

Tropism Analysis Package: Interactive Machine Learning Software to Identify Viral Host Factors Through Single-Cell Host-Virus mRNA Profiling

Authors

Kenny Pavan^{1,2}, Lamy Ben Ameer¹, Zach Goode¹, Emily Tiedemann¹, Elizabeth Kolb¹, Arpiar Saunders^{1#}

Affiliations

1. Vollum Institute, Oregon Health & Science University, Portland, OR, 97239, USA

2. Biomedical Engineering Graduate Program, Oregon Health and Science University, Portland, OR, 97239, USA

Correspondence: saundear@ohsu.edu

Methods

3.1 Package Software Structure

Developed as an object-oriented Python pip installable package, the TAP class implements a recursive categorical parsing technique to read the standard AnnData X layer cell by gene expression along with user-defined *obs* layer categories. The class can be instantiated with a variety of parameters all documented on the corresponding Github (url) to tune functionality of the html output containing the interactiveUI. The initial default steps include: data validation, conversion to raw expression counts, filtering of cells with low expression, then normalizing and log transforming the dataset, achieved using scanpy preprocessing functionality. Once completed, the dataset can utilize automated cell type calling using the included Cell Typist package or use manual cell type definitions provided in the AnnData object. All default Cell Typist models are included and can be accessed by using the runCell Typist parameter when the TAP class is instantiated.

Once categorical cell identifiers (cell type, replicates, etc...) are defined, all possible subsets of viral expression counts and categorical observations are then processed through the pipeline described in **Figure 1B** and stored for $O(1)$ access by the user interface. TAP considers the single-cell AnnData input as the following:

$$X \in \mathbb{R}^{n \times p}$$

Where n is the number of cells and p is the number of genes. It is expected that each cell i provided as input contains metadata, represented as m for categorical observations (e.g. cell type, replicates, etc) and C for containing all possible combinations of categories (Vector, Category, Vector + Category, Vector + Replicate + Category, etc) belonging to $C^{(i)}$ as defined below.

$$m_i = (m_i^{(1)}, m_i^{(2)}, \dots, m_i^{(k)}), m_i^{(j)} \in C^{(j)}$$

The TAP software then recursively iterates each unique categorical combination and performs threshold filtering to confirm non-empty subsets of data described as:

$$X_c = \{ x_i \in X \mid \forall j, m_i^{(j)} = c^{(j)} \} \text{ for each } c \in C$$

Where $c = (c^{(1)}, \dots, c^{(k)})$ is a unique categorical combination of a the Cartesian product of all selected metadata variables $C = (c^{(1)} \times \dots \times c^{(k)})$ and X_c is the subset of cells matching that particular combination. Each X_c subset is then passed through the highly configurable TAP analysis pipeline (**Figure 1B**) consisting of data anomaly validations, preprocessing, outlier removal, and infectivity clustering leading to important features for each X_c classification.

3.2 Infectivity Calling of Each Categorical Subset

TAP treats each categorical subset (X_c) of cells as an individual dataset and classification problem to be processed by its algorithmic ensemble pipeline. The first step is to frame each subset (X_c) into a classification problem by clustering infectivity, yielding binary labels: infected vs. uninfected. All infectivity calls consider only viral expression counts and total expression counts. TAP supports three infectivity calling strategies: (1) Simplistic infectivity thresholding accepts a scalar parameter that will denote all cells greater than that value will be considered infected. (2) The buffered infectivity thresholding option accepts two scalar values. A greater than value and a less than value. Depending on the vector and its properties, this option may provide the ability to capture more extreme transcriptional states of the two classes by setting the parameters with a buffered zone between greater than and less than values. (3) Lastly, the Scikit Learn Gaussian Mixture Model (GMM) implementation is included as a third option for clustering infectivity.²⁴ This option deploys the GaussianMixture class with the default parameters and `n_components = 2` to classify the infected and uninfected labels. Additionally, the TAP implementation of the GMM aims to increase the transcriptional distinctiveness of each label by only including cells with a 95% or greater probability of existing in each class.

