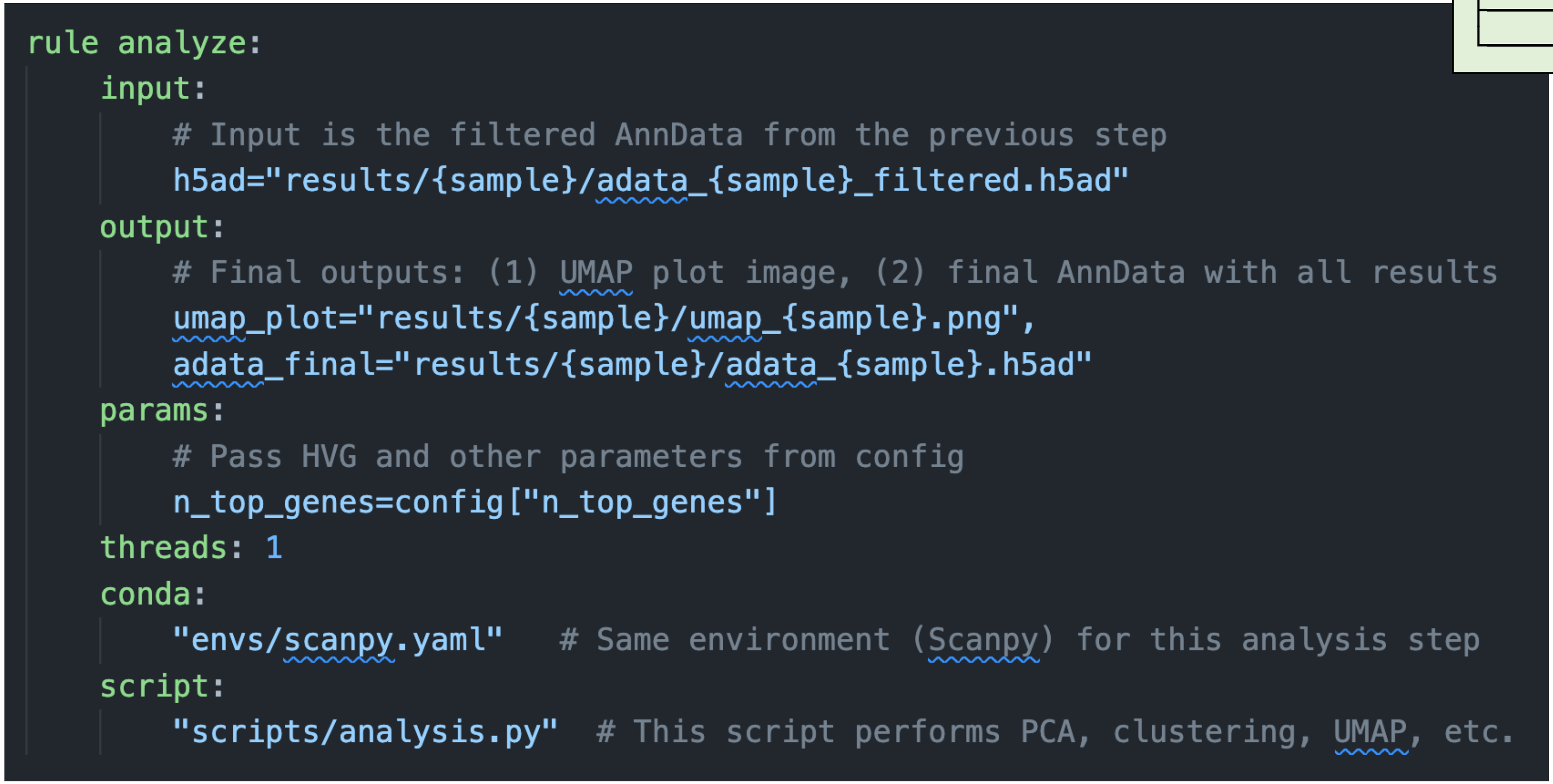
Snakefile:



```
rule preprocess:
    # Input: path to the 10x Genomics matrix directory for this sample.
    # We assume the 10x data (matrix.mtx, features.tsv, barcodes.tsv) are in data/{sample}/
    input:
        "data/{sample}/"   # directory with 10x data for the sample
    output:
        # Save intermediate AnnData after filtering & normalization
        "results/{sample}/adata_{sample}_filtered.h5ad"
    params:
        # Pass filtering parameters from config to the script
        min_genes=config["min_genes"],
        min_cells=config["min_cells"]
    threads: 1
    conda:
        "envs/scanpy.yaml"   # Use Scanpy conda environment for this rule
    script:
        "scripts/preprocess.py"   # This script will read input and apply preprocessing
```

