

Libraries used in Compass

--- Run as an administrator for installing all the libraries since admin permission is required.

#for activating any kernels or environments you can type

--- conda activate env_name

#for creating any kernels or environments you can type

--- conda -n env_name

#for deactivating kernels or environments, you can type

--- conda deactivate

#for viewing the environments present in anaconda type

--- conda env list

#for viewing all the libraries installed in anaconda type

--- conda list

#to view the version of anaconda that we are using type

--- conda --version

#rise env -- python=3.9 (needed for installing tensorflow since tf is compatible with only < python==3.9 version)

#to export any environment from one notebook to another use ===

--- conda env export > environment.yml

1.) conda install pandas

pandas - used for data analysis, data extraction and data preparation, offers data structures and operations for manipulating numerical tables and time series.

2.) conda install numpy

numpy- adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

3.) conda install tensorflow

tensorflow - used mainly for training and inference of neural networks. TensorFlow computations are expressed as stateful dataflow graphs.

4.) conda install matplotlib

matplotlib - plotting library for the Python programming language and its numerical mathematics extension NumPy. Used in scientific research as a tool for data visualization.

5.) conda install scipy

scipy - used for scientific computing and technical computing. The basic data structure used by SciPy is a multidimensional array provided by the NumPy module. NumPy provides some functions for linear algebra, Fourier transforms, and random number generation.

6.) conda install imageio

imageio - Imageio is a mature Python library that makes it easy to read and write image and video data. This includes animated images, video, volumetric data, and scientific formats

7.) conda install scikit-learn

scikit-learn - scikit-learn (formerly scikits.learn and also known as sklearn) is a free and open-source machine learning library for the Python programming language.[3] It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

8.) conda install keras

keras - Keras is an open-source library that provides a Python interface for artificial neural networks. Keras 3 is a full rewrite of Keras [and can be used] as a low-level cross-framework

language to develop custom components such as layers, models, or metrics that can be used in native workflows in JAX, TensorFlow, or PyTorch.

9.) conda install seaborn

seaborn - Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

10.) conda install ipykernel

ipykernel - this is mainly used for activating and showing that the kernel is properly working in the jupyter notebook. It provides the python kernel for jupyter notebook.

11.) conda install jupyter

jupyter - installs the **Jupyter** Notebook, JupyterLab, and the **IPython** Kernel. This is an empty metapackage for user convenience.

12.) conda install theano

theano – it is a Python library and optimizing compiler for manipulating and evaluating mathematical expressions, especially matrix-valued ones. In Theano, computations are expressed using a NumPy-esque syntax and compiled to run efficiently on either CPU or GPU architectures.

13.) conda install cmake ninja

Ninja - Ninja is a small build system with a focus on speed.

14.) conda install -c conda-forge libuv=1.39

Libuv - libuv is a multi-platform support library with a focus on asynchronous I/O. It was primarily developed for use by Node.js, but it's also used by Luvit, Julia, pyuv, and others.

15.) conda install pytorch

Pytorch - PyTorch is an open source machine learning (ML) framework based on the Python programming language and the Torch library. Torch is an open source ML library used for creating deep neural networks and is written in the Lua scripting language.

16.) conda install scikit-learn-intelex

Helps in accelerating Scikit-learn applications and still have full conformance with all Scikit-Learn APIs and algorithms. It is a free software AI accelerator that brings over 10-100X acceleration across a variety of applications.

17.) conda beautifulsoup4 docutils jinja2=3.1.4

wheel Or

conda install -c anaconda beautifulsoup4

BeautifulSoup – it is a Python library used for parsing HTML and XML documents. It creates a parse tree for parsed pages, making it easy to extract data from HTML, handle web scraping tasks, and navigate the structure of web pages. It's known for its ease of use and the ability to handle poorly formatted markup.

18.) conda install scrapy

scrapy – it is a free and open-source web scraping Python library. It's designed for large-scale web scraping. It comes included with all the tools needed to extract data from websites and process them into use-ready states. In addition to web scraping and crawling, Scrapy also allows you to use APIs to extract data directly from websites that offer it.

19.) conda install pycaret

pycaret - PyCaret is an open-source Python machine learning library that's based on the Caret machine learning library written in R. PyCaret offers features that automate and simplify standard practices and ML programs. It allows ML developers to spot-check a myriad of standard ML and DL algorithms on a classification or regression data set with a single command.

20.) conda install opencv

opencv - OpenCV is a library consisting of various programming functions, which makes it useful for real-time computer vision programs. It's able to process a variety of visual inputs from image and video data and identify objects, faces, and handwriting.0

21.) conda install polars

polars - It is a high-performance DataFrame library optimized for large data sets. It utilizes lazy evaluation and multi-threading for rapid data operations.

#libraries repository - <https://anaconda.org/conda-forge/repo>

22.) conda install bokeh

bokeh - it is a data visualization library in Python that provides high-performance interactive charts and plots. Bokeh output can be obtained in various mediums like notebook, html and server. It is possible to embed bokeh plots in Django and flask apps.

23.) conda install plotly

plotly - it is a web-based tool that supports many impressive visualizations. It is particularly suited for interactive web applications.

24.) conda install pydot

pydot - it is an interface to Graphviz and is ideal for generating complex graphs.

25.) conda install xgboost

xgboost - is an optimized distributed gradient boosting library known for its efficiency and flexibility.

26.) conda install LightGBM

Lightgbm - offers gradient boosting with decision tree-based algorithms and is known for its speed and efficiency.

27.) conda install catboost

Catboost – it is a gradient-boosting library known for its speed and quality. It supports numerical and categorical data.

28.) conda install pattern

pattern - Pattern is an all-in-one solution offering machine learning algorithms, data collection, and analysis. It supports data mining from sources like Google, Twitter, and Wikipedia.

29.) conda install fuel

Fuel – it acts as a data pipeline for machine learning models, offering out-of-the-box support for popular datasets and on-the-fly data preprocessors.

30.) conda install eli5

Eli5 - Eli5 helps debug machine learning models by offering visualization tools.

31.) conda install pybrain

Pybrain - Although inactive, PyBrain offers a range of machine-learning algorithms. It was designed for both beginners and advanced users.

32.) conda install Dist-Keras

Dist-keras - built on Keras and Apache Spark, Dist-Keras focuses on distributed deep learning.

33.) conda install statsmodels

statsmodel - Ideal for statistical analysis, StatsModels provides methods for various statistical models from simple linear regression to time-series analysis.

STEP 1 – INSTALLING THE LIBRARIES IN THEIR CORRECT VERSION FORMS.

conda install git #this is to install the github library and activate git

Requirements --- in notepad save it as requirements.txt

- python >= 3.10
- gurobipy >= 11.0.0
- numpy >= 1.12
- pandas >= 0.20
- scikit-learn >= 0.19

- `scipy >= 1.0`

Go to anaconda prompt and add the path as

```
--- cd C:\Users\user\Desktop\Harshini
```

```
--- pip install -r requirements.txt
```

```
--- python -m pip install git+https://github.com/wagnerlab-berkeley/Compass.git --upgrade
```

#This is to install compass library and clone the repository from github

```
---compass -h
```

#error will come since there needs to be a python version <3.9. Not explicitly mentioned by the developer, but Harshini researched it.

So type

```
---conda install python=3.8
```

#first remove all the other packages of python by typing `---conda uninstall python`.

This you have to do because otherwise python 3.9 or higher version will start running.

tensorflow 2.24 was installed before. Library gets deprecated because its not needed. Harshini says it hurts Compass workflow. Also Python 3.9 or less needs tensorflow 2.18 or less.

```
--- conda install tensorflow=2.18
```

STEP 2 – INSTALLING GUROBI LICENSE. GUROBI HAS ALL THE REACTIONS AND METABOLITES THAT WE NEED TO BUILD THE MODEL. WE HAVE LICENSE UPTO 23-11-2026.

#first place the gurobi license file in the user folder then copy the path

```
--- compass --set-license C:\Users\gurobi.lic
```

#to check if its working just type in anaconda prompt

```
--- python -c "import gurobipy; m=gurobipy.Model(); print('Gurobi is working')"
```

#now to install gurobi in anaconda prompt type

```
--- conda install -c gurobi gurobi
```

#now type this to check in what directory the libraries of gurobi and compass is present. These files will exist locally as well under the Compass install directory which can be found by running:

```
---compass --example-inputs --species homo_sapiens
```

This is an example input. It only shows the input files which is used for pre-processing this step is explicitly not required for installing or implementing anything.

<https://github.com/wagnerlab-berkeley/Compass/tree/master>

r

STEP 3- PREPARING THE INPUT FILES RELATED TO
GENE EXPRESSION AND SAMPLE NAMES.

Format	Species	Used In Compass As	Code file
.tsv	Human Th17 cells [SCP259] Single cell portal	Tab-separated gene expression matrix: rows = genes, columns = cells	Easy to inspect/edit; used in Compass via --expression.tsv
.mtx	Human Th17 cells	Matrix Market format with sparse matrix representation	Used with gene/sample metadata: --data-mtx expression.mtx genes.tsv sample_names.tsv
.h5ad	Mouse Th17 cells	Hierarchical format combining expression, metadata, and embeddings	Standard format from Scanpy/AnnData; passed via --data anndata_object.h5ad

HARSHINI RENAMED THE 'matrix.mtx' FILE PROVIDED IN COMPASS TO
expression.mtx. SHE ALSO RENAMED 'barcodes.tsv' to 'sample-names.tsv'.

Harshini generated the .h5ad file from GSE164999.

#now to check the **tsv expression file** you will have to place the expression.tsv file in the same
folder as where you are running it in anaconda prompt.

tsv expression file is in the form of [SHOWING AN EXAMPLE OF .TSV INPUT FILE]. Genes
are in rows and samples are in columns.

After implementing this code you will get 4 extra files

1.] *reaction.tsv* --- Main Output: Compass Score Matrix

Each Row = A Metabolic Reaction

- Example: **10FTHF5GLUtl_pos**
- **10FTHF6GLUtl** - The core reaction ID from the Recon2 model (e.g., transport of 10-formyltetrahydrofolate-) from cytosol to lysosome.
- **_pos** - Positive flux direction — Compass predicts this reaction flows forward under the modeled metabolic conditions.
- **Forward direction = dominant**
- **Reverse direction = minimal**

Each Column = A Cell or Pooled

Example:

WT-d_S141_L007_R1_001,
Ob-DHA-b_S132_L007_R1_001

- Each column is either a micropool of similar cells.
- Prefixes denote condition:
 - WT = Wild-type
 - Ob = Obese or other
 - DHA, EPA, 911, 1012 = different experimental treatments
- These columns contain Compass scores for each reaction in that sample.

1	
2	10FTHF5GLUtl_pos
3	10FTHF5GLUtm_pos
4	10FTHF6GLUtl_pos
5	10FTHF6GLUtm_pos
6	10FTHF7GLUtl_pos
7	10FTHF7GLUtm_pos
8	10FTHFtl_neg
9	10FTHFtl_pos
10	10FTHFtm_neg
11	10FTHFtm_pos
12	11DOCRTSLtm_neg
13	11DOCRTSLtm_pos
14	11DOCRTSLtr_neg
15	11DOCRTSLtr_pos
16	11DOCRTSTRNtm_neg
17	11DOCRTSTRNtm_pos

Sample

single Th17 cell or a

condition

B	C	D	E	F
WT-d S141 L007 R1 001	Ob-DHA-e S154 L007 R1 001	WT-1012-c S135 L007 R1 001	Ob-DHA-b S132 L007 R1 001	Ob-EPA-a S12

2.] *model.json.gz* or *model.json* — SBML/JSON Encoded Metabolic Model

- This is the metabolic network model (based on Recon2.2 or Recon3D) that Compass uses internally.
- It's either:
 - Auto-generated from the specified species like humans or mouse.
 - Or loaded from the provided ---model path.
- What it contains:
 - a.] List of metabolic reactions, metabolites, gene-reaction rules, and stoichiometry
 - b.] Stored in JSON format for compatibility with COBRApy and Compass

```

C: > Users > HP > AppData > Local > Temp > edebf84e-b0d1-46d0-a9b0-2f81d7935232_model.json.gz.232 > {} model.json > ...
1  {
2    "name": "RECON2_mat",
3    "reactions": {
4      "10FTHF5GLUtl_pos": {
5        "id": "10FTHF5GLUtl_pos",
6        "name": "5-glutamyl-10FTHF transport, lysosomal",
7        "subsystem": "Transport, lysosomal",
8        "upper_bound": 1000,
9        "lower_bound": 0,
10       "reactants": {
11         "10fthf5glu[c]": 1.0
12       },
13       "products": {
14         "10fthf5glu[l]": 1.0
15       },
16       "meta": {
17         "subsystem": "Transport, lysosomal",
18         "name": "5-glutamyl-10FTHF transport, lysosomal",
19         "keggId": NaN,
20         "confidenceEcoIDA": "3A0000000",
21         "confidenceScore": 2.0,
22         "sboTerm": 185.0,
23         "references": "PMID:11375437",
24         "ECNumber": NaN,
25         "notes": "carrier-mediated transport - but which and how is unknown IT"
26       },
27       "reverse_reaction": "",
28       "genes": {},
29       "gene_associations": null
30     },
31     "10FTHF5GLUtm_pos": {
32       "id": "10FTHF5GLUtm_pos",
33       "name": "5-glutamyl-10FTHF transport, mitochondrial",
34       "subsystem": "Transport, mitochondrial",
35       "upper_bound": 1000,
36       "lower_bound": 0,
37       "reactants": {

```

3.] *_tmp Folder --- Temporary Working Directory*

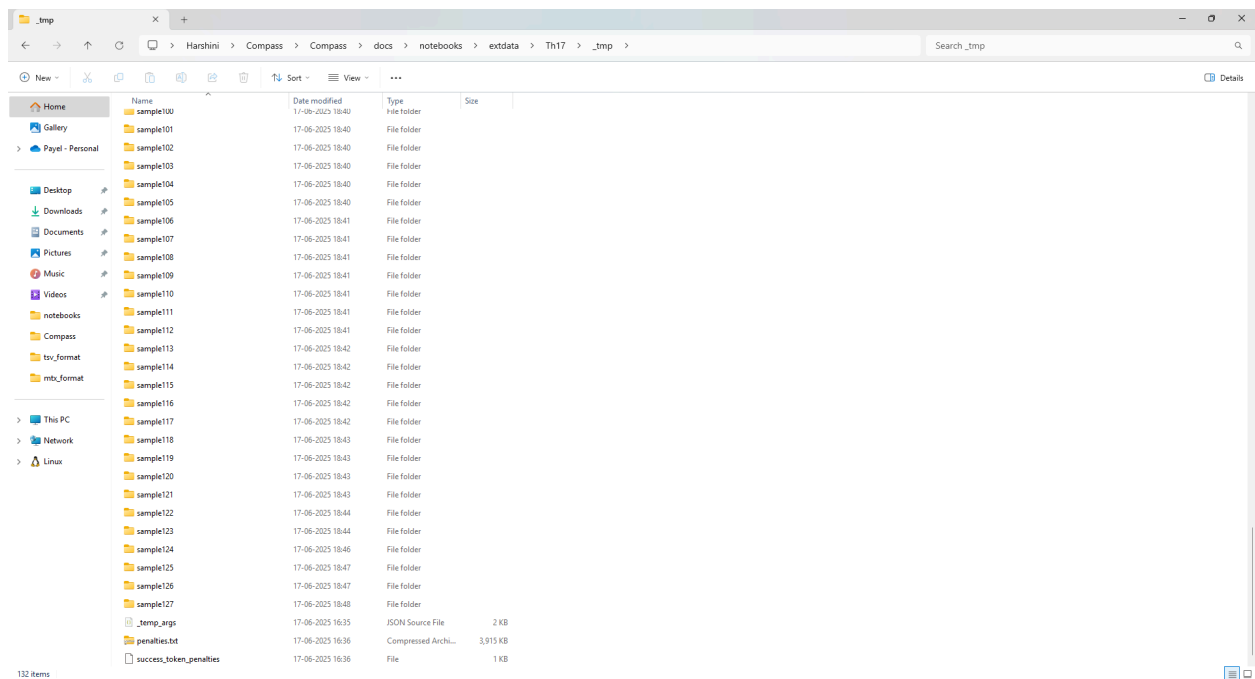
- This folder stores intermediate files during Compass execution.

What's inside:

- Temporary models for each cell
- Expression constraints per reaction
- Gurobi optimization solutions
- FBA solutions per reaction-cell pair

This directory allows Compass to:

- Run optimization per cell in parallel
- Store partial progress in case the run is interrupted
- Clean up memory



#for mtx formatted data use

#first you will have to go to

--- C:\Users\user\Desktop\Harshini\Compass\Compass\compass\Resources\Test-Data\mtx_format

#here change the directory from .tsv to .mtx format

--- compass --data-mtx expression.mtx genes.tsv sample_names.tsv --num-processes 10 species homo_sapiens

Before pre-processing, the following files are present in the folder

Name	Last commit message	Last commit date
..		
barcodes.tsv	Revert "Revert "Compass faster""	4 years ago
genes.tsv	Revert "Revert "Compass faster""	4 years ago
matrix.mtx	Untransposed the test mtx formatted matrix.	4 years ago

After pre-processing,

Name	Date modified	Type	Size
_tmp	17-06-2025 14:41	File folder	
compass	17-06-2025 14:35	Text Document	3 KB
expression.mtx	17-06-2025 11:00	MTX File	22,359 KB
genes	17-06-2025 11:00	TSV File	349 KB
sample_names	17-06-2025 11:00	TSV File	1 KB

PART OF STEP 4.

