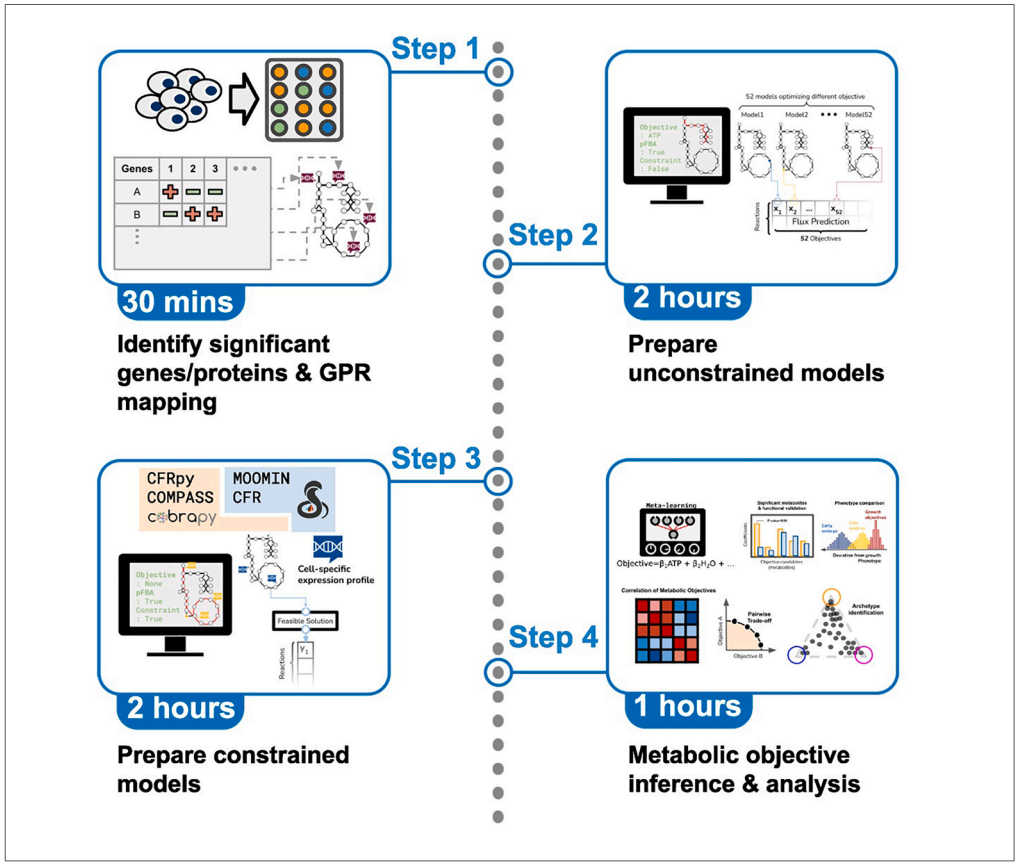
STAR Protocols

CellPress
OPEN ACCESS

Protocol

# Protocol for single-cell optimization objective and trade-off inference




Single-cell optimization objective and trade-off inference (SCOOTI) is a computational framework that integrates bulk and single-cell omics data with genome-scale metabolic modeling to infer metabolic objectives and trade-offs in biological systems. Here, we present a protocol for installing and running SCOOTI, using transcriptomics, proteomics, and metabolomics data to constrain metabolic models. We describe steps to interpret metabolic priorities across different cell states or conditions through clustering, dimensionality reduction, and trade-off analysis.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Da-Wei Lin, Cara Teixeira, Carolina Chung, Sayan Saha Roy, Amit Ghosh, Sriram Chandrasekaran

csriram@umich.edu

## Highlights

Steps to infer metabolic objectives from bulk and single-cell omics data

Construction of unconstrained and omics-constrained genome-scale metabolic models

Meta-regression-based inference of metabolic objective coefficients across conditions

Downstream analysis of metabolic trade-offs and objective priorities across cell states

Lin et al., STAR Protocols 7, 104373
March 20, 2026 
https://doi.org/10.1016/j.xpro.2026.104373

STAR Protocols

CellPress
OPEN ACCESS

Protocol

# Protocol for single-cell optimization objective and trade-off inference

Da-Wei Lin,[1,2,8] Cara Teixeira,[1,5] Carolina Chung,[3] Sayan Saha Roy,[6] Amit Ghosh,[7] and Sriram Chandrasekaran[1,3,4,5,9,*]

[1]Gilbert S. Omenn Department of Computational Medicine and Bioinformatics, Ann Arbor, MI 48109, USA

[2]Department of Statistics, University of Michigan, Ann Arbor, MI, USA

[3]Department of Biomedical Engineering, University of Michigan, Ann Arbor, MI, USA

[4]Program in Chemical Biology, University of Michigan, Ann Arbor, MI, USA

[5]Rogel Cancer Center, University of Michigan Medical School, Ann Arbor, MI 48109, USA

[6]School of Nano Science Technology, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal 721302, India

[7]School of Energy Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal 721302, India

[8]Technical contact

[9]Lead contact

*Correspondence: csriram@umich.edu
https://doi.org/10.1016/j.xpro.2026.104373

## SUMMARY

**Single-cell optimization objective and trade-off inference (SCOOTI) is a computational framework that integrates bulk and single-cell omics data with genome-scale metabolic modeling to infer metabolic objectives and trade-offs in biological systems. Here, we present a protocol for installing and running SCOOTI, using transcriptomics, proteomics, and metabolomics data to constrain metabolic models. We describe steps to interpret metabolic priorities across different cell states or conditions through clustering, dimensionality reduction, and trade-off analysis.**

## BEFORE YOU BEGIN

The simulation of metabolic networks has proven vital in characterizing the metabolic landscape in various cell types and disease models. However, standard methods in metabolic simulation such as flux balance analysis typically assume a common metabolic objective of biomass maximization across cell types, limiting their applicability to non-proliferating cell states. Thus, the advancement of metabolic modeling requires methods that account for the more complex objectives of non-proliferating cells. This protocol aims to assist users in applying the SCOOTI method to transcriptomics, proteomics, or metabolomics data to elucidate the balance and management of multiple metabolic objectives.[1] Overall, SCOOTI's pipeline involves the preparation of omics-constrained metabolic models to obtain flux predictions, inference of metabolic objectives, and analysis of metabolic objectives and trade-offs.

### Innovation

SCOOTI advances metabolic objective inference by integrating constraint-based modeling with a meta-regression framework. This framework learns sample-specific mixtures of metabolic goals. SCOOTI doesn't rely on a single global objective (such as biomass) or on manually exploring a small set of bi-objective combinations. Instead, SCOOTI infers interpretable "objective allocations" that show how different goals shape the observed flux state. These allocations provide condition-specific priorities that can be compared across cohorts, experimental conditions, or cellular states, offering a more nuanced view of metabolic strategy.

To make these capabilities accessible, SCOOTI is delivered as a streamlined, single-entry-point command-line tool. It unifies model construction, flux prediction, objective inference, and trade-off analysis into a coherent workflow. Users can run the full pipeline end-to-end or execute individual stages as needed. The tool produces standardized outputs and consistent logs to support reproducibility and large-scale batching. This design removes the need for manual bookkeeping, parameter sweeps, or ad-hoc scripting, making objective analysis easier to incorporate into routine computational pipelines.

SCOOTI is optimized for diverse data modalities, including bulk RNA-seq contrasts and single-cell state transitions. This flexibility allows the same framework to be applied from population-level summaries to fine-grained cellular trajectories. Together, these innovations provide a transparent, extensible, and scalable path from gene-expression changes to quantifiable metabolic goals and trade-offs, enabling more systematic discovery of metabolic strategies across biological systems.

## KEY RESOURCES TABLE

| REAGENT or RESOURCE | SOURCE | IDENTIFIER |
|---|---|---|
| Deposited data | | |
| Recon1 | Human Genome-scale Metabolic Model, Duarte et al.[2] | https://doi.org/10.1073/pnas.0610772104 |
| Shen2019 | Human Genome-scale Metabolic Model, Shen et al.[3] | https://doi.org/10.7303/syn17114770 |
| Recon2.2 | Human Genome-scale Metabolic Model, Swainston et al.[4] | https://doi.org/10.1007/s11306-016-1051-4 |
| Recon3D | Human Genome-scale Metabolic Model, Brunk et al.[5] | https://doi.org/10.1038/nbt.4072 |
| MCF10A quiescent cells | Bulk RNA-seq, Min et al.[6] | GSE122927 |
| Serum-starved fibroblasts + EGF/TPA | Bulk RNA-seq, Sharma et al.[7] | https://doi.org/10.1016/j.gene.2021.145842 |
| Contact-inhibited dermal fibroblasts | Bulk RNA-seq, Mitra et al.[8] | GSE117444 |
| Zygote, 2-cell, blastocyst embryos | scRNA-seq, Wang et al.[9] | GSE136714 |
| Software and algorithms | | |
| SCOOTI | Lin et al.[10] | https://github.com/sriram-lab/SCOOTI, https://doi.org/10.5281/zenodo.14201370 |
| CFRpy | Shen et al.[3] | https://github.com/sriram-lab/CFRpy |
| CFR-MATLAB | Shen et al.[3] | https://www.synapse.org/Synapse:syn17114770/wiki/588042 |
| DFA-MATLAB | Shen et al.[3] | https://www.synapse.org/Synapse:syn7253624/wiki/406160 |
| COBRA toolbox | openCOBRA | https://github.com/opencobra/cobratoolbox |
| COMPASS | Wagner et al.[11] | https://github.com/YosefLab/Compass/tree/master |
| MOOMIN | Pusa et al.[12] | https://github.com/htpusa/moomin/tree/master |
| UMAP | McInnes, Healy et al.[13] | https://umap-learn.readthedocs.io/en/latest/ |
| Gurobi | Gurobi Optimization, LLC | https://www.gurobi.com/ |
| scikit-learn | Pedregosa et al.[14] | https://scikit-learn.org/stable/index.htm |
| COBRApy | openCOBRA | https://cobrapy.readthedocs.io/en/latest/ |
| CPLEX | IBM | https://www.ibm.com/products/ilog-cplex-optimization-studio |
| HDBSCAN | McInnes, Healy, Astels | https://hdbscan.readthedocs.io/en/latest/ |
| Python 3.10.19 | Python Software Foundation | https://www.python.org/ |
| MATLAB | MathWorks | https://www.mathworks.com/ |

## STEP-BY-STEP METHOD DETAILS

### Installation of SCOOTI and virtual environment

Timing: 1–2 h

This section describes how to obtain the **SCOOTI** source code, set up the required software environment, and verify a successful installation. Users will clone the SCOOTI repository, create and activate

a Conda-based virtual environment, install Python dependencies, and confirm access to the SCOOTI command-line interface (CLI). Completion of these steps ensures that both the Python analysis components and the MATLAB-based metabolic modeling tools are correctly configured for running the demonstrations and analyses described in this protocol.

1. Clone SCOOTI with git.

```
$ git clone https://github.com/sriram-lab/SCOOTI.git.
```

2. Set up virtual environment and install SCOOTI.

```
$ cd SCOOTI

$ conda env create -f environment.yml

$ conda activate scooti

$ pip install -r requirements.txt # optional; make sure everything is installed

$ pip install -e .
```

*Note:* If you encounter errors during pip install -r requirements.txt, rerun with `pip install --no-build-isolation -r requirements.txt`.

3. Confirm the installation is correct.

```
$ python -c "from scooti._version import version; print(version)
```

4. Confirm the important scripts listed in the folder structure of SCOOTI. To run the demonstrations in this protocol, please make sure your current directory is at `SCOOTI/`.

```
SCOOTI/
|
+-- setup.py                          (Python packaging file)
+-- environment.yml                   (Conda environment setup)
+-- requirements.txt                  (pip-based dependency list)
+-- README.md                         (Project overview and usage)
|
+-- scooti/
|   +-- scooti.sh                        (Main CLI)
|   +-- run_flux.sh                      (Flux modeling CLI)
|   +-- run_trainer.sh                   (Training CLI)
|   +-- run_analyzer.sh                  (Analysis CLI)
|   +-- metabolicModels/                 (MATLAB modeling code)
|       +-- CFRinterface.m                  (transcriptomic/proteomic constraints)
```

```
|     +-- DFAinterface.m                  (metabolomic constraints)
|     +-- multiObj_CBM.m                   (main function)
|     +-- GEMs/                            (Metabolic models, genes, objectives)
|        +-- Shen2019.mat                     (Genome-scale metabolic model)
|
+-- examples/                          (Example configs and scripts)
+-- quickstart/                    (End-to-end example)
+-- identifySigGenes_demo/         (Generate DE gene lists)
+-- unconstrained_demo/            (Generate unconstrained models)
+-- constrained_demo/              (Generate constrained models)
+-- inference_demo/                (Generate objectives)
+-- analyze_demo/                  (Objective analysis demo)
+-- tradeoff_demo/                 (Trade-off analysis demo)
```

⚠ CRITICAL: Access SCOOTI via its command-line interface (CLI), `scooti`. This protocol uses the CLI end-to-end.

```
(scooti)$ scooti -h
Usage:
  scooti quickstart scooti
  quickstart-scembryo scooti [OPTIONS]
Options (each expects a JSON config; executed in given order):
  -U, --uncon PATH Unconstrained modeling via MATLAB engine (run_flux.sh)
  -C, --con PATH Constrained modeling (same engine; config sets constraint)
  -I, --infer PATH Training/inference (run_trainer.sh)
  -A, --analyze PATH Post-analysis (run_analyze.sh)
  -T, --tradeoff PATH Tradeoff analysis (run_tradeoff.sh)
  -h, --help Show this help and exit
Examples:
  scooti -U ./examples/unconstrained_demo/unconstrained_demo_config.json
  scooti -U uncon.json -C con.json -I infer.json -A analyze.json
  scooti -T ./examples/tradeoff_demo/tradeoff_config.json
  scooti quickstart-scembryo
```

### Quickstart to infer and analyze metabolic objectives

Timing: 1–4 h

This section provides a guided quickstart workflow for inferring and analyzing metabolic objectives using SCOOTI with precomputed metabolic flux predictions. Using a minimal example dataset comparing proliferating and quiescent cells, users will run objective inference through the SCOOTI command-line interface, configure input parameters via JSON files, and perform downstream analyses to visualize and interpret inferred objective coefficients and trade-offs. This quickstart is intended to familiarize users with the end-to-end SCOOTI workflow (Figure 1) before applying the method to custom datasets.