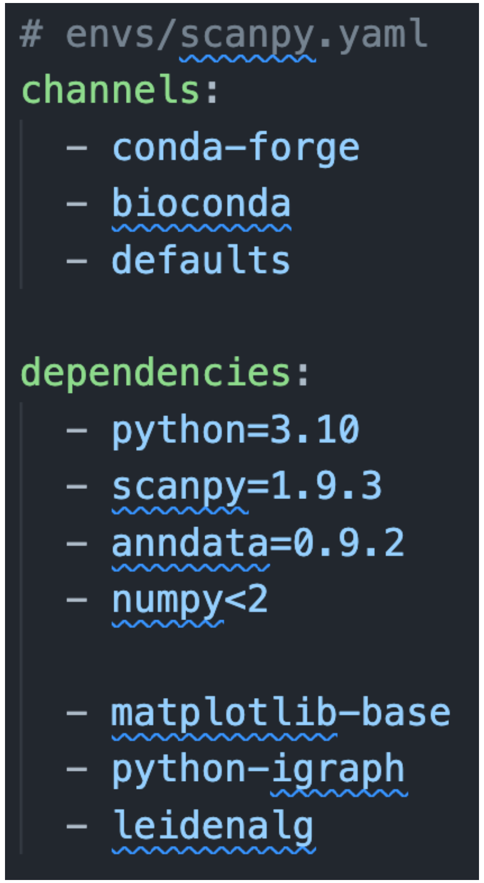
Snakefile:

```
snakemake_project/
├──── snakefile
├──── config.yaml
├──── envs/
│   └──── scanpy.yaml
├──── scripts/
│   ├──── preprocess.py
│   └──── analysis.py
├──── data
├──── results/
└──── README.md
```



```
rule analyze:
    input:
        # Input is the filtered AnnData from the previous step
        h5ad="results/{sample}/adata_{sample}_filtered.h5ad"
    output:
        # Final outputs: (1) UMAP plot image, (2) final AnnData with all results
        umap_plot="results/{sample}/umap_{sample}.png",
        adata_final="results/{sample}/adata_{sample}.h5ad"
    params:
        # Pass HVG and other parameters from config
        n_top_genes=config["n_top_genes"]
    threads: 1
    conda:
        "envs/scanpy.yaml"   # Same environment (Scanpy) for this analysis step
    script:
        "scripts/analysis.py"  # This script performs PCA, clustering, UMAP, etc.
```

```
snakemake_project/
├────── snakefile
├────── config.yaml
├────── envs/
│  └────── scanpy.yaml
├────── scripts/
│  ├────── preprocess.py
│  └────── analysis.py
├────── data
├────── results/
└────── README.md
```