#to clear the cache of a folder type this

```
---compass --precache --species homo_sapiens
```

#preprocessing notebook

```
--- conda install statsmodels
```

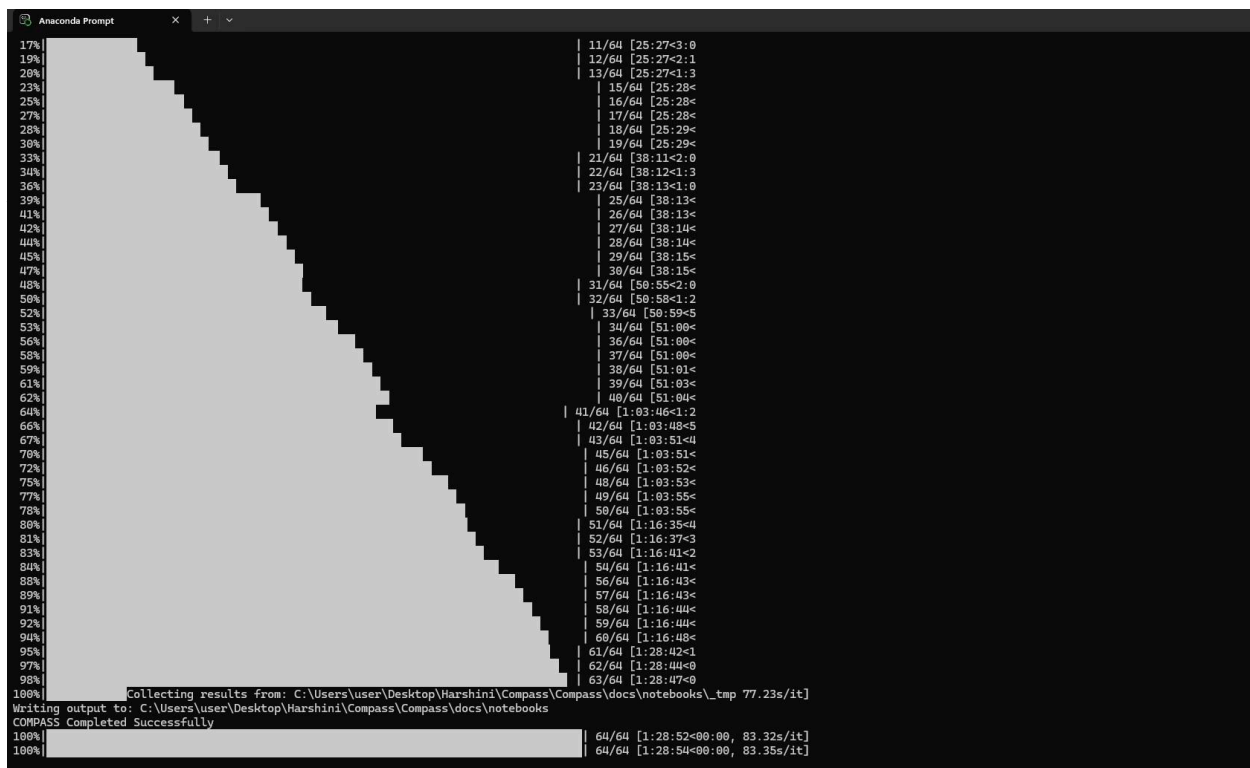
```
--- conda install scanpy
```

To run Compass on AnnData objects, use the following command:

```
--- cd C:\Users\user\Desktop\Harshini\Compass\Compass\docs\notebooks
```

```
--- compass --data anndata_object.h5ad --num-processes 10 --species homo_sapiens
```

```
--- conda install streamlit
```



Compass github

<https://github.com/wagnerlab-berkeley/Compass/tree/master>


[wagnerlab-berkeley / Compass](#)
Public

[Code](#)
[Issues](#)
[Pull requests](#)
[Actions](#)
[Projects](#)
[Security](#)
[Insights](#)

master
18 Branches
0 Tags

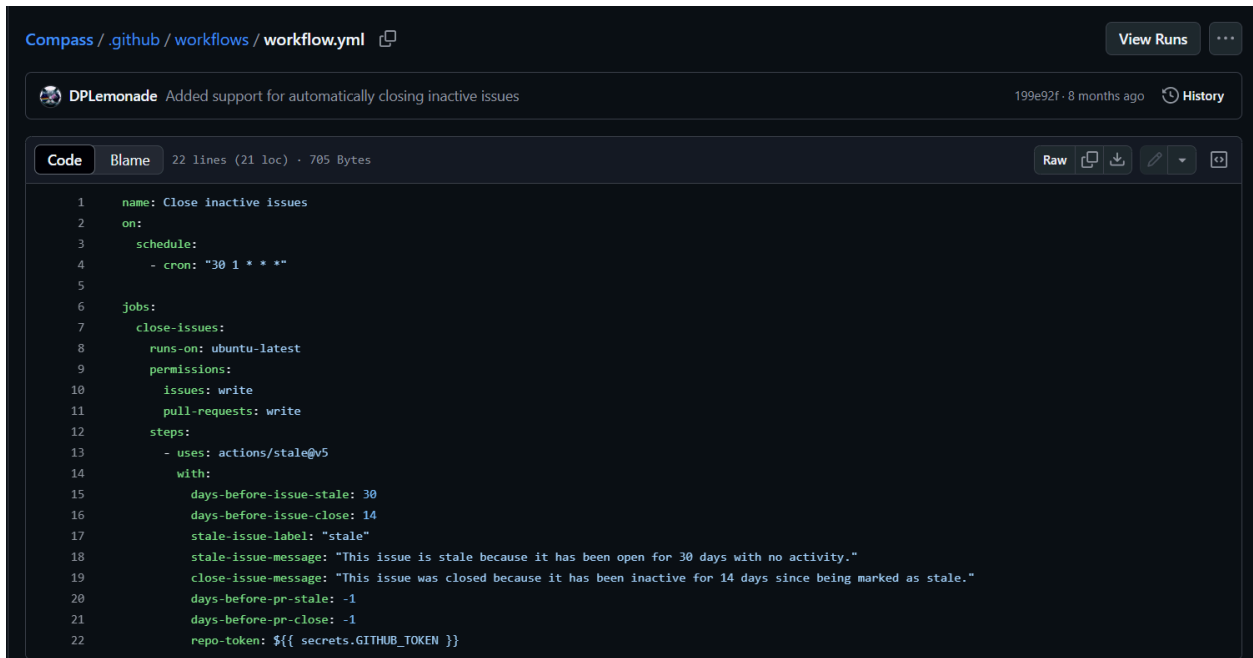
Code

File	Commit Message	Time Ago
 DPLemonade Fixed issues with Named-User License parsing 11350bd · 4 months ago 387 Commits		
.github/workflows	Added support for automatically closing inactive issues	8 months ago
compass	Fixed issues with Named-User License parsing	4 months ago
docs	Updated links in documentation to direct to new repository	9 months ago
tests	Big refactor Mflux -> Compass	7 years ago
.gitignore	Added support for Module-Compass	last year
.readthedocs.yaml	Added .readthedocs.yaml	last year
CHANGELOG.md	Added Module-Compass support for Human1	last year
LICENSE.md	Select a BSD-3 clause.	5 years ago
MANIFEST.in	Removed previous TSNE C extensions in favor of directly call...	4 years ago
README.md	Updated links in documentation to direct to new repository	9 months ago
pyproject.toml	Updated links in documentation to direct to new repository	9 months ago
setup.py	Updated links in documentation to direct to new repository	9 months ago

1.] .github/workflows ---folder

Workflow files use YAML syntax, and must have either a .yaml or .yml file extension. You must store workflow files in the .github/workflows directory of your repository. The file outlines the workflow's structure, including jobs, steps, triggers, and other configurations. The file specifies the jobs, steps, and runners involved in the workflow.

Compass/.github/workflows/workflow.yml



```
1  name: Close inactive issues
2  on:
3    schedule:
4      - cron: "30 1 * * *"
5
6  jobs:
7    close-issues:
8      runs-on: ubuntu-latest
9      permissions:
10       issues: write
11       pull-requests: write
12      steps:
13        - uses: actions/stale@v5
14          with:
15            days-before-issue-stale: 30
16            days-before-issue-close: 14
17            stale-issue-label: "stale"
18            stale-issue-message: "This issue is stale because it has been open for 30 days with no activity."
19            close-issue-message: "This issue was closed because it has been inactive for 14 days since being marked as stale."
20            days-before-pr-stale: -1
21            days-before-pr-close: -1
22            repo-token: ${{ secrets.GITHUB_TOKEN }}
```

---name: Close inactive issues

/ Describes what the job does — in this case, it manages stale GitHub issues.

---on:

schedule:

- cron: "30 1 * * *"

/Uses CRON syntax to run daily at 01:30 UTC, GitHub schedules this workflow automatically — no human or push trigger needed. Cron syntax incorporates logical operators, which are special characters that perform operations on the values provided in the cron field

---jobs:

close-issues:

runs-on: ubuntu-latest

/This defines a single job named close-issues, it will run on GitHub's Ubuntu Linux virtual machine

---permissions:

issues: write

pull-requests: write

/Grants write access to issues and pull requests so the bot can: Tag issues as stale, Comment on them, Close them.

--- steps:

- uses: actions/stale@v5

/Uses the official actions/stale GitHub Action (version 5). This action identifies and closes stale issues or pull requests.

---with:

```
days-before-issue-stale: 30
days-before-issue-close: 14
stale-issue-label: "stale"
stale-issue-message: "This issue is stale because it has been open for 30 days with no activity."
close-issue-message: "This issue was closed because it has been inactive for 14 days since being marked as stale."
days-before-pr-stale: -1
days-before-pr-close: -1
repo-token: ${{ secrets.GITHUB_TOKEN }}
```

Setting	Meaning
days-before-issue-stale: 30	Label issues as stale after 30 days of no activity
days-before-issue-close: 14	Close them 14 days later if still inactive
stale-issue-label: "stale"	Label added to stale issues
days-before-pr-stale: -1 / days-before-pr-close: -1	Pull requests are excluded (not affected)
repo-token	Authenticates the bot using GitHub's built-in token

RESULT

Every day at 1:30 AM UTC: GitHub checks for issues inactive for 30+ days, Tags them as stale, If no one replies in 14 more days → the issue is automatically closed with a message.

2.] .gitignore ---file

The .gitignore file is used in Git-based repositories to tell Git which files or folders to ignore; meaning: Don't track them, Don't stage them, Don't commit them and keep the repo clean and portable.

3.] .readthedocs.yaml --- file

Helps in building and writing the documentation. Configures building of docs on [ReadTheDocs](https://about.readthedocs.com/?ref=app.readthedocs.org)
-- <https://about.readthedocs.com/?ref=app.readthedocs.org>

4.] Changelog.md –file

Developer changelog; lists version history and major updates. All notable changes to this project will be documented in this file. It contains added files, modified files, deprecated files and deleted files history.

5.] Licese.md –file

Legal license = BSD 3-Clause License (permissive open-source)

//.md = markdown files

6.] Manifest.in

Specifies what files to include in the Python package. A manifest file is a file that contains metadata, or information about other files, and is often used to define the structure and

requirements of a software application, package, or project.

//.in = input files for macro processor

7.] Readme.md

This is where we document everything that appears in the interface when you open the github repository. It gives a project overview, installation and usage instructions.

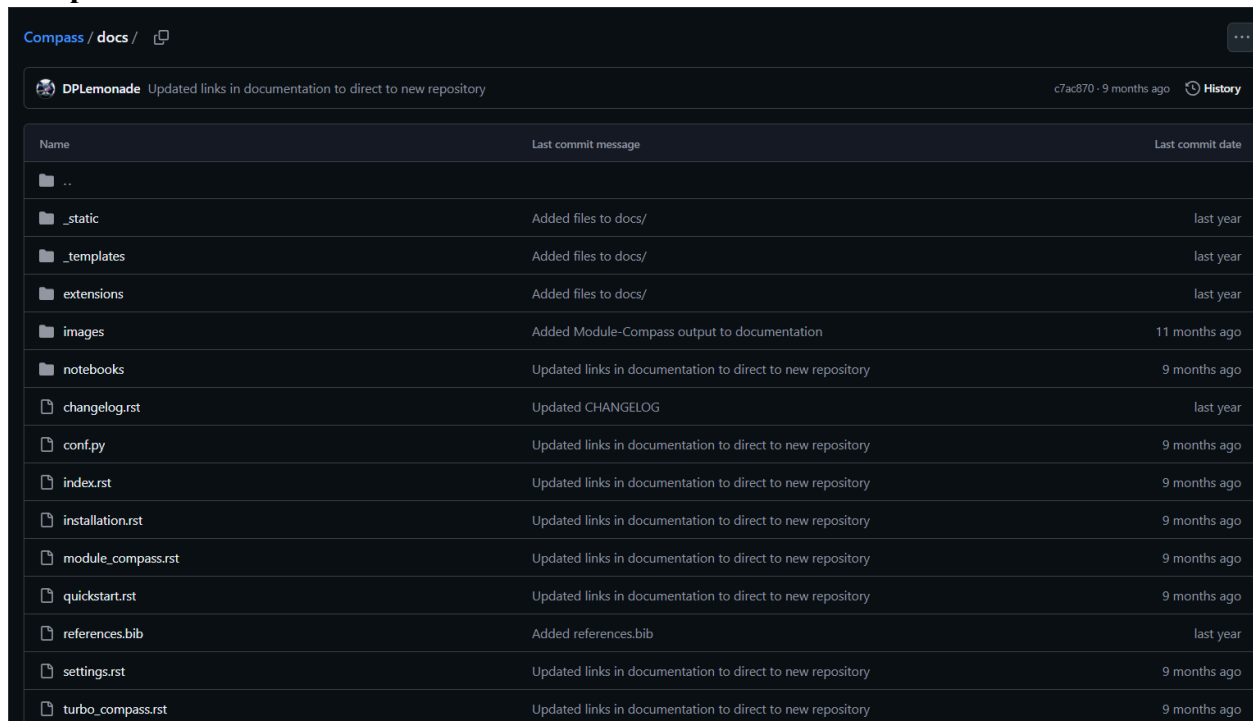
8.] setup.py

The main purpose of the setup script is to describe your module distribution to the Distutils, so that the various commands that operate on your modules do the right thing. The setup.py file is at the heart of a Python project. It describes all the metadata about your project.

9.] pyproject.toml

The pyproject.toml file acts as a configuration file for packaging-related tools (as well as other tools). In newer versions of python this version is used instead of **setup.py** due to version indifference.

Compass/docs/



Name	Last commit message	Last commit date
..		
_static	Added files to docs/	last year
_templates	Added files to docs/	last year
extensions	Added files to docs/	last year
images	Added Module-Compass output to documentation	11 months ago
notebooks	Updated links in documentation to direct to new repository	9 months ago
changelog.rst	Updated CHANGELOG	last year
conf.py	Updated links in documentation to direct to new repository	9 months ago
index.rst	Updated links in documentation to direct to new repository	9 months ago
installation.rst	Updated links in documentation to direct to new repository	9 months ago
module_compass.rst	Updated links in documentation to direct to new repository	9 months ago
quickstart.rst	Updated links in documentation to direct to new repository	9 months ago
references.bib	Added references.bib	last year
settings.rst	Updated links in documentation to direct to new repository	9 months ago
turbo_compass.rst	Updated links in documentation to direct to new repository	9 months ago

8.] Docs – folder

Documentation for how to use Compass. It includes example notebooks and markdown files for documentation building.

a.] _static --- folder

Compass/docs/_static /css/custom.css

The custom.css files help in making the website. Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML).

b.) `_templates` --- folder

Compass/docs/_templates/class.rst

The class.rst is the textual data used in python for technical documentation.

c.) `extensions` --- folder

Compass/docs/extensions/typed_returns.py

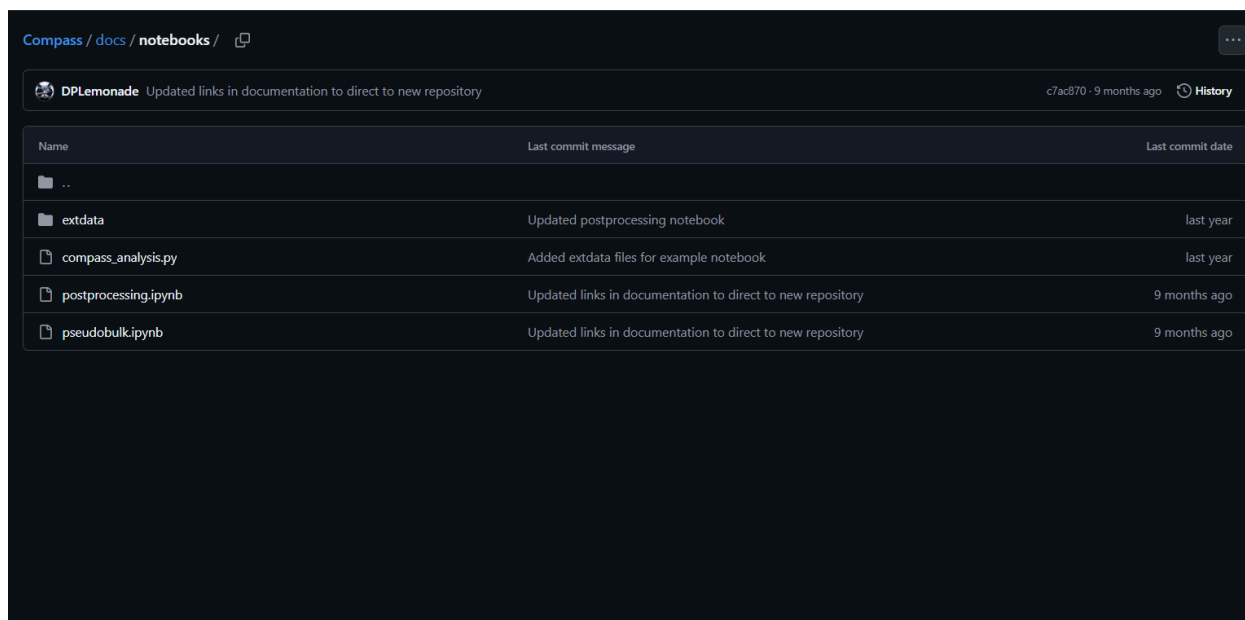
typed_returns.py = Extra Python scripts to extend Sphinx functionality. Sphinx is a documentation generator, meaning it's a tool that takes plain text source files (typically using reStructuredText) and transforms them into various output formats, most commonly HTML websites.

d.) `images` - folder

Compass/docs/images/

- **example_module_output.png** ---- Compass module output
- **glucose_reaction_scores_spearman.png** --- results of additional experiments for Turbo-Compass on glucose dataset
- **glucose_runtime.png** --- results of additional experiments for Turbo-Compass on glucose dataset
- **ibd_reaction_scores_spearman.png** --- Turbo compass image
- **ibd_runtime.png** --- Turbo compass image
- **input_ex.png** --- Turbo compass settings
- **output_ex.png** --- Turbo compass settings
- **th17_reaction_scores_spearman.png** --- Turbo compass image
- **th17_runtime.png** --- Added module-compass support for Human1

e.) `notebooks` – folder

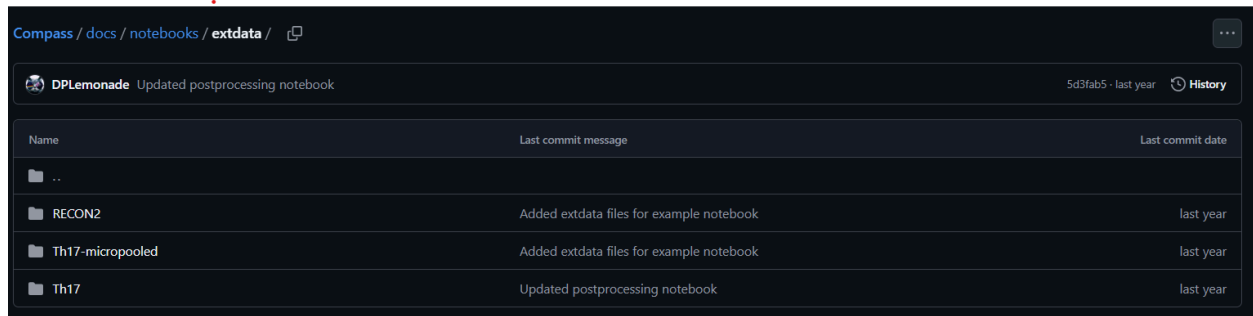


Name	Last commit message	Last commit date
..		
extdata	Updated postprocessing notebook	last year
compass_analysis.py	Added extdata files for example notebook	last year
postprocessing.ipynb	Updated links in documentation to direct to new repository	9 months ago
pseudobulk.ipynb	Updated links in documentation to direct to new repository	9 months ago

i.] extdata/

Contains example input files required by the notebooks, includes: Gene expression matrices, Cell metadata, Reaction metadata. Used by both postprocessing.ipynb and pseudobulk.ipynb.

