June 19, 2018

Robust and reproducible classification of rare cellular subsets/signatures (RCS) in single-cell technologies within a transfer learning framework.

Presented to the Division of Bioinformatics and Computational Biology (BCB) part of the Department of Medical Informatics and Clinical Epidemiology (DMICE) and the Oregon Health & Science University School of Medicine in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Eisa Mahyari

@eisamahyari

Bachelor of Science in Biology and Chemistry, Portland State University, 2006
Masters of Science in Biochemistry and Molecular Biology, Oregon Health & Science University, 2008

Oregon Health & Science University (OHSU)
Dept. Medical Informatics and Clinical Epidemiology (DMICE)
Div. Bioinformatics and Computational Biology (BCB)

OHSU, Oregon USA
June 2018

School of Medicine

Oregon Health & Science University

Certificate of Approval

This is to certify that the PhD Dissertation of

Eisa Mahyari

"Robust and reproducible classification of rare cellular subsets/signatures (RCS) in single-cell technologies within a transfer learning framework"

Has been approved

Dissertation Advisor – Shannon K. McWeeney Ph.D.

Committee Member – David M. Lewinsohn M.D.Ph.D.

Committee Member – Michael Mooney Ph.D.

Committee Member – Christina Zheng Ph.D.

Committee Member – Evan Lind Ph.D.

External Examiner – Andrew Adey Ph.D.

Acknowledgments

This dissertation has come to fruition with the combined efforts and support of my advising faculty, friends, and family. It is a culmination of personal and professional struggles as well as growth. There is no possible way to quantify the efforts of the instrumental individuals that have help lead me to this end-point. However, with certainty I can state that without the expertise and wisdom of my graduate school mentor and Ph.D. thesis advisor, Prof. Shannon McWeeney, this dissertation would not have been possible. At every stage of my scientific and professional development in this period, she was always keenly aware of my needs and thus was able to push for a continuum of excellence, as much as I could deliver. I am highly grateful to Prof. David Lewinsohn and his lab for expertise, patience, and data. I also like to thank Dr. Marielle Gold, who also was a significant contributor to my understanding of the immunobiology in the context of TB infection, treatment, and prophylaxis. Additionally, I am very appreciative of Dr. Mike Mooney for in-depth proofing of drafts and discussion in the development of this dissertation research. Combined, I am greatly appreciative of my dissertation advising committee for their support and feedback in improving my dissertation as well as guiding my professional development.

I want to thank my parents for their many sacrifices that has allowed me to follow my wish to peruse science and education, freely and unbounded. Emotionally dealing with the difficult graduate school period, filled with a roller-coaster of trials and tribulations, would not have been possible without my partner Marianna and our chihuahua Kenny.

Dissertation Committee:

Mentor: …….. Prof. Shannon K. McWeeney Ph.D.
Chair: ………. Prof. David M. Lewinsohn M.D. Ph.D.
Member: …….. Asst. Prof. Michael Mooney Ph.D.
Member: …….. Asst. Prof. Evan Lind Ph.D.
Member: …….. Asst. Prof. Christina Zheng Ph.D.
External Faculty: …….. Asst. Prof. Andrew Adey Ph.D.

Advising/Mentorship:
Shannon K. McWeeney Ph.D., David M. Lewinsohn M.D. Ph.D., and Marielle C. Gold Ph.D.

Extended Support:
Deborah A. Lewinsohn M.D. and Ted Laderas Ph.D.

Also, thank you to all of the DMICE staff, faculty, and fellows, in addition to the Lewinsohn lab members.

Funding:
2015-16:
OHSU Division of Pulmonary and Critical Care Medicine
NIH T32 grant: 5T32HL083808-07
2014-15:
OHSU Division of Pulmonary and Critical Care Medicine
NIH T32 grant: 2T32HL083808-06
2013-14:
OHSU Knight Cancer Institute
OHSU Molecular Microbiology and Immunology

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Rare cellular subsets are a marginally infrequent set of cells, represented by a transcriptional signature or an immunophenotype. The importance of RCS clinically in various clinical domains from oncology to infectious diseases has propelled research to define clinically actionable signatures. In this dissertation we asked if there is a robust and reproducible computational method to classify RCS; improving the downstream hypothesis testing for research or clinical needs. For example, being able to detect 1 leukemic cell from 10,000 healthy cells, has been shown to be prognostic of outcome in adult acute lymphoblastic leukemia. However, detection of specific RCS is not sufficient to save lives and also improve future therapy and prophylaxis. That is why we believe the potential future application of the methods researched an developed herein, needs to be combined in a systems approach; integrating multiple sources of evidence from various '-omics'-based analyses to provide targeted calls within a precision medicine framework. However, there are current limitations in computational methods that utilize single-cell data, especially in the context of RCS. Such limitations arise from the high-throughput nature of single-cell technologies that demand robust and generalizable computational methods that scale and can handle high-dimensionality.

In this investigation, we address two current limitations of computationally classifying cellular subsets: Transcriptionally, how well can RCS be classified? And how well does a classifier generalize across datasets, even if the sources/platforms differ? We assume that the signature or phenotype of interest has been previously characterized and data is available for training machine learning algorithms; parallel to human experts who train on previous literature and data. With this assumption, the primary machine learning methods of interest fall under the umbrella of supervised learning. However, two major assumptions of many of such methods is that A) the training and testing data are independent, but identically distributed (i.i.d) and B) the marginal frequency of the population of interest does not significantly differ between training and testing. Another machine learning domain, called transfer learning (TL) does not have these assumptions. Based on such theoretical considerations, we hypothesized that within a TL framework, both of the computational limitations i.e., RCS and generalizability can be researched and potentially addressed.