3.3 Ensemble Implementation and Feature Importance

A Random Forest (RF) classifier implemented by Scikit Learn is used by default and invoked in the FeatureSelection class, which is instantiated in the TAP class and trained on each subset of categorical cells (X_c) after clustering labels are determined. Each subset (X_c) is evaluated for class imbalance. The Imbalanced Learn package is utilized to provide TAP parameters for data balancing.

Once classification labels are assigned, TAP trains a Random Forest (RF) model on each categorical subset (X_c). The Random Forest (RF) classifier *RandomForestClassifier*, implemented by Scikit Learn is used by default and invoked in the *FeatureSelection* class, which is instantiated in the TAP class. Optional parameters also exist for feature and gene filtering prior to training, removing specified viral serotypes, marker genes, and other unwanted features. To address class imbalance, TAP supports several resampling techniques via the imbalanced-learn package, including SMOTE, ADASYN, Borderline-SMOTE, SVM-SMOTE, KMeans-SMOTE, and random oversampling.^{26,43–46} Subsets with insufficient class diversity (e.g. only one class present) are skipped automatically.

The default parameters for training the *RandomForestClassifier* on each subset (X_c) include `n_estimators=20`, `max_depth=2`, and `min_samples_split=2`, though all hyperparameters can be customized by the user. TAP also supports automated hyperparameter tuning via randomized grid search (*RandomizedSearchCV*), exploring combinations of tree depth, number of estimators, maximum features, and minimum samples per split. The *rfHyperParameterTune* parameter may be used to access the *RandomizedSearchCV* functionality, however, this will increase the runtime of the TAP build. Model performance is evaluated and stored using accuracy and AUC. Additionally, TAP can optionally compute permutation-based feature importances using Scikit Learn's `permutation_importance`, repeated two times, user-defined, or can be skipped entirely. This produces a ranked list of genes most predictive of infectivity for each categorical subset (X_c). All feature related metrics are stored in JSON format and for downstream composition of the user interface.

3.4 Visualization Development

3.4.1 Overview of Interface Construction

TAP constructs a user interface in three parts. (1) The `AnnData` object undergoes extensive preprocessing, infectivity clustering, and ensemble feature selection, while caching the data results and base64 encoding supporting images into JSON format (**Methods 3.1-3.3**). (2) The preprocessed cell by gene matrix is converted to a CSV format. (3) Both JSON and CSV results are converted to one line representations, then pushed into a singular HTML output file which serves as a template for the visualization. The TAP visualization interface is built as a fully self-contained HTML output using Bootstrap and custom vanilla JavaScript. It supports interactive exploration of infectivity classifications and gene or virus level features across categorical cell subsets (X_c).

3.4.2 Interactive User Interface Development

As the main component of the UI, a dynamic and responsive heatmap is generated directly client side by parsing all JSON and CSV components using the D3 javascript library.⁴⁷ The main heatmap is crafted by calculating the amount of columns necessary for each cell to fit in the heatmap container, then the amount of table columns and rows necessary to build the heatmap are generated, which is responsive to various browser sizes; however, is not intended to be mobile optimized. Each top category bar (horizontally displayed) and gene/vector (left column, vertically displayed) are assigned unique IDs. Javascript on-click listeners detect user click interactions with those identifiers and facilitate loading selection specific content from the JSON cache in each of the four panels below the main heatmap.

3.4.3 Selection Details and Metrics

When responding to an on-click event, the first actionable task is to rebuild the “Selection Details” lower left block of the web application. Javascript will set a selection state which denotes the user interactive choice ($C^{(i)}$), then use it as a cache key to O(1) lookup details of corresponding metadata in the embedded JSON. The JSON provided details are used to update all values in the “Selection Details” UI element. Additionally, an image of the cluster results for the specific cell subset (X_c) is shown. All clustering images described in **Methods 3.2** are base64 encoded and embedded into the JSON, so they can be rendered in HTML. This section augments the other UI components by providing contextual validation of each subset (X_c) and UI components including: RF model parameters, accuracy, AUC; along with clustering results, and selection details which can be used as a first step toward evaluating results.