- It contains RECON2, Th17-micropooled and Th17 folders built inside it.
- Compass/docs/notebooks/extdata/

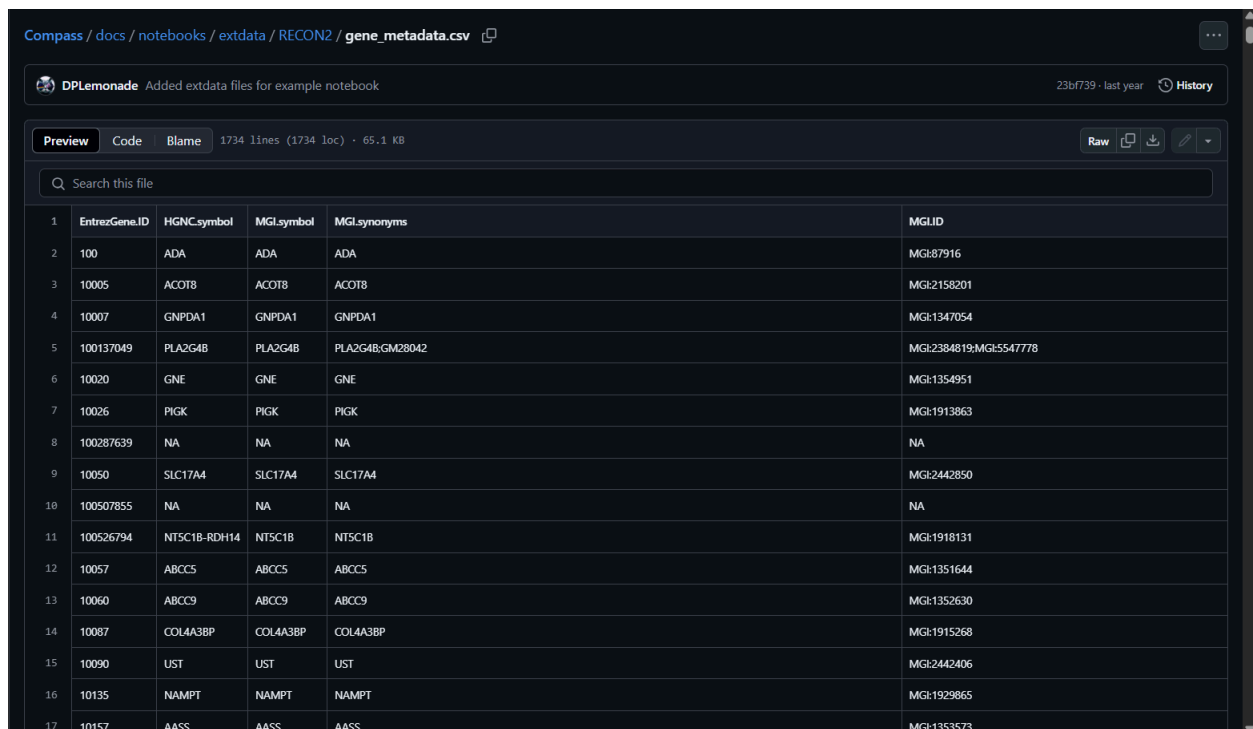


Name	Last commit message	Last commit date
..		
RECON2	Added extdata files for example notebook	last year
Th17-micropooled	Added extdata files for example notebook	last year
Th17	Updated postprocessing notebook	last year

[1] **RECON2** --- Contains Compass scores computed over the entire Recon2 model using a complete expression dataset (not necessarily Th17). Used as a reference or when exploring all metabolic reactions.

- gene_metadata.csv

It is a reference mapping file that links gene identifiers across different annotation systems. Here, HGNC.symbol is Human gene symbol, MGI.symbol is Mouse gene symbol, MGI.synonym is Mouse Genome informatics and MGLID is Mouse gene unique ID I MGI.



	EntrezGene.ID	HGNC.symbol	MGI.symbol	MGI.synonyms	MGLID
1	100	ADA	ADA	ADA	MGI:87916
2	10005	ACOT8	ACOT8	ACOT8	MGI:2158201
3	10007	GNPDA1	GNPDA1	GNPDA1	MGI:1347054
4	100137049	PLA2G4B	PLA2G4B	PLA2G4B;GM28042	MGI:2384819;MGI:5547778
5	10020	GNE	GNE	GNE	MGI:1354951
6	10026	PIGK	PIGK	PIGK	MGI:1913863
7	100287639	NA	NA	NA	NA
8	10050	SLC17A4	SLC17A4	SLC17A4	MGI:2442850
9	100507855	NA	NA	NA	NA
10	100526794	NT5C1B-RDH14	NT5C1B	NT5C1B	MGI:1918131
11	10057	ABCC5	ABCC5	ABCC5	MGI:1351644
12	10060	ABCC9	ABCC9	ABCC9	MGI:1352630
13	10087	COL4A3BP	COL4A3BP	COL4A3BP	MGI:1915268
14	10090	UST	UST	UST	MGI:2442406
15	10135	NAMPT	NAMPT	NAMPT	MGI:1929865
16	10157	AASS	AASS	AASS	MGI:1353573

It also helps in the following-

Map gene IDs between species (e.g., human vs mouse). RECON2 is human-based; Compass can run on mouse expression. Relate scRNA-seq gene IDs (like Ensembl or gene symbols) to metabolic reactions Support pathway labeling, subsystem annotation, or gene set enrichment

- metabolite_metadata.csv

It is in Human-readable labeling that is instead of h[c], Compass can display Proton (c). Filtering helps in ignoring currency metabolites (like H₂O, H⁺) based on high degree. It maps metabolites to external IDs like KEGG or HMDB.

Compass / docs / notebooks / extdata / RECON2 / metabolite_metadata.csv

DPLEmonade Added extdata files for example notebook 23bf739 · last year History

Preview Code Blame 5064 lines (5064 loc) · 318 KB Raw

Search this file

	metabolite_id	metabolite_no_compartment	compartment	metabolite_name	degree	kegg_id	hmdb_id
1	h[c]	h	[c]	proton	1210	C00282	HMDB01362
2	h2o[c]	h2o	[c]	Water	868	C00001	HMDB02111
3	na1[c]	na1	[c]	Sodium	483	C01330	HMDB00588

- reaction_metadata.csv

This file tells you what that reaction does — what metabolites, what pathway, what genes. It helps in grouping reactions into subsystems (e.g., Glycolysis) for summary plots. This connects Compass reaction penalties back to gene expression via associated_genes. reaction_metadata.csv is merged with Compass scores to label volcano plots, color by pathway, etc.

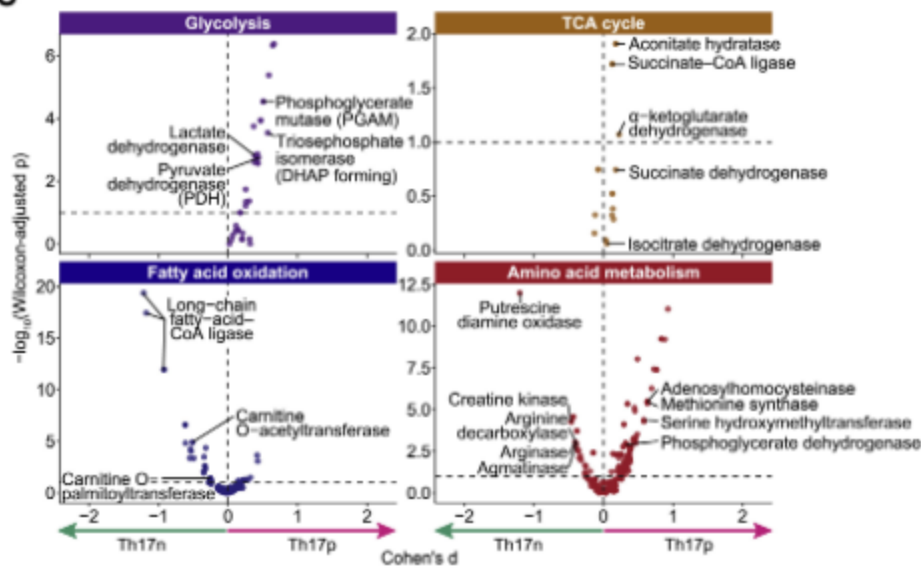
Preview Code Blame 14881 lines (14881 loc) · 1.68 MB Raw

	reaction_no_direction, reaction_name, formula, associated_genes, subsystem, EC_number, confidence
1	10FTHF5GLUtl, "5-glutamyl-10FTHF transport, lysosomal", "1.00 * 10-formyltetrahydrofolate-[Glu](5) [c] --> 1.00 * 10-formyltetrahydrofolate-[Glu](5) [l]"
2	No genes", NA, "Transport, lysosomal", N/A, 2
3	10FTHF5GLUtm, "5-glutamyl-10FTHF transport, mitochondrial", "1.00 * 10-formyltetrahydrofolate-[Glu](5) [m] --> 1.00 * 10-formyltetrahydrofolate-[Glu](5) [c]"
4	No genes", NA, "Transport, mitochondrial", N/A, 2
5	10FTHF6GLUtl, "6-glutamyl-10FTHF transport, lysosomal", "1.00 * 10-formyltetrahydrofolate-[Glu](6) [c] --> 1.00 * 10-formyltetrahydrofolate-[Glu](6) [l]"
6	No genes", NA, "Transport, lysosomal", N/A, 2
7	10FTHF6GLUtm, "6-glutamyl-10FTHF transport, mitochondrial", "1.00 * 10-formyltetrahydrofolate-[Glu](6) [m] --> 1.00 * 10-formyltetrahydrofolate-[Glu](6) [c]"
8	No genes", NA, "Transport, mitochondrial", N/A, 2
9	10FTHF7GLUtl, "7-glutamyl-10FTHF transport, lysosomal", "1.00 * 10-formyltetrahydrofolate-[Glu](7) [c] --> 1.00 * 10-formyltetrahydrofolate-[Glu](7) [l]"
10	No genes", NA, "Transport, lysosomal", N/A, 2
11	10FTHF7GLUtm, "7-glutamyl-10FTHF transport, mitochondrial", "1.00 * 10-formyltetrahydrofolate-[Glu](7) [m] --> 1.00 * 10-formyltetrahydrofolate-[Glu](7) [c]"
12	No genes", NA, "Transport, mitochondrial", N/A, 2
13	10FTHFtl, 10-Formyltetrahydrofolate lysosomal transport via diffusion, "1.00 * 10-Formyltetrahydrofolate [c] --> 1.00 * 10-Formyltetrahydrofolate [l]"
14	No genes", NA, "Transport, lysosomal", N/A, 2
15	11DOCRSLtm, 11-deoxycortisol intracellular transport, "1.00 * 11-deoxycortisol [c] --> 1.00 * 11-deoxycortisol [m]"
16	No genes", NA, "Transport, mitochondrial", N/A, 1
17	11DOCRSLtr, 11-deoxycortisol intracellular transport, "1.00 * 11-deoxycortisol [c] --> 1.00 * 11-deoxycortisol [r]"
18	No genes", NA, "Transport, endoplasmic reticular", N/A, 1
19	11DOCRSTRtm, 11-deoxycorticosterone intracellular transport, "1.00 * 11-deoxycorticosterone [c] --> 1.00 * 11-deoxycorticosterone [m]"
20	No genes", NA, "Transport, mitochondrial", N/A, 1
21	11DOCRSTRtr, 11-deoxycorticosterone intracellular transport, "1.00 * 11-deoxycorticosterone [c] --> 1.00 * 11-deoxycorticosterone [r]"
22	No genes", NA, "Transport, endoplasmic reticular", N/A, 1
23	13DAMPPOX, "1,3-Diaminopropane: oxygen oxidoreductase (deaminating)", "1.00 * trimethylenediaminium [c] + 1.00 * Water [c] + 1.00 * O2 [c] --> 1.00 * beta-Aminopropic
24	AOC1; AOC2; AOC3", AOC1; AOC2; AOC3, beta-Alanine metabolism, 1.4.3.6, 1
25	1a_24_25VITD2Hm, "1-alpha-Vitamin D-24,25-hydroxylase (D2)", "1.00 * 24R,25-Dihydroxyvitamin D2 [m] + 1.00 * proton [m] + 1.00 * Nicotinamide adenine dinucleotide phos
26	No genes", NA, Vitamin D metabolism, N/A, 2
27	1a_24_25VITD3Hm, "1-alpha-Vitamin D-24,25-hydroxylase (D3)", "1.00 * proton [m] + 1.00 * Nicotinamide adenine dinucleotide phosphate - reduced [m] + 1.00 * O2 [m] +
28	No genes", NA, Vitamin D metabolism, N/A, 2

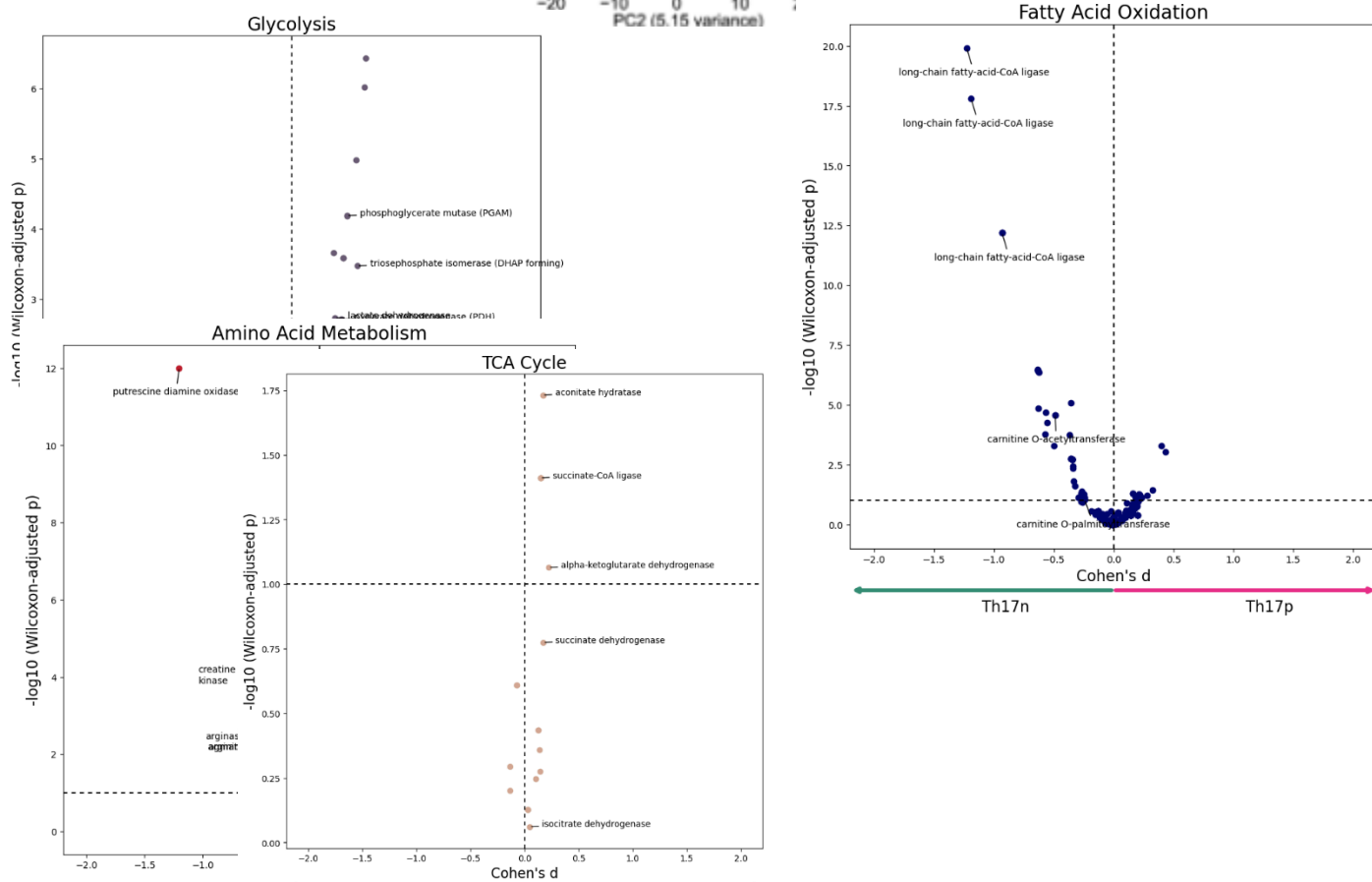
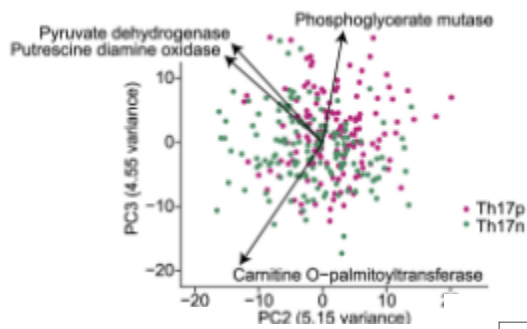
[2] Th17 [raw single cells]

- Single-cell Compass score matrix from mouse Th17 cells (no pooling).
- Each cell has one Compass score per reaction.
- Used to assess metabolic programs at single-cell resolution.
- Group comparisons (Th17n vs Th17p). Used in figure 2

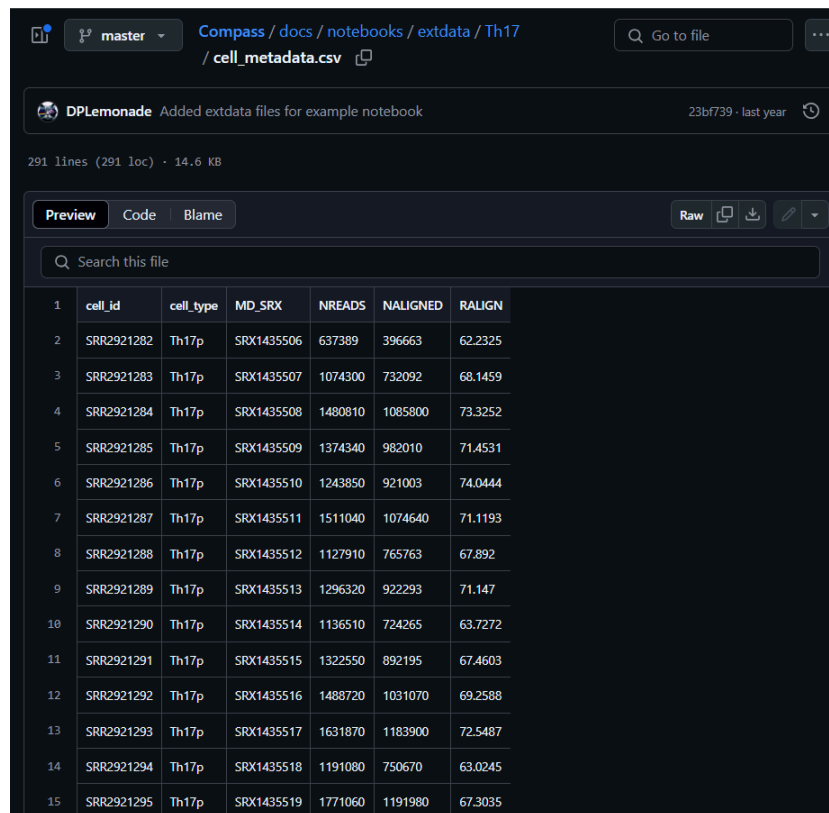
C



B



- cell_metadata.csv



291 lines (291 loc) · 14.6 KB

	cell_id	cell_type	MD_SRX	NREADS	NALIGNED	RALIGN
2	SRR2921282	Th17p	SRX1435506	637389	396663	62.2325
3	SRR2921283	Th17p	SRX1435507	1074300	732092	68.1459
4	SRR2921284	Th17p	SRX1435508	1480810	1085800	73.3252
5	SRR2921285	Th17p	SRX1435509	1374340	982010	71.4531
6	SRR2921286	Th17p	SRX1435510	1243850	921003	74.0444
7	SRR2921287	Th17p	SRX1435511	1511040	1074540	71.1193
8	SRR2921288	Th17p	SRX1435512	1127910	765763	67.892
9	SRR2921289	Th17p	SRX1435513	1296320	922293	71.147
10	SRR2921290	Th17p	SRX1435514	1136510	724265	63.7272
11	SRR2921291	Th17p	SRX1435515	1322550	892195	67.4603
12	SRR2921292	Th17p	SRX1435516	1488720	1031070	69.2588
13	SRR2921293	Th17p	SRX1435517	1631870	1183900	72.5487
14	SRR2921294	Th17p	SRX1435518	1191080	750670	63.0245
15	SRR2921295	Th17p	SRX1435519	1771060	1191980	67.3035

- linear_gene_expression.tsv
- reactions.tsv

[3] Th17-micropooled

To reduce noise, cells are pooled in small transcriptomic groups (“micropools”). Compass is rerun on the average expression per micropool.