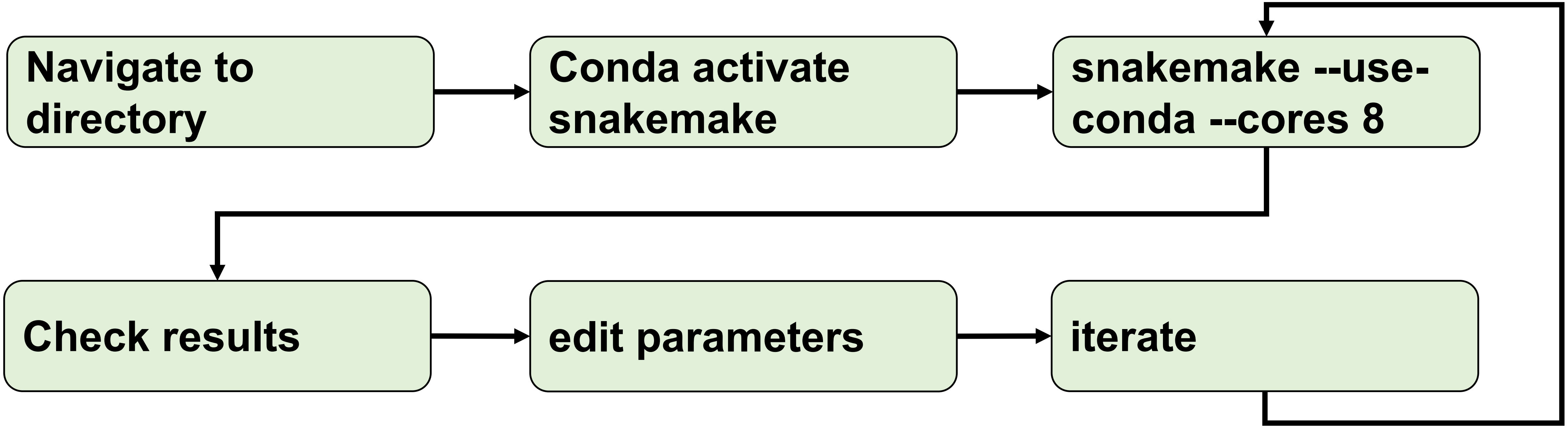


```yaml
# envs/scanpy.yaml
channels:
  - conda-forge
  - bioconda
  - defaults

dependencies:
  - python=3.10
  - scanpy=1.9.3
  - anndata=0.9.2
  - numpy<2

  - matplotlib-base
  - python-igraph
  - leidenalg
```

