A Cautionary Tale About Properly Vetting Datasets for Supervised Machine Learning Predicting Metabolic Pathway Involvement

Hunter N.B. Moseley, PhD

University of Kentucky

http://bioinformatics.cesb.uky.edu/

https://github.com/MoseleyBioinformaticsLab

Superfund Research Center

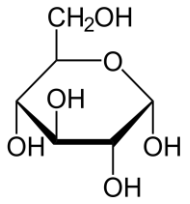
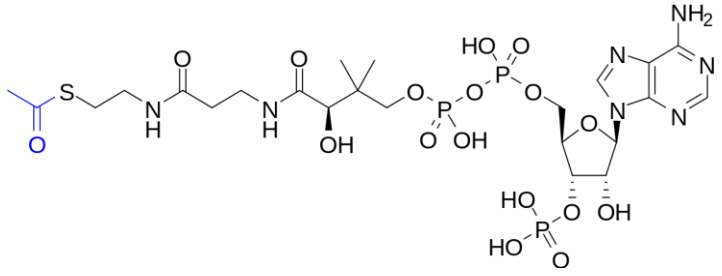
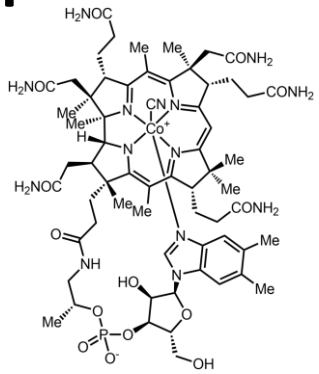
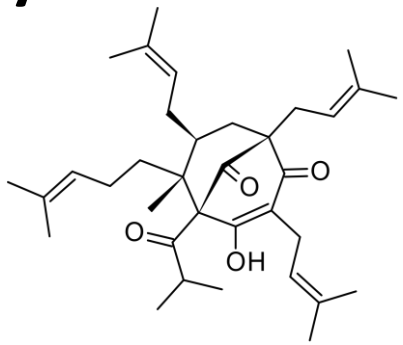
HealthCare MARKEY CANCER CENTER

Institute for Biomedical Informatics

Center for Clinical and Translational Science

What is metabolomics and why is it hard to analyze?

- **Metabolomics is the systematic detection and characterization of small biomolecules generated from metabolism that are present in a biological sample.**
- **In comparison to other omics, the detected biomolecules are very chemically diverse and hard to comprehensively detect.**

- **Current metabolic databases are quite incomplete.**
- **Detection by any single analytical method (nuclear magnetic resonance spectroscopy or mass spectrometry) is grossly incomplete.**
  - **Systematic analysis of metabolites is limited by metabolite detection, database completeness, and availability of standards for identification.**

Given the difficulty, why use metabolomics?

Metabolomics provides a culminating molecular phenotype representing a final product of gene regulation and expression.

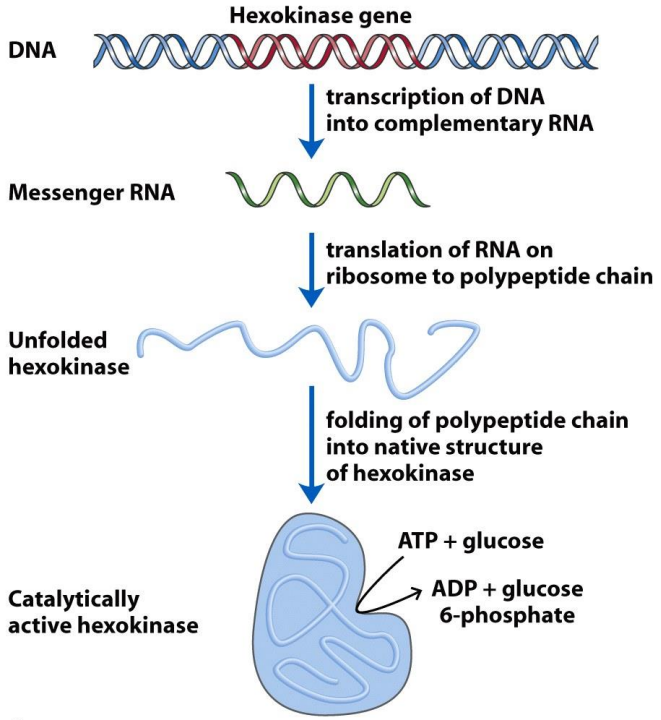


Figure 1-31
Lehninger Principles of Biochemistry, Fifth Edition

- **Allows a window into observing cellular and systemic metabolism.**
- **Changes in metabolism…**
  - **Reflect changes in cellular processes.**
  - **Typically occur on second and minute time scales.**
  - **Can be more easily achieved pharmacologically (via targeting enzymes).**
  - **Are a product of many disease processes.**
- **No model of a living system or process is complete without a metabolic component.**

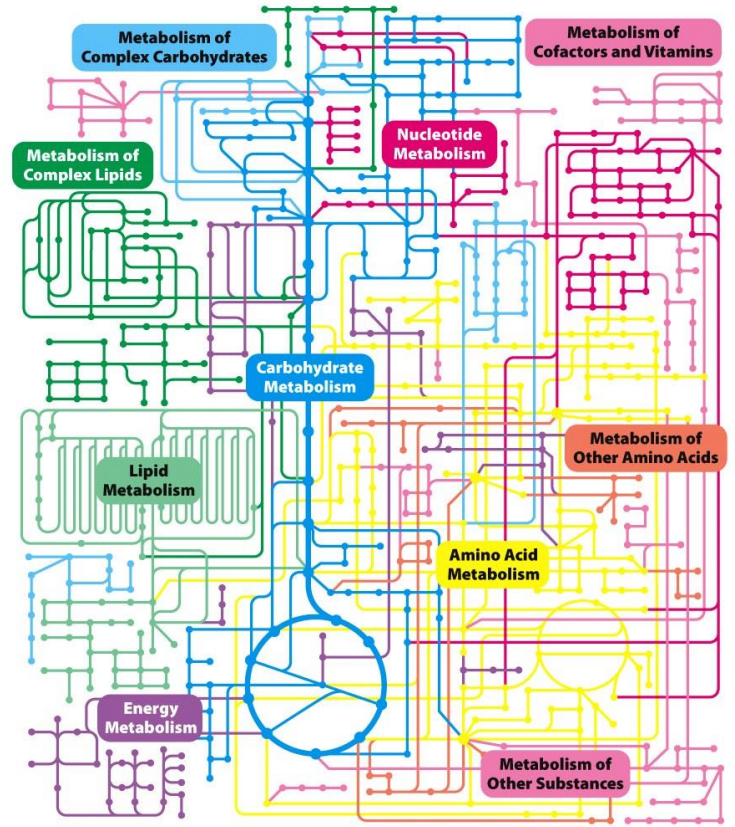
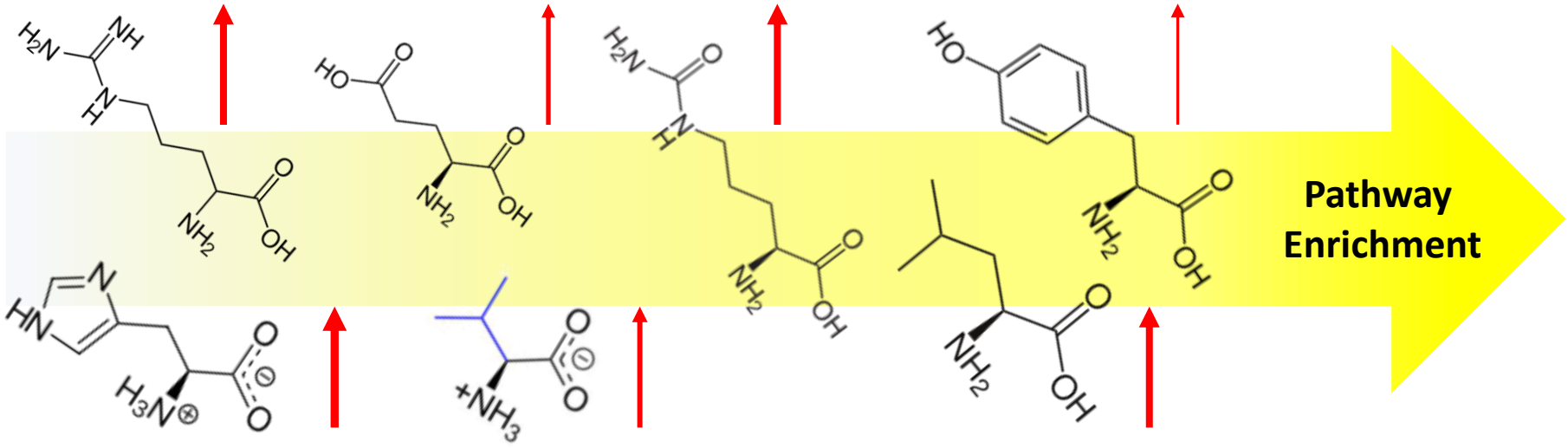
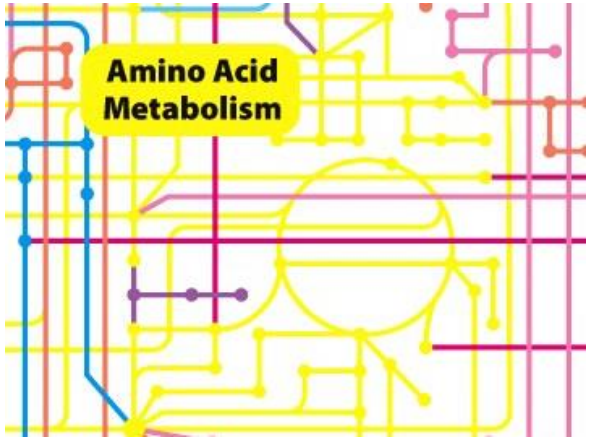


Figure 15.2
Biochemistry, Seventh Edition

Metabolome Mining is (Potentially) an Easier Approach.

