Perspective

Developing Machine Learning Applications for Population Genetic Inference: Ensuring Precise Terminology and Robust Implementation

Xin Huang[1, 2, *]

1 Department of Evolutionary Anthropology, University of Vienna, Vienna, Austria

2 Human Evolution and Archaeological Sciences (HEAS), University of Vienna, Vienna, Austria

* Corresponding author: xin.huang@univie.ac.at

Abstract

Machine learning applications for population genetic inference are emerging due to their potential to leverage large-scale genomic datasets, offering insights that traditional statistical methods may overlook. However, I have identified certain recurring issues. First, there is sometimes confusion between power and recall, and between the false discovery rate and one minus precision. These terms are specifically designed for hypothesis testing and are not appropriate for directly evaluating classification outcomes, as classification is a different task. Second, the lack of robustness in machine learning applications complicates their verification and application across different datasets, limiting their broader impact and slowing research progress. Robustness can be improved through strategies such as employing object-oriented programming for design, utilizing version control systems during development, and adopting package managers and workflow managers for distribution. I suggest by adhering to precise terminology and refining implementation practices, the impact of machine learning in population genetics can be maximized.

Keywords: Population genetics; Machine learning; Classification; Hypothesis testing; Software development.

Recent advances in machine learning, particularly in deep learning, have introduced innovative methodologies for population genetic inference, enabling the development of numerous applications for tasks such as inferring population structure, identifying genomic introgression, and investigating natural selection (Schrider and Kern 2018; Korfmann et al. 2023; Yelman and Jay 2023; Huang et al. 2024). These advancements capitalize on large-scale genomic data and sophisticated models, revealing intricate patterns that traditional statistical methods might overlook. While interdisciplinary approaches can inspire

novel applications, it is crucial to apply these methods with precision, particularly regarding the terminology across different fields. Such precision enhances communication among disciplines and helps prevent misunderstandings. As machine learning models grow increasingly complex, testing and validating these approaches becomes progressively challenging (Raff 2019). Therefore, ensuring the robustness of machine learning applications is essential for maximizing their impact. In this perspective, I will first address the misuse of statistical terminology when applying machine learning to population genetics, followed by an examination of the issues of non-robustness in certain machine learning applications for population genetic inference, along with proposed strategies to enhance reproducibility and reliability in the development of these applications.

Utilize Precise Terminology

Several recent machine learning applications for population genetic inference (Durvasula and Sankararaman 2020; Amin et al. 2023; Arnab et al. 2023; Lauterbur et al. 2023a; Zhang et al. 2023) have used power and the false discovery rate (FDR) to evaluate the performance of their classification models, equating power to recall (or the true positive rate) and FDR to one minus precision. However, the terms 'power' and 'FDR' seem to be used colloquially to describe performance in terms of aptitude or effectiveness, which can lead to confusion. These terms have specific meanings in hypothesis testing, particularly in frequentist inference, and should not be conflated with machine learning metrics, as they are not directly interchangeable. For a statistical test, power is defined as the probability of correctly rejecting the null hypothesis when it is false, while FDR, in the context of multiple testing, is defined as the expected proportion of false discoveries—not false positives—among the rejected null hypotheses (Benjamini and Hochberg 1995; Wasserman 2003; Descôteaux 2007; Shreffler and Huecker 2024).

Hypothesis testing uses sample data to assess whether a population characteristic is more likely to be consistent with the null hypothesis (H_0) or the alternative hypothesis (H_1), for the population from which the sample was drawn. To conduct a hypothesis testing, an appropriate test statistic is needed (Wasserman 2003; Alpaydin et al. 2014). For example, suppose we sample two groups of data $\{X_1, X_2, \ldots, X_n\}$ and $\{Y_1, Y_2, \ldots, Y_n\}$, a typical hypothesis task is to determine whether their means are equal. In this case, H_0 is $\mu_X = \mu_Y$, and H_1 is $\mu_X \neq \mu_Y$, where μ_X and μ_Y represent the true means of the populations from which the samples $\{X_1, X_2, \ldots, X_n\}$ and $\{Y_1, Y_2, \ldots, Y_n\}$ are drawn. If we further assume that both groups are approximately normally distributed, then we can calculate a t statistic based on the collected data to test our hypothesis. If such a test is performed multiple times, a multiple testing correction may be applied.