3.4.4 Ensemble Feature Importance, Differential Expression, and Highly Important Indicators

The two UI components, “Ensemble Features” and “Differential Expression” display gene lists constructed from each subset of cells (X_c). Each component uses javascript to draw a HTML table and populate it with results from **Methods 3.1-3.3**, based on the current user selection. All listed gene results are hyperlinked directly to the Gene Cards verbose description.⁴⁸ Ensemble feature results are displayed in descending order importance (highest on top) and include a barplot as a visual indicator, along with yellow, orange, and red pepper icons to indicate potential relevance to tropism (**Figure 1D**). These three icons are determined by the following logic: (1) A yellow icon denotes a particular gene that exists in both the Ensemble Features and the Differential Expression lists for the current selection. (2) An orange icon indicates a specific gene is present across all replicates in both lists; and (3) a red icon flags genes whose descriptions are fuzzy matches for keywords relating to receptors, plasma membrane, transmembrane, adhesion, or an extracellular molecule. The keyword matching is accomplished upstream of the UI during **Methods 3.1-3.3** using gene descriptions retrieved from the BioThings API.⁴⁹ The approach simply parses the descriptions for the relevant keywords. Each icon level is then indicated in the JSON for a particular gene contained with the subset selection (X_c).

3.4.5 Configurations and Parameters

Visual configurations options, including heatmap color themes and aesthetic visual UI styling were developed in pure javascript and are accessible via the gear icon in the interface. When selecting an option, Javascript updates html element IDs to apply the selected styles dynamically. The heatmap supports multiple themes that were developed to closely resemble popular Matplotlib color themes: Cividis, Viridis, Plasma, Inferno, Magma and a custom TAP theme that maps values from black to red.⁵⁰ Although these settings exist and are directly accessible in the UI, additional configuration options, including deeper parameter control over aesthetics and functionality and can be accessed in our documentation found at: <https://github.com/ArpiarSaundersLab/tap>

3.5 Replicate Mode

TAP includes the boolean parameter “*replicate_mode*” which can be used to process the dataset that considers replicates as individual entities. Replicate mode starts by examining categorical observations added by the user and searches for text indicating a category is part of a series of values (e.g. replicate_1, replicate_2, ... , replicate_n). Replicate mode also searches genes/vectors for potential replicates in the same manner (e.g. aav1_bc1, aav1_bc2, ... , aav1_bcn). Once each replicate is determined, they are processed in the main pipeline described in **Figure 1** and **Methods 3.1**, however, each subset (X_c) now includes an additional term specifically for each replicate. Once important features are determined individually for each replicate, they are then examined for the intersection of features existing in each replicate. Common features are denoted as described in **Methods 3.4.4**. Replicates are then represented in the UI as categories in the top bar (**Figure 1D,2E**) and parsed directly from the embedded JSON object.

3.6 Performance and Usability

Building the HTML output of TAP requires the entire pipeline (**Figure 1B**) to be run recursively through all categorical combinatorial subsets of cells. TAP requires ~10Gb of memory for every 10,000 cells in the input dataset. TAP is mutli-threaded and will use 2 CPUs by default. This value can be modified by utilizing the “*cpus*” parameter. Once the HTML output is built, the file can be opened in any modern

browser. During the building process, all results are cached in the final output which makes the UI experience fluid on most modern computers.

3.7 Multiplexed AAV Barcoding, Primary Cortical Cell Culture, and AAV transduction.

AAV vectors were based on a plasmid backbone encoding EGFP fused to a nuclear localization signal (NLS) under the control of the CAG promoter. A unique 300 bp synthetic barcode was inserted into the 3' UTR of each construct to allow transcript-based discrimination of capsid variants. Barcode sequences were designed to minimize secondary structure and avoid homopolymer runs. Each barcoded plasmid was cloned individually and sequence-verified via Sanger or whole-plasmid sequencing. Recombinant AAVs were produced using 11 distinct capsid variants (**Figure 2A, S1A**), each packaging an identical genome differing only in barcode identity. Production and titration were performed by the UNC NeuroTools Viral Vector Core Facility. Viral preparations were pooled in equimolar proportions to a final concentration of 1×10^{12} vg/mL.