- **"Metabolome mining is defined as the use of metabolite features, with chemical and other annotations, to derive metabolic information that is interpretable in a biological or biomedical context."**
  - https://www.mdpi.com/journal/metabolites/topical_collections/metabolome_mining
- **Identifying metabolites associated with specific metabolic pathways enables metabolic pathway enrichment analysis.**





But most metabolites detected in metabolomics experiments do not have metabolic pathway annotations!

Exploring Current State of the Art in Metabolic Pathway Involvement Prediction

Model / Feature Set	Accuracy (%)	Precision (%)	Recall (%)	F1
Hu et al. RF [1]	94.64	77.97	67.83	0.7254
Baranwal et al. GCN/RF [2]	97.58 ± .12	83.69 ± .78	83.63 ± .68	0.8366
Baranwal et al. GCN [2]	97.61 ± .12	91.61 ± .52	92.50 ± .44	0.9205
Yang et al. GAT [3]	97.50 ± .06	93.04 ± .28	93.22 ± .16	0.9313
Du et al. MLGL-MP [4]	98.64 ± 0.47	95.26 ± 2.25	94.21 ± 1.94	0.9473

Standard deviation of the model performance metrics across CV folds indicated by the ± symbol, if available from the publication.
RF – Random Forest; GCN – Graph Convolutional Network; GAT – Graph Attention Network;
MLGL-MP - Multi-Label Graph Learning framework enhanced by pathway interdependence for Metabolic Pathway prediction

[1] Hu L-L, Chen C, Huang T, Cai Y-D, Chou K-C. *PLoS ONE*. 2011 Dec 29;6(12):e29491.
[2] Baranwal M, Magner A, Elvati P, Saldinger J, Violi A, Hero AO. *Bioinformatics*. 2020 Apr 15;36(8):2547–53.
[3] Yang Z, Liu J, Wang Z, Wang Y, Feng J. *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE; 2020. p. 126–31.
[4] Du B-X, Zhao P-C, Zhu B, Yiu S-M, Nyamabo AK, Yu H, et al. *Bioinformatics*. 2022 Jun 24;38(Suppl 1):i325–32.

All of these methods used a Kyoto Encyclopedia of Gene and Genomes (KEGG) derived dataset with SMILES chemical structure representations (KEGG-SMILES dataset).

KEGG-SMILES Dataset(s) Used

Model / Feature Set	Data available	Code available	Dataset Size	Publication Date
Hu et al. RF [1]	No	No	3,137	December 2011
Baranwal et al. GCN/RF [2]	Yes	Yes	6,669*	April 2020
Baranwal et al. GCN [2]	Yes	Yes	6,669*	April 2020
Yang et al. GAT [3]	No	No	6,669*	December 2020
Du et al. MLGL-MP [4]	Yes	Yes	6,648*	June 2022

*Publications using the dataset originating with Baranwal et al.

Data Leakage Problem in Baranwal KEGG-SMILES Dataset

Label ID	Pathway Category	Number Of Compounds In Dataset (Original)	Fraction Of Dataset (Original)	Percentage Of Duplicates	Number Of Compounds In Dataset (De-duplicated)	Fraction Of Dataset (De-duplicated)
0	Carbohydrate metabolism	1126	0.169	67.05	371	0.075
1	Energy metabolism	750	0.113	72.80	204	0.041
2	Lipid metabolism	1066	0.16	38.93	651	0.132
3	Nucleotide metabolism	342	0.051	49.12	174	0.035
4	Amino acid metabolism	1440	0.217	54.37	657	0.133
5	Metabolism of other amino acids	597	0.09	59.80	240	0.049
6	Glycan biosynthesis and metabolism	325	0.049	64.00	117	0.024
7	Metabolism of cofactors and vitamins	948	0.143	44.83	523	0.106
8	Metabolism of terpenoids and polyketides	1483	0.223	35.13	962	0.195
9	Biosynthesis of other secondary metabolites	1906	0.287	35.78	1224	0.248
10	Xenobiotics biodegradation and metabolism	1452	0.218	32.58	979	0.199
N/A	Total Dataset	6,648	N/A	25.86	4,929	N/A

Over 25% of the dataset are complete duplicates! This creates a catastrophic data leakage problem for training!

The Good, the Bad, and the Ugly!

The Bad

- A catastrophic data leakage was created within the Baranwal KEGG-SMILES dataset.

The Ugly

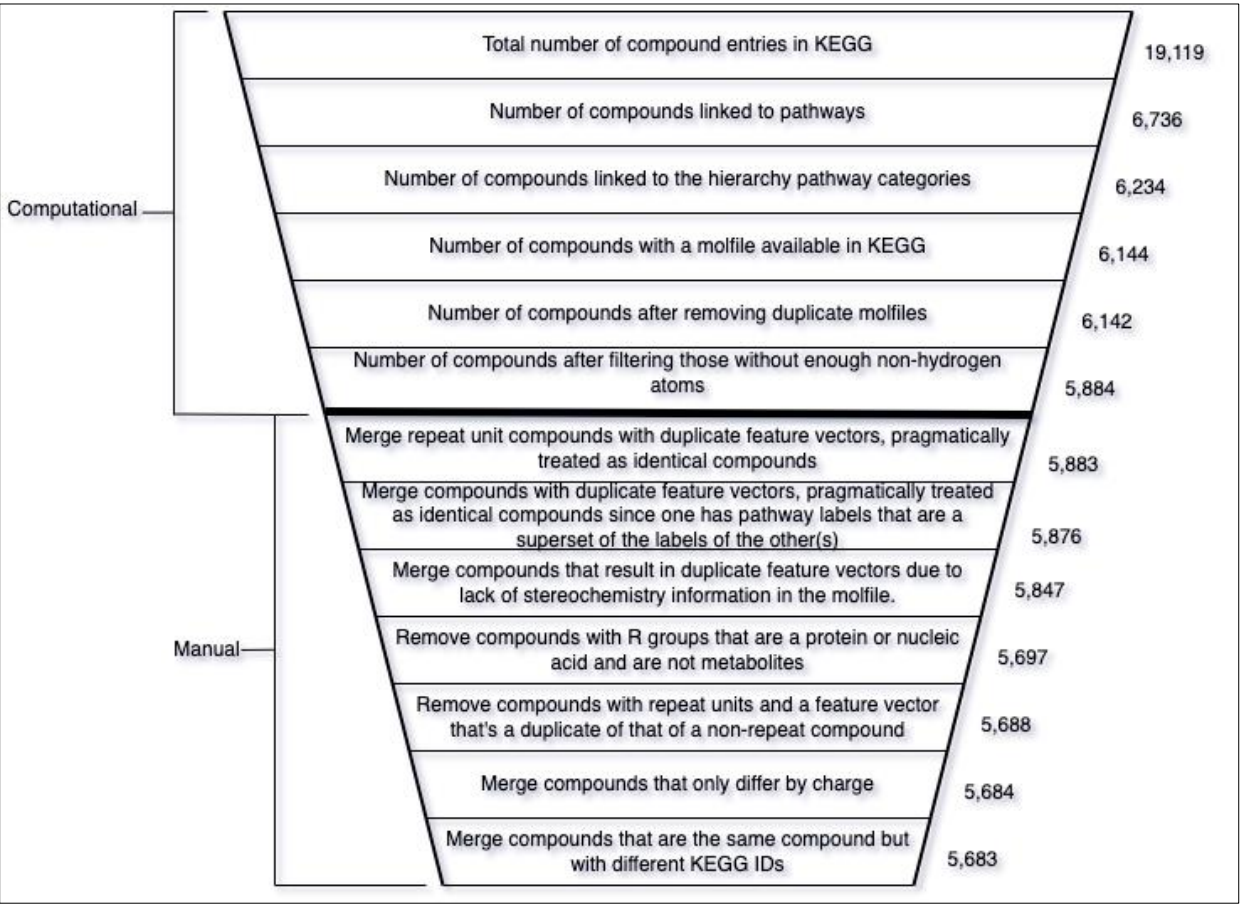
- This dataset affected at least 3 publications in highly reputable journals and conferences, since none of the authors properly vetted the dataset.