For the development of this framework, we leveraged a previously published TL algorithm, designed to classify specific immunophenotypes (IPs) by flow cytometry (FC) data. However, as the original code nor the datasets of the workshop manuscript were available, our initial goal was the implementation of an equivalent framework. The 'baseline' RTL version, closely adheres to the previously published approach [1]. Once implemented and validated with simulated and pilot FC data, to improve the overall automation of the pipeline and classification robustness and generalizability, we proposed and developed extensions to several of the modules in this framework i.e., the extended version. Finally, to improve the precision of the classification call in the context of RCS, we introduced the utility of the area-under the density curve (AUD) parameter transfer (PT) option; transferring the trained positive class marginal frequency (in addition to a small random Gaussian error) to aid the classification of the testing set. We plan for an open-source release both versions, combined as the RTL framework, in conjunction with the publication of its manuscript.

To benchmark the RTL framework, we utilized publicly available scRNASeq data with known cell cycle labels, paralleling our approach with a previously published assessment with 6 different classifiers. This meant that it was necessary to rank-normalize the expression matrices. However, such task-specific normalizations can introduce bias in the classification task. Therefore, we ran parallel classification runs using the log-scaled gene counts directly as well as conducting quantile and log-median-absolute-value normalization for comparison. This also aids in evaluating the hypothesis that in TL can adapt/generalize across differently sourced datasets without the need to perform task-specific normalization. Briefly, we demonstrate equal or higher $F1_{score}$ within the RTL framework than those previously published, however, confounded by the task-specific normalization methods performed prior to the classification. In our internal validation (i.e., leave-one-out 5-fold cross-validation), quantifying the performance of the RTL framework, the highest $F1_{score}$ and overall accuracy was achieved with the log-median-absolute-value (LogMAV) normalization. However, in our external validation (i.e., 5-fold CV on two datasets from differences sources, such that training is achieved on one and inference on the other), which quantifies the classifier generalizability, we found LogMAV normalization produced the lowest $F1_{score}$, regardless of the two versions of the RTL framework. Therefore, the next highest computed $F1_{score}$, was found on the log-scaled counts; in both of the internal and external validations. Next, in evaluating the classification of RCS down to 1% of the total, we achieved the highest $F1_{score}$ when utilizing the PT feature with the baseline version. In the RCS context, we find our extended implementation is biased towards precision as opposed to the baseline which generally has higher recall than precision; consequently, PT has a much greater impact on this extended version when classifying RCS.

Finally, to demonstrate and evaluate classification of a clinically valuable immunophenotype compared to manual gating in flow cytometry data, we utilized the publicly available HVTN080 dataset. We designed two train/test set schemes to evalu-

[1] The baseline version of the RTL framework is not an exact 1:1; to accommodate scRNASeq data as well as utility of R-specific libraries several changes were made.

ate the performance of the RTL framework. As expected, a higher $F1_{score}$ was achieved with the ensemble scheme where multiple trained classifiers, train on non-identical sets of FC-samples/files were used for inference on the same testing set to obtain a probability of association to the positive class of interests for all the cells/events. Comparing the baseline and extended versions in this context we observed that the latter produced calls with higher variance, thus on average lower performing the the baseline version.

In conclusion, we have made available an open-source TL framework, which can be applied to several different classification tasks. Proper evaluation is required to determine which version is appropriate and is ideal to utilize as well as pre-processing procedures such as normalization, projections, and transformations. In this dissertation we perform such evaluations in the context of single-cell analysis with scRNASeq and FC data.

Keywords and acronyms:
Rare Cellular Subsets/Signatures (RCS)
R-based Transfer Learning (RTL) framework
Flow Cytometry (FC)
Single Cells transcriptional Sequencing (scRNASeq)
Support Vector Machine (SVM)
Transfer Learning (TL)
Human Immunodeficiency virus (HIV)
Minimum Residual Disease (MRD)

1 Introduction

The capacity to study specific subsets of cells, especially rare cellular subsets (RCS) such as those associated with a disease-phenotype is at an inflection point; the high-throughput nature of such studies necessitates robust and generalizable computational methods to be developed and tested. Single-cell analysis has flourished since the early applications in Flow cytometry (FC) [14]. Currently, fluorescent-FC can optimally quantify up to 20 features on each of the single cells [15]. New mass-based cytometry technologies (e.g. CyTOF) have enabled measuring over 100 features on single cells [16]; also useful for barcoding samples in multiplexed runs. More recently, it is even possible to evaluate single cells transcriptionally (scRNASeq), with specialized applications that combine multiple technological platforms [17]. Which technology we use, usually depends on feasibility within the context of a specific hypothesis and budget. In the near future, such single-cell data (or a respective analyzed summary) will be used in the clinic, guiding evidence-based diagnostics, prognostics, and treatment optimization plans. For this integration to occur, it is expected that the underlying computational processes must have high performance statistics.

In a classification task, the scope of this work, high-performance means when a positive call is made, most are actually positive (high precision) with a high proportion of the total negatives (specificity) and positives (recall) identified correctly. As our major focus is on rare cellular subtypes/signatures (RCS), the selection of such a statistical measure becomes critical. For example, when the marginal frequency of the positive cases is 1%, 99% accuracy can be achieved by simply inferring all as negatives. Furthermore, such measures can be confounded by the inherent limitations (or assumptions) of a selected classification method, as well as various factors pertaining to the data acquisition (protocols, technology, reagents, etc) and the pre-processing procedures (cleaning, scaling, normalization, feature selection, etc). Herein, we explore these issues as they are pertaining within the scope of our investigation in classifying cellular subsets/signatures with a transfer learning (TL) framework. The main computational motivation for this research was the robust classification cellular subsets or signatures, specifically RCS, that are deterministic of disease and health. The immediate return from this TL framework is to conduct reproducible research that can be easily scaled to process high-throughput single-cell data. The long-term extension is the integration of such classification models within a precision medicine framework, integrated with additional -omics data and analyses to make systems based, targeted calls [1]; from stratification of patients in clinical trials to guiding optimal treatments and prophylaxis (Figure 1B).