Cortical tissue was isolated from E16.5 C57BL/6J mouse embryos and dissociated to obtain a mixed population of brain cell types, including neurons, astrocytes, oligodendrocyte precursor cells (OPCs), and microglia. Cells were cultured in Neurobasal medium supplemented with B27 and maintained at 37°C with 5% CO₂. At DIV7, cultures were transduced with the pooled AAV cocktail. At DIV14, cells were enzymatically dissociated using papain and filtered to obtain single-cell suspensions. Single-cell RNA-seq libraries were generated using the 10x Genomics Chromium Single-cell 3' v3.1 platform following the manufacturer's protocol.

Single-cell preprocessing was completed using the default Cell Ranger pipeline.⁵¹ Ambient RNA signal and duplicate cell captures were removed using Cell Bender with the default parameters, adjusted only for expected cells of each library.⁵² Barcodes were demultiplexed based on their unique 3' UTR barcode to determine which serotype transduced each individual cell. Using a custom script, barcodes were then added to the cell by gene matrix to be accessed downstream alongside the full gene panel. A standard Scanpy preprocessing pipeline was then followed for each dataset, along with PCA, integration using the Harmony²¹, followed by UMAP and manual cell type identification based on the Allen Brain Map Mouse Whole Cortex Markers.

3.8 Cell Typist Model Training

We provide automatic cell type prediction included with TAP using the Cell Typist package. Included with TAP are all default models provided by the package maintainers as well as two additional models we provide for convenience. The first model was trained using an adult mouse brain single-cell atlas²³ and the second is from a Rhesus macaque single-cell amygdala dataset⁵³. For both models, normalization and scaling were performed as recommended by Cell Typist. A grid parameter search was then deployed to determine optimal model training parameters of the following: *alpha*, *epochs*, *use_SGD*, *mini_batch*, *with_mean*, and *feature_selection*. Trained models were then assessed using confidence scores provided by Cell Typist. Models with the mean highest confidence scores were selected.