The Good (Silver Lining)

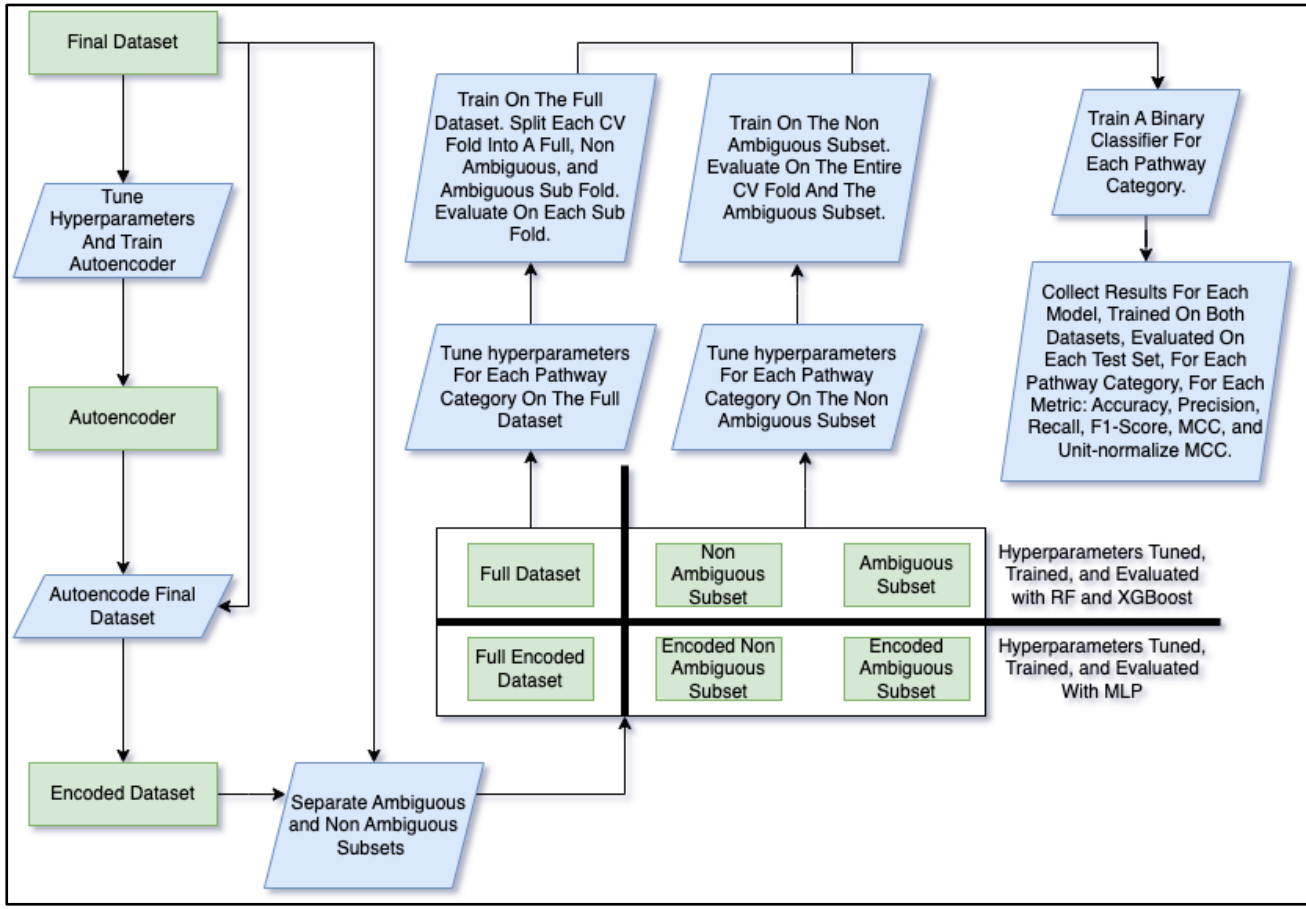
- Baranwal et al and Du et al followed many best practices for scientific reproducibility in computational research, enabling the detection of this catastrophically-flawed dataset and highly flawed results.
- These analyses are available in the following preprint and are under review:
  - Erik D. Huckvale and Hunter N.B. Moseley. "A cautionary tale about properly vetting datasets used in supervised learning predicting metabolic pathway involvement" bioRxiv 2023.10.03.560711 (2023).
- These findings prompted us to create a new benchmark dataset for metabolic pathway involvement prediction, which was recently published:
  - Erik D. Huckvale, Christian D. Powell, Huan Jin, and Hunter N.B. Moseley. "Benchmark dataset for training machine learning models to predict the pathway involvement of metabolites" ***Metabolites*** 13, 1120 (2023).