Name	Type	Compressed size
 cell_metadata.csv	Microsoft Excel Comma S...	6 KB
 cluster_metadata.csv	Microsoft Excel Comma S...	1 KB
 data_th17.tsv	TSV File	14,253 KB
 micropooled_data.tsv	TSV File	1,833 KB
 micropools.tsv	TSV File	1 KB
 reactions.tsv	TSV File	1,135 KB

ii.] compass_analysis.py

This should be placed in the same folder path as where you are implementing the jupyter notebooks. A custom helper script used by the notebooks. It contains `wilcoxon_test(...)`, `get_reaction_consistencies(...)`, `cohens_d(...)`. Instead of importing from the full Compass package, this isolates lightweight, analysis-specific code.

iii.] postprocessing.ipynb

The main downstream analysis notebook. Reproduces the Compass paper's Figure 2C–2E (reaction differential activity between Th17p and Th17n). Walks through: Importing `reactions.tsv`, `reaction_metadata.csv`, `cell_metadata.csv`. Helps in calculating activity scores, Running Wilcoxon tests, Volcano and strip plots

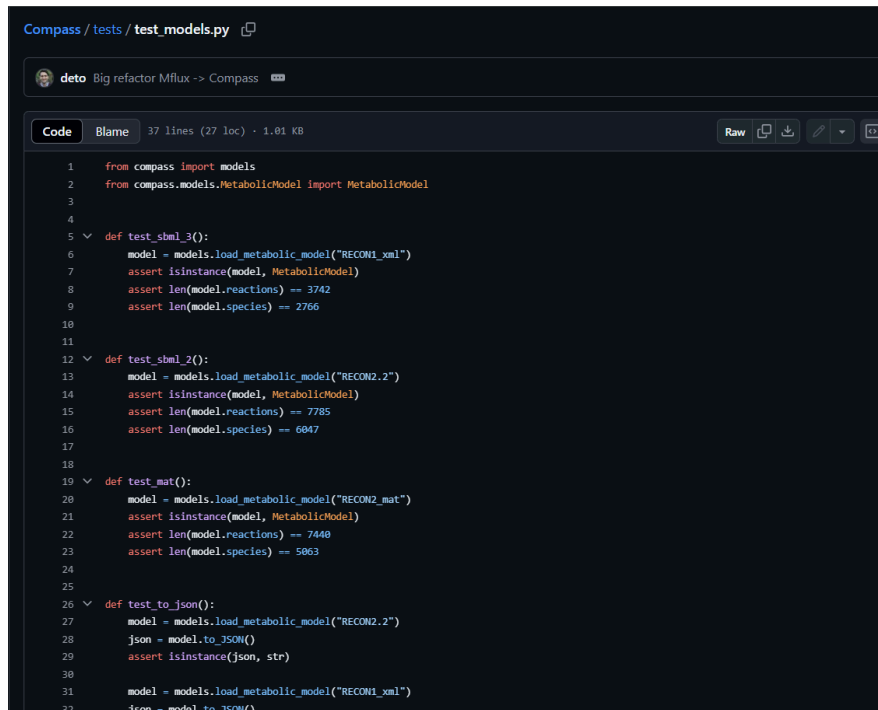
iv.] pseudobulk.ipynb

Group-wise Compass output analysis (bulk-like). Performs pseudobulk-style analysis (averaging cells per group).

	File	What it does
f.]	<code>index.rst</code>	Main entry point of the documentation — like a homepage for the docs
g.]	<code>conf.py</code>	Config file for Sphinx (defines extensions, theme, version info, etc.)
h.]	<code>installation.rst</code>	Step-by-step installation guide for Compass
i.]	<code>quickstart.rst</code>	Short guide to get up and running quickly
j.]	<code>module_compass.rst</code>	Autogenerated or manually written docs for the main Compass module
k.]	<code>turbo_compass.rst</code>	Likely refers to the newer Turbo-Compass module (faster variant)
l.]	<code>settings.rst</code>	Documentation of tunable Compass parameters
m.]	<code>changelog.rst</code>	Human-readable changelog, may duplicate CHANGELOG.md
n.]	<code>references.bib</code>	BibTeX file for citations ; used to reference literature in docs

.bib file format = bibliography file for citing the references

9.] tests --- folder



```
Compass / tests / test_models.py
deto Big refactor Mflux -> Compass

Code Blame 37 lines (27 loc) · 1.01 KB

1 from compass import models
2 from compass.models.MetabolicModel import MetabolicModel
3
4
5 def test_sbml_3():
6     model = models.load_metabolic_model("RECON1_xml")
7     assert isinstance(model, MetabolicModel)
8     assert len(model.reactions) == 3742
9     assert len(model.species) == 2766
10
11
12 def test_sbml_2():
13     model = models.load_metabolic_model("RECON2.2")
14     assert isinstance(model, MetabolicModel)
15     assert len(model.reactions) == 7785
16     assert len(model.species) == 6047
17
18
19 def test_mat():
20     model = models.load_metabolic_model("RECON2_mat")
21     assert isinstance(model, MetabolicModel)
22     assert len(model.reactions) == 7440
23     assert len(model.species) == 5063
24
25
26 def test_to_json():
27     model = models.load_metabolic_model("RECON2.2")
28     json = model.to_JSON()
29     assert isinstance(json, str)
30
31     model = models.load_metabolic_model("RECON1_xml")
32     json = model.to_JSON()
```

- compass/tests/test_models.py

This is a unit test script that ensures Compass can correctly load and parse metabolic models from different file formats and versions.

```
# Imports the model loading functions and the MetabolicModel class which defines the internal
# representation of a loaded model (reactions, species, etc.)
from compass import models
from compass.models.MetabolicModel import MetabolicModel

# Loads the RECON1 model from SBML XML format
def test_sbml_3():
    model = models.load_metabolic_model("RECON1_xml")
    assert isinstance(model, MetabolicModel)
    assert len(model.reactions) == 3742
    assert len(model.species) == 2766

# Load RECON2.2 from SBML
def test_sbml_2():
    model = models.load_metabolic_model("RECON2.2")
    assert isinstance(model, MetabolicModel)
    assert len(model.reactions) == 7785
    assert len(model.species) == 6047

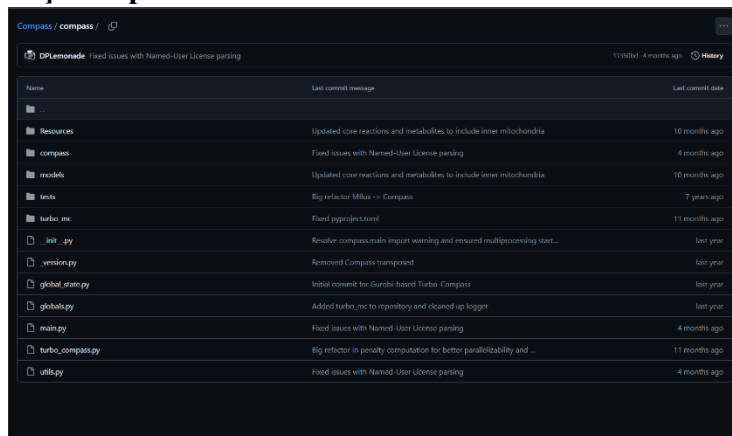
# Load RECON2 from MATLAB .mat
def test_mat():
    model = models.load_metabolic_model("RECON2_mat")
    assert isinstance(model, MetabolicModel)
    assert len(model.reactions) == 7440
    assert len(model.species) == 5063

# Convert model to JSON
def test_to_json():
    model = models.load_metabolic_model("RECON2.2")
    json = model.to_JSON()
    assert isinstance(json, str)

    model = models.load_metabolic_model("RECON1_xml")
    json = model.to_JSON()
    assert isinstance(json, str)

    model = models.load_metabolic_model("RECON2_mat")
    json = model.to_JSON()
    assert isinstance(json, str)
```

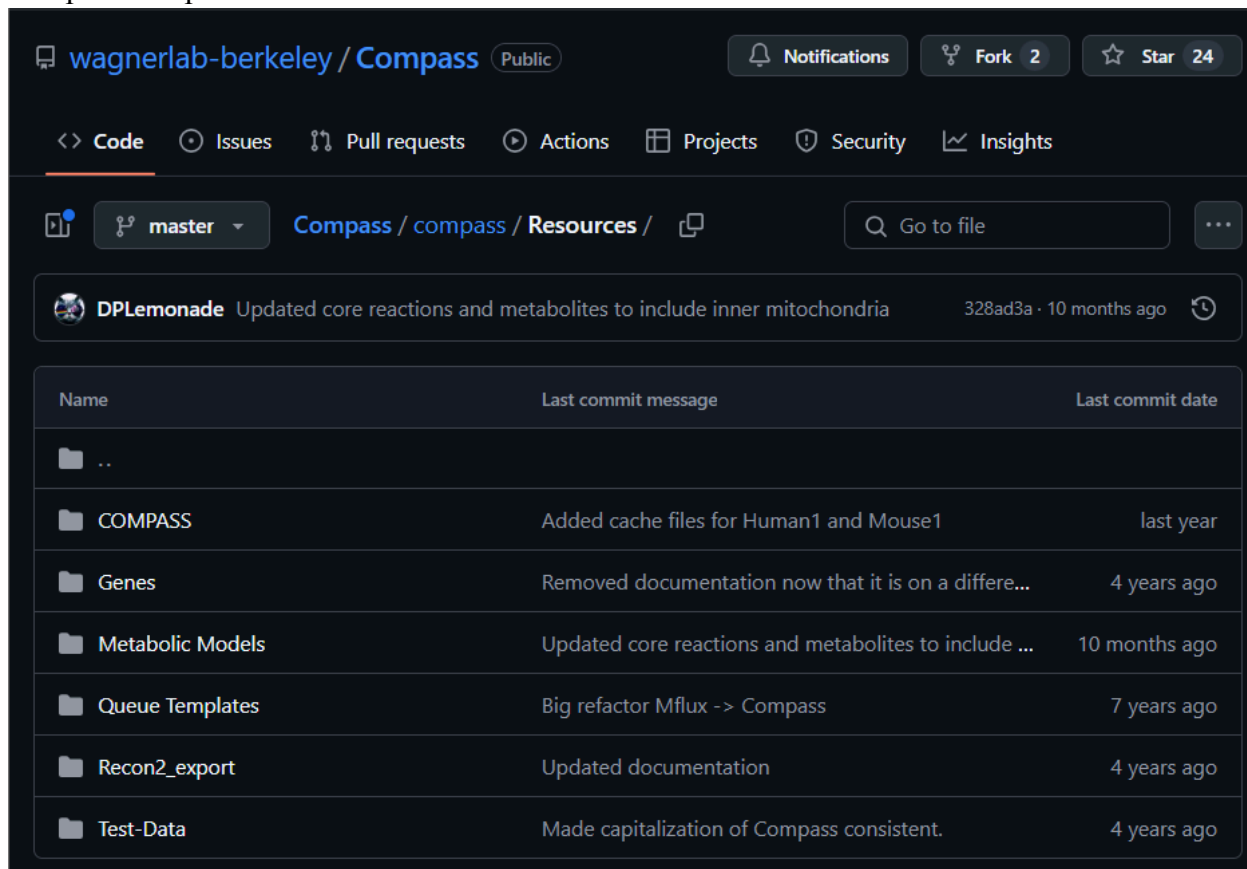
10.] Compass ---folder



Name	Last commit message	Last commit date
Resources	Updated core reactions and metabolites to include inner mitochondria	10 months ago
compass	Fixed issues with Named-User License parsing	4 months ago
models	Updated core reactions and metabolites to include inner mitochondria	10 months ago
tests	Big refactor Mflux -> Compass	7 years ago
turbo_mc	Fixed pyproject.toml	11 months ago
__init__.py	Reorder compass.yaml import warning and ensured multiprocessing start...	last year
version.py	Removed Compass transposed	last year
global_state.py	Initial commit for Gurobi-based Turbo Compass	last year
global.py	Added turbo_mc to repository and cleaned up logger	last year
main.py	Fixed issues with Named-User License parsing	4 months ago
turbo_compass.py	Big refactor in penalty computation for better parallelizability and ...	11 months ago
utils.py	Fixed issues with Named-User License parsing	4 months ago

a.] Resources

Compass/compass/Resources/



Name	Last commit message	Last commit date
..		
COMPASS	Added cache files for Human1 and Mouse1	last year
Genes	Removed documentation now that it is on a differe...	4 years ago
Metabolic Models	Updated core reactions and metabolites to include ...	10 months ago
Queue Templates	Big refactor Mflux -> Compass	7 years ago
Recon2_export	Updated documentation	4 years ago
Test-Data	Made capitalization of Compass consistent.	4 years ago

i.] Compass

Compass/compass/Resources/Compass

- Human1/default-media
- Mouse1/default-media
- Recon2.2/Nomedia
- Recon2_mat

Each of the above folder contains a 'preprocess.json' file. This file contains cache files for huma1 and mouse 1 files.

ii.] Genes

- HGNC.json
- HGNC.json.gz
- HOM_MouseHumanSequence.rpt.gz

Removed documentation now that it is on a different branch

iii.] Metabolic models

Name	Type
Human1	File folder
iHepatocytes2322_xml	File folder
iND750_xml	File folder
Mouse1	File folder
RECON1_mat	File folder
RECON1_xml	File folder
RECON2.2	File folder
RECON2_core_mat	File folder
RECON2_mat	File folder

Folder Name	Description
Human1	Modern human GEM from VMH (Virtual Metabolic Human); widely used in recent studies
iHepatocytes2322_xml	Liver-specific human metabolic model in SBML XML format
iND750_xml	GEM of <i>E. coli</i> , widely used in bacterial metabolism
Mouse1	Mouse metabolic model (used in Compass Th17 mouse data)
RECON1_xml	RECON1 model (early, curated human GEM), in XML (SBML) format
RECON1_mat	Same RECON1 model in MATLAB .mat format
RECON2.2	Updated human model (RECON2.2), one of the most used in metabolic modeling
RECON2_mat	RECON2 model in .mat format
RECON2_core_mat	Core version of RECON2 (subset of essential reactions/metabolites), MATLAB format

Suffix	Format	Usage in Compass
_xml	SBML (XML)	Platform-independent model file, easy to validate
_mat	MATLAB .mat	Loads faster in Python via <code>scipy.io</code> or <code>cobra.io</code>

We can load it using

```
--- model = models.load_metabolic_model("RECON2_mat")
```

Depending on the file structure, Compass will:

Parse the SBML or MATLAB file, Build a MetabolicModel object with: Stoichiometric matrix, Metabolites, Genes, Reactions and use it for downstream activity scoring based on single-cell gene expression.

If you are modelling...	Use this model
-------------------------	----------------

Human immune cells (T cells, etc.)	RECON2.2, Human1
Mouse T cells (e.g., Th17 Compass)	Mouse1
Liver-specific metabolism	iHepatocytes2322_xml
Core pathways only (fast)	RECON2_core_mat
<i>E. coli</i> metabolism	iND750_xml

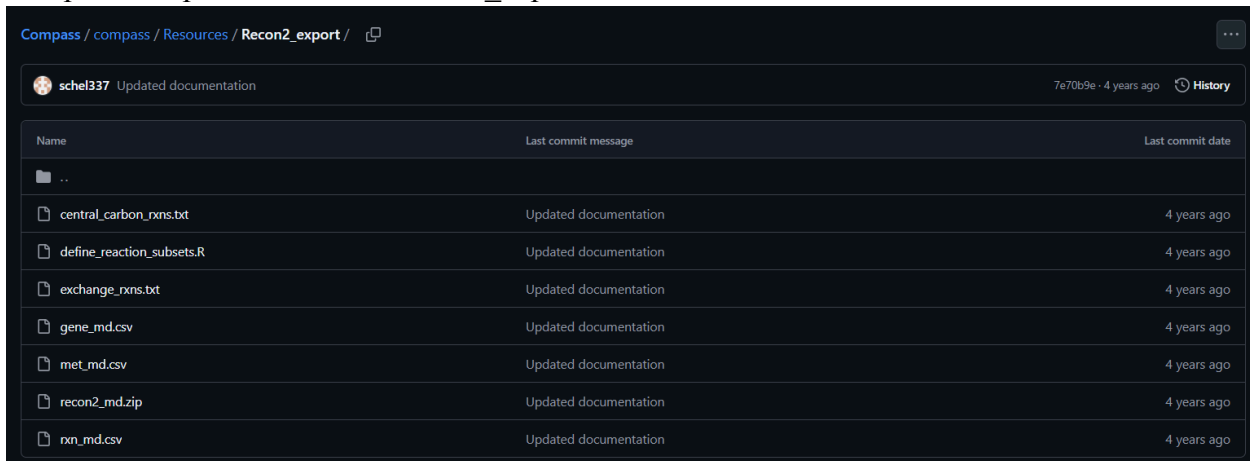
iv.] Queue templates

- CompassCollect.sh
- CompassSingleSample.sh

A .sh file is a shell script, which is a computer program designed to be executed by a Unix shell, a command-line interpreter. These files are essentially text files containing a sequence of commands that the shell can interpret and execute.

v.] Recon2_export

Compass/compass/Resources/Recon2_export/



Name	Last commit message	Last commit date
..		
central_carbon_rxns.txt	Updated documentation	4 years ago
define_reaction_subsets.R	Updated documentation	4 years ago
exchange_rxns.txt	Updated documentation	4 years ago
gene_md.csv	Updated documentation	4 years ago
met_md.csv	Updated documentation	4 years ago
recon2_md.zip	Updated documentation	4 years ago
rxn_md.csv	Updated documentation	4 years ago

- central_carbon_rxns.txt

List of reaction IDs involved in central carbon metabolism (e.g., glycolysis, TCA cycle, PPP). Used to subset or highlight key energy-producing pathways.

- define_reaction_subsets.R

R script for categorizing reactions into functionally meaningful subsystems or groups. Used to generate the .txt files above. Can be repurposed to define other subsets (e.g., lipid metabolism, amino acid synthesis).

- exchange_rxns.txt

Contains IDs of exchange reactions (inflow/outflow of metabolites). These are usually excluded from statistical tests since they don't represent intracellular activity. Compass uses this to filter non-informative reactions.

- gene_md.csv

Gene metadata for RECON2: Gene IDs (Entrez, Ensembl), Descriptions, gene-to-reaction mappings (GPR rules)

- met_md.csv

Metabolite metadata like Metabolite IDs (e.g., glc__D_c), Names, formulas, compartments (e.g., cytosol, mitochondria). Useful for labeling figures or filtering by compartment (e.g., cytosolic vs

mitochondrial ATP).

- rxn_md.csv

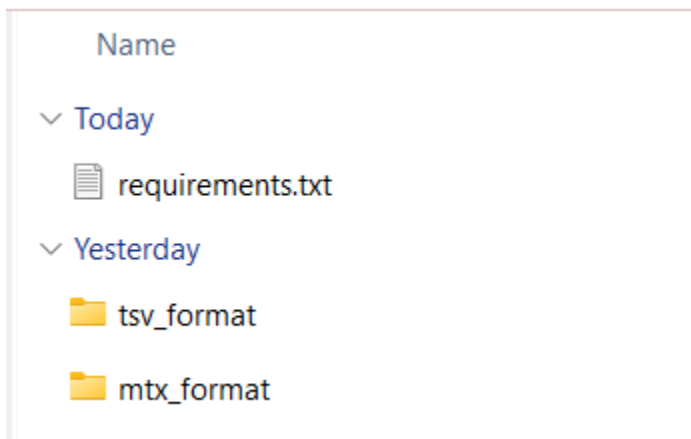
Reaction metadata like Reaction ID (e.g., r0281), Name, EC number, subsystem, stoichiometry
Compass uses this for annotation in: Volcano plots, Strip plots, Table outputs (e.g., cohens_d, pval, reaction_name, etc.)

- recon2_md.zip

A compressed archive containing all three metadata files (gene_md.csv, met_md.csv, rxn_md.csv) bundled for easy distribution.

vi.] Test-data

It contains .mtx and .tsv format file.