References

1. See, W.R., Yousefi, M., and Ooi, Y.S. (2024). A review of virus host factor discovery using CRISPR screening. *mBio* *15*, e03205-23. <https://doi.org/10.1128/mbio.03205-23>.
2. McDougall, W.M., Perreira, J.M., Reynolds, E.C., and Brass, A.L. (2018). CRISPR genetic screens to discover host–virus interactions. *Curr. Opin. Virol.* *29*, 87–100. <https://doi.org/10.1016/j.coviro.2018.03.007>.
3. Pillay, S., Meyer, N.L., Puschnik, A.S., Davulcu, O., Diep, J., Ishikawa, Y., Jae, L.T., Wosen, J.E., Nagamine, C.M., Chapman, M.S., et al. (2016). An essential receptor for adeno-associated virus infection. *Nature* *530*, 108–112. <https://doi.org/10.1038/nature16465>.
4. Dudek, A.M., Zabaleta, N., Zinn, E., Pillay, S., Zengel, J., Porter, C., Franceschini, J.S., Estelien, R., Carette, J.E., Zhou, G.L., et al. (2020). GPR108 Is a Highly Conserved AAV Entry Factor. *Mol. Ther.* *28*, 367–381. <https://doi.org/10.1016/j.ymthe.2019.11.005>.
5. Dhungel, B.P., Xu, H., Nagarajah, R., Vitale, J., Wong, A.C.H., Gokal, D., Feng, Y., Tabar, M.S., Metierre, C., Parsania, C., et al. (2025). An alternate receptor for adeno-associated viruses. *Cell* *188*, 4924–4935.e23. <https://doi.org/10.1016/j.cell.2025.06.026>.
6. Li, B., Clohisey, S.M., Chia, B.S., Wang, B., Cui, A., Eisenhaure, T., Schweitzer, L.D., Hoover, P., Parkinson, N.J., Nachshon, A., et al. (2020). Genome-wide CRISPR screen identifies host dependency factors for influenza A virus infection. *Nat. Commun.* *11*, 164. <https://doi.org/10.1038/s41467-019-13965-x>.
7. Biering, S.B., Sarnik, S.A., Wang, E., Zengel, J.R., Leist, S.R., Schäfer, A., Sathyan, V., Hawkins, P., Okuda, K., Tau, C., et al. (2022). Genome-wide bidirectional CRISPR screens identify mucins as host factors modulating SARS-CoV-2 infection. *Nat. Genet.* *54*, 1078–1089. <https://doi.org/10.1038/s41588-022-01131-x>.
8. Park, R.J., Wang, T., Koundakjian, D., Hultquist, J.F., Lamothe-Molina, P., Monel, B., Schumann, K., Yu, H., Krupczak, K.M., Garcia-Beltran, W., et al. (2017). A genome-wide CRISPR screen identifies a restricted set of HIV host dependency factors. *Nat. Genet.* *49*, 193–203. <https://doi.org/10.1038/ng.3741>.
9. Su, C., Xu, Z., Shan, X., Cai, B., Zhao, H., and Zhang, J. (2023). Cell-type-specific co-expression inference from single cell RNA-sequencing data. *Nat. Commun.* *14*, 4846. <https://doi.org/10.1038/s41467-023-40503-7>.
10. van Dam, S., Vösa, U., van der Graaf, A., Franke, L., and de Magalhães, J.P. (2018). Gene co-expression analysis for functional classification and gene–disease predictions. *Brief. Bioinform.* *19*, 575–592. <https://doi.org/10.1093/bib/bbw139>.
11. Wolf, F.A., Angerer, P., and Theis, F.J. (2018). SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.* *19*, 15. <https://doi.org/10.1186/s13059-017-1382-0>.
12. Virshup, I., Rybakov, S., Theis, F.J., Angerer, P., and Wolf, F.A. (2021). anndata: Annotated data. Preprint at bioRxiv, <https://doi.org/10.1101/2021.12.16.473007>.