New Benchmark Dataset for Metabolic Pathway Involvement Prediction

Dataset Creation Workflow



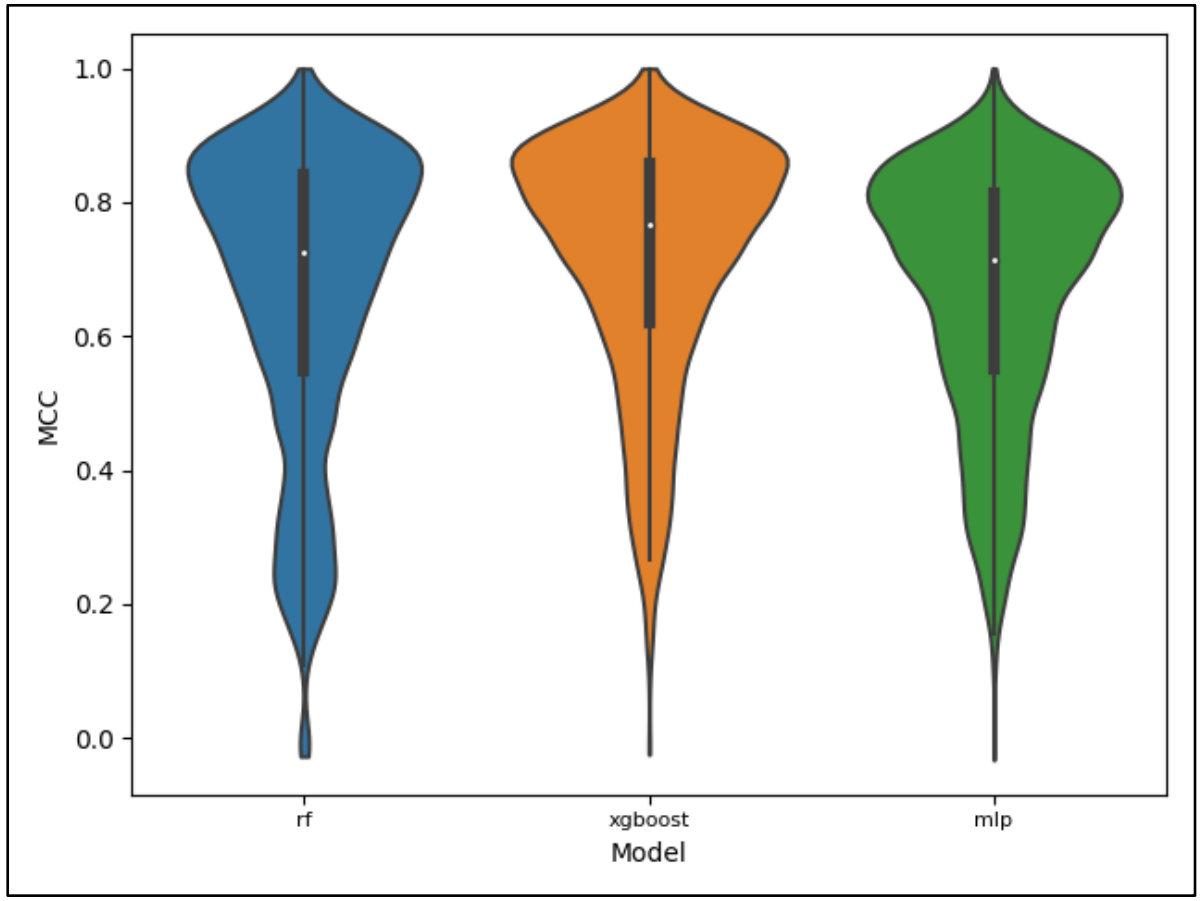
Machine Learning Workflow



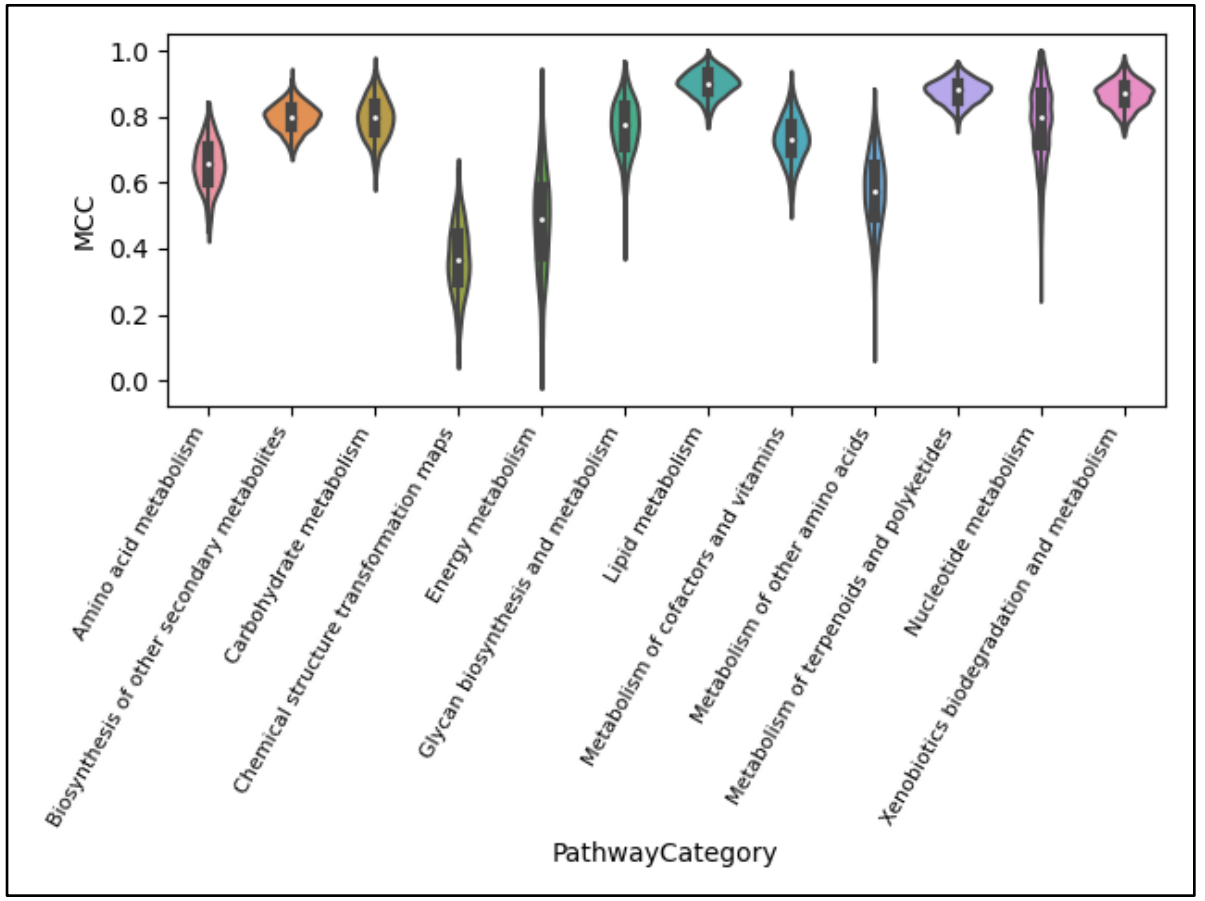
Erik D. Huckvale, Christian D. Powell, Huan Jin, and Hunter N.B. Moseley. "Benchmark dataset for training machine learning models to predict the pathway involvement of metabolites" ***Metabolites*** 13, 1120 (2023).

New Benchmark Dataset for Metabolic Pathway Involvement Prediction

Overall Model Performance Comparison



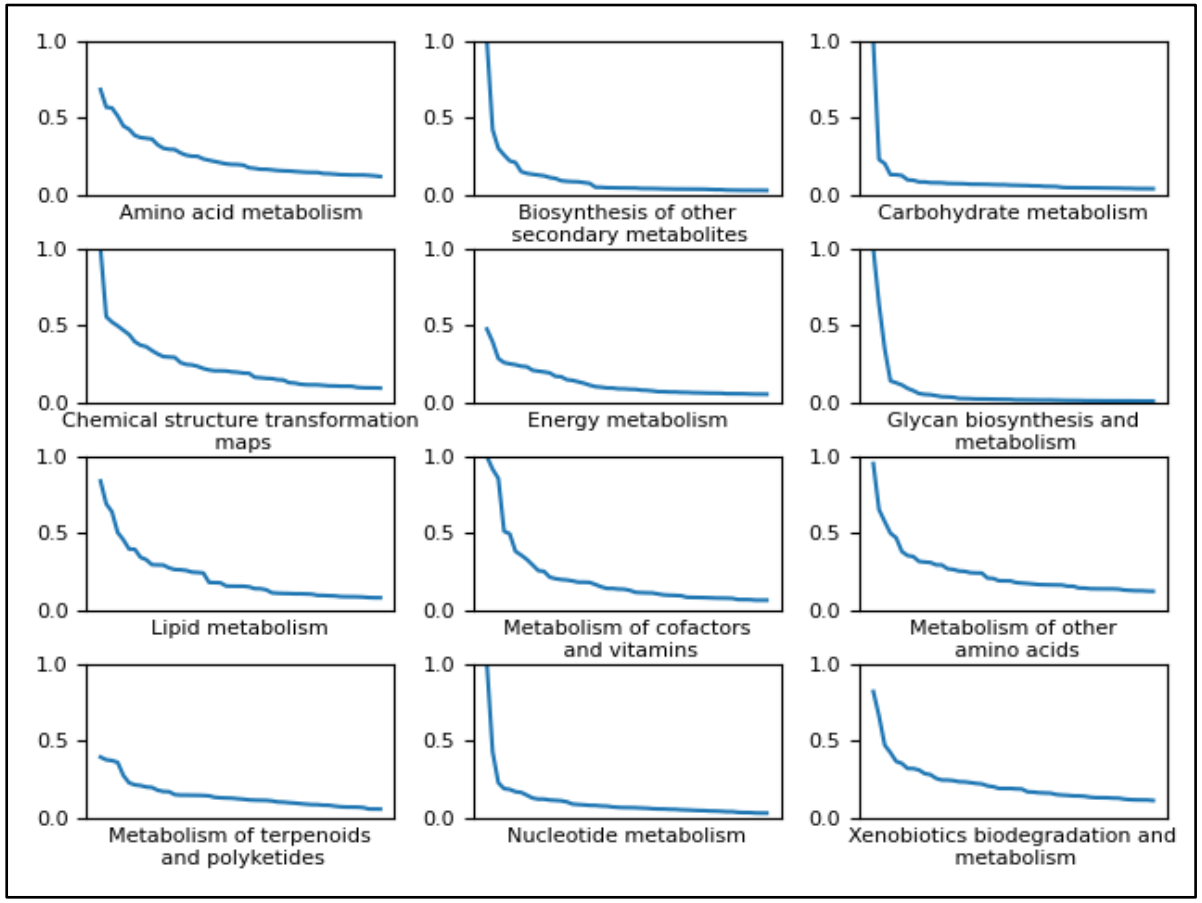
Pathway-Specific XGBoost Performance



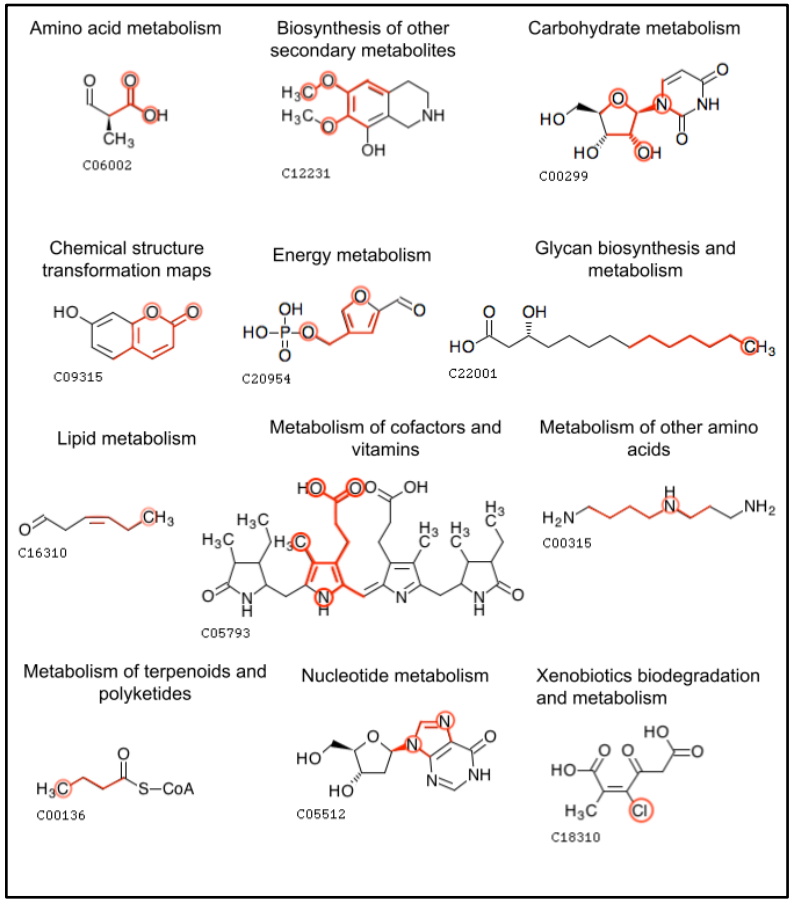
Erik D. Huckvale, Christian D. Powell, Huan Jin, and Hunter N.B. Moseley. "Benchmark dataset for training machine learning models to predict the pathway involvement of metabolites" ***Metabolites*** 13, 1120 (2023).