1.1 Motivation

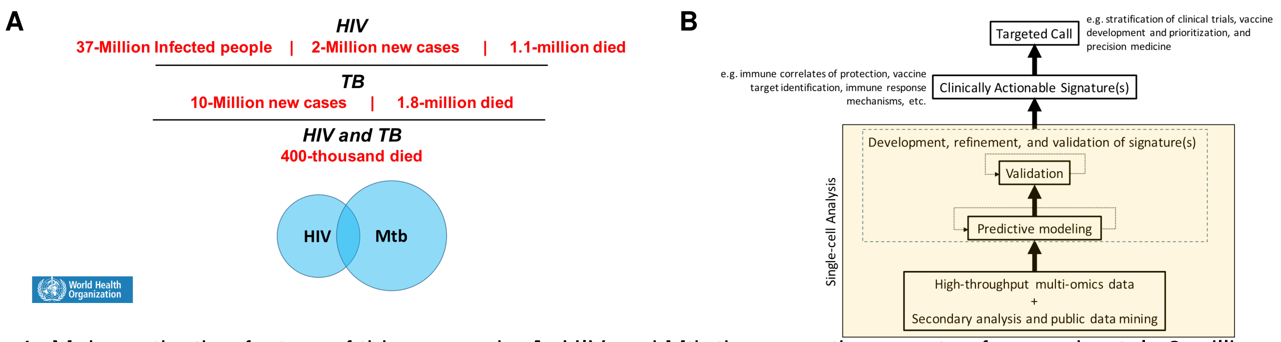

A
HIV
37-Million Infected people | 2-Million new cases | 1.1-million died
TB
10-Million new cases | 1.8-million died
HIV and TB
400-thousand died
HIV
Mtb
World Health Organization
B
Targeted Call
e.g. stratification of clinical trials, vaccine development and prioritization, and precision medicine
Clinically Actionable Signature(s)
e.g. immune correlates of protection, vaccine target identification, immune response mechanisms, etc.
Development, refinement, and validation of signature(s)
Validation
Predictive modeling
High-throughput multi-omics data + Secondary analysis and public data mining
Single-cell Analysis

Figure 1: Main motivating factors of this research. **A**: HIV and Mtb the causative agents of approximately 3 million deaths in 2015. **B**: Systems based integration of -omics based data towards targeted calls within a precision medicine framework; adapted from [1].

Computationally, we are motivated by a current bottleneck in classifying rare cellular subsets (RCS). Clinically several challenges have motivated the study of cellular subsets/signatures, especially RCS as determinants of health and disease. For example, according to the World Health Organization (WHO), the human immunodeficiency virus (HIV) and Mycobacterium tuberculosis (Mtb) were the causative agents of 3-million deaths worldwide in 2015 [18, 19]. Access to healthcare and medicines has been critical in reducing the rates of new infections as well as improving the prognosis of infected people in many parts of the world. However, even in the developed world, specific communities of people are not as fortunate. Furthermore, new studies for both pathogens have shown resurgence of drug-resistant genotypes [20]. Therefore, there is a major motivation to unravel the pathophysiology of such infectious diseases, both at the scope of the population and at the individualized scope, so that we can develop and disseminate targeted treatments and vaccines to eradicate them.

Specific immunophenotypes (cellular subsets/signatures) in various pathologies such as those caused by HIV or Mtb have been shown to be clinically important; useful for diagnostic and prognostic [21, 22, 23, 24] needs. However, there is still work to be done integrating and translating such findings into clinically actionable targets or calls within the context of precision medicine. This is now feasible with recent -omics based technologies and computational/analytical methods which have enabled the unraveling of disease/health mechanisms and molecular networks (e.g. phenomics, epigenomics, genomics, transcriptomics, proteomics) [25].

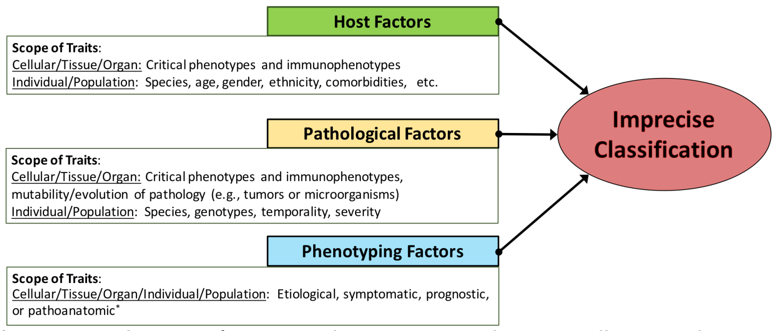

Host Factors
Scope of Traits:
Cellular/Tissue/Organ: Critical phenotypes and immunophenotypes
Individual/Population: Species, age, gender, ethnicity, comorbidities, etc.
Pathological Factors
Scope of Traits:
Cellular/Tissue/Organ: Critical phenotypes and immunophenotypes, mutability/evolution of pathology (e.g., tumors or microorganisms)
Individual/Population: Species, genotypes, temporality, severity
Phenotyping Factors
Scope of Traits:
Cellular/Tissue/Organ/Individual/Population: Etiological, symptomatic, prognostic, or pathoanatomic*
Imprecise Classification

Figure 2: Three Major factors and examples at various scopes that contribute to imprecise classification. An example of miss-classification may be diagnosing active vs. latent disease due to unspecific clinical definitions pertaining to host, pathological, or phenotyping factors; highly problematic for clinical trial recruitment.