13. Comparative analysis of structure, expression and PSD95-binding capacity of Lrfrn, a novel family of neuronal transmembrane proteins (2006). *Gene* 380, 72–83. <https://doi.org/10.1016/j.gene.2006.05.014>.
14. Cagno, V., Tseligka, E.D., Jones, S.T., and Tapparel, C. (2019). Heparan Sulfate Proteoglycans and Viral Attachment: True Receptors or Adaptation Bias? *Viruses* 11, 596. <https://doi.org/10.3390/v11070596>.
15. Ojala, D.S., Sun, S., Santiago-Ortiz, J.L., Shapiro, M.G., Romero, P.A., and Schaffer, D.V. (2018). In Vivo Selection of a Computationally Designed SCHEMA AAV Library Yields a Novel Variant for Infection of Adult Neural Stem Cells in the SVZ. *Mol. Ther.* 26, 304–319. <https://doi.org/10.1016/j.ymthe.2017.09.006>.
16. Bauer, A., Nolden, T., Nemitz, S., Perlson, E., and Finke, S. (2015). A Dynein Light Chain 1 Binding Motif in Rabies Virus Polymerase L Protein Plays a Role in Microtubule Reorganization and Viral Primary Transcription. *J. Virol.* 89, 9591–9600. <https://doi.org/10.1128/jvi.01298-15>.
17. Feige, L., Young, K., Cerapio, J.P., Kozaki, T., Kergoat, L., Libri, V., Ginhoux, F., Hasan, M., Ameer, L.B., Chin, G., et al. (2025). Human brain cell types shape host-rabies virus transcriptional interactions revealing a preexisting pro-viral astrocyte subpopulation. *Cell Rep.* 44. <https://doi.org/10.1016/j.celrep.2025.116159>.
18. Tan, G.S., Preuss, M.A.R., Williams, J.C., and Schnell, M.J. (2007). The dynein light chain 8 binding motif of rabies virus phosphoprotein promotes efficient viral transcription. *Proc. Natl. Acad. Sci.* 104, 7229–7234. <https://doi.org/10.1073/pnas.0701397104>.
19. Jespersen, N.E., Leyrat, C., Gérard, F.C., Bourhis, J.-M., Blondel, D., Jamin, M., and Barbar, E. (2019). The LC8-RavP ensemble Structure Evinces A Role for LC8 in Regulating Lyssavirus Polymerase Functionality. *J. Mol. Biol.* 431, 4959–4977. <https://doi.org/10.1016/j.jmb.2019.10.011>.
20. Pavan, K., and Saunders, A. (2025). AnnSQL: a Python SQL-based package for fast large-scale single-cell genomics analysis using minimal computational resources. *Bioinforma. Adv.* 5, vbaf105. <https://doi.org/10.1093/bioadv/vbaf105>.
21. Xu, C., Prete, M., Webb, S., Jardine, L., Stewart, B.J., Hoo, R., He, P., Meyer, K.B., and Teichmann, S.A. (2023). Automatic cell-type harmonization and integration across Human Cell Atlas datasets. *Cell* 186, 5876–5891.e20. <https://doi.org/10.1016/j.cell.2023.11.026>.
22. Conde, C.D., Xu, C., Jarvis, L.B., Rainbow, D.B., Wells, S.B., Gomes, T., Howlett, S.K., Suchanek, O., Polanski, K., King, H.W., et al. (2022). Cross-tissue immune cell analysis reveals tissue-specific features in humans. *Science*. <https://doi.org/10.1126/science.abl5197>.
23. Langlieb, J., Sachdev, N.S., Balderrama, K.S., Nadaf, N.M., Raj, M., Murray, E., Webber, J.T., Vanderburg, C., Gazestani, V., Tward, D., et al. (2023). The molecular cytoarchitecture of the adult mouse brain. *Nature* 624, 333–342. <https://doi.org/10.1038/s41586-023-06818-7>.
24. Pedregosa, F., Pedregosa, F., Varoquaux, G., Varoquaux, G., Org, N., Gramfort, A., Gramfort, A., Michel, V., Michel, V., Fr, L., et al. Scikit-learn: Machine Learning in Python. *Mach. Learn. PYTHON*.
25. Maximum Likelihood from Incomplete Data via the EM Algorithm on JSTOR <https://www-jstor-org.liboff.ohsu.edu/stable/2984875?seq=1>.