New Benchmark Dataset for Metabolic Pathway Involvement Prediction

XGBoost Feature Importance Per Pathway

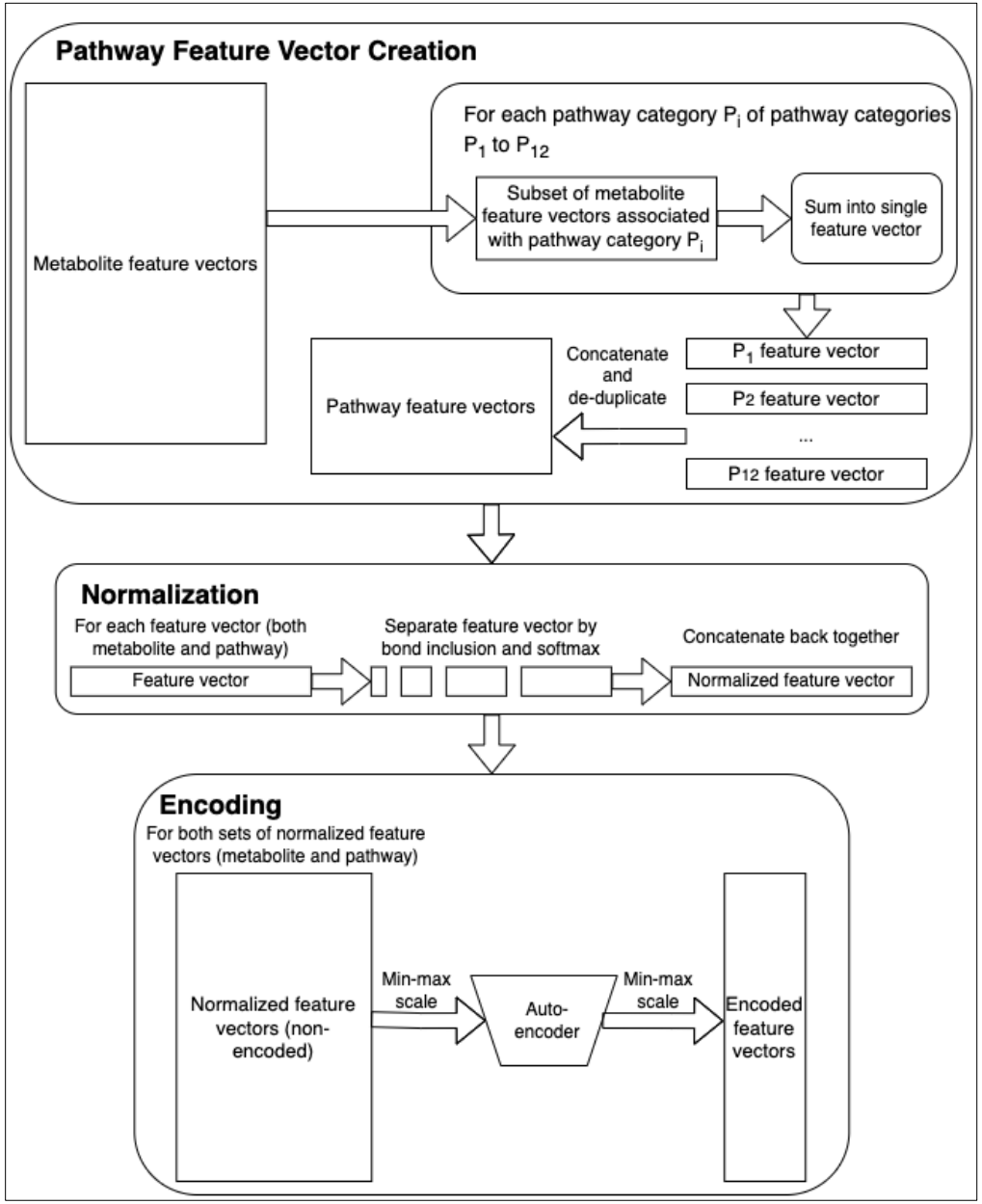


Most Important Pathway-Specific Feature

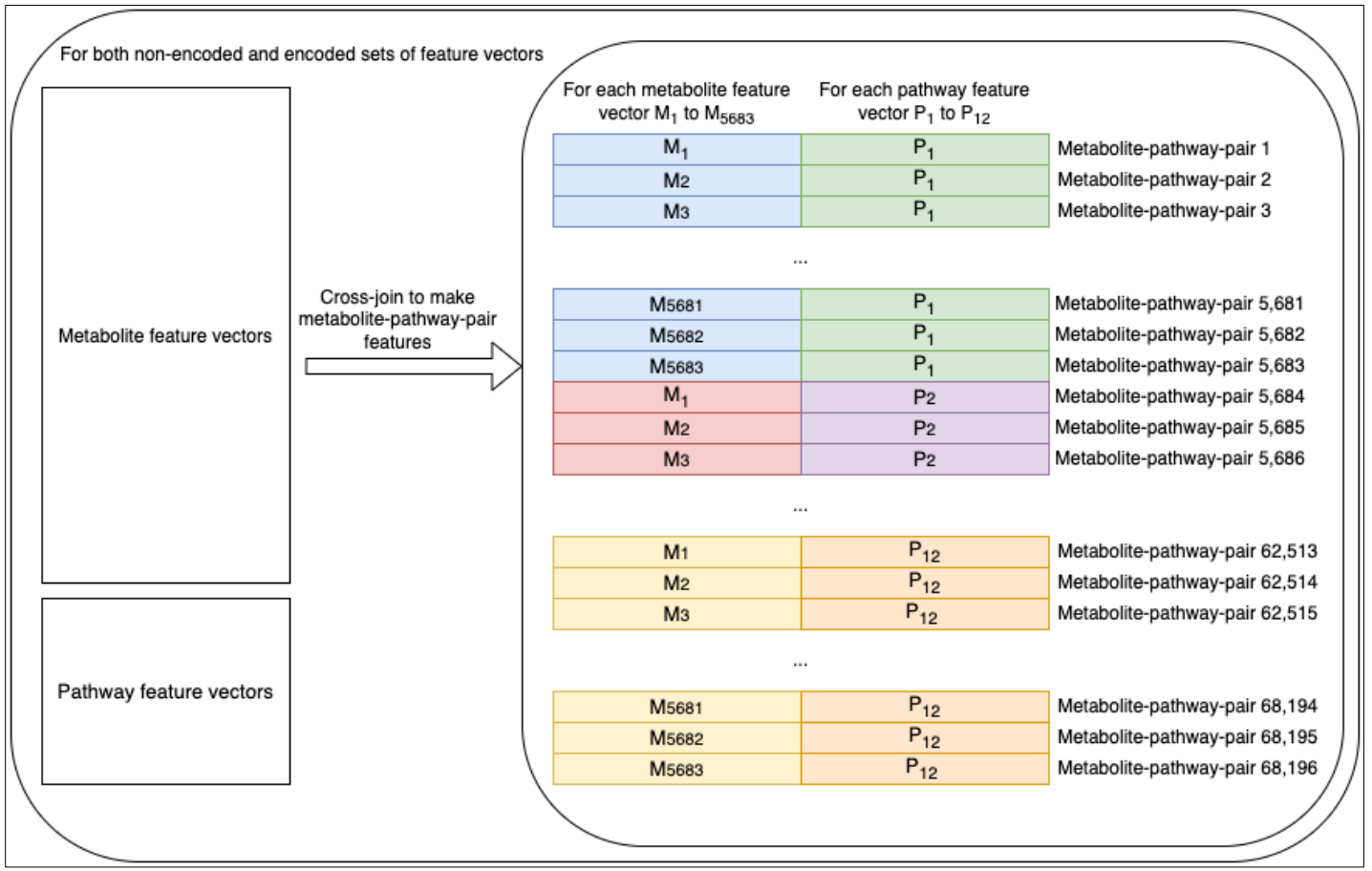


Erik D. Huckvale, Christian D. Powell, Huan Jin, and Hunter N.B. Moseley. "Benchmark dataset for training machine learning models to predict the pathway involvement of metabolites" ***Metabolites*** 13, 1120 (2023).