One of the current major limitations of clinical trials and medicine in practice is that many patients do not respond to treatments or prophylaxis. Fiscally, this translates to over $300 billion annually in the U.S. alone [26]. Heterogeneity is a significant contributor to this problem: A) Host/Patient factors: age, gender, ethnicity, etc. B) Pathological factors: species, genotype, temporality, severity, mutability, etc. C) Phenotyping factors: Etiological, symptomatic, prognostic, or pathoanatomic [27, 1]. Therefore, there is a major need for high-throughput multi-omics data combined with secondary analysis from public data to identify relevant signatures and develop predictive models that can eventually identify clinically actionable signatures; the highly valuable return of such efforts would be making targeted calls such as the stratification of patients for clinical trials or the development and prioritization of treatments or vaccines [1]. Current efforts to overcome such limitations is supported by the results observed in targeted cancer therapy; pathological molecular profiling and molecular-based patient stratification.

Single-cell technologies have had a recent rebirth due to modern computing capabilities. Shifting away from profiling molecular targets on proteins and sugars on or inside the cells towards transcriptional profiling of single cells. That said, every research hypothesis or clinical diagnosis does not require transcriptional analysis of single cells. However, it has been shown that in many pathologies such as cancer, complex diseases, and infectious diseases many cell types are involved; both epigenetic and genetic changes in these cell types can be quantified by transcriptional analysis and correlated to the pathology.

1.2 Single-cell technologies

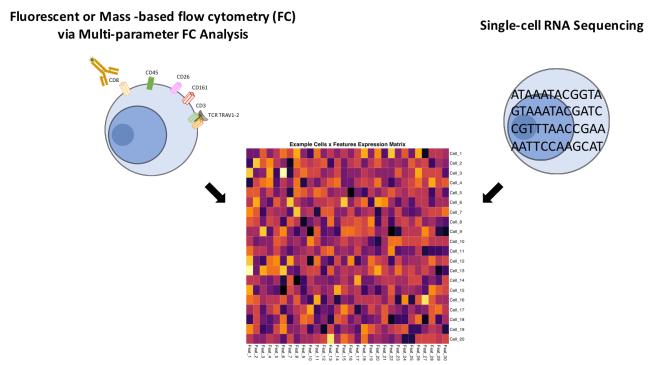

Fluorescent or Mass -based flow cytometry (FC) via Multi-parameter FC Analysis
Single-cell RNA Sequencing
CD8
CD45
CD26
CD161
CD3
TCR TRAV1-2
Example Cells x Features Expression Matrix
ATAAATACGGTA
GTAAATACGATC
CGTTTAACCGAA
AATTCCAAGCAT

Figure 3: single-cell technologies and scope: Flow cytometry (FC) and single-cell RNA sequencing (scRNASeq) can be summarized similarly as a per-cell expression matrix of features.

1.2.1 Fluorescent and Mass-labeled Flow Cytometry (FC)

Flow-cytometric based experiments begin with the acquisition of isolated cells in suspension and end with the analysis of '.fcs' files [28] produced during acquisition. Once the cells of interest are in appropriate aliquots, antibodies tagged with fluorophores or metal beads are used to label the cells based on phenotypic features of interest; usually with cell-surface markers combined with other internal targets of interest. Laminar flow within the fluidics system of the flow cytometer pushes individual cells forward in a queue, eventually passing the platform's detector. Multi-parameter analysis then enables discrimination of single-cells based on the quantified labels [29]. Traditionally, this approach has been a manual process called 'gating' that defines boundaries surrounding density-clouds in bi-axial scatter plot; these decisions are based on *a priori* definitions of immunophenotypes acquired from the literature. Computational methods automate this task in robust, scalable, and reproducible frameworks [30]. Many of the limitations of manual or automated gating that fall under the umbrella of confounding factors (technical or biological), can be overcome by standardization of the entire experimental and analytical protocols. Additionally, multiplexing significantly improves the signal to noise ratio for high-throughput demanding experiments by minimizing the noise across the samples. Overall, the goal is to significantly improve the quality of analysis by increasing the statistical power that can be achieved [31]. Although much effort has been put on ways to improve such cohort analysis in FC, several challenges remain; for example the difficulty of classifying rare cellular subtypes (RCS), manually or computationally.

In fluorescent FC, the lasers in the flow cytometer excite the antibody-bound fluorophores, which causes them to emit a spectrum (wavelength) of light to return to the lower energy state. The more features per cell in a given experiment, the more fluorochromes are needed, but the overlap in the emission spectra from all these fluorochromes set significant limitations such as an upper-bound limit on the optimal number of fluorochromes per experiment. In the context of RCS, the fluorochrome(s) used to identify them must have an emission spectrum that is bright, low variant, and with minimal to zero interference from other fluorochromes emission spectra in the experiment. Ideally, more than one marker can be used to improve the classification by reducing the false positives. Mass cytometry (CyTOF) overcomes this limitation as the metals with different masses are accelerated and their time of flight (TOF) to a detector is measured. Furthermore, there is no auto-fluorescence and there are a large set of metal-tags to choose from. However, during experiment design, it should be noted that the metal elements naturally have isotopes with varying purity which can cause overlapped signals.