26. SMOTE: Synthetic Minority Over-sampling Technique | Journal of Artificial Intelligence Research <https://www.jair.org/index.php/jair/article/view/10302>.
27. Lemaître, G., Nogueira, F., and Aridas, C.K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *J. Mach. Learn. Res.* *18*, 1–5.
28. Zheng, X., Wu, B., Liu, Y., Simmons, S.K., Kim, K., Clarke, G.S., Ashiq, A., Park, J., Li, J., Wang, Z., et al. (2024). Massively parallel in vivo Perturb-seq reveals cell-type-specific transcriptional networks in cortical development. *Cell* *187*, 3236–3248.e21. <https://doi.org/10.1016/j.cell.2024.04.050>.
29. Saunders, A., Huang, K.W., Vondrak, C., Hughes, C., Smolyar, K., Sen, H., Philson, A.C., Nimesh, J., Wysoker, A., Kashin, S., et al. (2022). Ascertaining cells' synaptic connections and RNA expression simultaneously with barcoded rabies virus libraries. *Nat. Commun.* *13*, 6993. <https://doi.org/10.1038/s41467-022-34334-1>.
30. Zhang, R., Xu, G., Cao, L., Sun, Z., He, Y., Cui, M., Sun, Y., Li, S., Li, H., Qin, L., et al. (2019). Divergent engagements between adeno-associated viruses with their cellular receptor AAVR. *Nat. Commun.* *10*, 3760. <https://doi.org/10.1038/s41467-019-11668-x>.
31. Walkey, C.J., Snow, K.J., Bulcha, J., Cox, A.R., Martinez, A.E., Ljungberg, M.C., Lanza, D.G., Giorgi, M.D., Chuecos, M.A., Alves-Bezerra, M., et al. (2025). A comprehensive atlas of AAV tropism in the mouse. *Mol. Ther.* *33*, 1282–1299. <https://doi.org/10.1016/j.ymthe.2025.01.041>.
32. Tervo, D.G.R., Hwang, B.-Y., Viswanathan, S., Gaj, T., Lavzin, M., Ritola, K.D., Lindo, S., Michael, S., Kuleshova, E., Ojala, D., et al. (2016). A Designer AAV Variant Permits Efficient Retrograde Access to Projection Neurons. *Neuron* *92*, 372–382. <https://doi.org/10.1016/j.neuron.2016.09.021>.
33. Klimczak, R.R., Koerber, J.T., Dalkara, D., Flannery, J.G., and Schaffer, D.V. (2009). A Novel Adeno-Associated Viral Variant for Efficient and Selective Intravitreal Transduction of Rat Müller Cells. *PLOS ONE* *4*, e7467. <https://doi.org/10.1371/journal.pone.0007467>.
34. Zinn, E., Pacouret, S., Khaychuk, V., Turunen, H.T., Carvalho, L.S., Andres-Mateos, E., Shah, S., Shelke, R., Maurer, A.C., Plovie, E., et al. (2015). In Silico Reconstruction of the Viral Evolutionary Lineage Yields a Potent Gene Therapy Vector. *Cell Rep.* *12*, 1056–1068. <https://doi.org/10.1016/j.celrep.2015.07.019>.
35. Wickersham, I.R., Finke, S., Conzelmann, K.-K., and Callaway, E.M. (2007). Retrograde neuronal tracing with a deletion-mutant rabies virus. *Nat. Methods* *4*, 47–49. <https://doi.org/10.1038/nmeth999>.
36. Nam, J., Mah, W., and Kim, E. (2011). The SALM/Lrfr family of leucine-rich repeat-containing cell adhesion molecules. *Semin. Cell Dev. Biol.* *22*, 492–498. <https://doi.org/10.1016/j.semcdb.2011.06.005>.
37. Choi, Y., Nam, J., Whitcomb, D.J., Song, Y.S., Kim, D., Jeon, S., Um, J.W., Lee, S.-G., Woo, J., Kwon, S.-K., et al. (2016). SALM5 trans-synaptically interacts with LAR-RPTPs in a splicing-dependent manner to regulate synapse development. *Sci. Rep.* *6*, 26676. <https://doi.org/10.1038/srep26676>.