In contrast, classification is a predictive task that focuses on assigning a label or category to each newly observed data point, rather than making inferences about population characteristics. Unlike hypothesis

testing, classification does not rely on a test statistic to make decisions. Instead, it employs an algorithmic approach to learn patterns from the data and apply those patterns to new observations. For example, consider a dataset $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$, where $\{X_1, X_2, \ldots, X_n\}$ are instances and $\{Y_1, Y_2, \ldots, Y_n\}$ are their corresponding classes; classification uses these data to find a function $f: X \rightarrow Y$ that predicts the class Y_{n+1} of a new instance X_{n+1}, where $Y_{n+1} = f(X_{n+1})$ (Wasserman 2003). Since no hypothesis testing is involved in this prediction, I suggest that when applying techniques such as classification to fields like population genetics, it is important and necessary to adhere to the established terminology specific to those techniques.

Several metrics to assess the performance of a classifier exist, such as error, which represents the proportion of incorrectly classified instances out of the total number of instances (Alpaydin et al. 2014). Error is equal to one minus accuracy, where accuracy is the proportion of correctly classified instances (Alpaydin et al. 2014). However, accuracy is often an inadequate metric for evaluating performance, which is why receiver operating characteristic (ROC) curves have become popular in the machine learning community (Fawcett 2006). To conduct ROC analysis, various performance metrics are calculated from elements organized in a confusion matrix, which may sometimes be mistakenly equated with the outcomes of hypothesis testing or multiple testing due to their similar structure (Figure 1).

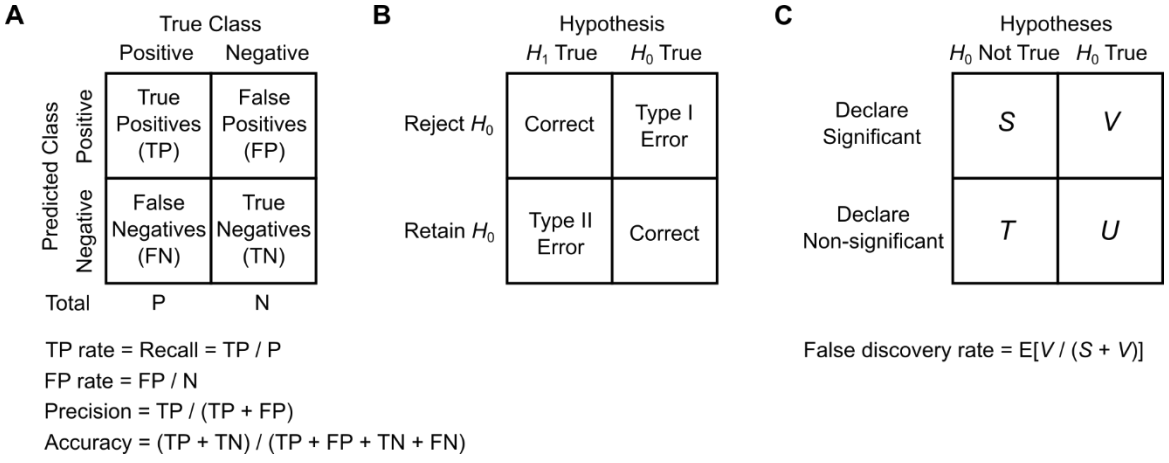

Figure 1 Outcomes in different scenarios. (A) Binary classification outcomes. (B) Hypothesis testing outcomes. (C) Multiple testing outcomes.

One should note that 'positive' and 'negative' in machine learning are defined on a case-by-case basis. For example, when detecting introgressed fragments, 'positive' can refer to the detection of introgressed fragments, whereas when detecting introgression deserts, 'positive' can refer to the detection of these deserts. Instead, conventional statistical terminology emphasizes concepts like rejecting or failing to reject the null hypothesis, identifying significant results, or making discoveries, without framing these outcomes