Novel Metabolite-Pathway Paired Feature Model

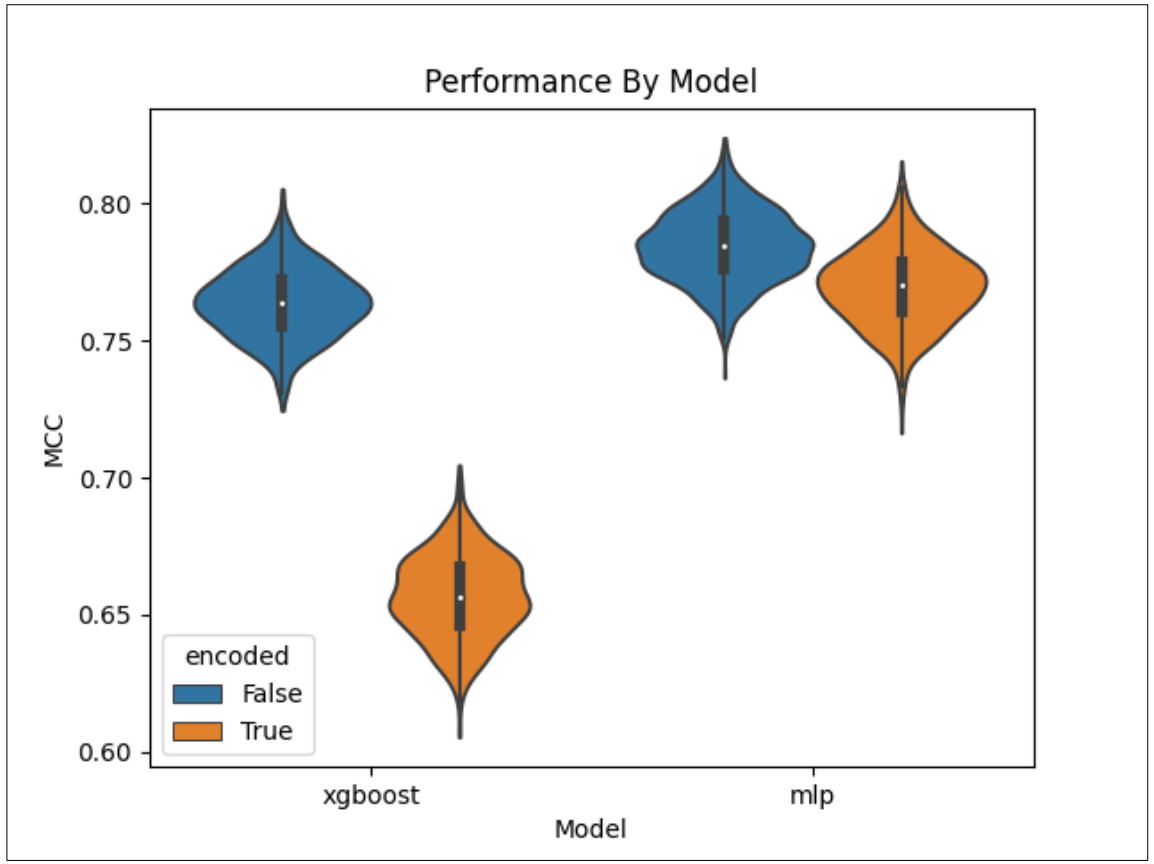


Metabolite-Pathway Feature Fusion

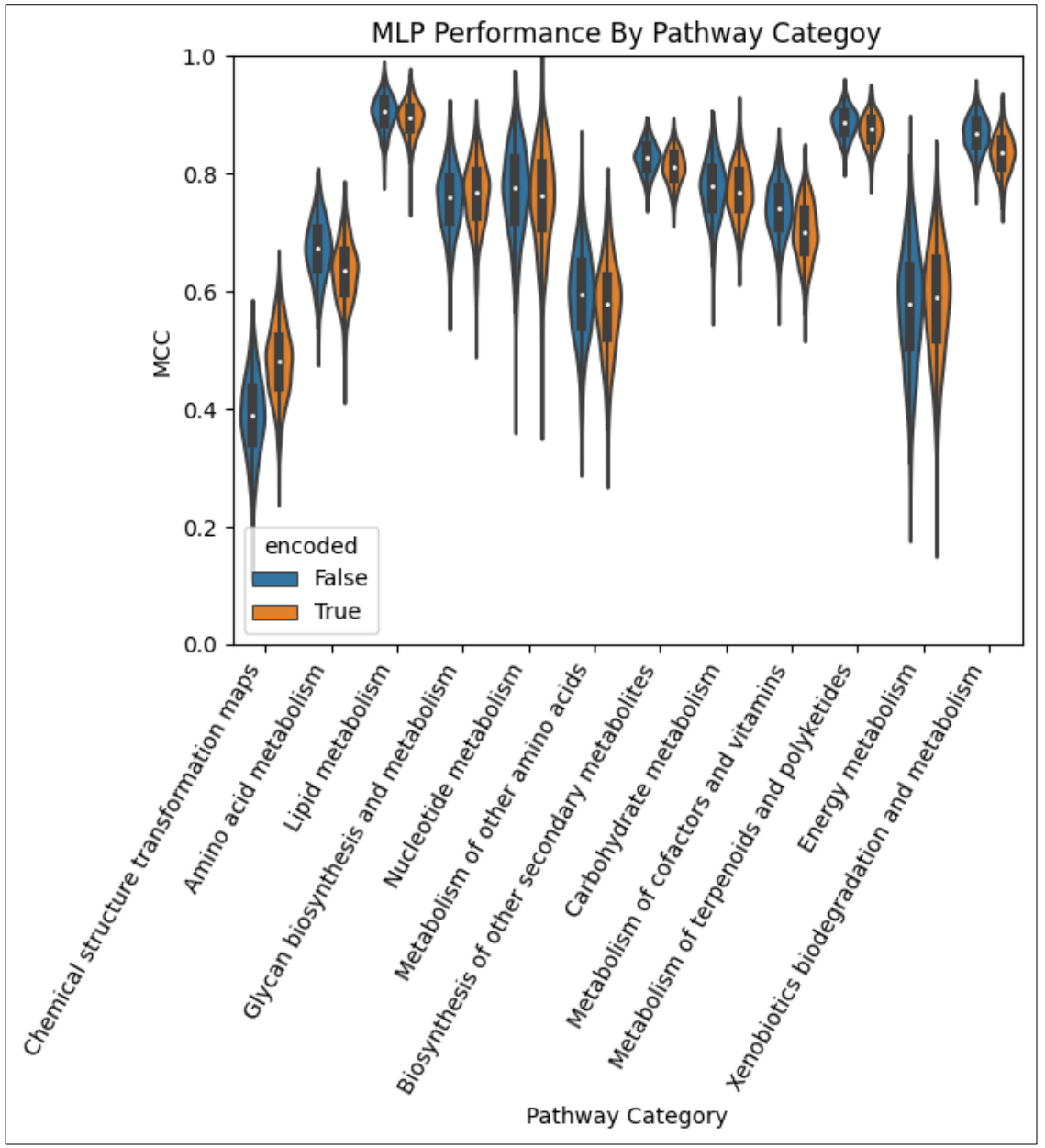


Erik D. Huckvale and Hunter N.B. Moseley. "Predicting The Pathway Involvement Of Metabolites Based on Combined Metabolite and Pathway Features" *bioRxiv* 2024.04.01.587582 (2024).