- mtx_format

Name	Type	Compressed size
 barcodes.tsv	TSV File	1 KB
 genes.tsv	TSV File	103 KB
 matrix.mtx	MTX File	3,785 KB

File	Description
matrix.mtx	Sparse expression matrix: rows = genes, columns = cells.
genes.tsv	Gene annotation: contains gene IDs and names. Matches MTX rows.
barcodes.tsv	Cell barcodes: unique identifiers for cells. Matches MTX columns.

When loaded together:

matrix.mtx → Raw gene expression counts (sparse)

genes.tsv → Gene names like Cd3e, Ifng, Il17a

barcodes.tsv → Cell IDs like AAACCTGAGCAGCGTA-1

- tsv_format

Name	Type	Compressed size
 expression.tsv	TSV File	1,753 KB
 knn.tsv	TSV File	1 KB
 knn_distances.tsv	TSV File	4 KB
 random_latent.tsv	TSV File	5 KB

[expression.tsv] ← Gene-level input (raw or normalized expression)



Compass infers penalties per reaction per cell using RECON model + Gurobi



[random_latent.tsv] ← Stores simulated or real latent penalty/activity matrix

If --penalty-diffusion knn is enabled:



[knn.tsv] ← Neighbor graph structure

[knn_distances.tsv] ← Quantifies distance between cells



Smoothed penalty matrix is computed and used

 _init_.py	Resolve compass.main import warning and ensured...	last year
 _version.py	Removed Compass transposed	last year
 global_state.py	Initial commit for Gurobi-based Turbo-Compass	last year
 globals.py	Added turbo_mc to repository and cleaned up logg...	last year
 main.py	Fixed issues with Named-User License parsing	4 months ago
 turbo_compass.py	Big refactor in penalty computation for better paral...	11 months ago
 utils.py	Fixed issues with Named-User License parsing	4 months ago

b.] compass

Compass/compass/compass

Name	Last commit message	Last commit date
..		
<code>__init__.py</code>	Big refactor Mflux -> Compass	7 years ago
<code>algorithm.py</code>	Fixed issues with Named-User License parsing	4 months ago
<code>cache.py</code>	Ordered cache for deterministic results	last year
<code>microclustering.py</code>	Initial commit for Gurobi-based Turbo-Compass	last year
<code>penalties.py</code>	Big refactor in penalty computation for better paral...	11 months ago
<code>torque.py</code>	Fixed torque-queue submission and added --only-...	4 years ago

i.] `__init__.py`

Package initialiser. It enables import of the module via –

from compass import algorithm

ii.] `algorithm.py`

This is one of the main files in Compass. Main penalty scoring logic. It implements:

Reaction-wise optimization, Gurobi LP formulation and Expression-to-penalty conversion.

iii.] `cache.py`

It optimises cache intermediate results for Model parsing, Expression vector and reaction GPR rules. It is used for faster runs.

iv.] `microclustering.py`

It supports pseudobulk analysis or clustering of similar cells for: KNN diffusion,

Pre-aggregation, Group-level statistics.

v.] `penalties.py`

Implements how reaction penalties are calculated. It is refactored to support: Parallelization (e.g., joblib/multiprocessing), KNN-based penalty smoothing and Batching per cell or reaction.

vi.] `torque.py`

Job scheduler integration. It adds support for running Compass jobs on a Torque cluster. This is useful in large-scale Compass runs like on 100k+ cells.

c.] models

Name	Last commit message	Last commit date
..		
MetabolicModel.py	Added support for Module-Compass	last year
__init__.py	Added Module-Compass support for Human1	last year
geneSymbols.py	Updated metadata file processing for meta subsystems	11 months ago
importCommon.py	Big refactor Mflux -> Compass	7 years ago
importMATLAB.py	Added support for Module-Compass	last year
importSBML2.py	Big refactor Mflux -> Compass	7 years ago
importSBML3.py	Added Module-Compass support for Human1	last year
partitionModel.py	Added support for Mouse1 based Module-Compass	last year
partitionModel_Human1.py	Updated core reactions and metabolites to include ...	10 months ago
partitionModel_Mouse1.py	Updated core reactions and metabolites to include ...	10 months ago
partitionModel_RECON2.py	Updated metadata file processing for meta subsystems	11 months ago

Filename	Purpose
MetabolicModel.py	Core class representing the metabolic model. Parses reactions, metabolites, and genes from RECON or Human1/Mouse1.
__init__.py	Initializes the package so it can be imported (from compass.models import...).
geneSymbols.py	Handles gene symbol normalization and mapping between human/mouse gene IDs.
importCommon.py	Shared functions for importing models in multiple formats (MAT, SBML, etc.).
importMATLAB.py	Reads .mat MATLAB model files like RECON2_mat or Human1_mat.
importSBML2.py	Loads SBML Level 2 format models (e.g., older RECON or iND750 versions).
importSBML3.py	Loads SBML Level 3 models (e.g., Human1 or newer mouse models).
partitionModel.py	Generic logic for partitioning a model into meta-subsystems.
partitionModel_Human1.py	Human1-specific partitioning: maps reactions to meta-subsystems like Glycolysis, FAO, etc.
partitionModel_Mouse1.py	Mouse1-specific partitioning of reactions into biological subsystems.
partitionModel_RECON2.py	Custom RECON2 partitioning into Compass-compatible meta-subsystems.

d.) tests

Compass/compass/tests

<> Code	Issues	Pull requests	Actions	Projects	Security	Insights
master	Compass / compass / tests /	Go to file				
deto	Big refactor Mflux -> Compass	29d1143 · 7 years ago				
Name	Last commit message	Last commit date				
..						
Network1.py	Big refactor Mflux -> Compass	7 years ago				
Network2.py	Big refactor Mflux -> Compass	7 years ago				
Network3.py	Big refactor Mflux -> Compass	7 years ago				

These test scripts define metabolic models with controlled structures to validate core components of the Compass algorithm like Gurobi-based flux inference, reaction stoichiometry, and gene expression constraints.

File	Network Structure	Tests What
Network1.py	Simple linear chain : $M1 \rightarrow R1 \rightarrow M2 \rightarrow R2 \rightarrow M3$	Basic Compass LP logic, reversible/irreversible reaction handling, exchange constraints
Network2.py	Branched network with combination: $M1 \rightarrow R1 \rightarrow M3$ $\searrow$ $\qquad \qquad \qquad \searrow R3 \rightarrow M5 \rightarrow E5$ $M2 \rightarrow R2 \rightarrow M4 \nearrow$	Stoichiometry-dependent flux limits (R3 needs 2 M4 + 1 M3), reaction splitting, exchange balance
Network3.py	Circular loop or feedback: $M1 \rightarrow R1 \rightarrow M2 \rightarrow R2 \rightarrow M3 \rightarrow R3 \rightarrow M1$	Detects cyclic dependencies, checks Compass’s ability to constrain infeasible loops or infeasible paths

e.] turbo_mc

Compass / compass / turbo_mc /	
DPLemonade	Fixed pyproject.toml
Name	Last commit message
..	
iterative_models	Fixed pyproject.toml
models	Removed scikit-surprise dependency
__init__.py	Added turbo_mc to repository and cleaned up l
matrix_manipulation.py	Fixed pyproject.toml
metrics.py	Added turbo_mc to repository and cleaned up l

Filename	Purpose	Functionality
<code>__init__.py</code>	Package initializer	Makes <code>turbo_mc</code> importable as a Python module
<code>matrix_manipulation.py</code>	Matrix utilities	Matrix slicing, conversion, masking, padding
<code>metrics.py</code>	Statistical utilities	Evaluation metrics like mean, variance, confidence intervals
<code>plotting.py</code>	Plot helpers	Custom visualizations (likely UMAP, PCA, heatmaps for penalties)
<code>utils.py</code>	General-purpose utilities	Logging, file reading, environment setup
<code>wilcoxon.py</code>	Statistical testing	Fast Wilcoxon rank-sum test implementation used for differential penalty scoring
 <code>iterative_models/</code>	Contains iterative penalty refinement models	Implements models that iteratively adjust penalty scores for improved inference
 <code>models/</code>	Gurobi solver-based Turbo models	Core ML/Gurobi-backed model classes

f.] `__init__.py`

Marks this folder as a Python package. It makes everything in `compass/` importable.

g.] `_version.py`

Stores the current version of Compass (for pip install or CLI). It contains:

```
__version__ = "1.2.0"
```

h.] `global_state.py`

Manages runtime configuration, such as: Gurobi solver settings, Parallelism flags and Logging levels

i.] `globals.py`

It holds shared constants and parameters. Examples:

- List of supported RECON models
- Default metabolite exclusions (e.g., H_2O , ATP)
- Subsystem-to-color mappings for plots

j.] `main.py`

This is the main script run when you execute Compass via CLI.

k.] `turbo_compass.py`

This is the core engine of Compass. It Contains the main metabolic optimization loop. It includes: Preprocessing of gene expression data, Construction of LPs for each reaction, Gurobi LP solving, Penalty smoothing (if `--penalty-diffusion knn`) and Output to `reactions.tsv`.

l.] `utils.py`

General helper functions reused across the pipeline. It includes- CSV/TSV I/O, Argument parsing, Data structure transformations and Reaction/matrix checks.

Gurobi --- <https://docs.gurobi.com/current/>

Gurobi is a powerful mathematical optimization software, often referred to as a solver, developed by Gurobi Optimization, LLC. It's used to solve complex problems involving decision-making, particularly in areas like logistics, finance, and operations research. Gurobi excels at finding optimal solutions to linear, quadratic, and mixed-integer programming problems.

At the heart of the Compass algorithm is linear programming. Compass computes a penalty score for each reaction in each cell by solving a constrained optimization problem using the Gurobi linear solver. In order to use Compass, you need to obtain a Gurobi WLS license, which is free for academic use.

- Gurobi is the required solver for Compass.
- It's used to solve LP problems formulated on RECON models.

To install gurobi in anaconda prompt type—

```
--- python -m pip install gurobipy
```

// If you get a python or any other libraries version clash you can install a specific version. To install a specific version, you can type

```
--- python -m install gurobipy---12.0.2
```

// With Gurobi 12, gurobipy can be used with Python 3.9, 3.10, 3.11, 3.12, and 3.13.

//for offline installation

```
--- python -m pip install --find-links <wheel-dir> --no-index gurobipy
```

//if you want to install gurobi with conda then type

```
--- conda install -c gurobi gurobi
```

//use this command to install a specific version

```
--- conda install -c gurobi gurobi=12.0.2
```

Note – if you have installed using conda from starting make sure to continue with conda only, don't switch to pip. This can cause version imbalance and python interpreter may not know what to use.

If you want more information on how gurobi constraints are used, you can visit this website.

- https://www.gurobi.com/jupyter_models/?_difficulty_level=introductory
- https://colab.research.google.com/github/Gurobi/modeling-examples/blob/master/milp_tutorial/introduction_to_modeling.ipynb

//Where to place the gurobi file

Windows: C:\gurobi and your home directory (C:\Users\yourusername)

Linux: /opt/gurobi and your home directory (/home/yourusername)

macOS: /Library/gurobi and your home directory (/Users/yourusername)

//Note that if you place the gurobi.lic license file in your home directory, the license will only be available for your account on the computer.

//Storing the license file in an alternative location

--- If it is necessary to store the license file in a different directory, you will need to set the system environment variable GRB_LICENSE_FILE to point to the exact location of the file. For example, if the license file gurobi.lic is in the directory D:\work, then set GRB_LICENSE_FILE to D:\work\gurobi.lic.

--- On Windows systems, environment variables are created and modified through the Control Panel. Searching for Environment Variables from the Control Panel search box will lead you to the appropriate screen. You will need to add a new System variable named GRB_LICENSE_FILE, and set it to the location of your license file (e.g., C:\gurobi\gurobi.lic).

// How do I set environment variables for Python/Jupyter?

--- The Jupyter extension of VSCode has built-in functionality to read .env files located in the workspace directory. It will read .env files in the workspace directory. Both the Jupyter Notebook execution and the Interactive Window will be affected by the .env file.

// Payal ma'am asked an interesting question on how gurobi optimizer and Recon model is correlated ?

--- The answer to this question can be found in this paper -

<https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-020-3410-4>

FastMM is efficient for both flux variability and knockout analysis. We applied FastMM and COBRA 3.0 (using Gurobi solver) to analyze the consistent general human metabolic model Recon 2.03 (5317 reactions and 2960 metabolites) [13, 14] on a CPU of Intel Xeon E5-2640 2.60GHz. COBRA 3.0 has made a large improvement on computational efficiency compared with COBRA 2.0 (Table 1). For flux variability analysis, FastMM and COBRA 3.0 spent 80 s, and 183 s, respectively. For genome wide single gene knockout analysis, FastMM and COBRA 3.0 spent 9 s, and 19 s, respectively. For genome wide double gene knockout analysis, FastMM and COBRA 3.0 spent 260, and 118,570 s, respectively. These results showed that FastMM is 2~400 times faster than COBRA 3.0 (Table 1). The results of FVA and single gene knockout analysis using FastMM are also consistent with the results of COBRA 3.0 (Supplementary Table S1-S3).

// What Gurobi Generates in General ?

--- When called inside Python (e.g., via gurobipy), Gurobi returns:

Output	Description
Objective value	Minimum or maximum value of the objective function (e.g., metabolic penalty or flux)
Decision variables	The optimized values for the variables (e.g., fluxes through reactions)
Solver status	Whether the solution was optimal, infeasible, unbounded, etc.
Dual variables	Shadow prices / Lagrange multipliers (in advanced usage)
Timing info	Time taken to solve, number of iterations

//In Compass: What Gurobi Does Step-by-Step ?

--- 1.] Minimize: penalty for flux through reaction R_i

Subject to: $S \cdot v = 0$ (mass balance)

$v_{\min} \leq v \leq v_{\max}$

gene expression constraints (bounds based on expression)

metabolite knockout analysis, respectively. All the six core programs call GLPK or Gurobi solver and use a two-phase primal simplex method to iteratively solve the following standard FBA linear programming (LP):

$$\begin{aligned} & \max \text{ or } \min C^T V \\ & \text{Subject to : } S \times V = 0 \\ & V_l < V < V_u \end{aligned} \tag{1}$$

Where S is the stoichiometry matrix with m rows and n columns. m and n are the number of metabolites and reactions, respectively. C^T is a vector of the coefficient factors of the objective function(s). In most cases, the vector just contains one non-zero element. V_l and V_u is the lower and upper bounds of the flux. For knockout analysis, V_l and V_u can be different from wide type when there is a gene or a metabolite knockout.

To perform efficient knockout analysis, we employed an algorithm to reduce the number of LPs, which was similar as Fast-SL [12]. Firstly, we solved a LP to minimize the sum of reaction fluxes while the wild type objective function ($C^T V$) was optimized, termed as f , this LP can be written as following:

$$\begin{aligned} & \min \sum |V| \\ & \text{Subject to : } S \times V = 0 \\ & V_l < V < V_u \\ & C^T V = f \end{aligned} \tag{2}$$

2.] Gurobi solves this LP where it computes a feasible flux distribution v that minimizes the penalty and returns the objective value (a proxy for how active that reaction is in the cell).

3.] Compass stores this value. Here, Gurobi doesn't generate a file. Compass saves the output to reactions.tsv, a matrix of metabolic activity.

// How pathways are considered in Th17p and Th17n in figure 2 volcano plot ?

--- We need 2 files to analyze 1.] reactions.tsv 2.] reaction_metadata.csv

The reactions.tsv contains gene reactions like-

```
10FTHF5GLUtl_pos
10FTHF5GLUtm_pos
...
```

We'll use reaction_metadata.csv to map reactions to subsystems (pathways)

Example 1: Methionine and Cysteine Metabolism

Reaction ID: 2AMACSULT_pos

Reaction Name: 2-Aminoacrylate sulfotransferase

Subsystem (Pathway): Methionine and cysteine metabolism

This falls under amino acid metabolism, which was grouped and visualized as one of the major metabolic programs in Th17 cells.

Example 2: Fatty Acid Oxidation

Reaction ID: 2MB2COAc_pos

Reaction Name: Transport of 2-methylcrotonoyl-CoA into cytosol

Subsystem (Pathway): Fatty acid oxidation

Part of the FAO pathway, which was significantly enriched in non-pathogenic (Th17n) cells according to COMPASS analysis.

Example 3: Amino Acid Metabolism

Reaction ID: 3SALATAi_pos

Reaction Name: 3-sulfinyl-alanine transaminase (irreversible)

Subsystem: Methionine and cysteine metabolism

Again, this belongs to the amino acid metabolism super-pathway which is important in modulating immune suppression and cell signaling.

Example 4: 10FTHF5GLUtl_pos

Reaction Name: Transport of 10-formyltetrahydrofolate-glutamate5 to cytosol

Subsystem: Folate metabolism

Pathway Group: Falls under Amino Acid/1-Carbon metabolism

Folate-linked one-carbon reactions support nucleotide biosynthesis and redox balance, especially active in proliferative Th17p cells.

Example 5: 10FTHF5GLUtm_pos

Reaction Name: Transport of 10-formyltetrahydrofolate-glutamate5 to mitochondria

Subsystem: Folate metabolism

Pathway Group: Amino acid metabolism / mitochondrial redox

Transport-linked mitochondrial 1C cycles are active in Th17 differentiation and cytokine regulation.

Example 6: 3SALATAim_pos

Reaction Name: 3-sulfino-alanine transaminase (mitochondrial, irreversible)

Subsystem: Methionine and cysteine metabolism

Group: Amino Acid Metabolism

Cysteine-linked redox and transamination reactions are upregulated in Th17n for antioxidant balance.

Example 7: 3HBCOARc_pos

Reaction Name: (R)-3-Hydroxybutyryl-CoA reduction (cytosolic)

Subsystem: Fatty acid oxidation

FAO is a key metabolic hallmark of Th17n (non-pathogenic) cells.

Example 8: 2MB2COAc_pos

Reaction Name: 2-methylbutanoyl-CoA cytosolic transport

Subsystem: Fatty acid oxidation

Supports mitochondrial lipid metabolism → energy homeostasis in regulatory T cells.

Example 9: 2AMACSULT_pos

Reaction Name: 2-aminoacrylate sulfotransferase

Subsystem: Methionine and cysteine metabolism

Transsulfuration and methyl-donor pathways are dynamically modulated in T cell activation.

Example 10: ACACCTC2_pos

Reaction Name: Acetyl-CoA carboxylase (Cytosolic)

Subsystem: Fatty acid biosynthesis

Control point for fatty acid synthesis vs oxidation balance which is strongly altered in Th17n vs Th17p cells.

Compass output settings

Compass has many command-line settings like --lambda, --penalty-diffusion, --data, etc. Where exactly do these settings apply in the pipeline? And where do they affect the output files like reactions.tsv, penalties.txt.gz, or _tmp/?"

1. INPUT STAGE — --data, --data-mtx

Option	Purpose	Affects...
--data	Uses .tsv expression file	Gene expression → penalties
--data-mtx	Uses .mtx + genes.tsv + samples.tsv	Same as above

Feeds gene expression matrix into penalty scoring system for each gene-reaction association.

2. MODEL STAGE — --model, --species, --media, --and-function

Option	Purpose	Affects...
--model	Load metabolic model (e.g. RECON2 mat)	Reaction list
--species	Match gene IDs (e.g. mouse/human)	Expression to gene mapping
--and-function	Aggregates AND gene rules (min, mean, median)	Penalty value for reactions

Controls how Compass translates gene expression into reaction feasibility (penalty).

3. PENALTY SCORING STAGE — --only-penalties, --penalties-file

Option	Purpose	Output File
--only-penalties	Run only first stage (no LP), saves penalties	_tmp/penalties.txt.gz
--penalties-file	Manually specify a penalty matrix	Custom loading

This produces the raw penalty matrix. Each value means:

"How hard is it for the cell to run this reaction?"