The antibodies that tag a specific target molecules in FC analysis (and fluorescent microscopy) have specific binding affinities based on their sequence and 3-dimensional topology. This means, even for a major marker such as CD3, a ubiquitously used T cell marker in FC, due to the biological diversity of this molecule relative to the clone(s) producing the antibody, we expect a variable binding affinity curve. Within a single sample, there are additional sources of unwanted experimental variation. For example, the relative intensity of a marker on a given cell, is the sum of the fluorochrome emissions at the moment of detection. Therefore the amount and location of the target molecule's expression in the context of the hypothesis becomes critical. Finally, in the context of RCS, this translates to the target marker(s) used must be well-expressed in the population of interest and experimentally distinguishable from controls.

Currently, both platforms of FC are used for medical and research needs. More recently, mass cytometry [32] has been identified as a valuable tool in translational clinical research [33], but there are some limitations pertaining this instrument. Primarily, reproducibility is the key challenge to research that utilizes CyTOF. That is because generally CyTOF has been used as an exploratory expedition [34, 35, 36, 37]. This, of course, is a problem statistically speaking as the exploration confounds the utility of the generated dataset; mandating multiple datasets to be generated to validate any findings via hypothesis testing. The Second major limitation and a source of bias for exploratory research is that FC will only capture the set of probes previously decided in an experimental protocol; in other words, potentially many targets may not be quantified, simply because the antibodies to those targets are not present in the cocktail used; this is analogous to the criticisms of microarray-based transcriptional profiling compared to next-generation RNA sequencing. That said, there still a major applied benefit to high-throughput clinical or research use of CyTOF because of its multiplexing capacity, mixing multiple samples within a single run by barcoding.

1.2.2 High-throughput single-cell RNA-sequencing (scRNASeq)

High-throughput RNA-sequencing at the single-cell level i.e., scRNASeq, is a recent and major technological advancement predicated on a few decades of gene expression profiling of bulk cells [38]. The benefit of using next-generation sequencing is that it overcomes the limitations and biases microarrays posed on transcriptomic fingerprinting, i.e. because sequence-based technologies do not use probes [25]. Two recent reviews comprehensively discuss scRNASeq methods, instruments, utility, impact, and current limitations [39, 40]. Briefly, there are many medical tests or research hypotheses that can still be answered with previous technologies such as bulk-cell and non-sequencing transcriptional profiling. However, when we consider that for each bulk sample (e.g., tissue sample) we are measuring the mean expression of any gene/transcript for that entire sample, it is evident that A) we lose being able to discriminate cellular heterogeneity B) if the cells of interest are low in frequency, their entire signal is diluted and C) we lose the capacity to quantify transcriptional kinetics amongst the potential cellular subtypes. That is why only when a research hypothesis is justified to use scRNASeq, it is best to utilize this technology. We refer to [25] as a review of several sequencing platforms; there are important considerations such as cross-platform comparisons that are not trivial and may affect the downstream analyses and hypothesis testing.

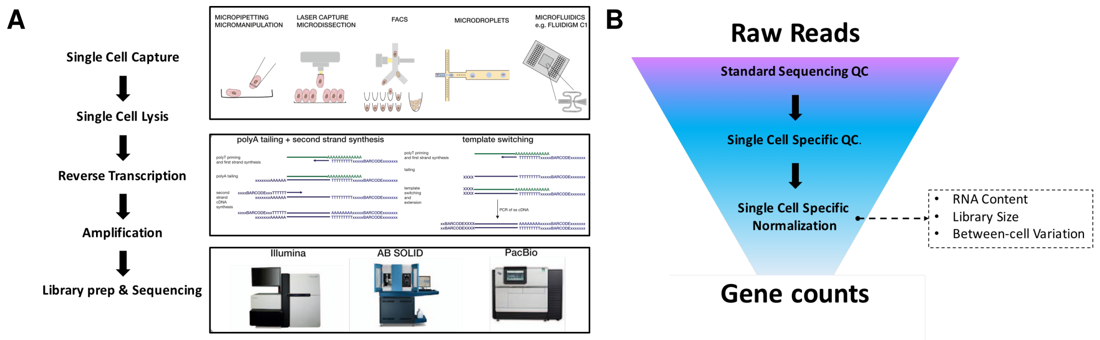


Figure 4: An overview of the acquisition and pre-processing involved in scRNASeq, adapted from [2]. **A** From single cells suspended in media to sequenced reads per cells. **B** From reads to gene counts, several QC steps remove anomalous and problematic cells. The counts are normalized to remove unwanted technical and biological noise based on the specific protocol utilized. In the context of RCS of interest, such procedures could be bias to remove rare cells as they may be considered anomalous.

Briefly, summarized in Figure 4, the current approach to scRNASeq starts with experimentally isolating the cells of interest; using methods that minimize tissue-isolation transcriptional changes such as magnetic-activated cell sorting (MACS), laser capture dissection (LCM), or fluorescence-activated cell sorting (FACS). The cells are then lysed and their mRNA molecules [2] are captured and reverse transcription converts them to the more stable respective cDNA molecules. Next, the cDNA molecules are amplified and profiled by next-generation sequencing [2]. As with other high-throughput -omics data, the first step in the bioinformatics pipeline is quality control. That is because the multiplex process may capture a range of cells: from a perfect, healthy, unstressed and undamaged single cell to nothing at all. Therefore, low-quality

[2]polyadenylated fraction of mRNA

cells need to be removed so that they don't bias the downstream processes. For an in-depth and recent review of the scRNASeq pre-processing pipelines, we refer to [41].