in terms of ‘positive’ or ‘negative’ (Figure 1B). The null hypothesis typically represents a statement of no effect, such as no difference between groups, which does not align naturally with the binary language of ‘positive’ or ‘negative.’ This is because statistical results often involve complex interpretations that cannot be easily reduced to such terms. For instance, a statistically significant result does not necessarily imply scientific significance, and a non-significant result does not necessarily indicate the absence of an effect—it may simply mean the effect was not detectable with the given data (Wasserman 2003). In this context, a ‘discovery’ typically refers to the rejection of the null hypothesis (Soric 1989). The original definition of FDR describes it as ‘the expected proportion of falsely rejected hypotheses,’ without employing the term ‘positive’ (Benjamini and Hochberg 1995). This suggests that, in hypothesis testing, conclusions are made carefully and conditionally based on the data, while in classification, the focus is on confidently assigning labels to instances, reflecting the fundamental difference in how results are interpreted.

Moreover, FDR is the expected value of the false discovery proportion (FDP; Wasserman 2003), a random variable $Q = V / (S + V)$ (Figure 1C). Although FDP, $V / (S + V)$, resembles the formula for FP / (TP + FP), it is important to note that S and V are unobserved random variables; only the total number of tests declared significant ($S + V$) is known after experimentation and analysis of the collected data (Benjamini and Hochberg 1995). This differs from common metrics for evaluating classification model performance, such as recall and precision, which are calculated using elements (TP, FP, TN, and FN) directly observed in the data (Figure 1A). These metrics directly measure performance based on prediction outcomes, without requiring a statistical hypothesis. Hence, the claim that FDR represents ‘the proportion of false positives among all positives detected’ is inaccurate (Lauterbur et al. 2023a; Romieu et al. 2024).

FDP, precision, and recall are dataset-dependent metrics, meaning they vary based on the characteristics of the data and the model performance, which are often beyond direct control. In contrast, the power of a statistical test remains theoretically consistent for a given null and alternative hypothesis, as long as test conditions—such as sample size, significance level, and effect size—are held constant. This is because power reflects the probability of detecting an effect when one exists, and these test conditions can be controlled in advance (Shreffler and Huecker 2024). Although we can adjust the significance level in hypothesis testing to influence the power of the test, this differs from how thresholds are used in classification to determine whether an instance belongs to the positive or negative class based on the probability output from a classifier. The significance level controls the probability of making a type I error (Figure 1B) in deciding whether to reject the null hypothesis based on the entire dataset, and adjusting it involves balancing the risk of making a type I error with the power of the test. Conversely, the

threshold in machine learning is a flexible parameter used to classify individual instances, and adjusting it balances precision and recall, impacting how confidently we can classify each instance. For example, when detecting introgressed fragments in genomes, we might choose a threshold that prioritizes high precision to obtain highly confident results for candidate segments. Additionally, power analysis is typically conducted during the experimental design phase, before data collection, to ensure that the study is adequately powered to detect the expected effects (Descôteaux 2007). Although retrospective power analysis can be conducted after experimentation, it is a controversial practice and may be fundamentally flawed (Thomas 1997; Hoenig and Heisey 2001). These fundamental differences highlight why directly equating FDR with one minus precision, or power with recall (or the true positive rate), is inappropriate.

Consequently, misusing these metrics can lead to misunderstandings about the distinct focuses, workflows, and methodologies, thereby hindering effective communication and collaboration across disciplines. However, these approaches can be complementary. For example, machine learning can introduce novel methods for controlling FDR, offering new perspectives beyond traditional approaches like the Benjamini-Hochberg procedure and independent hypothesis weighting (Xia et al. 2017). Although hypothesis testing can be employed to evaluate the performance of classification algorithms or to compare the efficacy of different classifiers (Dietterich 1998; Demšar 2006; Alpaydin et al. 2014), it is crucial to clearly define the hypothesis, as hypothesis testing requires a well-defined hypothesis (Wasserman 2003). For example, when assessing the performance of a classifier, one might test whether the probability p that the classifier makes an error is less than or equal to a specified value p_0. In this scenario, H_0 would be $p \leq p_0$, while H_1 would be $p > p_0$; statistical tests, such as the binominal test, can then be utilized to evaluate this hypothesis (Alpaydin et al. 2014). In this context, we could discuss the power of these statistical tests rather than the power of the classifier.