⚠ CRITICAL: To use SCOOTI, users must provide predicted metabolic fluxes, which typically represent flux solutions inferred from omics data using genome-scale metabolic models (GEMs). In this quickstart tutorial, we assume that such flux predictions are already available. To perform objective inference, SCOOTI also requires a set of ideal objective flux distributions as reference objectives. For convenience, SCOOTI provides a default set of 52 precomputed ideal metabolic objectives, which users can directly use when starting from omics-based flux data alone. Users are not required to generate these ideal objective fluxes themselves for the quickstart. Alternatively, advanced users may supply their own ideal objective fluxes, with any user-defined objectives, or generate flux predictions using custom omics-integration pipelines rather than relying on SCOOTI's default resources. If you do not yet have flux predictions, you may follow along using the provided example flux file included with SCOOTI.

*Note:* `Unconstrained models` refer to flux predictions generated by optimizing a single objective without incorporating omics-based constraints. `Constrained models` are flux predictions generated under omics-derived constraints, without optimizing any specific objective function.

5. Minimum configurations for objective inferences.
   a. Make sure the command $ `scooti -h` works.

   *Note:* This demo section uses a minimal example dataset comparing proliferating vs quiescent cells to get familiar with the workflow. For instance, gene expression data from quiescent cell experiments have been processed to generate flux data including constrained models and unconstrained models in `SCOOTI/examples/quickstart/cell_quiescence/constrained_models/` and `./unconstrained_models/`.

   b. To start running objective inference with our prepared flux models, one can use the command below.

```
(scooti)$ scooti quickstart
```

   c. Create or revise a JSON file before running the model inference. This file is required for SCOOTI to read data and infer metabolic objectives. We have provided an example JSON file to demonstrate a basic configuration at `./examples/quickstart/inference_reproduce/reproduce_inference_config.json`.

   *Note:* Users are encouraged to revise the file to customize their models.

```
# reproduce_inference_config.json

# Complete list of configuration options

# Current path: /home/user/SCOOTI/

{

"unconModel":
```

```
"./examples/quickstart/cell_quiescence/unconstrained_models/",
"conModel": "./examples/quickstart/cell_quiescence/constrained_models/",
"savePath":
"./examples/quickstart/cell_quiescence/inference_reproduce/out/regression_models/",
"kappaArr": "0.1",
"rhoArr": "10",
"dkappaArr": "-1",
"expName": "inference_reproduce",
"unconNorm": "T",
"conNorm": "F",
"medium": "DMEMF12",
"method": "cfr",
"model": "recon1",
"inputType": "flux",
"clusterPath": "",
"objListPath": "",
"rank": "F",
"stackModel": "F",
"sampling": "F",
"learner": "L",
"geneKO": "F", # only required for KOs
"geneListPath": "", # only required for KOs
"learningRate": 0.001, # not necessary for regressions
"epo": 5000, # not necessary for regressions
"fileSuffix": "_fluxes.csv.gz"
}
```

d. Define paths to your unconstrained models (unconModel) and constrained models (`conModel`) directories (created in step 17–22).

*Note:* The flux models in these directories should be either CSVs or compressed CSVs (`.csv.gz`). To make SCOOTI load/save flux models, the suffix of the files should be defined in the `fileSuffix` in the configuration file.

e. Define a path to save regression models or outputs (e.g., savePath for the results).

*Note:* Ensure that other parameters (such as solver settings) are configured as needed. Full parameter control using JSON files is described in **Steps 24–25**. In this quick start tutorial, we assume that users have **already obtained flux solutions inferred from omics data** (e.g., transcriptomics, proteomics, or metabolomics) using a genome-scale metabolic model (GEM). These omics-derived fluxes constitute the **primary input** to SCOOTI. To enable objective inference, SCOOTI requires a set of **ideal objective flux distributions** as reference objectives. For convenience, SCOOTI **provides a default set of 52 precomputed ideal metabolic**

**objectives**, which can be directly used when users only have omics-based flux data. Importantly, users may **replace or extend this default set** by supplying their **own ideal objective fluxes**, with **any number of objectives**, depending on the biological system and research question. Instructions for specifying custom objective sets or using alternative GEMs are provided in **Steps 39–42**.

6. Run a quick inference of metabolic objectives.
   a. Although we have welcomed users to execute the command
   to launch an inference-and-analysis demo, an inference-only run can be launched by.
   This command will perform a meta-regression to infer the contribution (coefficient) of each candidate objective to the observed flux distribution. This yields a set of **metabolic objective coefficients** for the sample or condition.

```
$ scooti quickstart
```

```
$ bash examples/quickstart/inference_reproduce/run_reproduce_inference.sh
```

   b. A quick analysis of metabolic objectives can be done by using the deviation from the biomass objective (set `distance` to true) (Figures 2A and 2B), objective allocation (default analysis) (Figure 3C), proportion of objective selection (set `compare` to true) (Figure 3B), and significant objectives (set `compare` to true) (Figure 3A). To perform these analyses, we need a configuration file like this:

```
# current directory is /home/user/SCOOTI/

# ./examples/quickstart/analyze_reproduce/analyzer_reproduce_demo.json

{

"flux_paths": {"exp": "./examples/quickstart/cell_quiescence/constrained_models/"},

"coef_paths": {"exp": "./examples/quickstart/inference_reproduce/out/regression_models/"},

"save_root_path": "./examples/quickstart/analyze_reproduce/out/",

"engine": "legacy",

"reduction": "auto",

"GEM_path": "./scooti/metabolicModel/GEMs/Shen2019.mat",

"uncon_model_path": "./SCOOTI/examples/quickstart/cell_quiescence/unconstrained_models/",

"col_map": {},

"samplingFlux_path": "",

"sel_para": "k0.1_r10",

"prefix": "analyze_reproduce",

"medium": "DMEMF12",

"labels": {"mode": "contains", "contains_pattern": "P", "true_label": "Proliferation", "false_label": "Quiescence"},

"get_coef": {"metType_cluster": false},

"coef_analysis": {"unknown_clustering": false, "clustering": true, "entropy": true, "distance": true, "compare": true, "umap_para": [5, 50], "method": "average", "ref_col": null}

}
```

c. After preparing the configuration file, run this command to generate analytical plots.

```
$ bash examples/quickstart/analyze_reproduce/run_analyze_reproduce.sh
```

```
# Example output of quickstart

(scooti)$ ~/sriram-lab/SCOOTI$ scooti quickstart

[scooti] Logging to: /SCOOTI/logs/scooti-20251201-040619-17839.log

[reproduce-all] 1/2 Inference ...

[inference-reproduce] Running meta-regression with: /SCOOTI/examples/quickstart/inference_reproduce/reproduce_inference_config.json

[inference-reproduce] unconstrained dir: /SCOOTI/examples/quickstart/cell_quiescence/unconstrained_models/

[inference-reproduce] constrained dir: /SCOOTI/examples/quickstart/cell_quiescence/constrained_models/

[inference-reproduce] medium='DMEMF12' fileSuffix='_fluxes.csv.gz'

[inference-reproduce] Scanning unconstrained metadata in: /SCOOTI/examples/quickstart/cell_quiescence/unconstrained_models/

[inference-reproduce] Found medium tags in unconstrained: 53 DMEMF12

testing bug...

Start processing the unconstrained models...

Start processing the unconstrained models...

Returning reaction fluxes...

Finishing up the data loading of unconstrained models...

Start processing the constrained models...

False Ordinary constrained models Start processing the data of constrained models...

Start collecting filenames... 100%|----------| 60/60 [00:00<00:00, 39138.14it/s]

Start merging tables 100%|----------| 30/30 [00:00<00:00, 319.19it/s] ...
```

```
Finishing up loading the constrained models...

Start loading models... (3757, 52) (3757, 30)

Check the size of the models: (1515, 52) (1515, 30) (1515, 52) (1515, 30)

./examples/quickstart/inference_reproduce/out/regression_models//flux_rfelm_inference_reproduce_cfr_True_False_recon1_DMEMF12_k[0.1]_rho[10.0].csv Start training regression models...

Mapping...

100%|----------| 30/30 [00:00<00:00, 275941.05it/s]

fitting linear meta model... (5, 1) (52, 1) ...

[inference-reproduce] Done.

Outputs in: /SCOOTI/examples/quickstart/inference_reproduce/out/regression_models/

[reproduce-all] 2/2 Analysis (legacy) ...
```

```
[analyze-reproduce] Running analysis with: /SCOOTI/examples/quickstart/analyze_repro-
duce/analyze_reproduce_config.json

[analysis] coef_paths[exp] -> examples/quickstart/inference_reproduce/out/regression_
models/flux_sl_inference_reproduce_cfr_True_False_recon1_DMEMF12_k[0.1]_rho[10.0].csv
[analysis] init:

save_root_path=./examples/quickstart/analyze_reproduce/out/ medium=DMEMF12 prefix=analyze_
reproduce

[analysis] get_coef ...

[analyze-reproduce] Done. Outputs under ./examples/quickstart/analyze_reproduce/out/

[reproduce-all] Done.
```

*Note:* SCOOTI offers more analytical methods for both flux predictions and objective coefficients. Please refer to the last steps 28-29 of this protocol (e.g., Figures 2 and 3).

*Note:* Different versions of MATLAB and Gurobi can generate different flux solutions and lead to different objective coefficients while the overall trends should be preserved.

### Identifying significant genes or proteins

Timing: 30 min

This section describes how to identify significantly up- and down-regulated genes or proteins from bulk or single-cell omics data for use as constraints in genome-scale metabolic modeling. Using representative examples, users will generate condition-specific gene or protein lists that reflect differential regulation between biological states, such as proliferation and quiescence. These lists serve as inputs for constructing constrained metabolic models in subsequent steps of the SCOOTI workflow.

*Note:* There are several methods to identify differentially expressed genes or proteins. Here we demonstrate the procedure using a representative approach. In the context of quiescence-proliferation transitions, we identify metabolic genes that are significantly up- or down-regulated between cell-type transitions. These gene lists will be used to constrain the metabolic models in subsequent steps.

7. Determine significantly regulated genes (example: cell quiescence data).

*Note:* To understand SCOOTI's workflow, identify significantly up- and down-regulated metabolic genes from single-cell or bulk omics data. For example, compare gene expression in cell quiescence to find metabolic genes that are up-regulated in quiescent states relative to proliferative states, and vice versa.

a. If it is bulk-omic data with biological replicates, we can directly initiate the instance of a Python object `findSigGenes` with the path to access the data.
b. If it is single-cell data, we recommend converting data into Market Exchange Format (MEX), which is compatible with the function for `10x_genomics`.[15,16]

*Note:* To convert the data table, we offer two functions in the Python class `findSigGenes`. The first function `table_to_10xformat` can read the data from either Excel or CSV files. However, if additional preprocessing of the data table is required, users may want to apply `Pandas.DataFramedf_to_10xformat` to convert the data from `Pandas.DataFrame` to MEX format.

```
# Python

# find regulators

fr = findSigGenes(dataset_path)

# read single-cell data

adata = fr.read_scRNAseq()

# optional

fr.merge_adata(rename_cellNames=True)

# get processed expression data

genedf = fr.get_genedf(transpose=False)
```

*Note:* It is highly recommended to apply batch normalization, standardization, and quantile normalization.

8. To demonstrate the logic, we have provided a minimal configuration in `SCOOTI/example/identifySigGenes_demo/identify_siggenes_config.json` identifying significant genes from cell quiescence datasets.

```
# identify_siggenes_config.json

{

"out_dir": "./examples/identifySigGenes_demo/out/",

"fdr_alpha": 0.05,

"fc_threshold": 1.0,

"sharma_csv":  "/SCOOTI/examples/example_omics/cell_quiescence/sharma_21_prolif_pmid_
34274479.csv",

"min_csv": "/SCOOTI/examples/example_omics/cell_quiescence/min_18_GSE122927_ReadCount.
csv",

"qp_csv": "/SCOOTI/examples/example_omics/cell_quiescence/johnson_18_GSE117444_prolif_
qui_count.csv",

"sharma_groups": {

"EGF": {"exp": [2,6,10], "ctrl": [1,5,9,15]},

"TPA": {"exp": [8,12], "ctrl": [1,5,9,15]},

"H89-EGF": {"exp": [13,16], "ctrl": [7,11,17]},

"H89-TPA": {"exp": [14,18], "ctrl": [7,11,17]}},

"min_pairs": [

{"label": "p21_2E2", "regex": "(?=.*p21_high)(?=.*2E2)|(?=.*p21_low)(?=.*2E2)"},

{"label": "p21_3B6", "regex": "(?=.*p21)(?=.*3B6)"},

{"label": "SerumStarvation_2E2","regex": "(?=.*SerumStarvation)(?=.*2E2)|(?=.*2E2)(?=.
*Control)"},

{"label": "SerumStarvation_3B6","regex": "(?=.*SerumStarvation)(?=.*3B6)|(?=.*3B6)(?=.
*Control)"},

{"label": "Meki_2E2", "regex": "(?=.*Meki)(?=.*2E2)|(?=.*2E2)(?=.*Control)"},
```

```
{"label": "Meki_3B6", "regex": "(?=.*Meki)(?=.*3B6)|(?=.*3B6)(?=.*Control)"},
{"label": "CDK46i_2E2", "regex": "(?=.*CDK46i)(?=.*2E2)|(?=.*2E2)(?=.*Control)"},
{"label": "CDK46i_3B6", "regex": "(?=.*CDK46i)(?=.*3B6)|(?=.*3B6)(?=.*Control)"},
{"label": "ContactIn_2E2", "regex": "(?=.*ContactIn)(?=.*2E2)|(?=.*2E2)(?=.*Control)"},
{"label": "ContactIn_3B6", "regex": "(?=.*ContactIn)(?=.*3B6)|(?=.*3B6)(?=.*Control)"}
],
"qp_split": {"ctrl_n": 3, "exp_n": 3}
}
```

9. For each biological transition or condition of interest (e.g., proliferation vs quiescence; labeled with or without "_P_"), begin with identifying genes that are significantly up-regulated or down-regulated.
   a. Use appropriate statistical thresholds, such as an adjusted `p-value < 0.05` and a minimum fold-change or Z-score cutoff (e.g., `Z-score>1` for up-regulated, `Z-score<-1` for down-regulated).
   b. Ensure that the gene identifiers used match those in the genome-scale metabolic model, as they will later be mapped to corresponding reactions.
10. For each transition, prepare two gene lists: one containing up-regulated genes and one containing down-regulated genes. Save these lists as CSV files using a consistent naming convention:
   a. `SAMPLE_upgenes.csv` for up-regulated genes and `SAMPLE_dwgenes.csv` (or `SAMPLE_downgenes.csv`) for down-regulated genes.
   b. For example, for the cell proliferation datasets, you would generate `SerumStarvation_3B6_P_upgenes.csv` and `SerumStarvation_3B6_P_dwgenes.csv`
   c. For its corresponding quiescent states, generate `SerumStarvation_3B6_upgenes.csv` and `SerumStarvation_3B6_dwgenes.csv`.
   d. Each file should contain one gene identifier per line, with no header row.