4. LP OPTIMIZATION STAGE — Gurobi

Option	Purpose	Affects...
--num-threads	Gurobi threading	Runtime only
--lpmethod	Chooses LP algorithm (e.g. simplex, barrier)	Solver performance

Solves 1 LP per reaction per cell. Result: reaction penalty.

5. DIFFUSION (SMOOTHING) STAGE — --lambda, --penalty-diffusion, --num-neighbors

Option	Purpose	Affects File
--lambda	Smoothing factor. 0 = no sharing, >0 = shared between neighbors.	reactions.tsv
--penalty-diffusion	knn (default) or gaussian smoothing kernel	reactions.tsv
--latent-space	Define how cells are compared (e.g. PCA, tSNE, etc)	knn graph
--input-knn	Use precomputed kNN graph instead of computing new	Neighbor selection

Smooths the raw penalty values to give smoothed Compass scores, output in reactions.tsv.

6. OUTPUT STAGE — `--output-dir`, `--list-*`, etc.

Option	Output	Format
<code>--output-dir</code>	Main output directory	Includes <code>reactions.tsv</code> , etc.
<code>--list-reactions</code>	Lists reactions & their subsystems	For filtering or debug
<code>--list-genes</code>	Lists genes used by the model	For preprocessing

Output Files and Where Each Option Affects Them

File	Affected By Options	Description
<code>reactions.tsv</code>	<code>--lambda</code> , <code>--penalty-diffusion</code> , <code>--output-dir</code> , <code>--select-reactions</code>	Main output: smoothed Compass scores
<code>_tmp/penalties.txt.gz</code>	<code>--only-penalties</code> , <code>--penalties-file</code> , <code>--model</code> , <code>--species</code> , <code>--and-function</code>	Raw reaction penalties before LP/diffusion
<code>list-reactions.txt</code>	<code>--list-reactions</code>	Lists all reactions in the model

Mapping

Stage	Controlled By	Output Files
Penalty Computation	<code>--data</code> , <code>--model</code> , <code>--species</code> , <code>--and-function</code>	<code>_tmp/penalties.txt.gz</code>
LP Optimization	(internal step)	Also part of <code>reactions.tsv</code>
Diffusion	<code>--lambda</code> , <code>--penalty-diffusion</code> , <code>--latent-space</code> , <code>--knn</code>	<code>reactions.tsv</code>
Output Control	<code>--output-dir</code> , <code>--only-penalties</code> , <code>--list-*</code>	Folder structure

<code>micropools.tsv</code> , etc.	<code>--microcluster-*</code>	Optional pseudobulk-style inputs
<code>knn.tsv</code> , <code>latent.tsv</code>	<code>--input-knn</code> , <code>--latent-space</code>	Used to build diffusion neighborhood graph

Output Settings | `--output-dir` | Output folder | `--output-dir results/` |
`--temp-dir`	Temp folder for partial results	
`--list-genes`	Output gene list used by model	
`--list-reactions`	Output reactions and subsystems	
`--select-reactions`	Limit Compass to specific reactions	
`--select-subsystems`	Limit Compass to specific subsystems	

Model Settings | `--model` | Choose RECON or Human1/Mouse1 model | `--model RECON2_mat` |
`--species`	Match gene names to `homo_sapiens` or `mus_musculus`	
`--media`	Placeholder for defining nutrient conditions	
`--and-function`	Logic for AND in GPR: `min`, `median`, `mean` (default)	
`--calc-metabolites`	Compute metabolite uptake/secretion	
`--no-reactions`	Skip reaction penalty calculation	

Turbo-Compass | `--turbo` | Enable fast mode using matrix screening | |
 | `--turbo-increments`, `--turbo-min-pct`, `--turbo-max-iters` | Advanced turbo parameters | |

Penalty Diffusion | `--penalty-diffusion` | Share penalties across cells: `knn` (default), `gaussian` | `--penalty-diffusion knn` |
`--lambda`	Strength of penalty smoothing (0–1), use 0 for bulk	`--lambda 0.25`
`--num-neighbors`	K for KNN smoothing	
`--input-weights`	Input custom sample weights	
`--input-knn`, `--input-knn-distances`	Precomputed neighbor graphs (indices/distances)	
`--output-knn`	Save computed KNN graph	
`--latent-space`	Input latent space (e.g., PCA, VAE) for KNN smoothing	

Computing | `--num-processes` | Number of parallel samples processed (recommended to set > 1) | `--num-processes 8` |
`--num-threads`	Threads per process for Gurobi (default 1)	
`--only-penalties`	Only compute raw penalties (not full Compass scores)	
`--precache`	Force rebuilding the model cache	

Microcluster | `--microcluster-size` | Aggregate cells into metacells | |
 | `--microcluster-file`, `--microcluster-data-file` | Output microcluster labels and expression | |

Testing/Hidden | `--test-mode` | Run only on 100 reactions and 50 metabolites (for debugging) | |
`--glucose`	Adjust glucose media level	
`--penalties-file`	Use custom penalties file	
`--config-file`	Use job config file	
`--lpmethod`	Gurobi LP method: -1 auto, 1 dual simplex, 2 barrier	
`--save-argmaxes`, `--detailed-perf`	Experimental/debugging flags	



Compass scoring system

Both `penalties.txt.gz` and `reaction.tsv` are penalty scores.

But:

- `penalties.txt.gz` = raw unsmoothed scores
- `reaction.tsv` = smoothed scores using λ
- Compass Activity Score = $-\log(\text{reaction.tsv} + 1)$
(only used during postprocessing for statistical tests)

Reaction.tsv = Smoothed Penalty Scores

- ◆ What it contains:
 - Per-cell scores for each reaction (e.g., R1_pos, R2_neg, ...)
 - After applying λ -smoothing (penalty diffusion)
 - From the Compass documentation:

“Compass first computes raw penalties per reaction per cell, and then optionally applies diffusion via a parameter λ to smooth scores between similar cells. The result is output to `reactions.tsv`.”

- From the postprocessing.ipynb notebook:

```
reaction_penalties = pd.read_csv("extdata/Th17/reactions.tsv", sep="\t")
```

This is treated as the input to generate activity scores after smoothing — confirming it’s already diffused.

penalties.txt.gz = Raw (Pre-smoothing) Penalty Scores

- ◆ What it contains:
 - Output **before λ is applied**
 - Unprocessed or per-cell raw LP-derived penalties
 - From `compass/__main__.py` and `penalty_diffusion.py`:

If smoothing is enabled (lambda > 0), apply diffusion to penalties

Otherwise, use raw penalty file: _tmp/penalties.txt.gz

- And from the official documentation:

“--only-penalties: Compute only the raw penalties and save to penalties.txt.gz. This skips smoothing, useful for parallelizing or inspecting raw flux feasibility.”

So:

We use reactions.tsv for downstream analysis like activity scoring, volcano plots, etc.

penalties.txt.gz is internal and only needed if you're bypassing or customizing Compass' smoothing.

Activity Score = $-\log(\text{penalty} + 1)$

Location:

- GitHub notebook: [postprocessing.ipynb](#)
- Function: `get_reaction_consistencies()`

Code from repo:

```
df = -np.log(compass_reaction_penalties + 1)
```

This converts **penalty** into an **activity score**. Why?

- High penalty → low activity
- Low penalty → high activity

Step 1: Gene Expression → Penalty Matrix (P)

STAR Methods, Page 28

“We convert R into a penalty matrix $P_{m \times n}$ by point-wise inversion. Whereas R represents gene expression support that a reaction is functional in the cell, P represents the lack thereof (which will be used in a linear program below).”

This means:

- R is the reaction-by-cell matrix of gene expression support.
- P is its **inverse** — high penalty means **low gene expression**.
- This **penalty matrix** PPP is saved to `_tmp/penalties.txt.gz`.
penalties.txt.gz = penalty matrix P

Step 2: Linear Programming to Minimize Total Penalty

STAR Methods, Page 28

“Then, we solve a linear program for every reaction r in every cell iii to find the minimal resistance of cell i to carry maximal flux through r . ”

This LP is:

Minimize: $\sum p_i \times |v_i|$

Subject to: $S \cdot v = 0$
 $v_{\min} \leq v \leq v_{\max}$

Where:

- S is the stoichiometric matrix.
- v is the flux vector.
- p_i is the penalty from matrix PPP.

The result of this LP = penalty to activate a reaction in a specific cell.

Step 3: Smoothing (λ -Diffusion) Across Cells

STAR Methods, Page 28

“The computation of R and P occurs also for the neighborhood of each cell to smooth results and mitigate single-cell technical noise.”

“Compass further mitigates data sparsity effects through information-sharing on a k -nearest neighbors graph...”

Compass computes a smoothed score using the λ formula:

$$\text{SmoothedScore}_{r,c} = (1 - \lambda) \times \text{Score}_{r,c} + \lambda \times \sum (w_{c,c'} \times \text{Score}_{r,c'})$$

Where:

- $\lambda \in [0,1]$, controls how much influence neighbors have.
- This is the diffused penalty score per reaction per cell.

This smoothed matrix is what gets written to `reaction.tsv`.

Step 4: Log Transformation into Compass Activity Score

This is done not in the paper, but in their official notebook:

```
activity_score = -np.log(penalty_score + 1)
```

Used in:

- `postprocessing.ipynb`
- Volcano plots (Wilcoxon p-value vs Cohen's d)

So:

- **penalties.txt.gz**: Raw LP penalty
- **reaction.tsv**: λ -smoothed penalty
- **Log-transformed score**: Used downstream, but **not saved as a file** unless user does it manually

File	Contains	From Where	Description
penalties.txt.gz	Penalty matrix Pm×nP	STAR Methods, Page e8	Inverse of expression support (unsmoothed)
reaction.tsv	Smoothed penalties (after λ diffusion)	STAR Methods, Page e8 + GitHub	Final Compass “score” (still a penalty)
	Log-transformed score = Activity Score	GitHub: postprocessing.ipynb only	Used in volcano plots, not stored by default

Algorithm 2: Compass

input : Gene expression matrix $\{G\}_{g \times n}$
GSMM
Pre-computed maximal fluxes $\{v_r^{opt} : r \in \mathcal{R}\}$

output : Compass scores matrix $\{C\}_{m' \times n}$ ($m' \leq m$)

parameters: Smoothing parameter $\lambda \in [0, 1]$ (here, $\lambda = 0.25$)
Nearest neighbor parameter k (here, $k = 10$)
Optimality slack parameter $\omega \in [0, 1]$ (here, $\omega = 0.95$)
Penalty function $p(x)$ (here, $p(x) = 1/(1+x)$)
Meta-reaction merging threshold $\rho \in [0, 1]$ (see below)

- 1 Compute with the procedures described above:
- 2 a reaction expression matrix $\{R(G)\}_{m \times n}$
- 3 a neighborhood reaction expression matrix $\{R^N(G)\}_{m \times n}$
- 4 Transform reaction expressions to penalties:
- 5
$$P := p(R(G))$$
$$P^N := p(R^N(G))$$
- 6 $\hat{P} := (1 - \lambda)P + \lambda P^N$
- 7 **foreach** $r \in \mathcal{R}, c \in \mathcal{C}$ **do**
- 8 Let $\hat{P}^{(c)} = (\hat{P}_{1,c}, \dots, \hat{P}_{m,c})$
- 9 $y_{r,c} := \underset{v \in \mathbb{R}^n}{\text{minimize}} \hat{P}^{(c)} \cdot v$
 s.t.
 (i) $S \cdot v = 0$
 (ii) $\alpha \leq v \leq \beta$
 (iii) $v_{rev(r)} = 0$
 (iv) $v_r \geq \omega \cdot v_r^{opt}$
- 10 **end**
- 11 Let $C^{raw} = \{y_{r,c}\}_{m \times n}$
- 12 Compute meta-reaction scores $C^{meta-raw} = \{y_{r,c}'\}_{m'' \times n}$ ($m'' \leq m$) as described below
- 13 Use Algorithm 3 to scale $C^{meta-raw}$ and obtain C
- 14 Remove constant rows from C , defined as rows in which the difference between largest and smallest score is less than $\varepsilon = 10^{-3}$
- 15 **return** C

Compass scoring system explained via all the formulas mentioned in the paper**1. Penalty Score Optimization (Compass Core LP)**

$$\text{Penalty Score} = \text{minimize } \sum (p_i \times |v_i|)$$

Subject to:

$$S \cdot v = 0$$

$$v_{\square_i} \leq v \leq v_{\square_{ax}}$$

Where:

- p_i = penalty for reaction i (inversely related to gene expression)
- v_i = flux through reaction i
- S = stoichiometric matrix

The output score is the minimal penalty to enable flux through r . Lower = more likely active.

2. Activity Score from Penalty (Log-transformed)

After raw penalties are computed, Compass uses:

$$\text{Activity Score}_{r,c} = -\log_e(\text{Penalty Score}_{r,c} + 1)$$

Where:

- Penalty Score_{r,c} = raw penalty of reaction *r* in cell *c*
- Higher activity score = more active reaction

3. Lambda-Smoothing (Neighborhood Sharing)

Single-cell data is sparse. To reduce noise, Compass allows borrowing information from nearby cells using λ -smoothing.

$$\text{Smoothed Score}_{r,c} = (1 - \lambda) \times s_{r,c} + \lambda \times \sum (w_{c,c'} \times s_{r,c'})$$

Where:

- $s_{r,c}$ = activity score for reaction *r* in cell *c*
- λ = smoothing factor (e.g., 0.25)
- $w_{c,c'}$ = similarity weight between cells *c* and *c'*
- $\sum$ over $c' \in \text{neighborhood of } c$ (KNN or Gaussian)

λ Value	Interpretation	Use Case
0.0	No smoothing — each cell independent	Pseudobulks or bulk RNA-seq
0.25	Light smoothing	Default for single-cell
1.0	Fully neighbor-averaged	Extremely sparse datasets (rare)

KNN Options:

- `--penalty-diffusion knn`: Use fixed K neighbors
- `--penalty-diffusion gaussian`: Use continuous weight kernel

4. Cohen's d (Effect Size between Two Groups)

$$\text{Cohen's } d = (\bar{x}_1 - \bar{x}_2) / s_{\text{pooled}}$$

Where:

- $\bar{x}_1, \bar{x}_2$ = mean activity scores in each group
- s_{pooled} = pooled standard deviation

5. Wilcoxon Rank-Sum Test

(Used for *p*-value calculation — nonparametric):

Wilcoxon *p*-value = rank-sum test between activity scores of group 1 vs group 2

To compare activity scores of each reaction across two groups (e.g., Th17p vs Th17n):

- Inputs: score matrix (cells $\times$ reactions), labels
- Output: *p*-value, adjusted *p*-value (FDR), and Cohen's *d*

Proper co-relation between reaction.tsv and penalties.txt

Compass formulates an optimization problem (via Gurobi) for each reaction in each cell, asking: "What is the minimal penalty needed to allow flux through this reaction, based on this cell's gene expression?"

The penalty is based on gene expression and GPR rules:

- If the genes are highly expressed $\rightarrow$ low penalty
- If genes are low or not expressed $\rightarrow$ high penalty

What is in `reactions.tsv` ?

The `reactions.tsv` file contains a matrix:

- **Rows**: Reactions (e.g., r0281_pos, r0401_neg, etc.)
- **Columns**: Cells (e.g., cell_001, cell_002, ...)
- **Values**: The computed Compass penalty scores

So, the relation is:

Each value in `reactions.tsv` is *the penalty score* for a particular reaction in a particular cell.

$$\text{reactions.tsv}[r,c] = \text{penalty}_{r,c}$$

Where:

- r : reaction ID (e.g., `r0281_pos`)
- c : cell ID (e.g., from expression matrix)
- $\text{penalty}_{r,c}$: solution of the Compass optimization for that reaction and cell

Important:

- These are raw penalties: lower = more active
- You can convert them to activity scores using:

$$\text{activity}_{r,c} = -\log(\text{penalty}_{r,c} + 1)$$

But `reactions.tsv` stores only the raw (unsmoothed, unlogged) penalties unless postprocessed.

File/Data	What it is	Interpretation
reactions.tsv	Raw penalty matrix from Compass	Lower = more likely active reaction
Penalty Score	Output of LP optimization per reaction & cell	Used to fill values in <code>reactions.tsv</code>
Activity Score	$-\log(\text{penalty} + 1)$	Optional transformation for plotting

<https://dspace.library.uvic.ca/server/api/core/bitstreams/2e818d3e-df84-4fb5-a7d8-5f4a61a9dd9a/content>

---Wagnerlab thesis

Cellular metabolic activity modeling

We modeled the metabolic state of individual cells by the Compass algorithm (Wagner et al., 2021). We extracted the normalized gene expression matrix as input and then ran Compass with default settings. The output of Compass is a `reactions.tsv` file, where the numbers correspond to penalties for each reaction per cell. We employ a negative log to have scores. If the score is great, the reaction is more active.

<https://elifesciences.org/reviewed-preprints/101731?>

reaction.tsv is the smoothened penalty score and penalty.txt is the raw penalty score

What if both the values of `-pos` and `-neg` match like in the below example ?