Running the analysis



Navigate to directory

Conda activate snakemake

snakemake --use-conda --cores 8

Check results

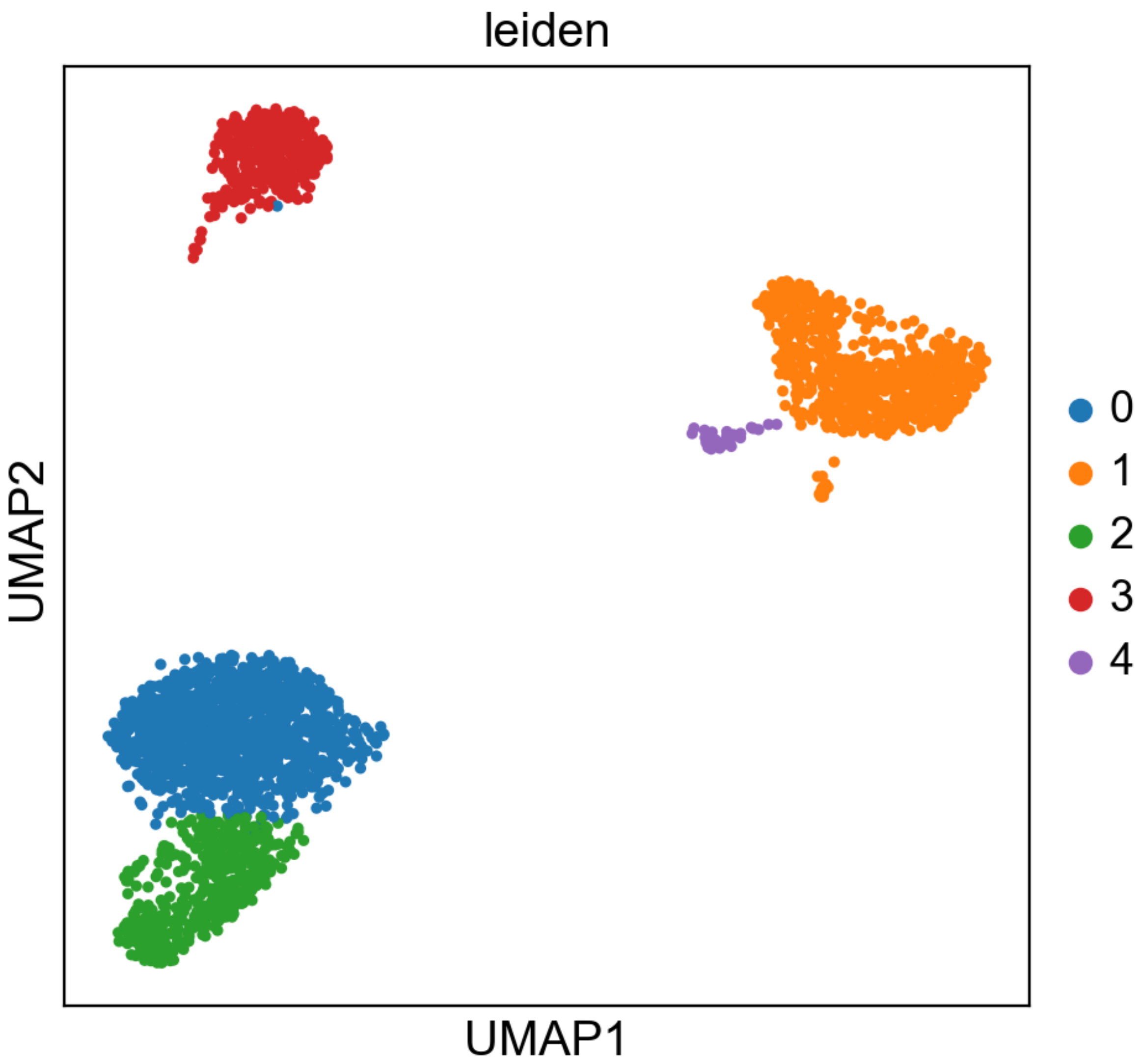
edit parameters

iterate

Results

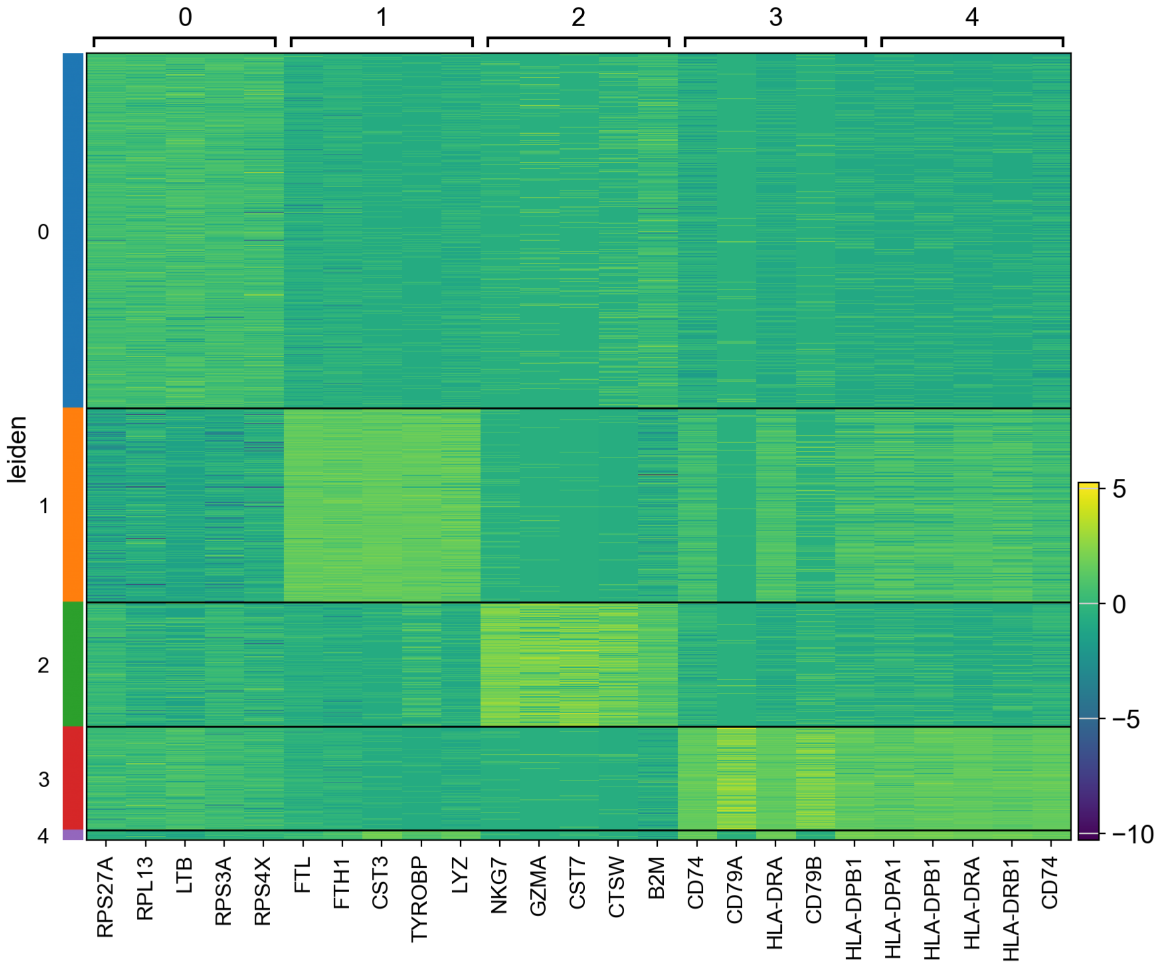
- Identified 6 clusters
- Resolution parameter