   *Note:* We suggest two approaches for inferring metabolic objectives based on gene expression data. First, contrastive analysis is useful for identifying genes that are differentially expressed between two distinguishable stages (e.g., comparing the quiescent to the proliferative states). Second, column-wise expression analysis is recommended when focusing on genes that are highly or uniquely expressed in a specific condition or cell type (i.e., examining the expression profile within a single column of the matrix).

11. Place all gene list CSV files in a designated directory (e.g., a data/subfolder within your project).

   ⚠ CRITICAL: Maintaining consistent and informative filenames is crucial, as SCOOTI will use these filenames to associate gene lists with specific conditions or samples. When used in SCOOTI, the up-gene lists are treated as activation constraints (i.e., forcing the associated reactions to carry flux), while the down-gene lists are interpreted as inhibition constraints (i.e., limiting or turning off associated reactions).

   *Note:* Once all gene list files are prepared and saved, SCOOTI will load them as inputs to construct condition-specific constrained metabolic models. For each condition or transition, the software will modify the base genome-scale metabolic model by applying gene-level constraints derived from the up- and down-regulated lists. This results in a tailored network that reflects the expected metabolic state based on the corresponding transcriptomic profile.

12. To run this demonstration, we can call the bash file.

```
$ bash examples/identifySigGenes_demo/run_identify_siggenes.sh
```

*Note:* We did not standardize the workflow to pre-process transcriptomics/proteomics and obtain significant gene/protein lists. This demonstration only shows our logic to prepare gene/protein constraints.

### Prepare unconstrained models optimizing ideal objectives

Timing: 2 h to 1 day

This section describes how to generate unconstrained metabolic flux models by optimizing a genome-scale metabolic model for each candidate metabolic objective independently. Using SCOOTI, users will configure and run flux balance analyses without applying omics-based constraints, producing a set of idealized flux distributions. These unconstrained flux models serve as reference predictors for subsequent meta-regression–based inference of metabolic objectives.

13. Generate unconstrained flux models using SCOOTI:
    a. Run the SCOOTI module to create unconstrained flux models for each candidate's metabolic objective.

    *Note:* To show more cases for users to understand, we will switch to inferring metabolic objectives during embryogenesis.

    b. In this context, "unconstrained" means that no omics-based constraints (e.g., gene expression) are applied—only the default bounds of the model are used.
    c. SCOOTI will optimize the genome-scale metabolic model for each of the 52 objectives independently (e.g., maximizing ATP, NADH, glutathione synthesis) using a solver like Gurobi.
    d. These optimized flux distributions will serve as predictors for meta-regression analysis.
14. Set up configuration for unconstrained model generation:
    a. Define paths to:
        i. COBRA Toolbox (`COBRA_path`).
        ii. Genome-scale metabolic model file (`GEM_path`}.
        iii. Objective candidate list (`obj_candidate_list_file`).
        iv. Output directory (`save_root_path`).

```
# example output
To get started, type doc. For product information, visit www.mathworks.com.
| | COnstraint-Based Reconstruction and Analysis
| | The COBRA Toolbox - 2025
| | Documentation:
| | http://opencobra.github.io/cobratoolbox
Checking if git is installed ... Done (version: 2.43.5).
Checking if the repository is tracked using git ...
Done. Checking if curl is installed ...
Done. Checking if remote can be reached ...
Done.
Initializing and updating submodules (this may take a while)...
Entering 'binary'
```

b. Set `model_name` to specify the model (e.g., Recon1), and medium to the appropriate condition (e.g., KSOM).

*Note:* Genome-scale metabolic model Shen2019.mat is a modified version of Recon1; thus, the model name should be set to "Recon1" in the configurations.[2,3]

c. Use `prefix_name = 'model'` to indicate that the run is unconstrained.
d. Below is an example configuration JSON:

```
# /SCOOTI/examples/unconstrained_demo/unconstrained_model

{

"jj": 1,

"COBRA_path": "~/cobratoolbox",

"GEM_path": "./scooti/metabolicModel/GEMs/Shen2019.mat",

"model_name": "Recon1",

"obj_candidate_list_file": "./scooti/metabolicModel/GEMs/obj52_metabolites_shen2019.csv",

"input_obj_tb": "",

"paraLen": 1,

"random_para": 0,

"init_objective": 1,

"genekoflag": 0,

"rxnkoflag": 0,

"FSflag": 0,

"pfba": 1,

"medium_perturbation": 0,

"data_dir": "",

"prefix_name": "model",

"medium": "KSOM",

"uplabel": "upgenes",

"dwlabel": "dwgenes",

"simulation": "CFR",

"constraint": 0,

"save_root_path": "./examples/unconstrained_demo/out/unconstrained_models/", "CFR_mo-
del_path": "",

"pairwise_CFR_model": 0,

"extraWeight": 0,

"algorithm": "cfr",

"data_series": "",

"prefix_series": "",
```

```
"medium_series": ""
}
```

15. Verify model generation:
    a. Ensure that SCOOTI completes this step without errors and reports non-zero values to the objective fluxes. Since it optimizes all objectives independently, this step may take some time.
    b. If an optimization fails or returns infeasible results, check out your metabolic model setup or solver configuration.
16. Run the flux prediction pipeline:
    a. Execute the shell script to begin the flux prediction step, as shown example below:

```
$ scooti -U examples/unconstrained_demo/unconstrained_demo_config.json
```

*Note:* The output of this step is a folder of flux distributions (one per objective), stored in `./scooti/examples/unconstrained_demo/out/unconstrained_models/`.

```
# example output of "scooti -U"
./examples/unconstrained_demo/out/unconstrained_models/
CFR_ct1_obj11_data
1
0
1
1
0
0
0
1
1
1
1
1
1
{6x1 cell}
./scooti/metabolicModel/GEMs/Shen2019.mat upgenes dwgenes cfr
[CFR] Flux results saved to ./examples/unconstrained_demo/out/unconstrained_models/
[Aug620251706]CFR_ct1_obj11_data1_fluxes.csv The CFR result has been saved in ./examples/
unconstrained_demo/out/unconstrained_models/ [Aug620251706]CFR_ct1_obj12_data1
```

*Note:* The default set of objectives covers major metabolic pathways and functions identified from the Recon1 model. If your system potentially involves additional objectives, you can augment this list (see step 41).

### Construct omics-constrained models

Timing: 2 h to 1 day

This section describes how to incorporate omics-derived constraints into genome-scale metabolic models and generate condition-specific flux predictions using SCOOTI. Using gene or metabolite lists derived from prior steps, users will configure constraint integration methods, apply transcriptomic or metabolomic constraints, and run flux balance analysis to produce constrained flux distributions. These constrained models capture condition-dependent metabolic states and serve as inputs for downstream inference of metabolic objectives and trade-off analysis.

17. Integrate omics constraints and generate flux predictions:
    a. Use significant gene lists obtained in step 12 to constrain the metabolic model for each condition or cell type.
    b. For each condition (e.g., developmental stage or cell type), reactions corresponding to up-regulated genes are kept active, and down-regulated ones are constrained (e.g., zero flux).
    c. Run flux balance analysis (FBA) for each candidate objective on the constrained model to generate flux distributions reflecting the metabolic state.
    d. Specify the paths to access the COBRA environment and metabolic model. The model name is required for SCOOTI to recognize the input.
18. Configure constraint method and parameter settings.
    a. Provide gene list paths using the `data_dir` parameter.
    b. Select the constraint integration method in the `simulation` argument:
        i. **CFR (Constraint Flux Regulation)**: Integrates transcriptomics/proteomics.[17,18]
        ii. **DFA (Dynamic Flux Analysis):** Integrates time-course metabolomics.[17–19]
    c. Use `DFA_kappa = -1` to deactivate DFA mode. Set `CFR_kappa` and `CFR_rho` to positive values to control constraint strength.
    d. Set `DFA_kappa > 0` to switch to DFA mode (applicable when metabolomics data is provided).
    e. Adjust CFR/DFA parameters if needed; e.g., `CFR_rho = '10,1,0.1'` to test multiple strengths.
    f. Set `paraLen = 3` to enable parameter scanning; use `random_para = 1` to randomly select values from given ranges.
    g. Set `init_objective = 1` to omit objective functions (recommended before inferring metabolic objectives); use `init_objective = 2` to assign biomass objective.
    h. Once users properly set `paraLen`, `CFR_rho`, `CFR_kappa`, and `DFA_kappa`. SCOOTI will automate parameter scanning (e.g., over `CFR_rho`, `CFR_kappa`, or `DFA_kappa`). A sample configuration script is shown below:

```
{
"COBRA_path": "~/cobratoolbox",
"GEM_path": "./scooti/metabolicModel/GEMs/Shen2019.mat",
"model_name": "Recon1",
"obj_candidate_list_file":
"./scooti/metabolicModel/GEMs/obj52_metabolites_shen2019.csv",
"input_obj_tb": "",
"paraLen": 9,
"random_para": 0,
"init_objective": 1,
```

```
"genekoflag": 0,

"rxnkoflag": 0,

"FSflag": 0,

"pfba": 1,

"medium_perturbation": 0,

"data_dir": "./examples/example_sigGenes/scEmbryo/",

"prefix_name": "scan_demo",

"medium": "KSOM",

"uplabel": "upgenes",

"dwlabel": "dwgenes",

"simulation": "CFR",

"constraint": 1,

"save_root_path": "./examples/constrained_demo/out/scan_constrained_models/", "CFR_model_path": "",

"pairwise_CFR_model": 0,

"extraWeight": 0,

"algorithm": "cfr",

"data_series": "",

"prefix_series": "",

"medium_series": "",

"CFR_kappa": "10,1,0.1",

"CFR_rho": "10,1,0.1",

"DFA_kappa": -1

}
```


i. Once the script is prepared, run the batch job using:

```
$scooti -U examples/constrained_demo/constrained_scan_demo_config.json
```

j. In contrast to parameter scanning, users can use the `jj` argument in the configuration file to index each parameter combination.

19. Enable *in silico* perturbation options.
    a. genekoflag – enable single-gene knockout simulation.
    b. `rxnkoflag` – enable single-reaction knockout simulation.
    c. `medium_perturbation` – enable single-metabolite depletion simulation.
    d. `pfba` – if set to 1, executes parsimonious flux balance analysis (pFBA), which minimizes total flux.
20. Integrate multi-omics sequentially (optional; see steps 33–38).
    a. First constrain the model with transcriptomics, then with proteomics.
    b. Use `CFR_model_path` to load previously constrained models.
    c. Set `pairwise_CFR_model = 1` if the omics data come from the same sample; set to 0 to use a Cartesian product across omics layers.
    d. Use `extraWeight` to adjust relative importance of each omic layer (e.g., `extraWeight = 1, 1000` to emphasize proteomics over transcriptomics).

21. Set global SCOOTI parameters.
    a. Choose method for constraining models with omics data using `algorithm` (e.g., `CFR`, `MOOMIN`; see more in step 40).
    b. Set the gene list suffixes: `late_stage` for up-regulated, `early_stage` for down-regulated genes (e.g., `early_stage='dwgenes'` and `late_stage='upgenes'`.
    c. Choose an appropriate medium: e.g., `DMEMF12` for cultured cells, `KSOM` for embryos.
    d. Set `save_root_path` to define the output directory (e.g., `constrained_models/`).
    e. Use a meaningful prefix to track conditions.

```
# constrained_demo_config.json

# Complete list of configuration options

{

"jj": 1,

"COBRA_path": "~/cobratoolbox",

"GEM_path": "./scooti/metabolicModel/GEMs/Shen2019.mat",

"model_name": "Recon1",

"obj_candidate_list_file":

"./scooti/metabolicModel/GEMs/obj52_metabolites_shen2019.csv",

"input_obj_tb": "",

"paraLen": 1,

"random_para": 0,

"init_objective": 1,

"genekoflag": 0,

"rxnkoflag": 0,

"FSflag": 0,

"pfba": 1,

"medium_perturbation": 0,

"data_dir": "./examples/example_sigGenes/scEmbryo/",

"prefix_name": "example",

"medium": "KSOM",

"uplabel": "upgenes",

"dwlabel": "dwgenes",

"simulation": "CFR",

"constraint": 1,

"save_root_path": "./examples/constrained_demo/out/constrained_models/", "CFR_model_-
path": "",

"pairwise_CFR_model": 0,

"extraWeight": 0,

"algorithm": "cfr",

"data_series": "",

"prefix_series": "",
```

```
"medium_series": "",
"CFR_kappa": 0.1,
"CFR_rho": 0.01,
"DFA_kappa": -1
}
```

22. Run SCOOTI to generate constrained models.
    a. Execute the SCOOTI pipeline using provided inputs and configuration.

```
$scooti -U examples/constrained_demo/constrained_demo_config.json
```

    b. Outputs will be grouped by condition and saved under the path specified by `save_root_path`.
23. Validate constrained flux models.
    a. Ensure that each condition has a complete set of flux predictions (e.g., from 1C to 2C and from 2C to BC). The middle panel of Figure 1 shows a preview of the predicted fluxes during embryogenesis.

```
# example output
{'SLC27A2'}
{'SLC27A4'}
{'SLC28A3'}
{'SLC2A6'}
{'SLC35A2'}
{'SLC35B2'}
{'SLC35B4'}
{'SLC35D1'}
{'SLC36A1'}
{'SLC38A5'}
{'SLC4A2'}
{'SLC4A4'}
{'SLC4A8'}
{'SLC5A10'}
{'SLC5A6'}
{'SLC7A1'}
{'SLC7A6'}
{'SMS'}
{'SPCS1'}
{'SRD5A1'}
{'SRM'}
{'ST3GAL4'}
```

```
{'SUCLG1'}

{'SUV39H1'}

{'SUV39H2'}

{'SYNJ2'}

{'TALDO1'}

{'TK1'}

{'TYMS'}

{'UAP1'}

{'UCK2'}

{'UCP2'}

{'UCP3'}

{'UMPS'}

{'UQCRFS1'}

{'UXS1'}

{'XYLT1'}

[CFR] Flux results saved to ~/examples/constrained_demo/out/constrained_models/
[Aug620251719]CFR_ctl_obj1_data2_fluxes.csv
```

b. If results are missing or the solver fails, revisit steps 17-22 and re-check constraints and solver setup.