Develop Robust Implementation

Machine learning is fundamentally dependent on code, making robust implementation crucial for ensuring the accuracy and validity of results. A robust implementation encompasses both reliable programs and reproducible analyses. Such an implementation not only allows other researchers to replicate experiments and verify outcomes but also facilitates the application of established machine learning approaches to different datasets. This, in turn, enhances our understanding of evolution across various species and populations, which is certainly a motivating intention of the researchers developing such methods. More generally, there is a reproducibility crisis in life sciences (Collins and Tabak 2014; Baker 2016), which can only be addressed by proper implementations. Conversely, a lack of robustness forces potential users to painstakingly examine the code and analysis, possibly leading to reinventing the wheel, which almost certainly slows down the pace of research and causes unnecessary duplication of

efforts. As noted by Nekrutenko et al. (2018), truly usable software in evolutionary biology is currently in short supply, and this issue could be exacerbated in the context of machine learning applications.

For example, a recent study by Romieu et al. (2024) evaluated the performance of several methods for detecting adaptive introgression, including two newly developed machine learning approaches, genomatnn (Gower et al. 2021) and MaLAdapt (Zhang et al. 2023). This study found that the Q95 statistic (Racimo et al. 2017) outperformed other methods, which differs from the results reported in the original studies for genomatnn and MaLAdapt. This discrepancy is likely due to the reliance on pre-trained models from the original studies, which were trained using simulated data based on specific human demographic models that differ from the new dataset, particularly since it originates from a different species. As a result, this mismatch may lead to model misspecification when the pre-trained models are applied to the new data. The current implementation of these methods makes training new models on novel datasets difficult, restricting their application to pre-trained models or a limited set of predefined human demographic models for simulating training data. This may lead to reduced performance when applied to diverse datasets, especially those from different species.

Hence, this suggests that non-robust implementation could lead to inconsistencies in model performance, particularly when new datasets are involved, ultimately hindering the broader applicability of these methods. Several recommendations to enhance the reproducibility of machine learning applications in life sciences have been proposed (Heil et al. 2021; Walsh et al. 2021). For instance, Heil et al. (2021) proposed a gold standard for reproducibility and emphasized that the entire analysis is reproducible with a single command, requiring minimal effort to replicate. Building on these recommendations and best practices from fields such as software development, I propose additional practical approaches to further promote the development of robust machine learning applications in population genetics (Table 1).

Table 1 Practical approaches for implementing robust machine learning applications

Phase	Design	Development	Distribution
Approach	• Object-oriented programming for organizing the code • Unified Modeling Language for mapping relation among different modules	• Version control tools for documenting development history • Unit tests for ensuring code functionality • Code review for improving code quality	• Package managers for maintaining dependencies and facilitating installation • Workflow managers for automating and managing complex tasks

To develop reliable code, developers can start with object-oriented programming (OOP), a paradigm that efficiently organizes complex software by structuring code around objects representing real-world entities

or concepts. OOP encapsulates data and behavior within self-contained modules, allowing for independent development, testing, and maintenance. Through inheritance, OOP enables the construction of complex modules from simpler ones, increasing code reusability and reducing redundancy. These principles—encapsulation and inheritance—enhance code flexibility and extensibility, thereby improving reliability by making it easier to modify, extend, and maintain. Familiarity with OOP is also beneficial for understanding machine learning frameworks like PyTorch (Paszke et al. 2019), which are built on OOP concepts. Additionally, Unified Modeling Language can aid in complex software design by visualizing the structure and behavior of software systems.

During code development, developers can utilize version control tools such as Git, which enable tracking changes and automatically documenting the development history of the application—a practice recommended by DOME (Walsh et al. 2021). In addition to writing code, developers should implement unit tests to ensure that individual components function correctly and to maintain the integrity of the codebase as it evolves. Developers can use Git and deposit their code on various platforms, such as GitHub, Bitbucket, and Hugging Face (specialized for machine learning). These platforms provide automated workflows for running unit tests, streamlining the coding and testing process, and ensuring continuous integration. Furthermore, these platforms facilitate collaboration, which can improve code quality through peer review, collective problem-solving, and user feedback. For instance, the development guidelines of the PopSim Consortium (Adrion et al. 2020) require that code contributions undergo peer review for validation in their GitHub repository, exemplifying how collaboration on platforms like GitHub can enhance code quality and reliability. This collaborative approach aligns with recommendations to employ reproducibility collaborators for validating machine learning studies (Heil et al. 2021).