--- when both `_pos` (forward) and `_neg` (reverse) reactions have the same Compass score, then

reactions.tsv																
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
1		exTreg_Lymph_extT	Tcon_Lymph_Tcon	exTreg_Spleen_extT	Tcon_Spleen_Tcon	Treg_Spleen_Treg	Treg_Lymph_Treg	Tcon_Spleen_Tcon	Treg_Spleen_Treg	Treg_Spleen_Treg	exTreg_Spleen_extT	Tcon_Spleen_Tcon	exTreg_Lymph_extT	Tcon_Lymph_Tcon	Treg_Lymph_Treg	
2	10TFH5GLU_pos	3909.471436	3838.883612	3873.138702	3792.300493	3768.952214	3783.142027	3853.969002	3820.482439	3804.406129	3826.188005	3813.803734	3950.607001	3849.203436	3788.60129	
3	10TFH5GLU_pos	6303.033315	6302.407427	6311.430357	6312.237547	6289.036696	6296.697287	6309.400192	6289.899059	6297.096444	6302.074707	6308.624978	6313.902994	6314.181235	6302.407427	
4	10TFH5GLU_pos	4745.072618	4627.977304	4682.529086	4554.304459	4514.143257	4540.124563	4653.483774	4600.292075	4573.873361	4609.067804	4588.792972	4807.148954	4646.179412	4548.751436	
5	10TFH5GLU_pos	6690.806151	6688.847396	6701.495365	6702.385355	6670.530465	6680.493375	6698.852181	6671.710552	6681.228424	6688.426768	6697.612505	6708.147302	6706.061375	6688.847396	
6	10TFH7GLU_pos	5302.140072	5154.039766	5222.122676	5062.307104	5012.27062	5044.779587	5186.493621	5120.165166	5086.851516	5130.987671	5105.452464	5378.178924	5177.456729	5055.481436	
7	10TFH7GLU_pos	6950.184192	6947.391513	6962.811696	6963.626437	6925.95485	6937.649111	6959.746387	6927.446252	6938.636695	6947.342959	6958.20573	6969.029363	6968.251002	6947.081436	
8	10TFH8_neg	3909.471436	3838.883612	3873.138702	3792.300493	3768.952214	3783.142027	3853.969002	3820.482439	3804.406129	3826.188005	3813.803734	3950.607001	3849.203436	3788.60129	
9	10TFH8_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
10	10TFH9_neg	10746.24926	10676.63728	10695.02871	10673.28395	10657.62743	10663.66604	10683.42475	10668.34791	10679.46996	10696.74799	10675.16995	10650.42657	10671.39615	10668.34791	
11	10TFH9_pos	23051.82226	22959.3785	22709.89574	22665.09234	22427.08616	22455.69038	22683.86022	22561.59501	22447.31284	22791.35222	22609.72988	23294.81925	22660.24793	22473.08616	
12	11DOCR1SLM_neg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
13	11DOCR1SLM_pos	71363.30495	70168.65007	69737.40103	69061.32597	67995.39279	68652.09578	69677.89202	68955.51796	69032.52084	70051.08256	69083.91214	72696.9635	69702.17397	68704.30495	
14	11DOCR1SLM_pos	71363.30495	70168.65007	69737.40103	69061.32597	67995.39279	68652.09578	69677.89202	68955.51796	69032.52084	70051.08256	69083.91214	72696.9635	69702.17397	68704.30495	
15	11DOCR1SLM_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
16	11DOCR1STRNm_r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
17	11DOCR1STRNm_r	74145.11136	72952.05926	72634.82083	71983.46221	70837.67415	71531.96229	72497.47755	71959.82503	71943.26301	72997.7863	71914.12508	75434.78066	72439.24732	71623.11136	
18	11DOCR1STRNm_r	74145.11136	72952.05926	72634.82083	71983.46221	70837.67415	71531.96229	72497.47755	71959.82503	71943.26301	72997.7863	71914.12508	75434.78066	72439.24732	71623.11136	
19	11DOCR1STRNm_r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
20	13DAMPPOX_pos	48912.90381	47493.74169	48560.72082	46987.0113	46818.97272	46797.52517	47556.59821	47270.33573	47252.88253	48208.67987	47351.33192	50330.62113	47388.44659	47080.11136	
21	1MNCAM_pos	60.4264009	61.23242008	61.6681991	59.95030133	59.19160987	59.77242632	60.950631	60.06054362	59.91680319	60.31149839	60.48497416	62.9969034	60.91410667	59.86891	
22	1PPDCR_pos	75.01190136	73.38598848	74.76626802	73.40816766	73.13444062	73.09669299	74.60866428	73.57217636	73.75998396	73.5594181	73.92532241	75.21721142	74.08169584	73.4391	
23	1a2SDHVITD2TRM_r	68452.7241	67049.35002	66784.60208	66068.92209	65249.03085	65649.9285	66605.3675	66138.21991	66204.03546	67235.84048	66132.38384	69546.08391	66695.63295	65747.91	
24	1a_24_2SVITD2Hm_r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
25	1a_24_2SVITD2Hm_r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
26	1a_2SVITD2Hm_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
27	1a_2SVITD2Hm_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
28	24NPHe_neg	17.02858344	17.04147234	17.06498634	16.99860212	16.94739976	17.02609509	17.01632236	16.94904362	17.00734455	17.02125768	17.0418326	17.1314356	17.09693545	17.0671	
29	24NPHe_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
30	24_2SDHVITD2L_pos	13.63670756	13.59355363	13.55885048	13.5661937	13.36725003	13.59355363	13.5116777	13.47193119	13.40380481	13.55424376	13.60655955	13.83724405	13.68737276	13.561	

Interpretation	What Compass Does	What we Should Do
Reaction is equally feasible in both directions	Compass does not choose one over the other both are kept	We must decide during post-processing whether to: keep both, combine, or drop one
Often means gene expression supports reversibility	Compass reports both as-is	Use mean, take max, or collapse into one reaction if needed

Why This Happens:

Compass doesn't prioritize one direction unless:

- Expression data or constraints make one direction more feasible (lower penalty).
- Otherwise, both are treated symmetrically, especially when:
 - Reaction is truly reversible in the GSMM (RECON).
 - Lambda smoothing causes forward and reverse to equalize.

Depending on your analysis goal:

Option A: Keep both

- If directionality matters (e.g. kinetic modeling).

Option B: Use the max direction

- $\text{final_score} = \max(\text{pos_score}, \text{neg_score})$

Option C: Combine into one

- if $|\text{abs}(\text{pos_score} - \text{neg_score})| < \text{threshold}$:
 $\text{final_score} = (\text{pos_score} + \text{neg_score}) / 2$

Option D: Drop both

- If bidirectional reactions confuse interpretation, we can exclude them where scores are equal.

What Does "Direction" Mean in Compass?

Compass splits each reversible metabolic reaction into two unidirectional reactions:

Reaction ID	Meaning
-------------	---------

R1_pos	Forward direction of R1
R1_neg	Reverse direction of R1

This is required because most solvers (like Gurobi) can't directly handle reversibility — so Compass splits each into `_pos` and `_neg`.

Direction = Which way the reaction is more likely to run

→ Based on gene expression and model constraints

Examples:

Reaction	<code>_pos</code> score	<code>_neg</code> score	Meaning	Direction Chosen
R1	1.2	5.6	Forward is easier	<code>_pos</code>
R2	7.5	1.0	Reverse is easier	<code>_neg</code>
R3	3.0	3.0	Equally feasible	None — undecided

In the official *YosefLab / Compass* GitHub repository, a user raised this exact question:

“For reversible reactions (having both pos and neg), it is not clear to me which one I should use ... I would like to get one effect size score per reaction so that I can plot them in Recon Map.”

— *User inquiry* in issue [#47]

<https://github.com/YosefLab/Compass/issues/47>

Differential activity analysis of positive and negative reactions #47 New issue

Open

Hrovatin opened on Sep 15, 2021

I am trying to analyse the results of compass. However, for reversible reactions (having both pos and neg) it is not clear to me which one I should use. Namely, often both directions would lead to large differential activity which would be in the same direction for both of them (but I would expect the opposite, namely if pos is upregulated then neg should be downregulated if both are differential). I assume that I can not just pick one direction with higher consistency across cells because some cells could have positive and the others negative reaction active.

At the end I would like to get one effect size score per reaction so that I can plot them in Recon Map.

schel337 on Sep 15, 2021 Contributor

That's interesting. I would also tend to think that there would not generally be a large differential for both directions of a reaction. I'm going to defer to Allon on how to interpret this, and I've pinged him on it.

allonw on Sep 17, 2021 Collaborator

That's a good question. The two directions can be significantly active in different directions (and then you've found an interesting metabolic control point!), but frequently, both directions of the reactions will be differential in the same directions, as you described. This occurs because of the presence of cycles in the network.

If your purpose is visualization, then the intuitive solution would be to draw one directed edge per reaction, based for example on the direction shown in KEGG maps, and pick the score corresponding to the chosen direction

Assignees
No one assigned

Labels
No labels

Type
No type

Projects
No projects

Milestone
No milestone

Relationships
None yet

Development
Code with agent mode
No branches or pull requests

--- Chapter 3.4.1 — "Post-processing of Compass forward and reverse reaction penalties"

“Compass outputs both forward and reverse penalties for each reaction, and thus, this posed a challenge when comparing the two systems (Compass vs scFEA). To compare the bidirectional reaction penalties produced by Compass ... I tested two methods of converting these reaction penalty pairs to a single penalty measure.”

They continue:

The first method uses only the forward (positive) penalty (“positive penalty only”).

The second method, more akin to OUR logic, uses the ratio of forward to reverse penalties:

“To calculate the ratio ... I took the inverse of the reverse (‘negative’) penalty and multiplied the value by the forward (‘positive’) penalty. This yielded a ratio for which forward movement through a reaction was expected to be favorable (lower penalty ratio)”

<https://www.biorxiv.org/content/10.1101/2024.01.15.572211v1.full.pdf>

They use this code logic

```
if abs(pos_score - neg_score) < threshold:
    direction = "reversible"
elif pos_score < neg_score:
    direction = "forward"
else:
    direction = "reverse"
```

$$R(p_{fwd}, p_{rev}) = \begin{cases} p_{fwd} & p_{rev} = 0 \\ \frac{p_{fwd}}{p_{rev}} & \text{otherwise} \end{cases}$$

where $\begin{cases} p_{fwd} \text{ is the forward (positive) reaction penalty} \\ p_{rev} \text{ is the reverse (negative) reaction penalty} \end{cases}$

This yielded a ratio for which forward movement through a reaction was expected to be favorable (relative to the reverse reaction) for a lower penalty ratio, and more restricted for a higher penalty ratio. For each of these Compass post-processing methods, we evaluated their success using the downstream similarity measures.

436
437
438
439

How to mention the model’s name in the code before pre-processing ?

--- intial code

```
compass --data expression.tsv --num-processes 10 --species homo_sapiens
```

--- where the model is mentioned

```
compass --data expression.tsv \
    --model RECON2_mat \
    --species homo_sapiens \
    --num-processes 10
```

Model Name	Description
RECON1_mat	Human Recon1
RECON2_mat	Default human metabolic model
RECON2.2	Extended version of Recon2
Human1	Updated human model
Mouse1	Mouse GSMM
Mouse2	Extended mouse model (in development)

Question – If the mouse gets orthologs because it needs to get converted to human datasets to match ?

--- Why Mouse Data Needs Ortholog Mapping in Compass

Compass uses human metabolic models by default — like `RECON2_mat`, `RECON2.2`, or `Human1` — which expect human gene IDs (e.g., `ENSG00000141510` for TP53).

But if your dataset is from mouse (e.g., `ENSMUSG00000059552`), then Compass cannot directly map mouse genes to reactions in a human model.

Solution: Ortholog Mapping (Mouse → Human)

This is why:

Compass requires `--species` to be set to `mus_musculus`

When you do this, Compass:

- automatically maps mouse genes to their human orthologs
- uses this mapping to activate reactions in the human model

This ortholog information is internally built into Compass (for supported models like RECON).

Step	What Happens
1	You input mouse scRNA-seq data
2	Set <code>--species mus musculus</code>
3	Compass maps mouse genes → human orthologs
4	Then runs optimization using a human model (e.g., RECON2)
5	This allows using a human GSMM with mouse data

You do NOT need to manually convert Ensembl IDs yourself if you use `--species mus_musculus`.

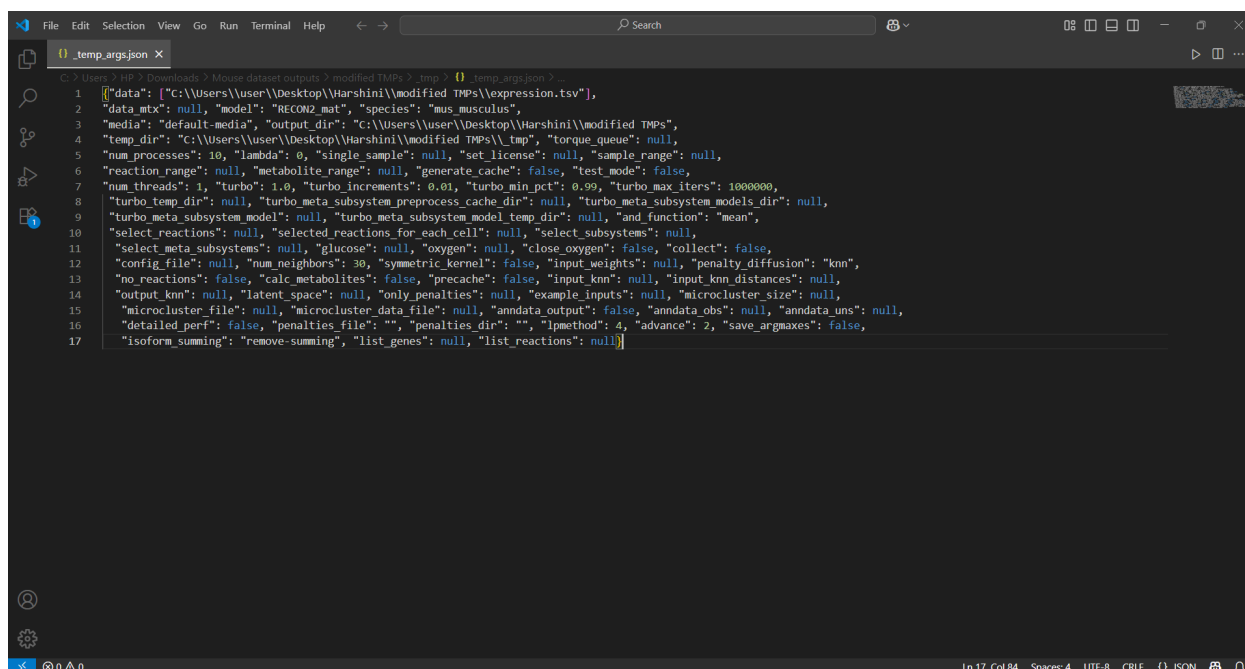
But if you're using a **custom model for mouse** (like `Mouse1`), then set:

`--model Mouse1 --species mus_musculus`

and the mapping is direct (no orthologizing needed).

Question – What is stored in `_temps_args.json` ?

C:\Users\HP\Mouse dataset outputs\modified TMPs_tmp



```
1 {"data": [{"C:\\Users\\User\\Desktop\\Harshini\\modified TMPs\\expression.tsv"},
2 "data_mtx": null, "model": "RECON2_mat", "species": "mus_musculus",
3 "media": "default-media", "output_dir": "C:\\Users\\User\\Desktop\\Harshini\\modified TMPs",
4 "temp_dir": "C:\\Users\\User\\Desktop\\Harshini\\modified TMPs\\_tmp", "torque_queue": null,
5 "num_processes": 10, "lambda": 0, "single_sample": null, "set_license": null, "sample_range": null,
6 "reaction_range": null, "metabolite_range": null, "generate_cache": false, "test_mode": false,
7 "num_threads": 1, "turbo": 1.0, "turbo_increments": 0.01, "turbo_min_pct": 0.99, "turbo_max_iters": 1000000,
8 "turbo_temp_dir": null, "turbo_meta_subsystem_preprocess_cache_dir": null, "turbo_meta_subsystem_models_dir": null,
9 "turbo_meta_subsystem_model": null, "turbo_meta_subsystem_model_temp_dir": null, "and_function": "mean",
10 "select_reactions": null, "selected_reactions_for_each_cell": null, "select_subsystems": null,
11 "select_meta_subsystems": null, "glucose": null, "oxygen": null, "close_oxygen": false, "collect": false,
12 "config_file": null, "num_neighbors": 30, "symmetric_kernel": false, "input_weights": null, "penalty_diffusion": "knn",
13 "no_reactions": false, "calc_metabolites": false, "precache": false, "input_knn": null, "input_knn_distances": null,
14 "output_knn": null, "latent_space": null, "only_penalties": null, "example_inputs": null, "microcluster_size": null,
15 "microcluster_file": null, "microcluster_data_file": null, "anndata_output": false, "anndata_obs": null, "anndata_uns": null,
16 "detailed_perf": false, "penalties_file": "", "penalties_dir": "", "lpmethod": 4, "advance": 2, "save_argmaxes": false,
17 "isoform_summing": "remove-summing", "list_genes": null, "list_reactions": null}]
```

--- This is a Compass configuration JSON object, generated by Compass when you ran it via the command line or a GUI wrapper. It shows exactly how Compass was executed on your system.

INPUT PARAMETERS

Key	Value	Meaning
"data"	["...expression.tsv"]	Gene expression data input (TSV format)
"data_mtx"	null	You are not using .mtx format input
"model"	"RECON2_mat"	Metabolic model used (Human Recon2)
"species"	"mus_musculus"	Input gene IDs are mouse genes — Compass will map to human orthologs
"media"	"default-media"	Standard simulated media conditions
"output_dir"	Path	Where final outputs (like reactions.tsv) are saved
"temp_dir"	Path	Where Compass stores intermediate outputs
"isoform_summing"	"remove-summing"	Controls how gene isoforms are handled in GPR parsing (this skips legacy summing)

COMPUTING PARAMETERS

Key	Value	Meaning
"num_processes"	10	Runs 10 processes in parallel (each handles a cell/sample)
"num_threads"	1	Each process uses 1 thread (default Compass behavior)
"lambda"	0	No penalty smoothing — each cell is processed independently
"penalty_diffusion"	"knn"	If $\lambda > 0$, penalty smoothing is done via k-nearest neighbors
"num_neighbors"	30	Would control neighborhood size if $\lambda > 0$
"precache"	false	Does not rebuild the model/media cache
"test_mode"	false	Not using debug mode (i.e., real analysis)

TURBO SETTINGS

Compass's Turbo mode is a faster approximation method.

Key	Value	Meaning
"turbo"	1.0	Turbo mode enabled at minimum SR^2 of 1.0 (conservative)
"turbo_increments"	0.01	Refinement increments
"turbo_min_pct"	0.99	Requires high precision per cell
"turbo_max_iters"	1000000	Max iterations allowed

UNSET PARAMETERS (all set to null/false)

These are features we did not use in this run:

- --select-reactions
- --select-subsystems
- --calc-metabolites
- --no-reactions
- --latent-space
- --input-knn
- --microclusters

We used Compass in standard reaction-scoring mode, without metabolite analysis, custom subsystems, or latent-space smoothing.

--- Comparison of reactions.tsv output files from both .mtx and .tsv

.tsv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		WT_d_S141_L007_F	Ob.DHA-a_S154_L0	WT-1012-d_S135_Li	Ob.DHA-b_S132_L0	Ob.EPA-a_S122_L0	Ob-c_S136_L007_R	Ob-1012-c_S140_LC	WT.EPA-a_S151_L0	WT.DHA-d_S143_L0	WT.DHA-b_S128_L0	WT.EPA-a_S117_L0	WT-1012-d_S145_Li	Ob-1012-d_S149_LC	WT.DHA-a_S117_L0
2	10TFHF5GLUL_pos	3245.920242	3187.43206	3246.467277	3188.677093	3211.270974	3190.912164	3206.441802	3220.727882	3216.444713	3250.974571	3241.51168	3227.029274	3195.275133	3237.251
3	10TFHF5GLUL_pos	3831.211621	3766.520729	3805.872198	3769.50818	3771.581046	3770.41378	3774.769417	3767.43139	3800.195909	3826.672747	3818.99752	3797.201769	3766.359683	3809.12
4	10TFHF5GLUL_pos	3516.628002	3429.344935	3501.343638	3426.899974	3449.28768	3429.281288	3451.02384	3474.071609	3470.925383	3517.708195	3505.983525	3481.770242	3434.798425	3494.80
5	10TFHF5GLUL_pos	3908.416993	3829.360034	3877.120744	3832.00219	3831.935889	3828.132917	3839.943035	3853.665863	3868.795778	3901.498741	3886.508504	3865.828049	3826.151227	3879.23
6	10TFHF7GLUL_pos	3666.318983	3560.664847	3647.665449	3557.417422	3583.691834	3559.552858	3586.566237	3614.891816	3609.43601	3666.701712	3653.648746	3624.478819	3566.494977	3638.271
7	10TFHF7GLUL_pos	3960.487024	3873.259045	3925.531468	3876.407801	3874.726076	3869.365461	3885.860643	3899.404512	3915.119565	3951.983152	3932.382829	3913.055076	3868.458767	3927.26
8	10TFHF5_neg	3245.920242	3187.43206	3246.467277	3188.677093	3211.270974	3190.912164	3206.441802	3220.727882	3216.444713	3250.974571	3241.51168	3227.029274	3195.275133	3237.251
9	10TFHF5_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	10TFHFm_neg	5290.587972	5093.538903	5239.019153	5077.026288	5047.637455	5045.158985	5097.04458	5185.592277	5259.327788	5299.657873	5174.488322	5317.998046	5102.88369	5215.911
11	10TFHFm_pos	13181.90491	12602.35623	13089.17736	12596.70505	12631.51275	12518.48767	12636.86819	12902.83693	13122.93668	13274.14664	13151.15996	13080.39971	12673.00664	12965.81
12	11DOCRTSLm_neg	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	11DOCRTSLm_pos	52539.08806	51377.80655	52303.12911	51353.26855	51585.03833	51273.44751	51560.99359	52044.20391	52230.38808	52364.67125	52238.2789	52070.60184	51445.76168	52074.1
14	11DOCRTSLm_pos	52539.08806	51377.80655	52303.12911	51353.26855	51585.03833	51273.44751	51560.99359	52044.20391	52230.38808	52364.67125	52238.2789	52070.60184	51445.76168	52074.1
15	11DOCRTSLm_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	11DOCRTSLm_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	11DOCRTSLm_pos	55075.87423	54008.87272	54846.65679	53983.05996	54182.07569	53863.41017	54188.29593	54548.69664	54761.96404	54889.44987	54876.2326	54612.85545	54051.04481	54643.7
18	11DOCRTSLm_pos	55075.87423	54008.87272	54846.65679	53983.05996	54182.07569	53863.41017	54188.29593	54548.69664	54761.96404	54889.44987	54876.2326	54612.85545	54051.04481	54643.7
19	11DOCRTSLm_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	13DAMPPOX_pos	30849.15713	29683.19885	30547.16741	29718.39323	30009.55307	29701.75424	29777.47415	30314.85103	30490.84234	30609.90221	30335.81273	30426.27155	29685.86416	30566.11
21	11MNCAM_pos	47.58699037	45.93290368	47.3599724	45.91200316	46.42190006	46.18965695	46.33387724	47.12651113	47.27675398	47.50488878	47.30755465	47.03070898	46.16808972	47.19091
22	11PDOPR_pos	49.50434646	48.49029775	49.25853462	48.4765925	48.57380209	48.22147882	48.56482029	48.95756842	49.50529283	49.42622604	49.15664102	49.20485713	48.50435664	49.24721
23	1a25DHVITD3Trm_f	50981.44383	49693.81073	50563.92928	49745.62084	49761.32835	49470.93394	49910.04053	50290.78337	50478.05017	50723.95057	50487.29356	50456.18274	49807.92434	50414.1
24	1a_24_25VITD3Hm_f	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	1a_24_25VITD3Hm_f	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	1a_25VITD3Hm_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	1a_25VITD3Hm_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	24NPHm_neg	14.23038594	14.02296043	14.21459489	14.01298491	14.04885928	13.98169306	14.03146995	14.15018727	14.2156041	14.27039686	14.31928082	14.22591381	14.03196634	14.1791
29	24NPHm_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	24_25DHVITD2L_pos	11.08134861	10.82227165	11.00297161	10.84471561	10.9130969	10.8448043	10.86777207	10.94148837	10.9637581	11.03521587	10.97356365	10.95826499	10.86201045	10.9719