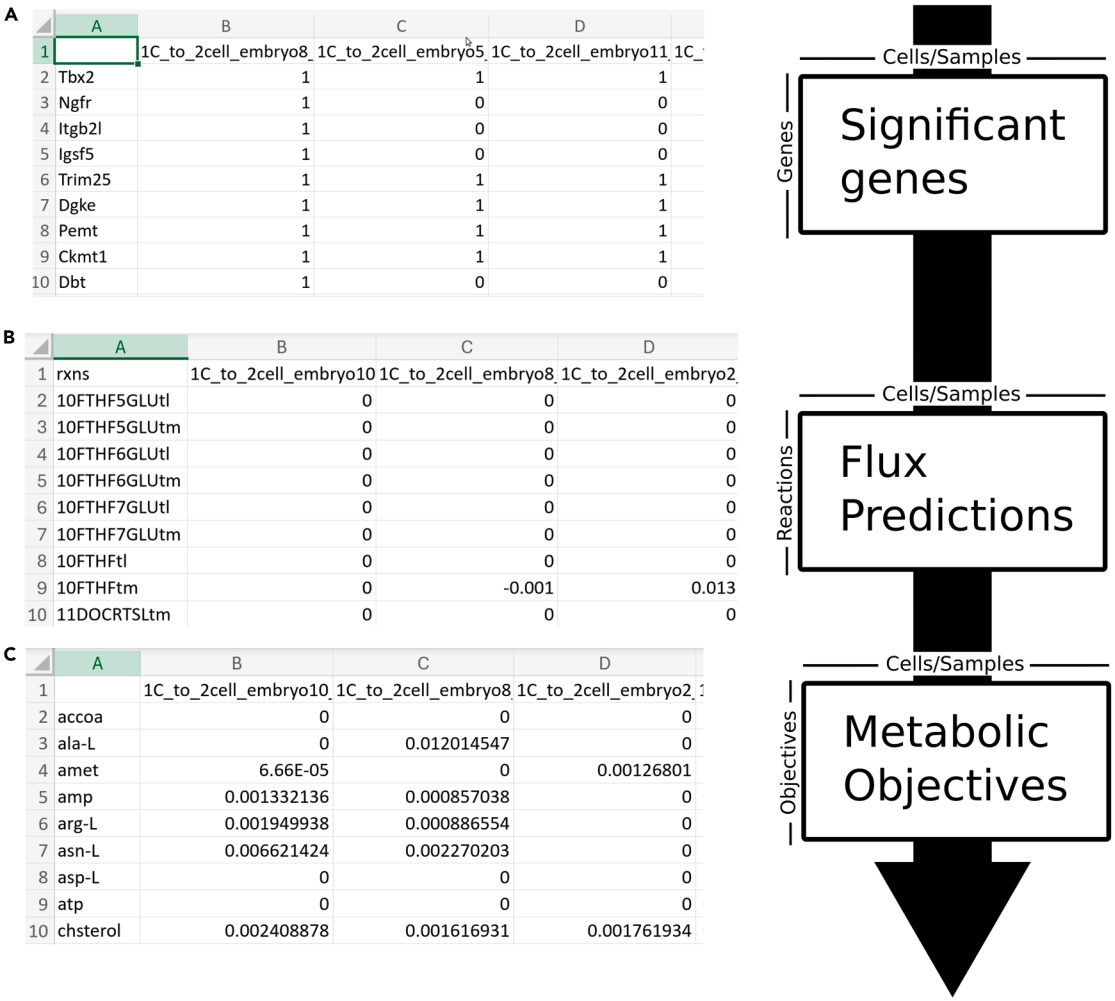
*Note:* When generating constrained models, there will be three files generated for each sample, including a JSON file of metadata, a CSV file of flux prediction, and a MAT file of a model with constraints (a reusable model). To infer the metabolic objectives, the previous two files are necessary, while the last file will be used in the section "Advanced uses: integrate multi-omics data simultaneously".

### Inference of metabolic objectives

Timing: 1 h to 1 day

This section describes how SCOOTI infers metabolic objectives by integrating unconstrained and omics-constrained flux predictions using a meta-regression framework. Users will configure regression inputs, specify regularization parameters, and run the SCOOTI inference pipeline to estimate objective coefficients that quantify the contribution of candidate metabolic objectives to observed flux distributions. The resulting coefficients provide a quantitative representation of metabolic priorities for each condition or sample.

24. Overview of inference process.
    a. With unconstrained flux predictions from step 16 and constrained flux distributions from step 23, SCOOTI infers metabolic objectives using meta-regression.
    b. In this approach, the constrained flux (representing the actual condition) is modeled as a linear combination of the unconstrained flux distributions, each corresponding to a single objective (e.g., ATP production, redox balance).
    c. The regression model computes a coefficient for each objective, representing its estimated contribution to the observed metabolic state.
25. Prepare input paths and regression settings.




A

| | A | B | C | D |
|---|---|---|---|---|
| 1 | | 1C_to_2cell_embryo8_ | 1C_to_2cell_embryo5_ | 1C_to_2cell_embryo11_ |
| 2 | Tbx2 | 1 | 1 | 1 |
| 3 | Ngfr | 1 | 0 | 0 |
| 4 | Itgb2l | 1 | 0 | 0 |
| 5 | Igsf5 | 1 | 0 | 0 |
| 6 | Trim25 | 1 | 1 | 1 |
| 7 | Dgke | 1 | 1 | 1 |
| 8 | Pemt | 1 | 1 | 1 |
| 9 | Ckmt1 | 1 | 1 | 1 |
| 10 | Dbt | 1 | 0 | 0 |

B

| | A | B | C | D |
|---|---|---|---|---|
| 1 | rxns | 1C_to_2cell_embryo10 | 1C_to_2cell_embryo8_ | 1C_to_2cell_embryo2_ |
| 2 | 10FTHF5GLUtl | 0 | 0 | 0 |
| 3 | 10FTHF5GLUtm | 0 | 0 | 0 |
| 4 | 10FTHF6GLUtl | 0 | 0 | 0 |
| 5 | 10FTHF6GLUtm | 0 | 0 | 0 |
| 6 | 10FTHF7GLUtl | 0 | 0 | 0 |
| 7 | 10FTHF7GLUtm | 0 | 0 | 0 |
| 8 | 10FTHFtl | 0 | 0 | 0 |
| 9 | 10FTHFtm | 0 | -0.001 | 0.013 |
| 10 | 11DOCRTSLtm | 0 | 0 | 0 |

C

| | A | B | C | D |
|---|---|---|---|---|
| 1 | | 1C_to_2cell_embryo10_ | 1C_to_2cell_embryo8_ | 1C_to_2cell_embryo2_ |
| 2 | accoa | 0 | 0 | 0 |
| 3 | ala-L | 0 | 0.012014547 | 0 |
| 4 | amet | 6.66E-05 | 0 | 0.00126801 |
| 5 | amp | 0.001332136 | 0.000857038 | 0 |
| 6 | arg-L | 0.001949938 | 0.000886554 | 0 |
| 7 | asn-L | 0.006621424 | 0.002270203 | 0 |
| 8 | asp-L | 0 | 0 | 0 |
| 9 | atp | 0 | 0 | 0 |
| 10 | chsterol | 0.002408878 | 0.001616931 | 0.001761934 |

**Figure 1. Example of data required in each step**

We use the analysis of embryogenesis as an example to demonstrate what the data and results are like in SCOOTI.

(A) Significant genes or proteins, providing contextual information, are required to generate constrained models, while this is optional if the flux data have been prepared ahead. The rows are genes, and the columns are samples or cells. In this example, there are 153183 genes and 183 single cells.

(B) Flux predictions in constrained or unconstrained models constrain reactions in rows and cells in columns. Based on the Recon1 model, this example generated 3757 reaction fluxes for the 183 single cells; however, the number of rows depends on the use of genome-scale metabolic models.

(C) The data of metabolic objectives contains metabolites, also known as single objectives, in rows and cells in columns. Unconstrained models determine the number of rows. In the default setting, as shown in this example, SCOOTI generates 52 rows for the 183 cells.

a. Set the following file paths in your configuration or script:
   i. `unconModel` – path to unconstrained flux predictions
   ii. `conModel` – path to constrained flux distributions
   iii. `savePath` – output directory for saving regression results (e.g., objective coefficient CSVs)
   iv. `geneListPath` – optional path to a file listing unique or significant genes used in the regression
b. Specify key parameters for the regression process:
   i. `kappa` – regularization strength or constraint weight (controls model sensitivity)
   ii. `rho` – additional penalty or shrinkage parameter (used for stability or sparsity tuning)

```
# demo_inference_config.json

# Complete list of configuration options

{

"unconModel": "./examples/unconstrained_demo/out/unconstrained_models/",

"conModel": "./examples/constrained_demo/out/constrained_models/",

"savePath": "./examples/inference_demo/out/regression_models/",

"kappaArr": "0.1",

"rhoArr": "0.01",

"dkappaArr": "0.1",

"expName": "inference_demo",

"unconNorm": "T",

"conNorm": "F",

"medium": "KSOM",

"method": "cfr",

"model": "recon1",

"inputType": "flux",

"clusterPath": "",

"objListPath": "",

"rank": "F",

"stackModel": "F",

"sampling": "F",

"learner": "L",

"geneKO": "F",

"geneListPath": "",

"learningRate": 0.001,

"epo": 5000,

"fileSuffix": "_fluxes.csv.gz"

}
```

26. Run SCOOTI's meta-regression pipeline.
    a. Once flux predictions are complete and configuration is prepared, run the regression module in SCOOTI using the appropriate Python script (e.g., regressorTraining.py).
    b. Ensure the pipeline runs successfully and generates output files such as:
        i. A CSV file listing objective coefficients for each condition or single cell.
        ii. Diagnostic logs or visualizations, if enabled.

```
$scooti -I examples/inference_demo/demo_inference_config.json
```

27. Interpret the results.

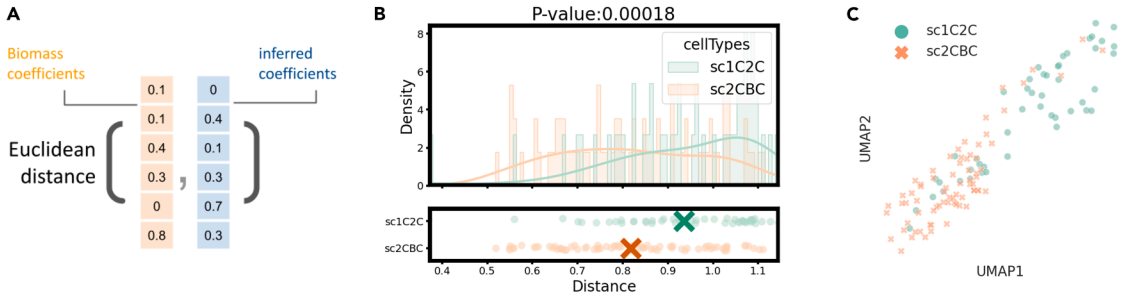



**Figure 2. Systematic view of the contrastive analysis of metabolic objectives for multiple conditions**

SCOOTI provides contrastive analyses of metabolic objectives at a systematic level that can be used in multiple conditions, cell types, or transitions.

(A) By setting the "distance" argument in "metaObjAnalyzer" to true, SCOOTI will compare inferred metabolic objectives with the default biomass objective using Euclidean distance.

(B) Statistical tests were applied for the distance from the biomass objective to the two groups of metabolic objectives, for example, two cell-type transitions during embryogenesis. A significant P-value shown on top of the figure indicates that the "sc2CBC" transition is relatively closer to the proliferative phenotype.

(C) UMAP of metabolic objectives provides another way to understand how the metabolic objectives are dissimilar in sample types.

a. For each sample or condition, SCOOTI outputs a vector of 52 objective coefficients (assuming the default candidate list), reflecting how much each objective contributes to the observed fluxes. In the `savePath`, there would be five coefficient CSV files including `flux_EN_*.csv`, `flux_lasso_*.csv`, `flux_LL_*.csv`, `flux_rfelm_*.csv`, `flux_sl_*.csv`. These coefficients are inferred by Elastic Net (EN), Lasso (lasso), Lasso–LARS (LL), linear regression with recursive feature selection (rfelm), and the super-learner (sl). The super-learner provides the final objective coefficients; the other models provide supplementary coefficients. The super-learner provides the final objective coefficients; the other models provide supplementary coefficients.

b. These coefficients can be used to profile metabolic priorities. For example, a sample with high coefficients for biomass precursor production and glutathione synthesis but low for others suggests a proliferative and antioxidant-focused state.

c. Coefficients are typically normalized or scaled so they can be compared across samples.

## Analyze metabolic objectives

Timing: 1–3 h

This section describes how to analyze and interpret inferred metabolic objective coefficients using SCOOTI's downstream analysis tools. Users will apply clustering, dimensionality reduction, distance-based metrics, and comparative analyses to explore metabolic heterogeneity, identify condition-specific objective patterns, and assess trade-offs between objectives. These analyses facilitate systematic comparison of metabolic programs across samples, conditions, or developmental stages.