Performance of Metabolite-Pathway Paired Model



- **0.013 Mathews Correlation Coefficient (MCC) standard deviation across 1000 CV folds is an order of magnitude improvement in robustness over the previous models.**



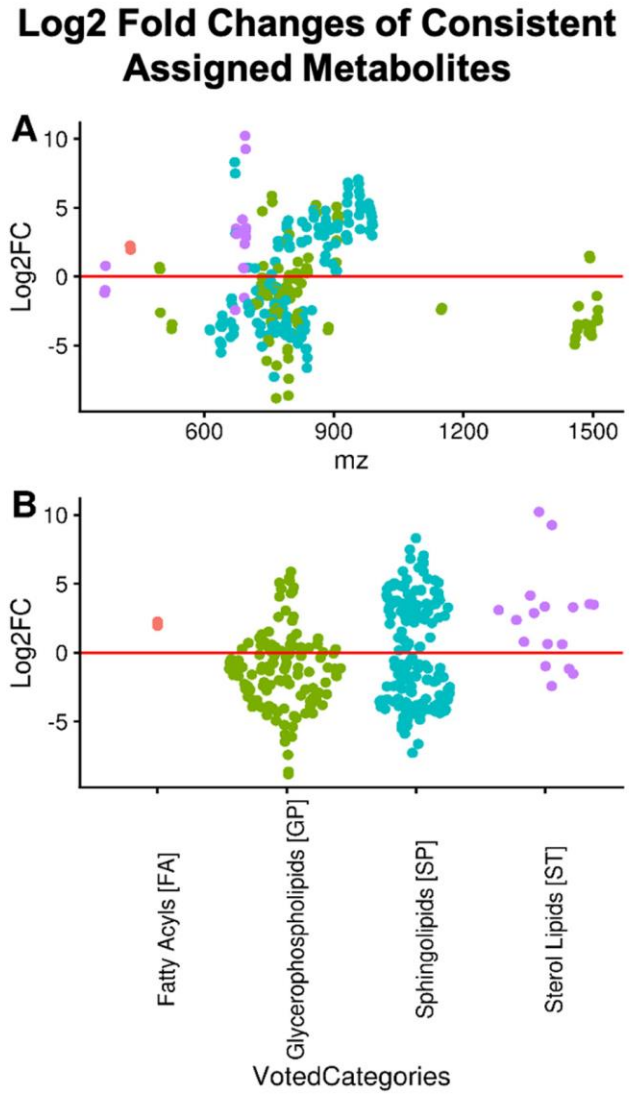
Untargeted lipidomics of non-small cell lung carcinoma demonstrates differentially abundant lipid classes in cancer vs non-cancer tissue

Joshua M. Mitchell, Robert M. Flight, and Hunter N.B. Moseley.

Metabolites 11, 740 (2021).

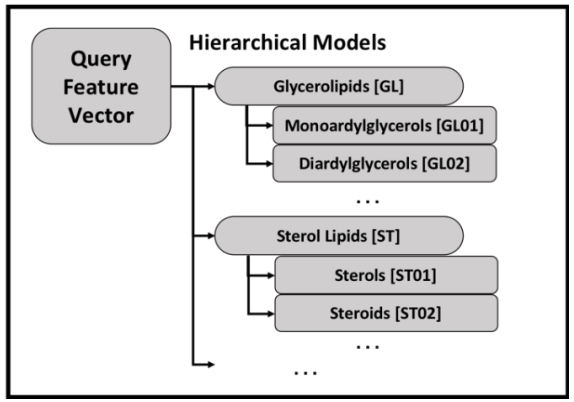
- Most untargeted approach to metabolomics which derives molecular formula from Fourier transform mass spectra using SMIRFE (US patent 10,607,723 B2).
- Resulting molecular formulas were classified into lipid categories and classes using a hierarchical set of Random Forest binary classifiers.
- High abundances of sterol esters were observed in NSCLC tissue, suggesting altered SCD1 or ACAT1 activity.
- Low abundances of cardiolipins were observed, suggesting altered human cardiolipin synthase 1 or lysocardiolipin acyltransferase activity which is known to confer apoptotic resistance.

Category	Total	More-Abundant Features			Less-Abundant Features		
		Expected	Observed	p-adjust	Expected	Observed	p-adjust
Fatty Acyls [FA]	12	2.989	2	1	3.947	0	1
Glycerophospholipids [GP]	205	51.055	37	1	67.424	88	**0.00503**
Prenol Lipids [PR]	5	1.245	0	1	1.644	0	1
Sphingolipids [SP]	281	69.983	79	0.09861	92.420	81	1
Sphingolipids [SP] – Low M/Z	33	8.219	3	1	10.854	16	0.141
Sphingolipids [SP] – High M/Z	248	61.764	76	**0.00967**	81.567	65	1
Sterol Lipids [ST]	23	5.728	13	**0.00643**	7.084	3	1



Machine learning methods are expanding the applications of annotation enrichment analysis.

- Predicts lipid category and class from features based on molecular formula using a hierarchical set of binary Random Forest classifiers.
- Uses LipidMaps database of known and theoretical lipids and the Human Metabolome Database for non-lipid examples.



Palmitic Acid $C_{16}H_{32}O_2$

1. Calculate Theoretical Monoisotopic Mass

Monoisotopic Mass = $AC_C * M_{12C} + AC_H * M_{1H} + AC_O * M_{16O} + N_N * M_{14N}$
Monoisotopic Mass = 16 * 12 + 32 * 1.0078250321 + 2 * 15.9949146196
Monoisotopic Mass = 256.240230266

2. Calculate Unsaturation

Unsaturation = 4*#C + 3*#N + 2*#O + 6*#P + 6*#S – (#H + #X) – 2*(#C+#N+#O+#P+#S-1)
Unsaturation = 4*16+2*2 – (32) – 2*(16+2-1)
Unsaturation = 2

3. Build Feature Vector

Feature = <mass, tens, ones, tenths, unsaturation, #X, #H+#X, #C, #N, #O, #P, #S, #H>
Feature = <**256.240230266**, 5, 6, 2, 2, 0, 32+0, 16, 0, 2, 0, 0, 32>
Feature = <**256.240230266**, 5, 6, 2, 2, 0, 32, 16, 0, 2, 0, 0, 32>

- Average accuracy >90%
- Average precision >83%

This is good enough for annotation enrichment analysis!

LMSD + LMISSD + HMDB_non_Lipid Model Performance (Category)

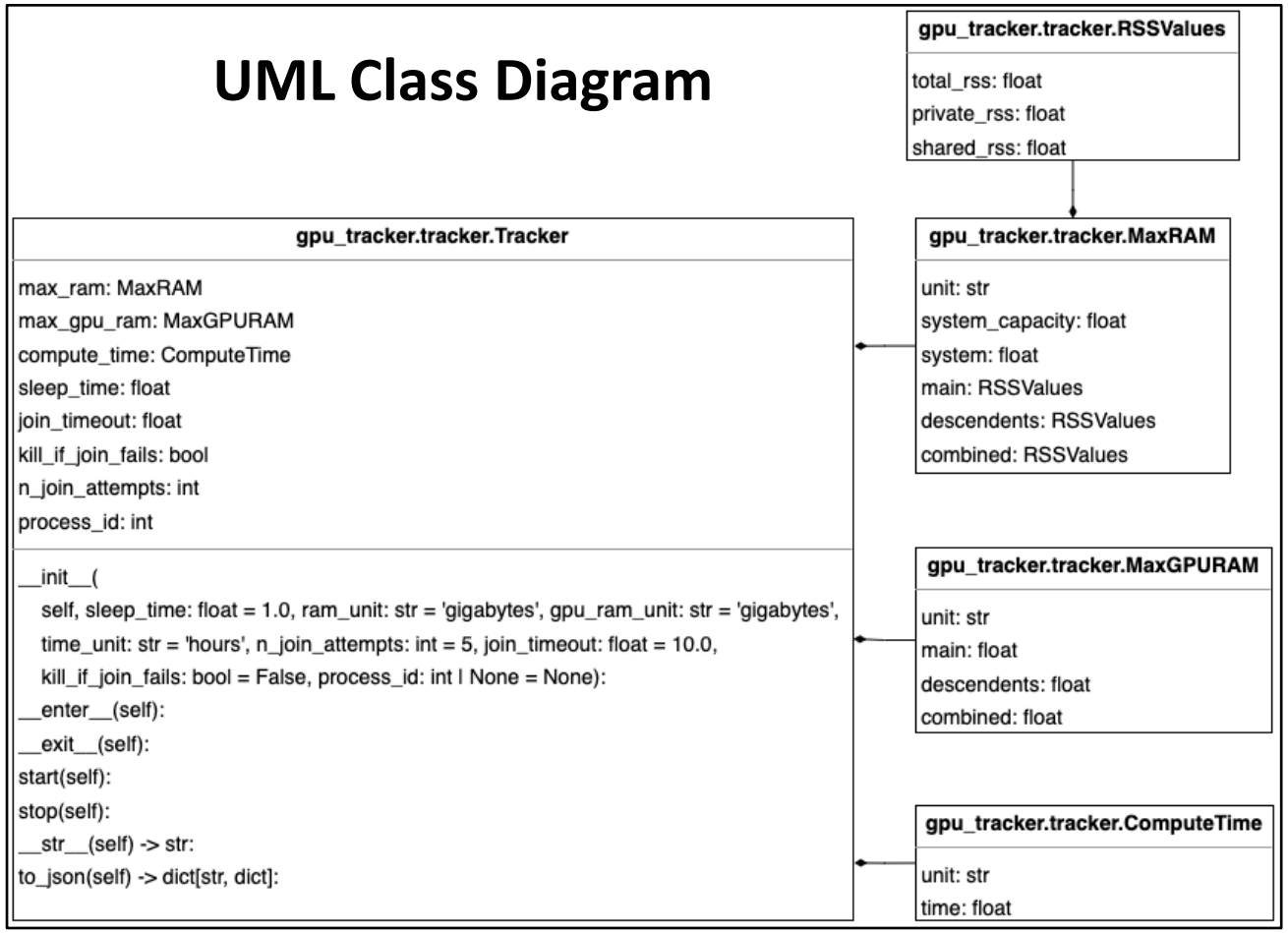
Category	Precision	Out-of-Bag Accuracy	Number of Entries	True Positives	False Positives
Fatty Acyls [FA]	0.837	0.939	2031	1659	322
Glycerolipids [GL]	0.995	0.993	2715	2696	14
Glycerophospholipids [GP]	0.979	0.979	9766	9706	206
Polyketides [PK]	0.768	0.933	1376	979	295
Prenol Lipids [PR]	0.985	0.983	473	259	4
Saccharolipids [SL]	1.000	0.998	102	99	0
Sphingolipids [SP]	0.976	0.976	3089	2875	72
Sterol Lipids [ST]	0.935	0.983	824	702	49
not_lipid	0.928	0.882	7587	6845	532