The gene-level expression matrix for each of the individual cells is the resulting data type after the pre-processing bioinformatics. If unique molecular identifiers (UMIs) were used, transcript molecules are counted directly, where as if the experiment lacked UMIs, counts are estimated by algorithms such as htseq-count; a popular tool that counts the overlap of reads with genes [42]. Alternatively, relative expression such as transcripts per million mapped reads (TPM), counts per million mapped reads (CPM), reads per kilobase per million mapped reads (RPKM) or fragments per kilobase per million mapped reads (FPKM) can be used. Overall, such counts are considered similar in terms of signal distribution to FC which is broadly attributed to having at least two modes indicating 'on' or 'off' states [43] supporting the application of similar computational tools, at the expression matrix scope, across these data platforms.

A critical next step in the scRNASeq processing pipeline, so that across the cells within an experiment, the expression data are comparable, three potential normalizations procedures are possible depending on the experimental design and protocol: A) Normalization for RNA content, B) normalization for library size, and C) normalization of between-cell variation. For example, extrinsic/artificial spike-ins [44], are commonly used in scRNASeq experiments to help determine cell-specific scaling factors. More specifically, because the spike-ins are added to the cell lysates at a known constant concentration, it can be concluded that when the ratio of reads mapped to the genome over the number of reads mapped to spike-ins is lower than other cells on average, that cell is anomalous and considered to be removed from downstream analysis; low RNA concentration is the most probable etiology.

Currently, a commonly used pipeline is a software package called Scater [45]. Not only does this process prove a uniform and fast computable data structure for scRNASeq datasets, in the commonly used R language, it also provides a pipeline to run pre-processing, quality control, normalization and compute pertinent summary statistics and visualizations. However, in the context of RCS, this process can be biased, and potentially remove them as aberrant events. Of note, is the consideration of technical dropouts, where the expression of a gene is not detected, but expected to be so; attributed to capture failure in the reverse transcription stage. This is an important consideration for experimental design and pre-processing. The scope of this dissertation relevant to the processing of the scRNASeq data is beyond addressing this limitation; we utilize a pre-processed dataset for benchmarking purposes of machine learning classification. In this dataset, the marginal frequency of the class labels are not rare, but rather as described in a later section, we derive rare labels by in-silico methods.

At the scope of the expression matrix, a current major difference between scRNASeq compared to FC is the number of cells assessed in each experiment, as well as the number of experimental samples. Furthermore, in the context of RCS, it is important to consider how the data was created and process to get to the expression matrix level. A common practice in the QC processing of scRNASeq data is removing low-expressing genes, which in fact belong to rare cells. In other words, there is a difference in the false positive/negative rates between the generation and processing of FC data represented by detection by fluorescence intensity or mass-based time-of-flight (TOF) compared to scRNASeq counts, a measure of expressed mRNA that is noisy and partially stochastic by nature. Combined, the transition towards high-dimensionality has introduced new statistical challenges and limitations, mandating the production of new or upgrading and adapting previously developed reproducible and scalable computational methods, at various scopes in the analysis pipelines. For this dissertation, as our goal is to benchmark our developed machine learning algorithm to previous methods, we utilize the processed expression matrices directly from the respective studies.

1.3 Frequent and Rare Cellular Subsets (RCS)

Rare cellular subsets (RCS) are distinct phenotypes/clusters/signatures of interest, common and important in biology [4]. For example, in a recent study, it was shown by detection of residual cancer cells across several time points during disease treatment was prognostic; in this minimum residual disease (MRD) context of adults with acute lymphoblastic leukemia (ALL), MRD was detected at 0.0001 frequency (0.01%) [3]. In another context, antigen-specific T cells may be the subset of interest which can be as rare as 1E-6 [3]. Evaluating the cytokine responses of these t cells means detecting the

[3]It is, in fact, the hallmark of the adaptive system to create many rare antigen-specific cells, that can expand when needed.

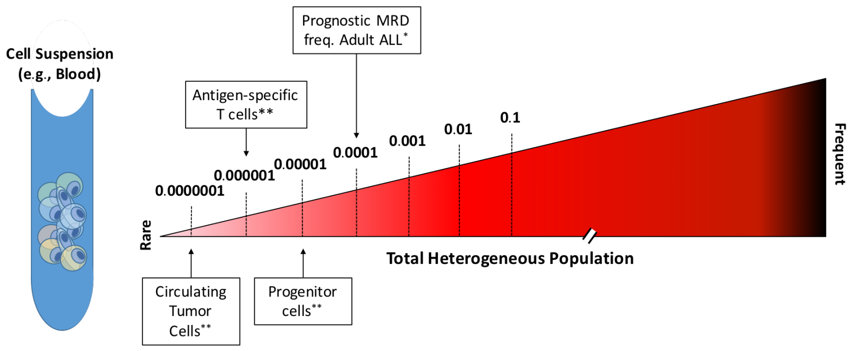

Cell Suspension (e.g., Blood)
Prognostic MRD freq. Adult ALL*
Antigen-specific T cells**
0.0000001
0.000001
0.00001
0.0001
0.001
0.01
0.1
Rare
Frequent
Total Heterogeneous Population
Circulating Tumor Cells**
Progenitor cells**

Figure 5: The definition of rare cellular subsets by frequency with examples of critical subsets. * [3], ** [4].

responding t cell populations at frequencies of 0.01 or less [46]; although usually, this means conducting multi-parameter analysis, which enriches for the rare cellular subsets of interest by gating out the non-relevant majority phenotypes. Other biologically important examples, such as progenitor cells, stem cells, and circulating tumors, they can have marginal frequencies as low as 1E-7 [4]. However, for practical exploration of such extremely rare subsets, a specific hypothesis based on a clinical need is required. There are several experimental methods to sort or enrich for a population of interest that minimize transcriptional (as well as phenotypic) changes. Therefore, a practical definition of an RCS is a subset with a marginal frequency in [0.0001, 0.01] [4]; in this study, our scope is within the upper limit of this range due to data availability and computational feasibility.