28. Apply the metabolic objective analyzer.

a. A quick access to SCOOTI's analytic methods is to set up configurations like the `./examples/analyze_demo/analyze_config.json`:

```
{
"flux_paths": {"exp": "./examples/constrained_demo/out/constrained_models/"},
"coef_paths": {"exp": "./examples/inference_demo/out/regression_models/"}, "save_root_-
path": "./examples/analyze_demo/out/",
```

```
"engine": "legacy",

"reduction": "auto",

"GEM_path": "./scooti/metabolicModel/GEMs/Shen2019.mat",

"uncon_model_path":  "./examples/unconstrained_demo/out/unconstrained_models/",  "col_
map": {},

"samplingFlux_path": "",

"sel_para": "k0.1_r0.01",

"prefix": "analyze_demo",

"medium": "KSOM",

"labels": {"mode": "regex", "regex": "(1C_to_2cell|2C_to_32cell)",

"regex_group": 1,

"group_map": {"1C_to_2cell": "1C2C", "2C_to_32cell": "2CBC"}},

"get_coef": {"metType_cluster": false},

"coef_analysis": {"unknown_clustering": false, "clustering": true, "entropy": true, "dis-
tance": true, "compare": true, "umap_para": [5, 50], "method": "average", "ref_col": null}

}
```

b. In the configuration, `sel_para` and `medium` should match the parameters used in preparing unconstrained and constrained models. Although `labels` are optional, properly setting up the mapping rule in this field can normalize the sample names and lead to better views of analyses.
c. `coef_analysis` defines what types of analyses will be executed. The details of each method will be explained later in this section.
d. To launch the analysis based on the example configuration, we can apply the `scooti -I` command followed by the path of the configuration file.

```
$ scooti -A examples/analyze_demo/analyze_config.json
```

e. If users are interested in developing or customizing SCOOTI, use SCOOTI's analysis module (e.g., the Python class `metObjAnalyzer.py`) to examine the inferred metabolic objective coefficients across cells or conditions. The code block below previews the minimal inputs of this class function.

```
#!/usr/bin/env python

-- coding: utf-8 --

import SCOOTI

from SCOOTI.metObjAnalyzer import metObjAnalyzer

initiate an instance with a Python object

moa = metObjAnalyzer(

flux_paths = {'exp':"./scooti_test/constrained_models/",},

coef_paths={'exp':f'./examples/example_regressionModels/text_run.csv',},    save_root_
path='./result_files/',

GEM_path='./metabolicModel/GEMs/Shen2019.mat',
```

```
uncon_model_path="./scooti_test/unconstrained_models/",
col_map={},
label_func=label_func,
samplingFlux_path='',
sel_para='k1_r1',
prefix='example',
medium='DMEMF12',
)
```

f. Get flux predictions using the class function `metObjAnalyzer.get_flux()`.

```
#!/usr/bin/env python
-- coding: utf-8 --
get flux data
moa.get_flux(
kappa=[10, 1, 0.1],
rho=[10, 1, 0.1],
rank=False,
)
```

g. Analyze fluxes using different kappa and rho parameter combinations and select the best separation of samples.

```
#!/usr/bin/env python
-- coding: utf-8 --
# Analyze fluxes and look for the best parameter combinations
moa.fluxAnalysis(
kappa_arr=[10, 1, 0.1],
rho_arr=[10, 1, 0.1],
)
```

h. Get metabolite coefficients. Setting `metType_cluster` to true enables SCOOTI to get coefficients for each type of metabolite (e.g., amino acids, lipids, etc.).

```
#!/usr/bin/env python
-- coding: utf-8 --
# get coefficients of metabolic objectives
moa.get_coef(metType_cluster=False)
```

i. SCOOTI offers the analysis of metabolic objectives with hierarchical clustering (set `clustering` to `True`), dimension reduction methods (set `clustering` to `True`), deviation from the biomass objective (set `distance` to `True`), objective allocation (default analysis), objective entropy (set `entropy` to `True`), proportion of objective selection (set `compare` to `True`), and significant objectives (set `compare` to `True`).

```
#!/usr/bin/env python

-- coding: utf-8 --

# analyze the coefficients with distance to biomass and statistical comparisons between groups

moa.coefAnalysis(clustering=False, entropy=False, distance=True, compare=True, umap_
para=[5, 50], method='average', ref_col='Proliferative',)

# function to change labels

def label_func(df): return pd.Series(df.columns).apply(lambda x: x.split('_')[0]).repla-
ce({'1C':'sc1C2C', '2C':'sc2CBC'})

# initiate the object

moa = metObjAnalyzer(

flux_paths={'

sc1C2C':'./fluxPrediction/sc1C2C/paraScan/',

'sc2CBC':'./fluxPrediction/sc2CBC/paraScan/',

}, # optional if you only want to analyze objectives

coef_paths={

'sc1C2C':'flux_sc1C2C_input_norm_outcome_nonorm_k0.1_r0.01.csv',
'sc2CBC':'flux_sc2CBC_input_norm_outcome_nonorm_k0.1_r0.01.csv',

},

save_root_path='/SCOOTI/analysis/',

GEM_path='Shen2019.mat',

uncon_model_path='unconstrained_models/',

col_map={'sc1C2C':'sc1C2C', 'sc2CBC':'sc2CBC'},

label_func=label_func,

sel_para='k0.1_r0.01',

prefix='scEmbryo',

medium='KSOM',

)

moa.get_coef()

> Columns with zero coefficients: Index([], dtype='object')
```

29. Guidance of Interpreting the results.

*Note:* After running these analyses, collect the figures generated (heatmaps, scatter plots, UMAPs, etc.) for interpretation. They will form the basis of understanding the metabolic programs in your system.

a. **Objective distance to biomass:** To quantify how different the inferred objectives are from the traditional biomass objective, set the `distance` argument to `True`. SCOOTI will calculate the Euclidean distance between each sample's objective vector and the default biomass objective vector. A smaller distance implies the cell's metabolism is closer to growth-oriented, while a larger distance indicates alternative objectives are prioritized.

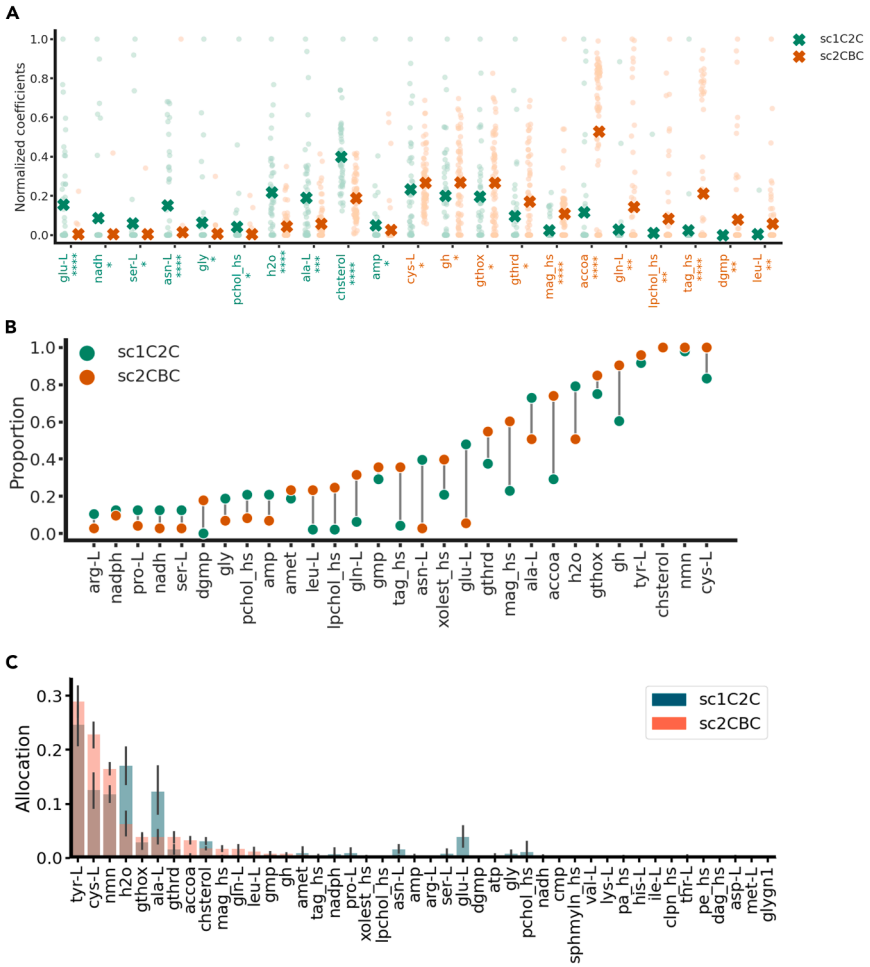



**Figure 3. Detailed view of contrastive analysis of metabolic objectives for multiple conditions**

By setting the compare argument in `metObjAnalyzer` to true, SCOOTI compared inferred metabolic objectives in two cell-type transitions during embryogenesis in this example and generated three different types of analyses.

(A) Each metabolite was statistically compared based on the coefficients. The strip plot showed coefficients rescaled from 0 to 1. Metabolites considered significantly higher in a sample type were marked by the color corresponding to the sample type on the X-axis.

(B) The proportion plot shows the count of non-zero coefficients of each metabolite normalized by the total number of samples. Instead of showing the magnitude of each coefficient, this plot emphasized the essentiality of each metabolite.

(C) The allocation plot focuses on the uses of different metabolites within each objective. In other words, each metabolic objective was normalized with the sum of coefficients to be a simplex.

b. **Objective allocation comparison:** If you have multiple conditions or a desire to compare objective usage, set `compare=True` in the analyzer. SCOOTI will then compare the metabolic objective coefficients between the two groups. For example, in our study, we compared the inferred objectives between two developmental transitions (1C-to-2C vs 2C-to-BC) to see how metabolic focus shifts. A statistically significant difference (e.g., via a permutation test or t-test, shown as a p-value) can highlight a metabolic shift. In the embryogenesis example, a significant p-value indicated that the 2C-to-BC transition was metabolically closer to a proliferative phenotype than the 1C-to-2C transition.

```
# analysis of metabolic objectives

moa.coefAnalysis(
```

```
clustering=True,
entropy=True,
distance=True,
umap_para=[5, 50]
)
Correlation size (121, 121)
# example output
UMAP1 UMAP2 Cell type cluster
3.680669 3.364651 sc1C2C 1
4.399888 6.746782 sc1C2C -1
3.204796 2.589123 sc1C2C 2
3.538360 0.614358 sc1C2C 2
3.384097 2.700004 sc1C2C 1
...
9.450109 0.952991 sc2CBC 0
8.954885 0.869949 sc2CBC 0
6.455922 1.344595 sc2CBC 4
9.599501 0.981174 sc2CBC 0
2.991922 0.447117 sc2CBC 2
[121 rows x 4 columns]
```

c. **Clustering by metabolite type:** Set `metType_cluster=True` to aggregate objective coefficients by metabolite categories. SCOOTI can group objectives related to specific metabolite classes (e.g., all amino acid-related objectives, lipid-related objectives, etc.) and compute combined coefficients for each class. This approach simplifies interpretation by seeing broad patterns, such as whether energy-related or redox-related objectives collectively change.

d. **Hierarchical clustering of objectives:** Enable `clustering=True` to produce a heatmap (clustermap) of the metabolic objective coefficients across samples. This unsupervised analysis will cluster samples with similar metabolic objective profiles. In practice, we found that using hierarchical clustering on objective coefficients cleanly separated cell states (e.g., G1 vs S vs G2 phase cells) even better than clustering on transcriptomic data alone. You can use this heatmap to identify groups of cells that share metabolic strategies.

e. **Identify trade-offs:** Look for pairs of objectives that show inverse relationships (trade-offs) in the data. This analysis is aided by the trade-off module in the next step (step 30), but even within `metObjAnalyzer` outputs, you might notice that certain objectives are rarely high simultaneously. For instance, in early embryos, we observed that reduced glutathione and biosynthetic precursors (biomass) had opposing trends. That is, cells strongly prioritizing glutathione tended to allocate less to biomass, and vice versa.

f. **UMAP or PCA visualization:** SCOOTI can project the metabolic objective profiles into lower-dimensional space. For example, using UMAP (Uniform Manifold Approximation and Projection) on the objective coefficients provides a 2D embedding of cells by metabolic phenotype. In our example, a UMAP of objectives showed clear segregation of cell cycle phases,

highlighting that metabolic objectives can distinguish cell states distinctly. Use the output to generate a UMAP or PCA plot (some of these may be automatically produced by `metObjAnalyzer`).

g. **Entropy of objectives:** If supported, compute the entropy of each cell's objective distribution (some analysis functions may do this when `entropy=True`). Entropy measures the dispersion of metabolic efforts. High entropy means a cell is balancing many objectives, whereas low entropy means the cell focuses on a few. In the embryogenesis study, cells in later stages had lower metabolic entropy than those in the one-cell stage, meaning they concentrated on fewer objectives as development proceeded.

h. **Comparative analysis:** If you set `compare=True` between two specific conditions or sets of cells, examine the output for each objective difference. SCOOTI can output a list of objectives with their differences and significance between the two groups. For example, we found that certain metabolites like cysteine, NMN (nicotinamide mononucleotide), cholesterol, and tyrosine were consistently selected as important objectives (>80% of cells) in both transitions, highlighting a core set of metabolic features, whereas other objectives differed between transitions.

*Note:* We suggested users choose the hyperparameters `kappa` and `rho` based on their research goals. Analysis of flux distributions in the constrained models could help users preliminarily understand the metabolic states.

*Note:* While setting `clustering` to `True` will generate hierarchical clustering and dimension reduction methods that default to UMAP, we highly recommend that users test parameters for `umap_para`, which determines the `n_neighbor` and `n_component` in UMAP. Another optional input `unknown_clustering` is another option to cluster samples with unknown labels.

### Advanced use: Identify metabolic trade-offs

Timing: 1–3 h

This section describes how to quantify metabolic trade-offs using SCOOTI's trade-off analysis module. Users will configure and run pairwise objective comparisons based on inferred objective coefficients to identify negatively associated objective pairs and visualize trade-off structure using correlation heatmaps, scatter plots, and optional Pareto-front analyses. These outputs support systematic evaluation of how cells or conditions balance competing metabolic priorities.