To further enhance reproducibility, developers can apply package managers for distributing code (Nekrutenko et al. 2018) and workflow managers for streamlining analysis. Package managers—such as Conda and Mamba—automate the installation of dependencies, manage different software environments, and ensure that all necessary packages are available and correctly configured. This is particularly important in machine learning, where packages evolve rapidly, and version conflicts are a common cause of reproducibility issues. By controlling the exact versions of all dependencies (Figure 2A), these tools help mitigate inconsistencies and ensure that the environment in which the code runs remains stable and reproducible. Workflow managers, such as NextFlow (Tommaso et al. 2017) and Snakemake (Mölder et al. 2021), allow developers to define and automate complex workflows, ensuring that each step in the analysis is executed in the correct order and under consistent conditions each time. This is notably valuable in machine learning, as it often involves multiple steps, such as data preprocessing, model

training, hyperparameter tuning, and performance evaluation (Figure 2B). Additionally, workflow managers can optimize the use of computing resources by distributing tasks across different machines and even integrating local and cloud servers (Huang et al. 2023). This capability is especially beneficial when working with massive datasets, ensuring that machine learning applications are both scalable and efficient. Together, package managers and workflow managers provide a robust framework for achieving the reliability and reproducibility required in modern machine learning applications. To demonstrate, I implemented a Snakemake workflow (see Data availability) to reproduce the results for detecting introgressed loci using pop_gen_cnn (Flagel et al. 2019). Users can replicate the entire analysis with a single command, aligning with the gold standard suggested by Heil et al. (2021).

A

```yaml
name: prml
channels:
    - conda-forge
    - bioconda
dependencies:
    - matplotlib=3.8.4
    - scikit-learn=1.5.0
    - snakemake=8.11.6
    - tensorflow=2.16.1
```

B

```
##### Target rules #####
rule all:
    input:
        "results/plots/pop_gen_cnn.original.model.performance.png",
        "results/plots/pop_gen_cnn.reproduced.model.performance.png",
##### Modules #####
include: "rules/download.smk"
include: "rules/train_pop_gen_cnn.smk"
include: "rules/test_pop_gen_cnn.smk"
```

Figure 2 An example illustrating how package managers and workflow managers can facilitate the implementation of robust machine learning applications for population genetic inference. (A) The dependencies and their versions for an application can be recorded in a YAML file, which is then used by the package manager Conda to create a virtual environment. This ensures a reproducible and consistent environment for executing the application. (B) Different steps in the Snakemake workflow, such as data downloading, model training, and testing, can be streamlined into a single file. The specific details of each step can then be documented in separate modules. This approach enables users to configure the appropriate environments for each step within the workflow. For instance, the training process in pop_gen_cnn uses Python 3 and TensorFlow 2 (Abadi et al. 2016), while the evaluation process requires Python 2 and TensorFlow 1 to maintain compatibility with the original model.

Besides source code, it is essential to make training and test datasets, model checkpoints, and trained models publicly available. This is particularly important because machine learning often relies on stochastic algorithms, where sharing model checkpoints and trained models helps in examining how randomness affects model performance. For supervised learning-based applications that utilize simulated data as training data—a new paradigm in population genetics (Schrider and Kern 2018)—sharing the simulation scripts and simulated datasets is crucial. These resources not only help users verify findings but also enable them to generate their own training data for similar research questions, for example, when working on different species. Moreover, it is vital to use established simulators, such as ms, msprime, and SLiM (Hudson 2002; Baumdicker et al. 2022; Haller and Messer 2023), and curated demographic

models, such as those from the PopSim Consortium (Lauterbur et al. 2023b), for generating simulated data. These tools have been extensively tested by the community, reducing the likelihood of producing unreliable datasets. As examples of such problems, customized versions of ms were used in two recent machine learning applications—ArchIE and IntroUNET—to simulate training data and detect introgressed fragments or alleles in genomes (Durvasula and Sankararaman 2019; Ray et al. 2024). In ArchIE, a variable representing the introgression proportion in the simulated data was introduced into ms, which causes this customized version of ms to fail when no introgression occurs between populations. Additionally, the recipient population—the population receiving genetic material during introgression—must be defined as the first subpopulation in the demographic model; otherwise, the variable cannot accurately report the introgression proportion, limiting its applicability across different species or population structures. Furthermore, while a four-population human demographic model was used to simulate the training data, a three-population model was reported in the paper, resulting in inconsistencies. In IntroUNET, modifications to the seed argument in ms introduced bugs, causing the same training data to be simulated periodically, which raises concerns about the actual performance of IntroUNET. These examples underscore the importance of transparency and the use of reliable, well-tested tools in the simulation process.