Mitchell et al., "Deriving Accurate Lipid Classification based on Molecular Formula" *Metabolites* 10, 122 (2020). https://doi.org/10.3390/metabo10030122

gpu_tracker: Python package for tracking and profiling GPU utilization in both desktop and high-performance computing environments

- Has both an API and CLI for profiling single and multiprocessing tasks that utilize GPUs.
- Provides both max RAM and GPU RAM utilization of a given computational task.
- Designed for Linux, but works on Windows and MacOS.
- CLI Example:

```
% gpu-tracker -e 'bash example-script.sh' --tu=seconds --gru=megabytes --ru=megabytes
Resource tracking complete. Process completed with status code: 0
Max RAM:
  Unit: megabytes
  System capacity: 67254.166
  System: 2458.182
  Main:
    Total RSS: 3.072
    Private RSS: 0.373
    Shared RSS: 2.699
  Descendents:
    Total RSS: 830.271
    Private RSS: 708.19
    Shared RSS: 122.081
  Combined:
    Total RSS: 831.537
    Private RSS: 708.563
    Shared RSS: 122.974
Max GPU RAM:
  Unit: megabytes
  Main: 0.0
  Descendents: 314.0
  Combined: 314.0
Compute time:
  Unit: seconds
  Time: 3.316
```



Erik D. Huckvale and Hunter N.B. Moseley. "gpu_tracker: Python package for tracking and profiling GPU utilization in both desktop and high-performance computing environments" *arXiv* arXiv:2404.01473 (2024).

Conclusions and Future Goals

- **We have developed a high-quality benchmark dataset for building metabolic pathway prediction models.**
- **We have developed the most robust models so far for metabolic pathway prediction.**
- **We are confident that achieving an average MCC > 0.9 will produce predicted metabolic pathway annotations useful for pathway enrichment analysis.**
- **Along the way we created a useful tool for profiling GPU utilization of HPC jobs, especially involving machine learning models.**

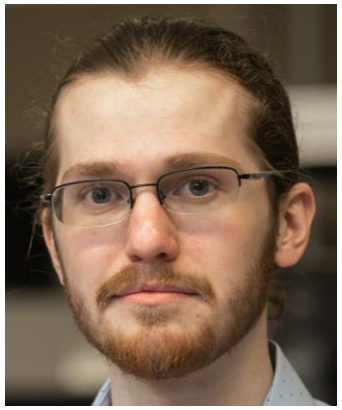
Acknowledgements

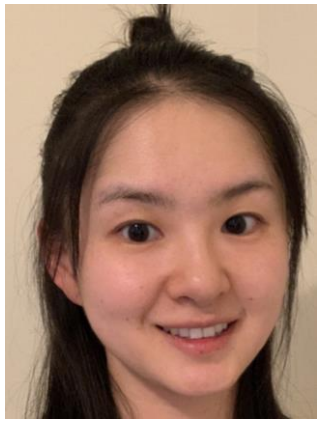
- Mr. Erik Huckvale did the majority of the KEGG-SMILES dataset evaluation and creation of a new benchmark dataset.
- Mr. Christian Powell started our metabolic pathway prediction project.
- Dr. Huan Jin provided atom coloring methodology used in the metabolic pathway prediction based on prior work done by Dr. Joshua Mitchell.
- Dr. Robert Flight provided feedback on the dataset evaluations.
- This work was supported by funding from:
  - NSF 2020026 (PI Moseley)
  - NIH NIEHS P42 ES007380 (UKSRC, PD Pennell)


Hunter Moseley

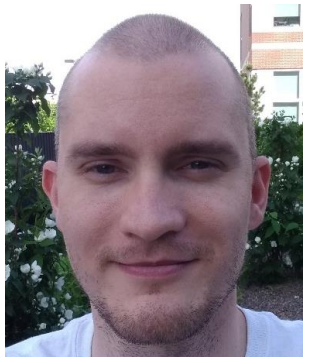

Robert Flight


Joshua Mitchell


Christian Powell


Huan Jin


Andrew Smelter


Travis Thompson


Erik Huckvale

kegg_pull: a Software Package for the RESTful Access and Pulling from KEGG

- **Promote the FAIR (Findable, Accessible, Interoperable, and Reusable) guiding principles of data stewardship with respect to KEGG.**
  - **Improves on the accessibility, interoperability, and reusability of the KEGG API.**
- **Makes the utilities of the KEGG API accessible to Python programmers through an application programming interface (API).**
- **Makes these utilities accessible to command line users either for shellbash scripting or for executing one-time commands without needing to write any script at all.**

```
Usage:
    kegg_pull -h | --help       Show this help message.
    kegg_pull -v | --version    Displays the package version.
    kegg_pull --full-help       Show the help message of all sub commands.
    kegg_pull pull ...          Pull, separate, and store an arbitrary number of KEGG entries to the local file system.
    kegg_pull entry-ids ...     Obtain a list of KEGG entry IDs.
    kegg_pull rest ...          Executes one of the KEGG REST API operations.
```

Erik Huckvale and Hunter N.B. Moseley. "kegg_pull: a Software Package for the RESTful Access and Pulling from The Kyoto Encyclopedia of Gene and Genomes" ***BMC Bioinformatics*** 24, 78 (2023).

kegg_pull Improvements Over Prior Packages

1. **Provides a command line interface.**
2. **Provides sleep and pull redundancy to handle KEGG's blacklisting.**

Sleep Time (seconds)	0.0	0.5	1.0	2.0	3.0	5.0	10.0
Percent Success	94.68	94.89	96.78	99.78	100.0	100.0	100.0
Pull Time (minutes)	12.99	16.03	14.69	10.82	8.51	8.44	8.7

Number of minutes spent attempting to pull all the entries in the KO KEGG database. Percent success is the percentage of the entries in the KO database that were successfully pulled while the others failed. Difference in pull time and percent success varies by the --sleep-time option on the kegg_pull CLI. All other options remained the same, including the use of multiprocessing. Values were collected on a 12 core (hyperthreaded) machine using 12 processes.

3. **Provides multiprocessing and URL construction speed improvements.**

Database Name	Multi-process Pull Time	Single Process Pull Time	Number Of Entries
Pathway	0.1	1.05	558
Compound	6.4	73.62	19,004
KO	8.32	74.0	25,458

The amount of time to pull and save all the entries of a given database on a single process (one core) compared to pulling across multiple processes (multiple cores). The above values result from running kegg_pull on a 12 hyper-threaded core machine using 12 processes for multiprocessing and one process for single-processing. The sleep time and all other options for each were also constant. Files were saved in a regular directory.

Database Name	One Entry At A Time				Ten Entries At A Time			
	Sleep Time 5 Seconds		Sleep Time 20 Seconds		Sleep Time 5 Seconds		Sleep Time 20 Seconds	
	Percent Success	Pull Time	Percent Success	Pull Time	Percent Success	Pull Time	Percent Success	Pull Time
Module	98.69	1.34	100.0	1.59	100.0	0.06	100.0	0.07
Pathway	98.75	1.6	100.0	1.67	100.0	0.11	100.0	0.09
Compound	99.24	63.62	100.0	61.36	100.0	6.77	100.0	7.21
KO	99.39	83.91	100.0	90.4	100.0	8.42	100.0	9.22

The amount of time (minutes) to pull all the entries from a given database and the success percentage when pulling one entry at a time (with the --force-single-entry flag set) compared to pulling ten entries (maximum allowed by KEGG) per request. Each of these are compared to a lower sleep time vs. a higher sleep time. Results were collected on a 12 (hyperthreaded) core machine on 12 processes with all other options consistent.