- target genes
- Cell type identification
- Explored 2 options



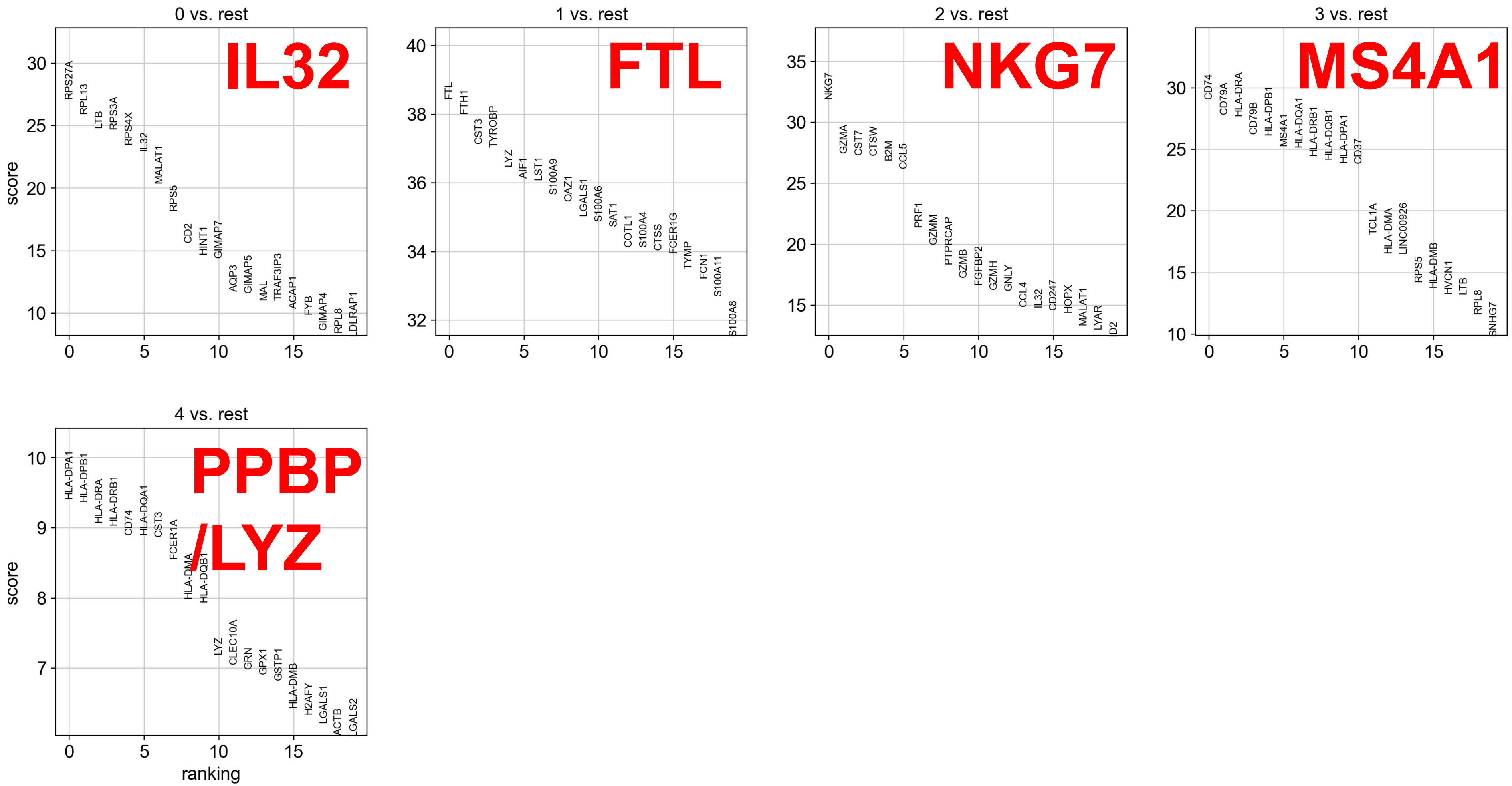
Results

```
sc.pl.rank_genes_group_heatmap()
```