38. Kremer, L.P.M., Cerrizuela, S., Dehler, S., Stiehl, T., Weinmann, J., Abendroth, H., Kleber, S., Laure, A., Andari, J.E., Anders, S., et al. (2021). High throughput screening of novel AAV capsids identifies variants for transduction of adult NSCs within the subventricular zone. *Mol. Ther. Methods Clin. Dev.* 23, 33–50. <https://doi.org/10.1016/j.omtm.2021.07.001>.
39. Raux, H., Flamand, A., and Blondel, D. (2000). Interaction of the Rabies Virus P Protein with the LC8 Dynein Light Chain. *J. Virol.* 74, 10212–10216. <https://doi.org/10.1128/jvi.74.21.10212-10216.2000>.
40. Tan, G.S., Preuss, M.A.R., Williams, J.C., and Schnell, M.J. (2007). The dynein light chain 8 binding motif of rabies virus phosphoprotein promotes efficient viral transcription. *Proc. Natl. Acad. Sci.* 104, 7229–7234. <https://doi.org/10.1073/pnas.0701397104>.
41. Poisson, N., Real, E., Gaudin, Y., Vaney, M.-C., King, S., Jacob, Y., Tordo, N., and Blondel, D. (2001). Molecular basis for the interaction between rabies virus phosphoprotein P and the dynein light chain LC8: dissociation of dynein-binding properties and transcriptional functionality of P. *J. Gen. Virol.* 82, 2691–2696. <https://doi.org/10.1099/0022-1317-82-11-2691>.
42. Buccitelli, C., and Selbach, M. (2020). mRNAs, proteins and the emerging principles of gene expression control. *Nat. Rev. Genet.* 21, 630–644. <https://doi.org/10.1038/s41576-020-0258-4>.
43. He, H., Bai, Y., Garcia, E.A., and Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 1322–1328. <https://doi.org/10.1109/IJCNN.2008.4633969>.
44. Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In *Advances in Intelligent Computing*, D.-S. Huang, X.-P. Zhang, and G.-B. Huang, eds. (Springer), pp. 878–887. https://doi.org/10.1007/11538059_91.
45. Adaptive SV-Borderline SMOTE-SVM algorithm for imbalanced data classification (2024). *Appl. Soft Comput.* 150, 110986. <https://doi.org/10.1016/j.asoc.2023.110986>.
46. Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE (2018). *Inf. Sci.* 465, 1–20. <https://doi.org/10.1016/j.ins.2018.06.056>.
47. Bostock, M., Ogievetsky, V., and Heer, J. (2011). D³ Data-Driven Documents. *IEEE Trans. Vis. Comput. Graph.* 17, 2301–2309. <https://doi.org/10.1109/TVCG.2011.185>.
48. Stelzer, G., Rosen, N., Plaschkes, I., Zimmerman, S., Twik, M., Fishilevich, S., Stein, T.I., Nudel, R., Lieder, I., Mazor, Y., et al. The GeneCards Suite: From Gene Data Mining to Disease Genome Sequence Analyses. <https://doi.org/10.1002/cpbi.5>.
49. Lelong, S., Zhou, X., Afrasiabi, C., Qian, Z., Cano, M.A., Tsueng, G., Xin, J., Mullen, J., Yao, Y., Avila, R., et al. (2022). BioThings SDK: a toolkit for building high-performance data APIs in biomedical research. *Bioinformatics* 38, 2077–2079. <https://doi.org/10.1093/bioinformatics/btac017>.
50. Hunter, J.D. (2007). Matplotlib: A 2D Graphics Environment. *Comput. Sci. Eng.* 9, 90–95. <https://doi.org/10.1109/MCSE.2007.55>.

51. Zheng, G.X.Y., Terry, J.M., Belgrader, P., Ryvkin, P., Bent, Z.W., Wilson, R., Ziraldo, S.B., Wheeler, T.D., McDermott, G.P., Zhu, J., et al. (2017). Massively parallel digital transcriptional profiling of single cells. *Nat. Commun.* 8, 14049. <https://doi.org/10.1038/ncomms14049>.
52. Fleming, S.J., Chaffin, M.D., Arduini, A., Akkad, A.-D., Banks, E., Marioni, J.C., Philippakis, A.A., Ellinor, P.T., and Babadi, M. (2023). Unsupervised removal of systematic background noise from droplet-based single-cell experiments using CellBender. *Nat. Methods* 20, 1323–1335. <https://doi.org/10.1038/s41592-023-01943-7>.
53. Totty, M.S., Juanes, R.C., Bach, S.V., Ameer, L.B., Valentine, M.R., Simons, E., Romac, M., Trinh, H., Henderson, K., Rosario, I.D., et al. (2024). Transcriptomic diversity of amygdalar subdivisions across humans and nonhuman primates. Preprint at bioRxiv, <https://doi.org/10.1101/2024.10.18.618721> <https://doi.org/10.1101/2024.10.18.618721>.