In conclusion, while machine learning holds great potential for addressing complex evolutionary problems, I believe that its full power can only be realized through precise terminology and robust implementation.

Acknowledgements

X.H. thanks Josef Hackl for testing ArchIE, IntroUNET, genomatnn, and MaLAdapt; Martin Kuhlwilm and Yungang He for their discussions and comments on the manuscript; and the Life Science Compute Cluster at the University of Vienna for providing computing resources.

Competing interests

X.H. declares no conflict of interests.

Data availability

The Snakemake workflow to reproduce the results for detecting introgressed loci using pop_gen_cnn with a single command can be found in https://github.com/xin-huang/prml, last accessed August 31, 2024.

References

Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, et al. 2016. TensorFlow: A system for large-scale machine learning. arXiv. https://arxiv.org/abs/1605.08695, last accessed August 31, 2024.

Adrion JR, Cole CB, Dukler N, Galloway JG, Gladstein AL, Gower G, Kyriazis CC, Ragsdale AP, Tsambos G, Baumdicker F, et al. 2020. A community-maintained standard library of population genetic models. *eLife* **9**: e54967.

Alpaydin E. 2014. *Introduction to Machine Learning 3rd edition*. Cambridge: MIT Press.

Amin MR, Hasan M, Arnab SP, DeGiorgio M. 2023. Tensor decomposition-based feature extraction and classification to detect natural selection from genomic data. *Mol Biol Evol* **40**: msad216.

Arnab SP, Amin MR, DeGiorgio M. 2023. Uncovering footprints of natural selection through spectral analysis of genomic summary statistics. *Mol Biol Evol* **40**: msad157.

Baker M. 2016. 1,500 scientists lift the lid on reproducibility. *Nature* **533**: 452–454.

Baumdicker F, Bisschop G, Goldstein D, Gower G, Ragsdale AP, Tsambos G, Zhu S, Eldon B, Ellerman EC, Galloway JG, *et al.* 2022. Efficient ancestry and mutation simulation with msprime 1.0. *Genetics* **220**: iyab229.

Benjamini Y, Hochberg Y. 1995. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J R Stat* **57**: 289–300.

Collins FS, Tabak LA. 2014. Policy: NIH plans to enhance reproducibility. *Nature* **505**: 612–613.

Demšar J. 2006. Statistical comparisons of classifiers over multiple data sets. *J Mach Learn Res* **7**: 1–30.

Descôteaux J. 2007. Statistical power: An historical introduction. *Tutor Quant Methods Psychol* **3**: 28–34.

Dietterich TG. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput* **10**: 1895–1923.

Durvasula A, Sankararaman S. 2019. A statistical model for reference-free inference of archaic local ancestry. *PLoS Genet* **15**: e1008175.

Durvasula A, Sankararaman S. 2020. Recovering signals of ghost archaic introgression in African populations. *Sci Adv* **6**: eaax5097.

Fawcett T. 2006. An introduction to ROC analysis. *Pattern Recognit Lett* **27**: 861–874.

Flagel L, Brandvain Y, Schrider DR. 2019. The unreasonable effectiveness of convolutional neural networks in population genetic inference. *Mol Biol Evol* **36**: 220–238.

Gower G, Picazo PI, Fumagalli M, Racimo F. 2021. Detecting adaptive introgression in human evolution using convolutional neural networks. *eLife* **10**: e64669.

Haller BC, Messer PW. 2023. SLiM 4: Multispecies eco-evolutionary modeling. *Am Nat* **201**: E127–E139.

Heil BJ, Hoffman MM, Markowetz F, Lee SI, Greene CS, Hicks SC. 2021. Reproducibility standards for machine learning in the life sciences. *Nat Methods* **18**: 1132–1135.