30. Perform quantitative trade-off analysis using `tradeoff_analysis.`

    a. Use the `tradeoff_analysis` function provided by SCOOTI to evaluate pairwise trade-offs between metabolic objectives. Trade-offs arise when two objectives cannot be simultaneously maximized, leading to a cellular compromise. Users can initiate the analyses referring to `./examples/tradeoff_demo/tradeoff_config.json`.

```
{
"coef_paths": {"exp": "./examples/inference_demo/out/regression_models/"}, "save_root_
path": "./examples/tradeoff_demo/out/",
"engine": "legacy",
"prefix": "tradeoff_demo",
"medium": "KSOM",
```

```
"coef_analysis": {"clustering": false, "entropy": true, "distance": false, "umap_para":
[10, 50], "compare": false},

"get_coef": {},

"tradeoff_analysis": {"pareto_mets": ["gh", "chsterol", "glutathione"],

"pareto3D": false,

"pareto2DAnalysis": false,

"traitAnalysis": false,

"normalize": true,

"corr_heatmap": true,

"corr_th": 0},

"labels": {"mode": "regex", "regex": "(1C_to_2cell|2C_to_32cell)",

"regex_group": 1,

"group_map": {"1C_to_2cell": "1C2C", "2C_to_32cell": "2CBC"}},

"sel_para": "k0.1_r0.01"

}
```

b. Set the `pareto_mets` argument to a list of target metabolites. This will generate scatter plots showing how each target metabolite trades off with others.
c. Enable the `traitAnalysis` flag to generate archetype-level summaries of metabolic trade-offs. These outputs may include correlation matrices, objective pair scatter plots, and Pareto front overlays.
d. To launch the trade-off analyses based on the example configuration, users can command:

```
$ scooti -T examples/tradeoff_demo/tradeoff_config.json
```

31. Interpret the output visualizations and identify trade-off patterns.
    a. Examine correlation matrices of objective coefficients. For example, Pearson correlation heatmaps can highlight negative (blue) or positive (red) associations. Strongly negative correlations may indicate functional trade-offs.
    b. Look for specific trade-off signatures. In early embryogenesis, a prominent negative correlation was observed between glutathione and biomass/cholesterol objectives—suggesting a trade-off between redox balance and growth-related metabolism.
    c. Review Pareto front visualizations. SCOOTI can plot each cell's normalized objective values relative to a theoretical Pareto front between two conflicting objectives. Points near the front suggest cells operating near optimal trade-off boundaries.
32. Analyze trait-specific clustering results (if applicable):
    a. Assess whether cells group based on their trade-off profiles. For instance, hierarchical density-based clustering of objective coefficients may reveal distinct metabolic states.
    b. In the embryogenesis dataset, two major clusters emerged: one enriched for glutathione (suggestive of pluripotency/redox emphasis), and another for biomass/growth objectives (indicative of proliferative states).
    c. Visualize the data using plots such as correlation heatmaps (e.g., Figure 4A), scatterplot matrices (Figures 4B and 4C), and Pareto front diagrams (Figure 4D) to synthesize your interpretation of system-level trade-offs.

```
# tradeoff analysis

moa.tradeoff_analysis(

pareto_mets=['gh', 'chsterol', 'glutathione'],

traitAnalysis=True,

)
```

## Advanced use: Integrate multi-omics data simultaneously

Timing: 1–3 h

This section describes how to integrate multiple omics layers (e.g., transcriptomics, proteomics, and metabolomics) to jointly constrain genome-scale metabolic models in SCOOTI. Users will apply constraints sequentially or in paired modes, configure weighting parameters to control the relative

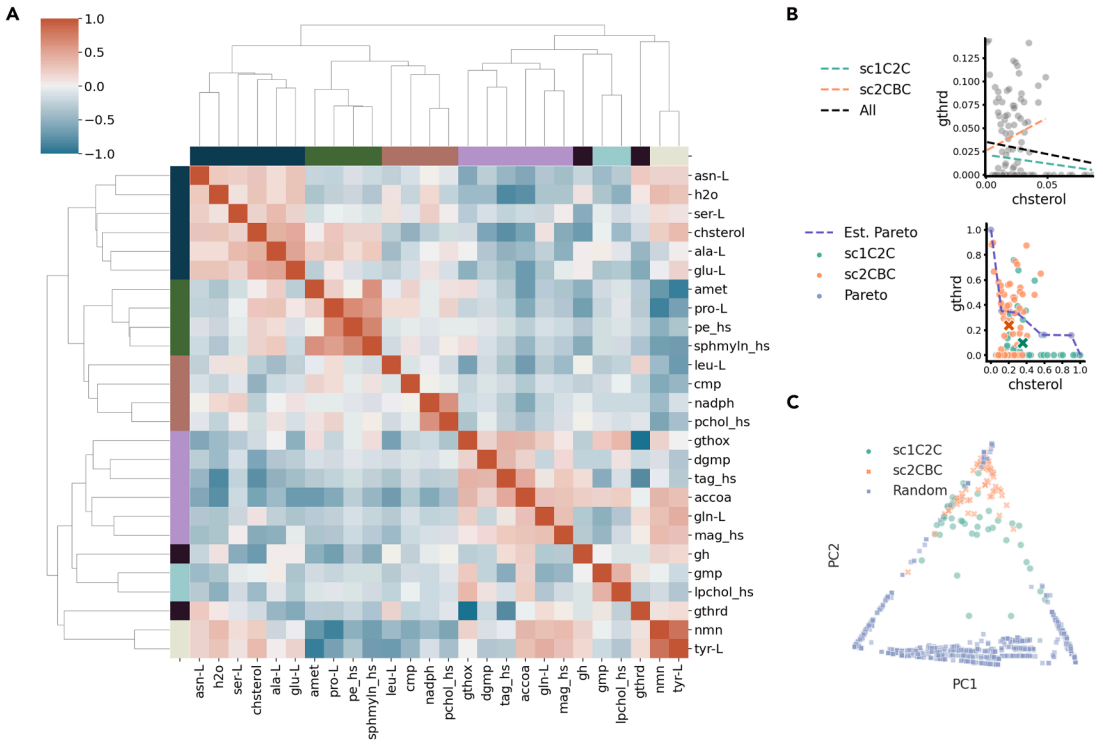



**Figure 4. Trade-off analysis of metabolic objectives**

(A) The trade-offs in each pair of metabolites can be found by the correlation of the metabolic objective coefficients. We mixed the metabolic objectives from the two cell-type transitions during embryogenesis to quantify the trade-offs emerging from embryogenesis. In this plot, negative correlations indicating the degree of trade-offs were shown in blue, while metabolites were clustered corresponding to the color bars as if they shared positive correlations.

(B) The relationships between each pair of metabolites are shown with scatter plots overlaid with curve-fitting trends (Top). This is useful for the comparison of tradeoffs in different conditions. Pareto front, on the other hand, provides a reference curve of mechanistically optimal conditions (Bottom). A greater deviation from the metabolic coefficients to the curve indicates a less optimized setup for the pair of metabolites. This reference curve was generated by the objective flux values of random samples of metabolic objective coefficients.

(C) We used PCA plots to visualize the simplex of metabolic objectives overlaying the mechanistic optimal fluxes, the same as the reference curve in (B). The vertices of the polygon are archetypes. The distance from each objective to the archetypes indicates how optimal an objective is and how the configuration of an objective is similar to the archetypes.

influence of each omics layer, and generate multi-omics–constrained models for downstream flux prediction and metabolic objective inference using the standard SCOOTI pipeline.

33. Set up multi-omics integration with weighting schemes.
    a. If both transcriptomic and proteomic data are available for the same samples, SCOOTI allows their integration to jointly constrain the metabolic model.
    b. Assign weights to reflect the expected regulatory strength of each data type. For example, set `extraWeight = 1000` for proteomics and `extraWeight = 1` for transcriptomics to emphasize the stronger control by protein levels.
    c. Provide both gene expression and protein abundance data as inputs, identifying significantly up- and down-regulated proteins between conditions. This allows the model to prioritize protein-level constraints when transcriptomic and proteomic data disagree.
    d. Run the model generation using these combined constraints; the rest of the SCOOTI pipeline remains unchanged.
34. Configure model paths and choose integration mode.
    a. Set the appropriate file paths, including `data_dir` for gene expression data, `save_root_path` for saving intermediate models, and the paths to COBRA Toolbox and the base metabolic model.
    b. Use the `pairwise_CFR_model` setting to define the integration strategy.
    c. Set `pairwise_CFR_model = 1` to integrate omics from the same sample (e.g., from the same petri dish and drug treatment).
    d. Set `pairwise_CFR_model = 0` to stack constraints by generating Cartesian products across omics layers.
    e. Set `extraWeight = 1` (or other appropriate values) to scale the relative influence of transcriptomics when stacking.

*Note:* We suggest that users test different `extraWeight` to see the differences in the constrained models. In the primary publication for SCOOTI, we hypothesize that the constraint effects of proteomics are higher than transcriptomics. Therefore, we applied 1,000 and 1 to `extraWeight` when we constrained the models with proteomics and transcriptomics, respectively. In addition, due to the different constraint-based methods used for metabolomics, users can ignore the size of `extraWeight` when integrating metabolomics data.

35. Apply transcriptomics constraints as the first layer.
    a. Begin by constraining the model with transcriptomics data using `simulation = CFR`.
    b. Set `data_dir` to access the transcriptomic gene expression data, and define `save_root_path` to store the intermediate transcriptome-constrained models.

```
# example configuration

{

"COBRA_path": "./cobratoolbox",

"GEM_path": "./metabolicModel/GEMs/Shen2019.mat",

"model_name": "Recon1",

"obj_candidate_list_file": "./metabolicModel/GEMs/obj52_metabolites_shen2019.csv", "data_dir":   "./examples/run_flux/example_sigGenes_from_transcriptome/",   "prefix_name": "example",

"medium": "DMEMF12",

"simulation": "CFR",

"save_root_path":   "./scooti_test/multiomic_constrained_models_from_transcriptome/", "CFR_model_path": "",
```

```
"pairwise_CFR_model": 1,

"extraWeight": 1,

"algorithm": "CFR",

}
```

36. Add proteomic constraints as a second layer.
    a. After generating transcriptome-constrained models, apply proteomic constraints on top of them.
    b. Save these as new models representing two-layer constrained metabolic networks

```
# example configuration

{

"COBRA_path": "./cobratoolbox",

"GEM_path": "./metabolicModel/GEMs/Shen2019.mat",

"model_name": "Recon1",

"obj_candidate_list_file": "./metabolicModel/GEMs/obj52_metabolites_shen2019.csv", "da-
ta_dir": "./examples/run_flux/example_sigGenes_from_proteome/",

"prefix_name": "example",

"medium": "DMEMF12",

"simulation": "CFR",

"save_root_path": "./scooti_test/multiomic_constrained_models_from_proteome/",

"CFR_model_path": "./scooti_test/multiomic_constrained_models_from_transcriptome/",

"pairwise_CFR_model": 1,

"extraWeight": 1E3,

"algorithm": "cfr",

}
```

37. Incorporate metabolomics constraints for full multi-omics modeling.
    a. To integrate metabolomics, load the previously constrained models and set simulation = DFA and algorithm = DFA.
    b. Keep other parameter settings the same. This final step adds the third omics layer and completes the multi-omics integration process.

```
# example configuration

{

"COBRA_path": "./cobratoolbox",

"GEM_path": "./metabolicModel/GEMs/Shen2019.mat",

"model_name": "Recon1",

"obj_candidate_list_file":
```

```
"./metabolicModel/GEMs/obj52_metabolites_shen2019.csv", "data_dir": "./examples/run_-
flux/example_sigGenes_from_proteome/",

"prefix_name": "example",

"medium": "DMEMF12",

"simulation": "DFA",

"save_root_path": "./scooti_test/multiomic_constrained_models_from_metabolome/",

"CFR_model_path": "./scooti_test/multiomic_constrained_models_from_proteome/", "pairwise_
CFR_model": 1,

"extraWeight": 1E6,

"kappa": -1,

"rho": -1,

"DFA_kappa": 1

}
```

38. Tune and interpret multi-omics model behavior.
    a. Integrated models offer a more accurate and biologically consistent representation of metabolic states.
    b. Remember that weighting parameters (e.g., 1000:1 proteomics-to-transcriptomics) may need adjustment depending on system context and data quality.

### Advanced use: Metabolic objective inference with customization

Timing: 1–3 h

This section describes how to customize key components of the SCOOTI workflow, including selection of genome-scale metabolic models, choice of omics integration algorithms, and specification of candidate objective metabolite lists. Users may also select alternative regression learners for objective inference and update configuration files accordingly. These options enable adaptation of SCOOTI to different organisms, model reconstructions, and research questions while preserving the core workflow.

39. Select and configure genome-scale metabolic models (GEMs).
    a. SCOOTI currently supports three GEMs: Recon1, Recon2.2, and Recon3D.[2,4,5]
    b. To use a custom GEM in constrained modeling, set the following parameters: `GEM_path = ./GEMs/Shen2019.mat` for Recon1 (Shen2019 includes acetylation-related reactions), `GEM_path = ./GEMs/Recon2.2.mat` for Recon2.2, or `GEM_path = ./GEMs/Recon3D.mat` for Recon3D.
    c. Set the corresponding `model_name` to Recon1, Recon2.2, or Recon3D (Figures S1C and S1D).
    d. Other pipeline settings do not require changes when switching GEMs.
    e. When inferring metabolic objectives with unconstrained models, set the `--model` flag (e.g., `--model Recon1`) to label outputs. However, users must regenerate unconstrained models using the selected GEM.
40. Choose omics integration algorithms.
    a. SCOOTI supports multiple algorithms for integrating omics data into metabolic models, including CFR and MOOMIN.[12]
    b. Use the algorithm parameter to switch between methods, e.g., `--algorithm CFR`.
    c. COMPASS is another supported integration method validated in our primary study, though users should refer to the original COMPASS publication for full usage details and guidance[11] (Figure S1A).

41. Customize the candidate metabolic objectives.
    a. By default, SCOOTI uses a curated list of 52 metabolites as objective candidates to balance biological interpretability and computational cost.
    b. To use a different list, set the `obj_candidate_list_file` parameter to a CSV file containing your chosen metabolites. For example, `obj_candidate_list_file=./obj52_metabolites_Recon3D.csv`.
    c. You may also prepare a fully custom list based on your specific research questions or dataset, compatible with any of the supported GEMs (Recon1, Recon2.2, or Recon3D).
    d. Be aware that using a larger or unrestricted list may substantially increase the computational load.

```
# example flux model configuration

{

"COBRA_path": "./cobratoolbox",

"GEM_path": "./metabolicModel/GEMs/Recon3D.mat",

"model_name": "Recon3D",

"obj_candidate_list_file":    "./metabolicModel/GEMs/allMets_metabolites_recon3d.csv",
"data_dir": "./examples/run_flux/example_sigGenes/",

"prefix_name": "example",

"medium": "DMEMF12",

"simulation": "DFA",

"save_root_path": "./scooti_test/constrained_models/",

}
```

42. Select the inference model for metabolic objective regression.
    a. SCOOTI uses a linear regression model by default for inferring metabolic objective coefficients.
    b. To switch to a single-layer neural network (ADALINE), change the `--learner` flag from `--learner L` (linear) to `--learner A` (ADALINE) (Figure S1B).
    c. This allows users to explore non-linear relationships in objective inference while maintaining model simplicity.

```
# example inference configuration

{

"unconModel": "./examples/unconstrained_models/",

"conModel": "./examples/example_fluxPrediction/",

"savePath": "./examples/example_regressionModels/",

"expName": "test_run",

"unconNorm": "T",

"conNorm": "F",

"medium": "DMEMF12",

"method": "cfr",
```

```
"model": "recon1",
"learner": "A",
"geneListPath": "",
"learningRate": 0.001, # essential
"epo": 5000 # essential
}
```

### Advanced use: Consider the uncertainty of metabolic objectives

Timing: 1–3 h

This section describes how to account for uncertainty in unconstrained flux predictions by generating multiple sub-optimal solutions for each objective and incorporating these solutions into objective inference. Users will run sampling to obtain replicate unconstrained flux sets and enable sampling-aware inference in SCOOTI. This workflow supports assessment of robustness of inferred objective coefficients to alternative feasible flux distributions.