Three main reasons why it is difficult to identify RCS from a mixture, manually or computationally. A) effect size, a standardized measure of difference between two populations and B) marginal frequency of the RCS of interest and C) total sample size; having enough 'N' representatives for proper determination of a classification boundary as well as training machine learning algorithms. In applied work, each population may be associated with a specific false positive or false negative rates. Since with high-throughput technologies we are almost always in high-dimensional (HD) space, feature selection is usually necessary as high-dimensionality confounds classification by introducing unwanted variance. The population of interest (i.e., phenotypes or signatures) may be gradients rather than discrete subsets. And there maybe confounding factors (technical and/or unwanted biology) that especially impede the detection of RCS.

1.3.1 Rare cellular subsets (RCS), rare-event/anomaly detection, and class-imbalanced classifiers

There are several approaches in applying machine learning algorithms to classify a rare subset within a sample. Generally, in the context of supervised learning, when the marginal frequency of a positive class is considerably smaller than the negative class, the inference is expected to favor the larger class. By design, such algorithms maximize the overall number of correct predictions because an equal cost of miss-classification is given to both classes, which can be expect to favor the larger subset in the case of class-imbalance. This problem can be even more difficult to address in high-dimensional data [47]; especially in biomedical research as the sample sizes tend to be small. As we demonstrated in our assessment of the 1D-simulated evaluation, also supported by a previous study in [48], there are three reasons that contribute to poor classification in the context of rare-event detection: A) effect size between the positive and negative class, i.e., data complexity B) the marginal frequency of the positive class and C) total sample size (used for training and inference). We also show in a later section that feature selection is crucial in classification tasks which we categorize under the umbrella of 'changing the effect size'. Finally, we note that the effect of confounding factors (technical or unwanted biological) that potentially impede machine learning classification, is amplified in the context of RCS.

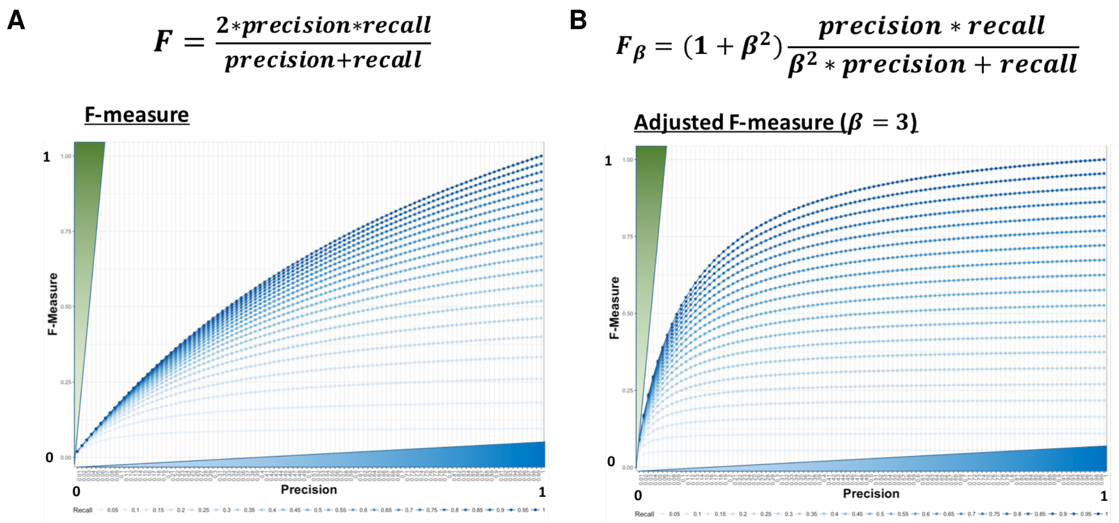
Because of the difficulty and cost associated with the development of datasets, there are several in silico approaches to creating class-imbalanced data from an existing dataset. For a review, we refer to the [49, 47, 48] articles. Briefly, the two methods explored in this dissertation are the random over-sampling and random under-sampling methods (Figure 48). The former, via random replication (sampling with replacement), is known to potentially increase the likelihood of over-fitting due to multiple exact replicas in the data. The latter, via random elimination of the majority class, has the limitation of excluding potentially valuable examples. Several additional methods exist, which were not explored in this dissertation research, but are potential future directions to attempted and evaluate: Tomek links, Condensed Nearest Neighbor (CNN) Rule, One-sided selection, CNN + Tomek links, Neighborhood cleaning rules, Smote, and others reviewed [49].

1.4 Supervised and unsupervised methods, an overview

Statistical learning (and inference) also known as machine learning are methods to describe how data is used to bring about higher understanding. Simply, sometimes the task is to use previous data to make a future prediction. For example, statistical regression is a mathematically-based method to examine and build models of data (such as the linear model $y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \cdots + \beta_m \cdot x_2$). Building such a model means training on data where y and **x** are known to find the optimal values for the coefficients β_m for all m features. Finally, on new data where y is not known, this model can be used to make an inference $\hat{y}$. Generally, methods that involve training and inference, fall under the supervised methods category. In an alternate scenario, if training data is not available, and the task is to find patterns in the data or to reveal higher understanding by exploring the architecture and relationships in the data, we would need to find an appropriate unsupervised method. We refer to [50] for a deep and highly organized exploration of the entire domain of machine learning methods.