.mtx

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		WT_d_S141_L007_F	Ob.DHA-a_S154_L0	WT-1012-d_S135_Li	Ob.DHA-b_S132_L0	Ob.EPA-a_S122_L0	Ob-c_S136_L007_R	Ob-1012-c_S140_LC	WT.EPA-a_S151_L0	WT.DHA-d_S143_L0	WT.DHA-b_S128_L0	WT.EPA-a_S117_L0	WT-1012-d_S145_Li	Ob-1012-d_S149_LC	WT.DHA-a_S117_L0
2	10TFHF5GLUL_pos	3245.920242	3187.43206	3246.467277	3188.677093	3211.270974	3190.912164	3206.441802	3220.727882	3216.444713	3250.974571	3241.51168	3227.029274	3195.275133	3237.251
3	10TFHF5GLUL_pos	3831.211621	3766.520729	3805.872198	3769.50818	3771.581046	3770.41378	3774.769417	3767.43139	3800.195909	3826.672747	3818.99752	3797.201769	3766.359683	3809.12
4	10TFHF5GLUL_pos	3516.628002	3429.344935	3501.343638	3426.899974	3449.28768	3429.281288	3451.02384	3474.071609	3470.925383	3517.708195	3505.983525	3481.770242	3434.798425	3494.80
5	10TFHF5GLUL_pos	3908.416993	3829.360034	3877.120744	3832.00219	3831.935889	3828.132917	3839.943035	3853.665863	3868.795778	3901.498741	3886.508504	3865.828049	3826.151227	3879.23
6	10TFHF7GLUL_pos	3666.318983	3560.664847	3647.665449	3557.417422	3583.691834	3559.552858	3586.566237	3614.891816	3609.43601	3666.701712	3653.648746	3624.478819	3566.494977	3638.271
7	10TFHF7GLUL_pos	3960.487024	3873.259045	3925.531468	3876.407801	3874.726076	3869.365461	3885.860643	3899.404512	3915.119565	3951.983152	3932.382829	3913.055076	3868.458767	3927.26
8	10TFHF5_neg	3245.920242	3187.43206	3246.467277	3188.677093	3211.270974	3190.912164	3206.441802	3220.727882	3216.444713	3250.974571	3241.51168	3227.029274	3195.275133	3237.251
9	10TFHF5_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	10TFHFm_neg	5290.587972	5093.538903	5239.019153	5077.026288	5047.637455	5045.158985	5097.04458	5185.592277	5259.327788	5299.657873	5174.488322	5317.998046	5102.88369	5215.911
11	10TFHFm_pos	13181.90491	12602.35623	13089.17736	12596.70505	12631.51275	12518.48767	12636.86819	12902.83693	13122.93668	13274.14664	13151.15996	13080.39971	12673.00664	12965.81
12	11DOCRTSLm_neg	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	11DOCRTSLm_pos	52539.08806	51377.80655	52303.12911	51353.26855	51585.03833	51273.44751	51560.99359	52044.20391	52230.38808	52364.67125	52238.2789	52070.60184	51445.76168	52074.1
14	11DOCRTSLm_pos	52539.08806	51377.80655	52303.12911	51353.26855	51585.03833	51273.44751	51560.99359	52044.20391	52230.38808	52364.67125	52238.2789	52070.60184	51445.76168	52074.1
15	11DOCRTSLm_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	11DOCRTSLm_pos	55075.87423	54008.87272	54846.65679	53983.05996	54182.07569	53863.41017	54188.29593	54548.69664	54761.96404	54889.44987	54876.2326	54612.85545	54051.04481	54643.7
17	11DOCRTSLm_pos	55075.87423	54008.87272	54846.65679	53983.05996	54182.07569	53863.41017	54188.29593	54548.69664	54761.96404	54889.44987	54876.2326	54612.85545	54051.04481	54643.7
18	11DOCRTSLm_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	13DAMPPOX_pos	30849.15713	29683.19885	30547.16741	29718.39323	30009.55307	29701.75424	29777.47415	30314.85103	30490.84234	30609.90221	30335.81273	30426.27155	29685.86416	30566.11
20	11MNCAM_pos	47.58699037	45.93290368	47.3599724	45.91200316	46.42190006	46.18965695	46.33387724	47.12651113	47.27675398	47.50488878	47.30755465	47.03070898	46.16808972	47.19091
21	11PDOPR_pos	49.50434646	48.49029775	49.25853462	48.4765925	48.57380209	48.22147882	48.56482029	48.95756842	49.50529283	49.42622604	49.15664102	49.20485713	48.50435664	49.24721
22	1a25DHVITD3Trm_f	50981.44383	49693.81073	50563.92928	49745.62084	49761.32835	49470.93394	49910.04053	50290.78337	50478.05017	50723.95057	50487.29356	50456.18274	49807.92434	50414.1
23	1a_24_25VITD3Hm_f	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	1a_24_25VITD3Hm_f	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	1a_25VITD3Hm_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	1a_25VITD3Hm_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	24NPHm_neg	14.23038594	14.02296043	14.21459489	14.01298491	14.04885928	13.98169306	14.03146995	14.15018727	14.2156041	14.27039686	14.31928082	14.22591381	14.03196634	14.1791
28	24NPHm_pos	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	24_25DHVITD2L_pos	11.08134861	10.82227165	11.00297161	10.84471561	10.9130969	10.8448043	10.86777207	10.94148837	10.9637581	11.03521587	10.97356365	10.95826499	10.86201045	10.9719

--- Both has the same values in their respective output file.

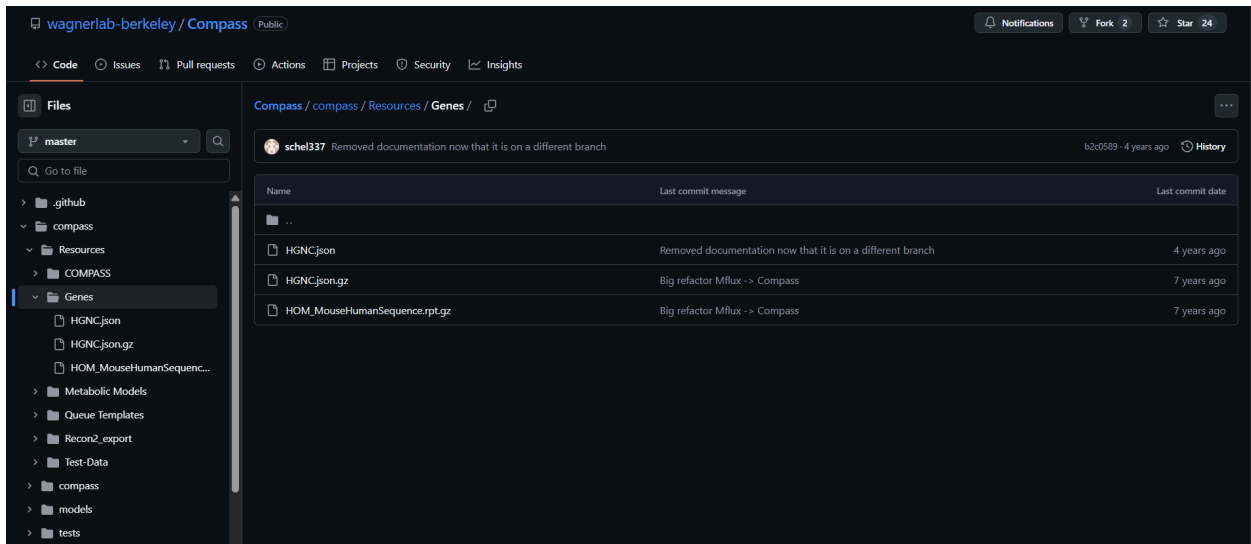
What does HOM_MouseHumanSequence.rpt.gz do ?

--- HOM_MouseHumanSequence.rpt.gz — is a reference file from the Mouse Genome Informatics (MGI) database used to map homologous genes between mouse and human.

What Does This File Do?

When you're analyzing mouse data but using a human metabolic model (like RECON2), COMPASS or other tools:

- Use this file to map mouse gene expression to human gene models
- Match via EntrezGene ID or gene symbol
- Resolve 1-to-1 orthologs for compatibility



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	HomoloGene ID	Common Organism	NCBI Taxon ID	Symbol	EntrezGene ID	Mouse MGI ID	HGNC ID	OMIM Gene ID	Genetic Location	Genomic Coordinates	Nucleotide RefSeq	Protein RefSeq	SWISS_PROT IDs
2	3	mouse, laboratory mouse	10090	Acadm	11364	MGI:87867			Chr3 78.77 cM	Chr3:153922357-NM_007382	NP_031408	P45952	
3	3	human	9606	ACADM	34		HGNC:89	607008	Chr1 p31.1	Chr1:75724347-NM_001286043	NP_001272973	P11310	
4	5	mouse, laboratory mouse	10090	Acadvl	11370	MGI:895149			Chr11 42.96 cM	Chr11:70010183-NM_017366	NP_059062	P50544	
5	5	human	9606	ACADVL	37		HGNC:92	609575	Chr17 p13.1	Chr17:7217125-NM_0000018	NP_001257376	P49748	
6	6	mouse, laboratory mouse	10090	Acat1	110446	MGI:87870			Chr9 29.12 cM	Chr9:53580522-NM_144784	NP_659033	Q8QZT1	
7	6	human	9606	ACAT1	38		HGNC:93	607809	Chr11 q22.3	Chr11:10812153-NM_0000019	XP_016873170	P24752	
8	7	mouse, laboratory mouse	10090	Acvr1	11477	MGI:87911			Chr2 33.05 cM	Chr2:58388644-XM_017315005	XP_017170495	P37172	
9	7	human	9606	ACVR1	90		HGNC:171	102576	Chr2 q24.1	Chr2:157736444-NM_001111067	XP_005246996	Q04771	
10	9	mouse, laboratory mouse	10090	Sgca	20391	MGI:894698			Chr11 59.01 cM	Chr11:94962791-NM_009161	XP_011247138	P82350	
11	9	human	9606	SGCA	6442		HGNC:10805	600119	Chr17 q21.33	Chr17:50165975-NM_001135697	XP_011523424	Q16586	
12	12	mouse, laboratory mouse	10090	Adsl	11564	MGI:103202			Chr15 37.95 cM	Chr15:80948490-NM_009634	NP_033764	P54822	
13	12	human	9606	ADSL	158		HGNC:291	608222	Chr22 q13.1	Chr22:40346500-NM_000026	NM_001152828	P30566	
14	13	mouse, laboratory mouse	10090	Aga	11593	MGI:104873			Chr8 29.13 cM	Chr8:53511727-XM_134151	XP_001191983	Q64191	
15	13	human	9606	AGA	175		HGNC:318	613228	Chr4 q34.3	Chr4:177430774-NM_000027	NM_001165459	P20933	
16	14	mouse, laboratory mouse	10090	Agt	11606	MGI:87963			Chr8 72.81 cM	Chr8:124556587-NM_007428	NP_031454		
17	14	human	9606	AGT	183		HGNC:333	106150	Chr1 q42.2	Chr1:230702523-NM_000029	NP_000020	P01019	
18	16	mouse, laboratory mouse	10090	Alad	17025	MGI:96853			Chr4 33.17 cM	Chr4:62509169-NM_001276446	XP_006537702	P10518	
19	16	human	9606	ALAD	210		HGNC:395	125270	Chr9 q32	Chr9:113386312-NM_001317745	NP_001304674	P13716	
20	17	mouse, laboratory mouse	10090	Alas2	11656	MGI:87990			ChrX 68.46 cM	ChrX:15054737-NM_009653	NM_001095916	P08680	
21	17	human	9606	ALAS2	212		HGNC:397	301300	ChrX p11.21	ChrX:55009055-NM_001037968	XP_005262052	P22557	

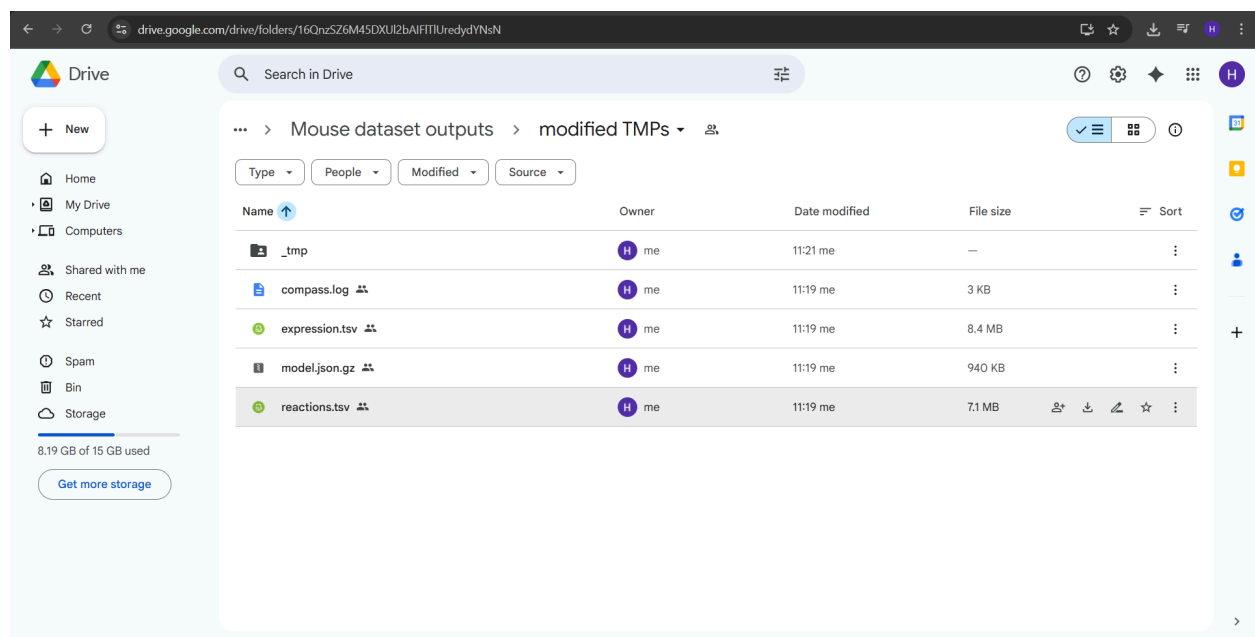
Mouse datasets

Mouse model	<i>FoxP3eGFP-CreERT2 ROSA26tdTomato ApoE^{-/-}</i>
Purpose	Lineage-tracing of FOXP3 ⁺ Tregs to identify exTregs
Tissue source	Spleen and lymph nodes

Sorted populations	- Tregs: tdTomato ⁺ eGFP ⁺ (active FOXP3) - exTregs: tdTomato ⁺ eGFP ⁻ (lost FOXP3)
Technology	Bulk RNA-seq (paired-end 50 bp reads)
Genome used	Mouse genome GRCm39
Read alignment	STAR v2.7.1 with --quantMode GeneCounts
Expression filtering	Genes with ≤ 100 total reads removed
DE analysis	DESeq2 with variance-stabilizing transformation (VST)
Use of mouse data	- Identify mouse exTreg signatures - Map mouse-human orthologs for SVM-based classification
Machine learning use	383 orthologous genes used to train an SVM model to classify exTregs vs Tregs
Public dataset link	GSE217010

To access all the output files, you can visit

https://drive.google.com/drive/folders/1nZYb_jlkt9dSkRPKXofbz1LEyb3iGUmk?usp=drive_link



reactions.tsv

	A	B	C	D	E	F	G	H	I	J	K	L
1	2DR1PP_pos	1	1	1	1	1	1	1	1	1	1	1
2	2HBO_pos	0.963141235	0.2977255018	0.4249690193	0.3082355901	0.303036646	0.2886007645	0.3088214429	0.3515411387	0.3133329645	0.5340120632	0.35695
3	2HBO_neg	0.963141235	0.2977255018	0.4249690193	0.3082355901	0.303036646	0.2886007645	0.3088214429	0.3515411387	0.3133329645	0.5340120632	0.35695
4	2HBi2_pos	0.2166234833	0.2384865637	0.2480108386	0.3039707343	0.1813124424	0.2029868569	0.3700824948	0.2040551455	0.2031693186	0.2355130125	0.28182
5	2HBi2_neg	0.2166234833	0.2384865637	0.2480108386	0.3039707343	0.1813124424	0.2029868569	0.3700824948	0.2040551455	0.2031693186	0.2355130125	0.28182
6	2HCO3_NaI_pos	0.5687554681	0.9143241881	1	0.9661078446	0.7849035087	0.8532036009	0.9115421278	0.7184356641	0.8931806488	0.9090394733	0.91066
7	2HCO3_NaI_neg	0.5687554681	0.9143241881	1	0.9661078446	0.7849035087	0.8532036009	0.9115421278	0.7184356641	0.8931806488	0.9090394733	0.91066
8	2MCIT_pos	1	1	1	1	1	1	1	1	1	1	1
9	2MCIT_neg	1	1	1	1	1	1	1	1	1	1	1
10	2OXOADOxm_r	1	1	1	1	1	1	1	1	1	1	1
11	2OXOADPTm_r	1	1	1	1	1	1	1	1	1	1	1
12	2OXOADPTm_n	1	1	1	1	1	1	1	1	1	1	1
13	34DHOXPEGO	0.4723419185	0.6875368548	0.4372382986	0.7425654387	0.5649408148	0.7376586534	0.6333924341	0.5844115211	0.7265554533	0.4753268159	
14	34DHOXPEGO	0.4723419185	0.6875368548	0.4372382986	0.7425654387	0.5649408148	0.7376586534	0.6333924341	0.5844115211	0.7265554533	0.4753268159	
15	34DHOXPEGL_r	1	1	1	1	1	1	1	1	1	1	1
16	11DOCRTSTRN	1	1	1	1	1	1	1	1	1	1	1
17	11DOCRTSTRN	1	1	1	1	1	1	1	1	1	1	1
18	11DOCRTSTRN	1	1	1	1	1	1	1	1	1	1	1
+ Sheet1												