Hoenig JM, Heisey DM. 2001. The abuse of power. *Am Stat* **55**: 19–24.

Huang X, Struck TJ, Davey SW, Gutenkunst RN. 2023. dadi-cli: Automated and distributed population genetic model inference from allele frequency spectra. bioRixv. https://doi.org/10.1101/2023.06.15.545182, last accessed August 31, 2024.

Huang X, Rymbekova A, Dolgova O, Lao O, Kuhlwilm M. 2024. Harnessing deep learning for population genetic inference. *Nat Rev Genet* **25**: 61–78.

Hudson RR. 2002. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics* **18**: 337–338.

Korfmann K, Gaggiotti OE, Fumagalli M. 2023. Deep learning in population genetics. *Genome Biol Evol* **15**: evad008.

Lauterbur ME, Munch K, Enard D. 2023a. Versatile detection of diverse selective sweeps with Flex-Sweep. *Mol Biol Evol* **40**: msad139.

Lauterbur ME, Cavassim MIA, Gladstein AL, Gower G, Pope NS, Tsambos G, Adrion J, Belsare S, Biddanda A, Caudill V, et al. 2023b. Expanding the stdpopsim species catalog, and lessons learned for realistic genome simulations. *eLife* **12**: RP84874.

Mölder F, Jablonski KP, Letcher B, Hall MB, Tomkins-Tinch CH, Sochat V, Forster J, Lee S, Twardziok SO, Kanitz A, et al. 2021. Sustainable data analysis with Snakemake. *F1000 Res* **10**: 33.

Nekrutenko A, Galaxy Team, Goecks J, Taylor J, Blankenberg D. 2018. Biology needs evolutionary software tools: Let's build them right. *Mol Biol Evol* **35**: 1372–1375.

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. *Adv Neural Inf Process Syst* **32**: 8026–8037.

Racimo F, Marnetto D, Huerta-Sánchez E. 2017. Signatures of archaic adaptive introgression in present-day human populations. *Mol Biol Evol* **34**: 296–317.

Raff E. 2019. A step toward quantifying independently reproducible machine learning research. *Adv Neural Inf Process Syst* **32**: 5485–5495.

Ray DD, Flagel L, Schrider DR. 2024. IntroUNET: Identifying introgressed alleles via sematic segmentation. *PLoS Genet* **20**: e1010657.

Romieu J, Camarata G, Crochet PA, de Navascués M, Leblois R, François Rousset. 2024. Performance evaluation of adaptive introgression classification methods. bioRixv. https://doi.org/10.1101/2024.06.12.598278, last accessed August 31, 2024.

Schrider DR, Kern A. 2018. Supervised machine learning for population genetics: A new paradigm. *Trends Genet* **34**: 301–312.

Shreffler J, Huecker MR. 2024. Type I and type II errors and statistical power. In *StatPearls*. Treasure Island: StatPearls Publishing.

Soric B. 1989. Statistical "discoveries" and effect-size estimation. *J Am Stat Assoc* **84**: 608–610.

Thomas L. 1997. Retrospective power analysis. *Conserv Biol* **11**: 276–280.

Tommaso PD, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. 2017. Nextflow enables reproducible computational workflows. *Nat Biotechnol* **35**: 316–319.

Walsh I, Fishman D, Garcia-Gasulla D, Titma T, Pollastri G, ELIXIR Machine Learning Focus Group, Harrow J, Psomopoulos FE, Tosatto SCE. 2021. DOME: Recommendations for supervised machine learning validation in biology. *Nat Methods* **18**: 1122–1127.

Wasserman L. 2003. *All of Statistics: A Concise Course in Statistical Inference*. New York: Springer.

Xia F, Zhang MJ, Zou J, Tse D. 2017. NeuralFDR: Learning discovery thresholds from hypothesis features. *Adv Neural Inf Process Syst* **30**: 1540–1549.

Yelmen B, Jay F. 2023. An overview of deep generative models in functional and evolutionary genomics. *Annu Rev Biomed Data Sci* **6**: 173–189.

Zhang X, Kim B, Singh A, Sankararaman S, Durvasula A, Lohmueller KE. *MaLAdapt* reveals novel targets of adaptive introgression from Neanderthals and Denisovans in worldwide human populations. *Mol Biol Evol* **40**: msad001.