43. Generate sub-optimal solutions using `OptGPSampler`.
    a. Although unconstrained models typically optimize one metabolic objective at a time, they may yield multiple sub-optimal solutions. Considering these solutions improves robustness in objective inference.
    b. SCOOTI addresses this uncertainty by leveraging the `OptGPSampler` to generate multiple sub-optimal flux predictions for each objective.
    c. To run sampling, use the following command to generate three sets of unconstrained models and save them to your desired directory:

```
$ python SCOOTI_sampler.py ./PATH_TO_SAVE_YOUR_MODELS/
```

    d. The command will create a directory structure as follows:

```
fluxPrediction/
| +-- scooti/
|    +-- Sample_1        (The first batch of samples)
|    +-- Sample_2        (The second batch of samples)
|    +-- Sample_3        (The third batch of samples)
```

    e. Perform objective inference with sampled unconstrained models.
    f. When running SCOOTI with sampled models, specify the flag `--sampling U` to indicate that inference should use the sub-optimal flux distributions.
    g. The pipeline will infer objective coefficients across all sampled solutions, improving generalization and uncertainty modeling.

### Advanced use: Identify gene essentiality from metabolic objectives

Timing: 1–3 h

This section describes how to evaluate gene essentiality using SCOOTI by comparing inferred metabolic objective coefficients between wild-type and gene-perturbed models. Users will generate constrained models with gene knockout simulations enabled, run objective inference across

perturbations, and analyze changes in objective-space metrics relative to wild-type. This analysis provides a structured way to prioritize genes whose perturbation strongly alters inferred metabolic priorities.

44. Prepare models for wild-type and gene knockout analysis.
    a. Follow the steps in the Generate constrained models section to create models for both wild-type and single-gene knockout strains.
    b. Enable gene knockout simulation by setting `genekoflag = 1` in your configuration.

```
# constrained_demo_config.json

# Complete list of configuration options

{

"COBRA_path": "./cobratoolbox",

"GEM_path": "./metabolicModel/GEMs/Shen2019.mat",

"model_name": "Recon1",

"obj_candidate_list_file": "./metabolicModel/GEMs/obj52_metabolites_shen2019.csv", "in-
it_objective": 1,

"genekoflag": 1,

"pfba": 1,

"data_dir": "./examples/run_flux/example_sigGenes/",

"prefix_name": "example",

"medium": "DMEMF12",

"uplabel": "upgenes",

"dwlabel": "dwgenes",

"simulation": "CFR",

"constraint": 1,

"save_root_path": "./scooti_test/KO_constrained_models/",

"algorithm": "cfr"

}
```

    c. Each constraint will produce a `.mat` file containing flux predictions for the wild-type model and single-gene knockouts. The data is structured as a matrix where rows represent reactions and columns correspond to the wild-type (WT) and each gene perturbation (e.g., Gene A, Gene B, etc.), as illustrated in Table 1.

45. Infer objectives under gene perturbations.
    a. Refer to the Infer metabolic objectives section and set `--geneKO T` to instruct SCOOTI to run inference for each knockout strain.

**Table 1. An example table with clear organization of flux predictions across wild-type and gene knockouts**

| Reactions | WT | Gene A | Gene B | Gene C |
|---|---|---|---|---|
| Rxn A | 0.20 | 0.00 | 0.01 | 0.18 |
| Rxn B | 1.50 | 1.20 | 1.40 | 1.55 |
| Rxn C | 0.00 | 0.00 | 0.02 | 0.01 |

b. This step computes objective coefficients under perturbation for comparison with the wild-type.

```
# demo_inference_config.json

# Complete list of configuration options

{

"unconModel": "./examples/unconstrained_models/",

"conModel": "./scooti_test/KO_constrained_models/",

"savePath": "./examples/example_regressionModels/",

"kappaArr": "10,1,0.1",

"rhoArr": "10,1,0.1",

"dkappaArr": "10,1,0.1",

"expName": "test_run",

"unconNorm": "T",

"conNorm": "F",

"medium": "DMEMF12",

"method": "cfr",

"model": "recon1",

"inputType": "flux",

"geneKO": "T",

}
```

46. Analyze distances and identify essential genes.
    a. Use the "Analyze metabolic objectives" section to calculate distances between inferred objectives from wild-type and knockout models.
    b. Apply a Z-score threshold (e.g., `Z = 1`) to determine significant changes in objectives, which may indicate gene essentiality.
    c. Genes with large Z-scored distances in objective space are considered essential for maintaining key metabolic functions.

### Alternative use: Preparing unconstrained and constrained models with CFRpy

Timing: 1 h

This section describes an alternative workflow for generating unconstrained and omics-constrained flux predictions in Python using CFRpy, while retaining SCOOTI for downstream objective inference and analysis. Users will install required Python dependencies, load a genome-scale metabolic model, apply CFR-based constraints from gene expression data, and export flux predictions in a format compatible with SCOOTI. This option supports users who prefer a Python-only environment for flux prediction.

47. Overview and prerequisites.
    a. While SCOOTI currently uses MATLAB to prepare unconstrained and constrained models, the rest of the pipeline is implemented in Python.

b. Users may prefer to use the Python-based CFRpy package to prepare models and flux predictions directly within Python.
c. CFRpy supports generation of both the JSON metadata file and the CSV file of predicted fluxes, making it compatible with the SCOOTI pipeline.
d. Ensure that a gene expression table is available. This table should contain log2 fold-change (log2fc) values for each gene (rows) across different conditions (columns), compared to a control.
e. SCOOTI identifies significant genes based on log2fc thresholds—refer to the section Identify significant genes or proteins for more details.

48. Install required Python packages.
    a. Install COBRAPy and GurobiPy with the following commands:

```
$ pip install cobra

$ pip install gurobipy
```

49. Run the CFRpy workflow to generate flux predictions.
    a. Load your COBRA model in Python using COBRApy by specifying the file path to an SBML (`.xml`) model.

*Note:* We provide an example model at `./scooti/metabolicModels/GEMs/Shen2019.xml`, `Recon2.2.xml`, and `Recon3D.xml`. Although there are MATLAB-format models, the model has been edited to keep them lightweight and not compatible with cobrapy.

    b. Run the `apply_cfr()` function in CFRpy, which requires a COBRA model object, a gene expression dataframe, an optional log2fc thresholds to define up- and down-regulated genes (default: −2 to 2).
    c. `apply_cfr()` returns a list of constrained COBRA model result objects for each condition.
    d. Use `get_fluxes()` to extract a dataframe of predicted fluxes from the result objects.
    e. Save the predicted fluxes to a CSV file for downstream metabolic objective inference.

```
#!/usr/bin/python3

import pandas as pd

import cobra

import CFRpy as cfr

from CFRpy.cfr_suite import *

# load model and df

cobra_model = cobra.io.read_smbl_model(FILE_PATH_TO_MODEL)

omics_df = pd.read_excel(FILE_PATH_TO_DATA, index_col=0)

# run CFRpy functions

results = apply_cfr(cobra_model, omics_df, (-2, 2))

result_df = get_fluxes(results)

result_df.to_csv('result_flux.csv', index=False)
```

*Note:* `apply_cfr()` will also output a CSV of binarized flux values (0 for down-regulated, 1 for up-regulated), a summary file reporting the number of regulated genes and calculated objective coefficients per condition.

## EXPECTED OUTCOMES

Using SCOOTI for contrastive analysis of metabolic objectives can suggest differences between conditions in terms of which metabolic goals are prioritized. For example, when applied to early embryonic development, SCOOTI found significant changes in metabolic strategy between developmental stages. For instance, one-cell embryos devoted a higher proportion of their metabolic flux toward glutathione synthesis (antioxidant function), whereas later-stage blastocyst cells shifted towards biomass production (growth-related). Cysteine, NMN, cholesterol, and tyrosine were frequently selected as important objectives across cells, indicating robust roles in early embryo metabolism.

Trade-off analysis can highlight pairs of objectives that cannot be maximized together. In our example, a prominent trade-off between glutathione and biosynthetic objectives (e.g., cholesterol/biomass) was identified in the metabolic objective space of embryonic cells. Early-stage embryonic cells optimize a trade-off that favors redox homeostasis, while later-stage cells favor proliferative growth. SCOOTI's outputs (Figure 4 in our study) showed that cells form two clusters corresponding to two metabolic strategies: one cluster with high glutathione objective (associated with pluripotency and stress resistance) and another with high growth objectives (associated with differentiation and proliferation). This analysis suggests a Pareto frontier where cells balance between "growth" and "maintenance" metabolic goals.

Another metric to analyze objectives is the metabolic entropy or diversity of objectives. In developmental progression or cell-state transitions that impose more order, SCOOTI shows a decrease in entropy (fewer objectives dominate). The entropy of metabolic objectives decreased from the one-cell to the blastocyst stage, meaning that later-stage cells focused on fewer metabolic tasks. This was visualized by plotting entropy values for each cell, which were significantly lower in the later transition (2C→BC) than in the earlier (1C→2C). Biologically, this aligns with the concept that as cells differentiate, they allocate resources to specific functions. When analyzing multiple distinct conditions (e.g., diseased vs healthy), one can expect heatmaps or UMAP plots of objective coefficients to segregate those conditions, highlighting metabolic differences. For instance, SCOOTI output for cell cycle phases resulted in a UMAP where G1, S, G2 phase cells clustered separately based on their metabolic objective profiles. Similarly, one might see that diseased cells cluster away from healthy ones on a metabolic objective UMAP, implying different metabolic priorities. For cases where a user has only one condition, SCOOTI's gene essentiality analysis may be beneficial, as it provides a list of genes with a significant impact on the metabolic objectives. By testing this idea using data from embryonic stem cells, SCOOTI correctly uncovered genes essential for maintaining the cells' metabolic objectives (such as those in the electron transport chain), which matched experimental CRISPR screen hits. In contrast, a biomass-maximization approach identified many false positives. As an example, SCOOTI indicated that inhibiting oxidative phosphorylation enzymes caused a significant shift in the objective profile (loss of pluripotency-related objectives), thereby labeling those genes as essential, in line with experimental data. After running SCOOTI, a user can compare inferred objectives to known biology. For example, one might expect that a cell type known to be non-proliferative (like a quiescent immune cell) will show a metabolic objective profile focusing on maintenance (high redox, nutrient storage objectives) rather than growth. SCOOTI's results should reflect that, as it did in our analyses of quiescent vs proliferating cells. If expected outcomes are not observed, follow the tips in the troubleshooting section.

## LIMITATIONS

The accuracy of SCOOTI depends on the quality of the underlying genome-scale metabolic model and the omics-derived constraints used for analysis. Missing pathways, incomplete reconstructions, or inaccurate reaction bounds can affect the inferred metabolic objectives and trade-offs. Consequently, SCOOTI cannot identify objectives associated with pathways that are absent or poorly

represented in the metabolic model, and users should employ the most up-to-date and system-appropriate reconstructions whenever possible.

SCOOTI infers metabolic objectives by comparing constrained and unconstrained ideal flux distributions, which requires biologically meaningful perturbations. When constraints are weak or metabolic differences between conditions or cells are minimal, inferred objectives may be noisy or uninformative. In such cases, grouping cells or focusing on conditions with clearer biological distinctions may improve interpretability, particularly for single-cell datasets.

Each cell or condition is modeled independently, which can lead to increased computational cost as dataset size grows. Unlike methods that aggregate or pool cells to reduce complexity, SCOOTI does not internally share computations across cells. As a result, runtime and memory usage scale with the number of analyzed cells and objectives. For large single-cell datasets, users may need to apply preprocessing steps such as clustering cells into metacells prior to running SCOOTI, as these strategies are not currently implemented within the framework.

SCOOTI also relies on a hybrid software environment in which flux balance analyses are performed in MATLAB using the COBRA Toolbox and a linear programming solver, while downstream analyses are conducted in Python. This split environment may pose accessibility challenges for some users and can introduce integration issues, particularly related to solver configuration. In addition, SCOOTI infers objectives from a predefined list of metabolites, which limits interpretation to the supplied objective set. If a cell's true metabolic goal is not included, it may be approximated by combinations of available objectives, and correlated objectives should be interpreted cautiously.

Finally, SCOOTI does not explicitly model continuous temporal dynamics. Objectives are inferred for discrete states or transitions rather than along a continuous trajectory, which may limit its applicability to systems with rapid or highly dynamic metabolic changes. Trade-offs identified by SCOOTI are also context-dependent and influenced by the chosen objective set, meaning that they may not represent universal metabolic principles. Despite these limitations, SCOOTI provides a systematic framework for exploring alternative metabolic objectives, particularly in non-dividing or specialized cell states where canonical growth-based assumptions may not apply.

## TROUBLESHOOTING

### Problem 1

Failure to generate constrained models. After providing the up/down gene lists, the pipeline produces no output for constrained flux models or throws an error during model generation.

### Potential solution

Ensure that the lists of significant genes/proteins are correctly formatted and not empty (step 12: Identifying significant genes or proteins). If the constrained model has no active constraints (for instance, if no genes were marked significant), the flux optimization might essentially replicate the unconstrained case or fail to solve. In case the output flux distributions for constrained models are empty or not generated, double-check that the file paths to the gene lists are correct and that the gene identifiers match those in the metabolic model (step 23: Construct omics-constrained models). It may help to use a more permissive threshold to get at least a few constraints, or to combine data (e.g., use bulk transcriptomics to derive constraints if single-cell signals are too weak). Verify that COBRA Toolbox is functioning and that the model can be loaded with constraints (step 23: Construct omics-constrained models) – if COBRA throws an error, it might be due to improper model setup.

### Problem 2

Unable to separate cell types or conditions with metabolic objective profiles. The expected clustering or separation by inferred objectives is not observed (e.g., a heatmap shows no clear grouping, or UMAP shows mixed cell types).

### Potential solution

SCOOTI provides multiple analysis views; try all three basic analyses (heatmap clustering, PCA/UMAP, and the regression diagnostics; step 27: Analyze metabolic objectives). If one visualization doesn't separate the groups, another might. Ensure that your input data indeed has differences – if the cells truly do not differ metabolically, SCOOTI cannot invent differences. It's also possible that technical noise is obscuring the patterns. In that case, consider grouping similar cells (as mentioned, use a tool like Seurat or SEACells to cluster cells and then run SCOOTI on cluster averages; step 12: Identifying significant genes or proteins).[20,21] Another approach is to increase the contrast by comparing more extreme conditions or supplementing additional omics (e.g., use proteomics if transcript differences are subtle; step 33–38: Advanced use: integrate multi-omics data simultaneously). If the objectives coefficients all appear very low or similar, it could be a symptom of too many objectives diluting the differences, reducing the objectives set to the most relevant ones might sharpen the contrast (step 41: Advanced use: metabolic objective inference with customization).

### Problem 3

Metabolic objectives have extremely low coefficients (near zero) for all objectives. The regression outputs objective weights that are all very small, suggesting the model didn't find a strong combination to explain the fluxes.