Results

`sc.pl.rank_genes_group()`



IL32

FTL

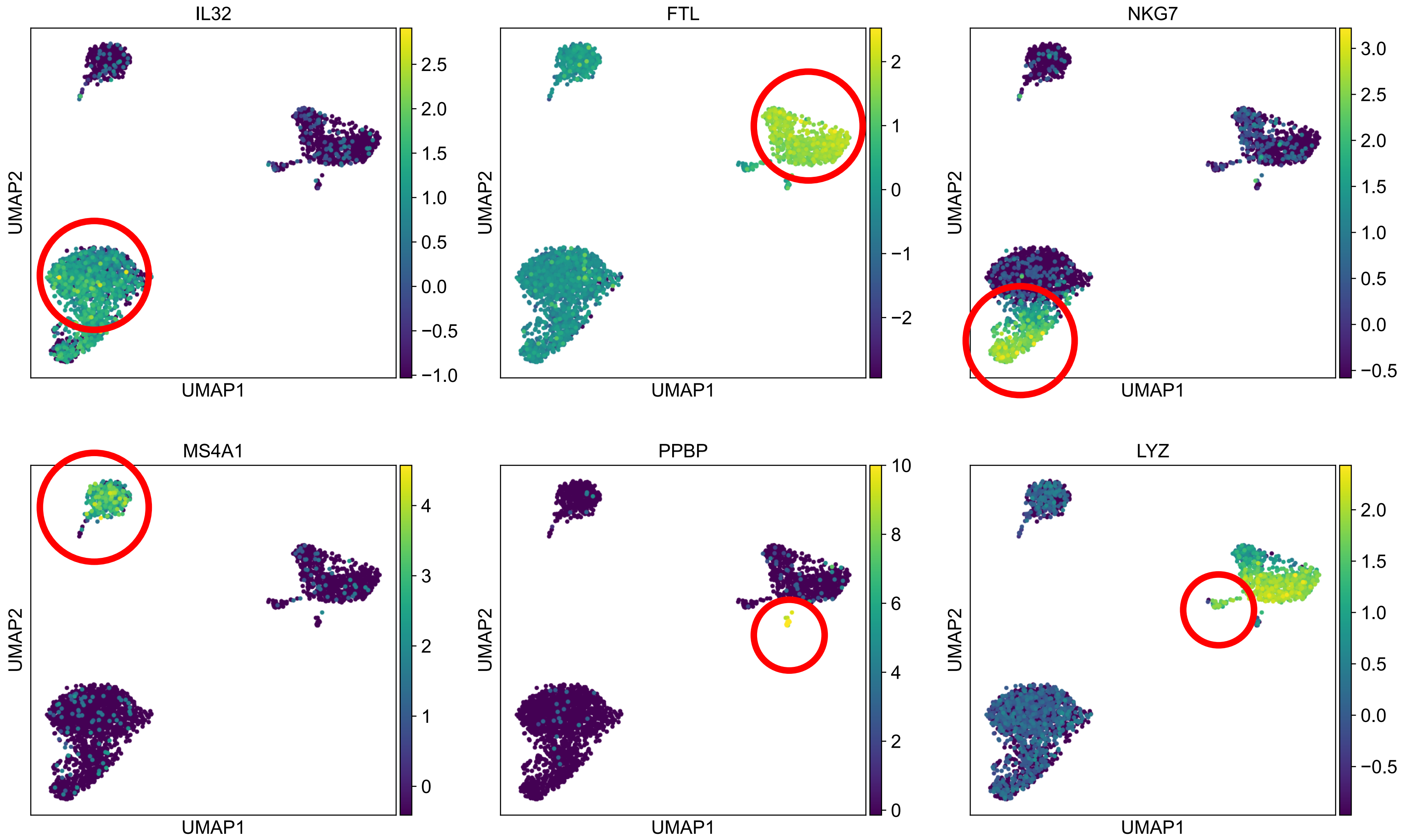
NKG7

MS4A1

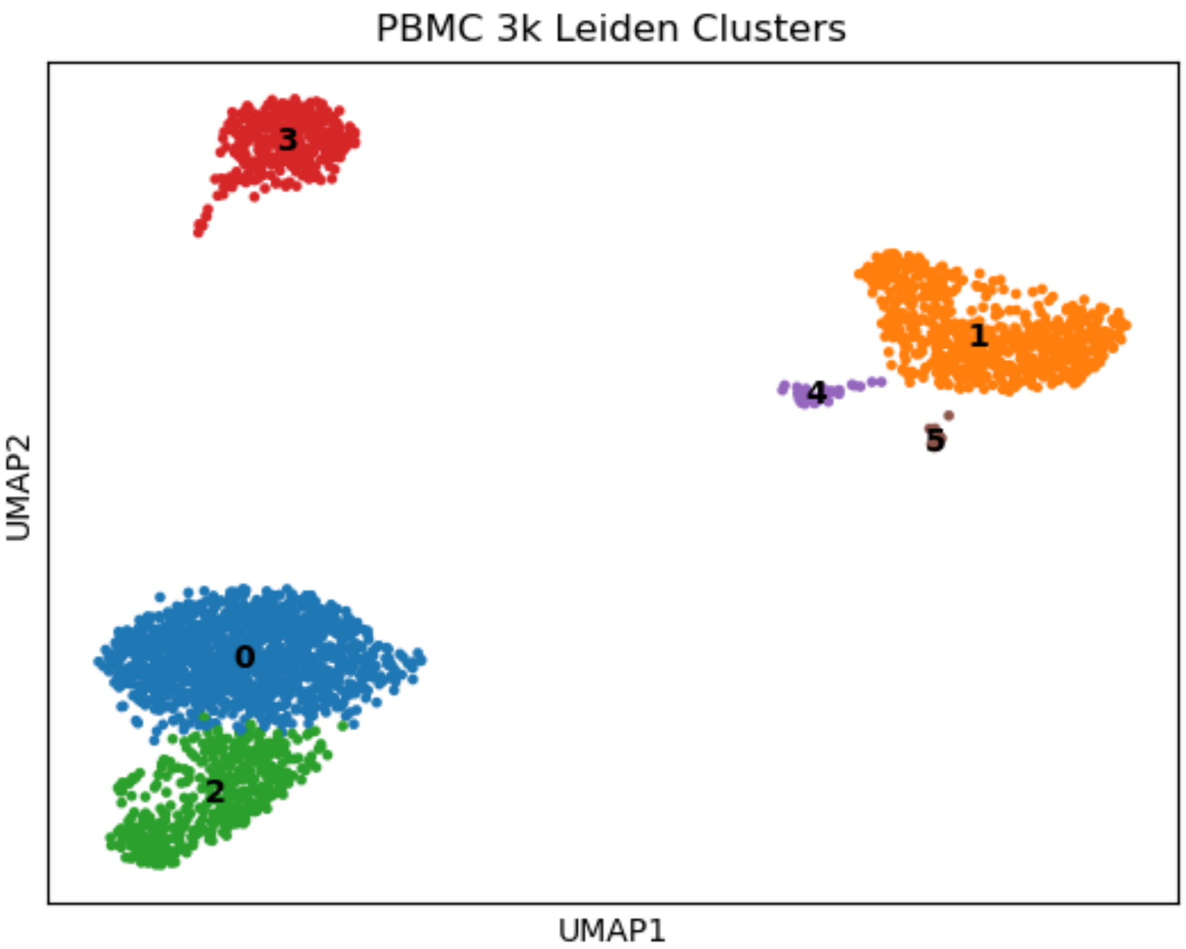
PPBP
/LYZ

Results

```
sc.pl.umap()
```



Results



0: IL32: CD4$^+$ T cells

1: FTL: CD14$^+$ Monocytes

2: NKG7: NK cells

3: MS4A1: B cells

4: PPBP: Platelets

5: LYZ: Dendritic cells

GitHub

- Initialized public repository
- Initially committed all data -> failed
- Wrote .gitignore to exclude large files
- Committed after each large change in code structure
- Wrote README for reproducibility
- Final commit today with the pptx slides

Thank you for your attention!