### Potential solution

This can occur if the constrained flux distribution is very similar to an unconstrained solution or if it lies near the origin in the objective space (steps 17–22: Prepare unconstrained models optimizing ideal objectives; step 23: Construct omics-constrained models). Recall that the objective coefficients are typically normalized (for example, they may sum to 1 or to the total flux). If all coefficients are near zero, possibly the regression was not performed correctly. Make sure the unconstrained flux distributions were provided as predictors (step 16: Prepare unconstrained models optimizing ideal objectives). If they were and the coefficients are still near zero, it means none of the 52 objectives explain the constrained flux well, indicating either a limitation of the objective set or an issue with the data (step 24: Inference of metabolic objectives). You might try allowing an intercept in the regression or including a generic "biomass" objective to capture any remaining flux (step 41: Advanced use: metabolic objective inference with customization). In our implementation, we usually see at least a few objectives with substantial coefficients. Another check: ensure that the flux distributions were scaled properly. If the flux values are extremely large or small, normalization issues could make coefficients tiny. Scale the flux data (e.g., unit normalize each objective's flux vector) and rerun the regression (step 24: Inference of metabolic objectives).

### Problem 4

The `findSigGenes.py` script (or similar) fails to identify significant genes. You run into errors or no output when extracting significant genes from single-cell data using the provided script.

### Potential solution

This often happens if SCOOTI's repository was cloned but not installed (step 1: Installation of SCOOTI and virtual environment). Some scripts may expect SCOOTI to be in the Python path. Installing SCOOTI as a package (as in step 1) should resolve this. If you already did, ensure you're running the script from the correct directory or providing the correct input format (the script might expect certain columns or file naming; step 12: Identifying significant genes or proteins). If the issue persists, you can manually perform differential expression with external tools and just provide the CSV lists to SCOOTI (step 12: Identifying significant genes or proteins). Also, check that you have the required Python libraries for that script (it may use `scanpy` or `anndata` for single-cell data). Installing any missing dependencies or running the script in the environment where those are installed will help (step 1: Installation of SCOOTI and virtual environment).

### Problem 5

Errors starting MATLAB for model preparation. When SCOOTI attempts to call MATLAB (for instance, to run COBRA Toolbox for generating models), it fails to launch or connect.

### Potential solution

Instead of relying on an interactive MATLAB session, use MATLAB in non-interactive (batch) mode (step 23: Construct omics-constrained models). For example, SCOOTI provides a MATLAB script to set up models; run that script manually in MATLAB to ensure it works. You can call MATLAB from the command line with a command to execute the script (e.g., using `-batch` or `-r` flag). Also, ensure that MATLAB's path is configured to include COBRA Toolbox and that you have the necessary solver (Gurobi or CPLEX) installed and licensed (step 23: Construct omics-constrained models). If using a MATLAB engine for Python, make sure the MATLAB engine API for Python is installed (`pip install matlabengine` after running MATLAB's `install_for_python` script; step 1: Installation of SCOOTI and virtual environment). Often, simply opening MATLAB separately and adding the COBRA paths, then saving the path, can solve recognition issues.

### Problem 6

MATLAB cannot find certain files or functions during model preparation. For example, it complains that a COBRA function or a model file is missing.

### Potential solution

This usually indicates a path issue. Make sure that when MATLAB is invoked, it knows where the SCOOTI models folder and the COBRA Toolbox are (step 23: Construct omics-constrained models). One way is to add `addpath('path/to/SCOOTI/models')` and similar commands at the top of the MATLAB script that SCOOTI runs. Another strategy is to run the model preparation steps inside MATLAB manually to debug (step 23: Construct omics-constrained models). If file recognition issues persist, verify the file names and cases (MATLAB is case-sensitive on Linux). As a workaround, you can use the COBRA Toolbox functions directly: for instance, use `changeRxnBounds` to apply gene constraints and `optimizeCbModel` to simulate flux – doing this step-by-step in MATLAB might isolate the problem (step 23: Construct omics-constrained models). The provided error messages in MATLAB will guide which file is not found; adjust the working directory or add paths accordingly. In summary, always launch MATLAB in the directory where the SCOOTI code resides, or modify the script to navigate to that directory before execution.

Additionally, if a specific COBRA function is not recognized, ensure you're using a compatible COBRA Toolbox version. We used COBRA Toolbox v3.0; newer versions may require different function calls. Consult the COBRA documentation for any deprecated functions and update the SCOOTI MATLAB scripts if needed.

### Problem 7

The flux distribution optimization or objective inference does not finish (hangs or crashes). SCOOTI gets stuck while solving or runs out of memory.

### Potential solution

This can happen with very large models or too many cells processed at once (steps 17–22: Prepare unconstrained models optimizing ideal objectives; step 23: Construct omics-constrained models). First, monitor memory usage: Gurobi can consume a lot of memory for sampling or repeated optimizations. If memory is an issue, try running fewer parallel jobs or splitting the work (process cells one by one rather than in a batch). If the solver is hanging on a particular optimization, consider using a different solver (if using CPLEX, try Gurobi or vice versa) or loosening solver tolerances to make the problem easier (step 23: Construct omics-constrained models). In COBRA, you can set `changeCobraSolverParams` for time limits or feasibility tolerances. Setting a time limit for each FBA problem (e.g., 60 seconds) may allow the pipeline to skip overly difficult optimizations. Also, check for

model issues: sometimes, certain constraints can make the model infeasible or unbounded, causing the solver to loop. If you suspect a certain condition's gene set is causing problems, test that case individually (step 12: Identifying significant genes or proteins). It might be necessary to remove a problematic constraint or metabolite if it consistently fails. Lastly, ensure you are not using an extremely high number of samples in flux sampling (if enabled). Reducing the sampling count will help performance (step 41: Advanced use: metabolic objective inference with customization). In summary, reduce the scale: fewer cells at a time, fewer objectives (if possible), or simplified models can be used to troubleshoot where it's failing, then gradually build back complexity.

### Problem 8

Package dependencies are missing or did not install during SCOOTI initialization.

### Potential solution

Continue to attempt to initialize SCOOTI and install each package that yields an error message using pip install or conda install if working in a conda environment (step 1: Installation of SCOOTI and virtual environment). By addressing the above problems, most issues encountered during the use of SCOOTI can be resolved. Each step of the pipeline (data preparation, model building, regression, analysis) has its own potential pitfalls, but careful checking of inputs and environment will usually identify the source of error. Once the pipeline runs smoothly, you can confidently interpret the results to gain novel insights into the metabolic objectives of your cells.

## RESOURCE AVAILABILITY

### Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Sriram Chandrasekaran (csriram@umich.edu).

### Technical contact

Technical questions on executing this protocol should be directed to and will be answered by the technical contact, Da-Wei Lin (daweilin@umich.edu).

### Materials availability

No materials have been generated in this study.

### Data and code availability

This paper analyzes existing, publicly available data. These identifiers for the datasets are listed in the key resources table.

- All original code for modeling metabolic fluxes and inferring metabolic objectives has been deposited at Zenodo and is publicly available as of the date of publication. DOIs are listed in the key resources table.
- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

## ACKNOWLEDGMENTS

We gratefully acknowledge the National Institutes of Health grant R35 GM13779501 (to S.C.).

## AUTHOR CONTRIBUTIONS

Conceptualization, S.C.; investigation, D.-W.L.; funding acquisition, S.C.; project administration, S.C.; supervision, S.C.; writing – original draft, S.C., D.-W.L., and C.T.; writing – review and editing, S.C., D.-W.L., and C.T.; software (CFRpy), C.C.; software (SCOOTI), D.-W.L. and C.T.; validation (reproducibility and software testing), S.S.R. and A.G.

## DECLARATION OF INTERESTS

The authors declare no competing interests.

## DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

Portions of the text in this manuscript were revised and refined with the assistance of OpenAI's ChatGPT to improve clarity, structure, and grammar. The content, interpretations, and conclusions are entirely those of the authors, who take full responsibility for the final manuscript.

## SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.xpro.2026.104373.

## REFERENCES

1. Islam, M.M., Shao, W., Batavia, M., Ford, R.M., Palsson, B.O., Nielsen, J., Maranas, C.D., Lee, S.Y., and Papin, J.A. (2025). Reframing the role of the objective function in its proper context for metabolic network modeling. Cell Syst. *16*, 101298. https://doi.org/10.1016/j.cels.2025.101298.

2. Duarte, N.C., Becker, S.A., Jamshidi, N., Thiele, I., Mo, M.L., Vo, T.D., Srivas, R., and Palsson, B.Ø. (2007). Global reconstruction of the human metabolic network based on genomic and bibliomic data. Proc. Natl. Acad. Sci. USA *104*, 1777–1782. https://doi.org/10.1073/pnas.0610772104.

3. Shen, F., Boccuto, L., Pauly, R., Srikanth, S., and Chandrasekaran, S. (2019). Genome-scale network model of metabolism and histone acetylation reveals metabolic dependencies of histone deacetylase inhibitors. Genome Biol. *20*, 49.

4. Swainston, N., Smallbone, K., Hefzi, H., Dobson, P.D., Brewer, J., Hanscho, M., Zielinski, D.C., Ang, K.S., Gardiner, N.J., Gutierrez, J.M., et al. (2016). Recon 2.2: from reconstruction to model of human metabolism. Metabolomics *12*, 109. https://doi.org/10.1007/s11306-016-1051-4.

5. Brunk, E., Sahoo, S., Zielinski, D.C., Altunkaya, A., Dr"ager, A., Mih, N., Gatto, F., Nilsson, A., Preciat Gonzalez, Aurich, M.K., et al. (2018). Recon3D enables a three-dimensional view of gene variation in human metabolism. Nat. Biotechnol. *36*, 272–281. https://doi.org/10.1038/nbt.4072.

6. Min, M., and Spencer, S.L. (2019). Spontaneously slow-cycling subpopulations of human cells originate from activation of stress-response pathways. PLoS Biol *17*, e3000178. https://doi.org/10.1371/journal.pbio.3000178.

7. Sharma, K.L., Jia, S., Beacon, T.H., Adewumi, I., López, C., Hu, P., Xu, W., and Davie, J.R. (2021). Mitogen-induced transcriptional programming in human fibroblasts. Gene *800*, 145842. https://doi.org/10.1016/j.gene.2021.145842.

8. Mitra, M., Johnson, E.L., Swamy, V.S., Nersesian, L.E., Corney, D.C., Robinson, D.G., Taylor, D.G., Ambrus, A.M., Jelinek, D., Wang, W., et al. (2018). Alternative polyadenylation factors link cell cycle to migration. Genome Biol. *19*, 176. https://doi.org/10.1186/s13059-018-1551-9.

9. Wang, Y., Yuan, P., Yan, Z., Yang, M., Huo, Y., Nie, Y., Zhu, X., Qiao, J., and Yan, L. (2021). Single-cell multiomics sequencing reveals the functional regulatory landscape of early embryos. Nat. Commun. *12*, 1247. https://doi.org/10.1038/s41467-021-21409-8.

10. Lin, D.-W., Zhang, L., Zhang, J., and Chandrasekaran, S. (2025). Inferring metabolic objectives and trade-offs in single cells during embryogenesis. Cell Syst. *16*, 101164. https://doi.org/10.1016/j.cels.2024.101164.

11. Wagner, A., Wang, C., Fessler, J., DeTomaso, D., Avila-Pacheco, J., Kaminski, J., Zaghouani, S., Christian, E., Thakore, P., Schellhaass, B., et al. (2021). Metabolic modeling of single Th17 cells reveals regulators of autoimmunity. Cell *184*, 4168–4185.e21. https://doi.org/10.1016/j.cell.2021.05.045.

12. Pusa, T., Ferrarini, M.G., Andrade, R., Mary, A., Marchetti-Spaccamela, A., Stougie, L., and Sagot, M.-F. (2020). MOOMIN – Mathematical explOration of 'Omics data on a Metabollc Network. Bioinformatics *36*, 514–523. https://doi.org/10.1093/bioinformatics/btz584.

13. McInnes, L., Healy, J., Saul, N., and Großberger, L. (2018). UMAP: Uniform Manifold Approximation and Projection. J. Open Source Softw. *3*. https://doi.org/10.21105/joss.00861.

14. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. *12*.

15. Zheng, G.X.Y., Terry, J.M., Belgrader, P., Ryvkin, P., Bent, Z.W., Wilson, R., Ziraldo, S.B., Wheeler, T.D., McDermott, G.P., Zhu, J., et al. (2017). Massively parallel digital transcriptional profiling of single cells. Nat. Commun. *8*, 14049. https://doi.org/10.1038/ncomms14049.

16. Wolf, F.A., Angerer, P., and Theis, F.J. (2018). Scanpy: large-scale single-cell gene expression data analysis. Genome Biol. *19*, 15. https://doi.org/10.1186/s13059-017-1382-0.

17. Chandrasekaran, S., Zhang, J., Sun, Z., Zhang, L., Ross, C.A., Huang, Y.C., Asara, J.M., Li, H., Daley, G.Q., and Collins, J.J. (2017). Comprehensive mapping of pluripotent stem cell metabolism using dynamic genome-scale network modeling. Cell Rep. *21*, 2965–2977. https://doi.org/10.1016/j.celrep.2017.11.056.

18. Shen, F., Cheek, C., and Chandrasekaran, S. (2019). Dynamic network modeling of stem cell metabolism. In Computational Stem Cell Biology Methods in Molecular Biology, P. Cahan, ed. (Humana), pp. 203–220. https://doi.org/10.1007/978-1-4939-9224-9_14.

19. Campit, S., and Chandrasekaran, S. (2024). Inferring metabolic flux from time-course metabolomics. In Metabolic Flux Analysis in Eukaryotic Cells Methods in Molecular Biology (Humana).

20. Yao, Z., van Velthoven, C.T.J., Nguyen, T.N., Goldy, J., Sedeno-Cortes, A.E., Baftizadeh, F., Bertagnolli, D., Casper, T., Chiang, M., Crichton, K., et al. (2023). SEACells: Modeling single-cell heterogeneity with structure-preserving metacells. Nat. Biotechnol. *41*, 90–100. https://doi.org/10.1038/s41587-022-01435-w.

21. Hao, Y., Stuart, T., Kowalski, M.H., Choudhary, S., Hoffman, P., Hartman, A., Srivastava, A., Molla, G., Madad, S., Fernandez-Granda, C., and Satija, R. (2024). Dictionary learning for integrative, multimodal and scalable single-cell analysis. Nat. Biotechnol. *42*, 293–304. https://doi.org/10.1038/s41587-023-01767-y.