1.4.1 The F1-Score

The $F_1 - Score$ is the harmonic mean between recall and precision. We use this measure as the key criteria to benchmark and compare between classification models across this research; although the other pertinent statistics are also presented. In the context of binary classification, the hope is that when a positive call is made, most are actual/true positives (high precision) with a high proportion of the total negatives (specificity) and positives (recall) identified correctly. A major limitation of the general $F_1 - Score$ is that knowing just it, does not help identify if the classification had better recall or precision. Furthermore, depending on the application if it is necessary to weight recall more than precision or vice versa we can use the adjust F-measure [5]; the difference between the two is illustrated in Figure 6.



$$F = \frac{2*precision*recall}{precision+recall}$$

$$F_\beta = (1+\beta^2)\frac{precision * recall}{\beta^2 * precision + recall}$$

Figure 6: Comparison of the range of the **A** regular $F_1 - Score$ vs. **B** the adjusted $F_1 - Score$ which weights the recall more than precision [5].

1.4.2 Demonstration of machine learning classification with 1D bimodal simulated data

The main classification goal in this dissertation involves large numbers of input features, i.e., high-dimensional data. Intuitive comprehension of such data can be limited and thus for demonstration and evaluation we simulated 1D bimodal data. Two parameters under control are the marginal frequencies and the effect size between the positive and negative class examples. We quantify the effect size using Cohen's D as in equation 4.

$$(length_x,\ length_y) = (length(dist_1)\ -\ 1,\ length(dist_2)\ -\ 1) \tag{1}$$

$$AbsMeanDifference = abs(mean(x) - mean(y)) \tag{2}$$

$$CommonSD = \sqrt{(length_x * var(x) + length_y * var(y))/(length_x + length_y)} \tag{3}$$

$$Cohens_d = AbsMeanDifference/CommonSD \tag{4}$$

The simulation of 1D bimodal data, involves combining two samples from two distributions at specified ratios (i.e., $N_1/N_2 = freq$), such that each distribution parameter such as mean (μ_1 and μ_2) and standard deviation (σ_1 and σ_2) can be defined; when combined this set defines the domain of each sample. As our intent is to produce reproducible and interpretable examples that are not hindered by sample distribution complexity, we chose to sample from Gaussian (Normal) distributions $N(\mu = int, \sigma = 1)$. Depending on the application more appropriate evaluations with alternate distributions and additional subsets and features can also be done as in [51].

We demonstrate using simulated 1D data that there are two major factors that confound the classification. First as in Figures 7A-1, 2, and 3, we simulated a thousand (1000) events, where the two class labels are proportional i.e 1:1, but, the positive class gets closer to the negative class, reducing its effect size as estimated by Cohen's D value reported. The classification results using three different commonly used supervised classifiers (GLM, SVM, and Random Forest) are reported in Table 1, where the reduction in effect size negatively impacts the classification statistics. Next, instead of altering effect size, we reduce the marginal frequency of the positive class (7:3, 9:1, and then 99:1) as reported in Figures 7A-4, 5, and 6. Again we see a drop in the classification statistics as the positive class becomes less frequent.

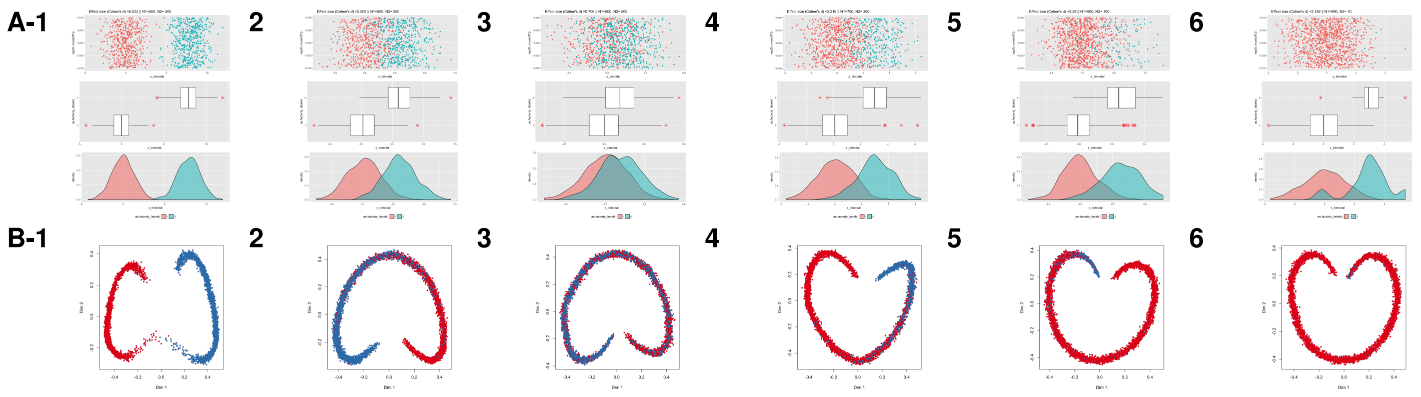


Figure 7: Simulated 1D bimodal samples varied by effect size and marginal frequency. **A** Each figure set contains (top to bottom): i- the class distribution, ii. box plots per class, and iii. estimated density curves per class. **B** Multi-dimensional Scaling (MDS) plot from the proximity matrix of an unsupervised random forest classifier.

	Sensitivity	Specificity	Precision	Recall	F1	Prevalence	Detection Prevalence	Balanced Accuracy	Accuracy	AccuracyLower	AccuracyUpper
Effect size (Cohen's D) $\approx$ 6.0											
1:1 $u_1 = 0, u_2 = 8$ w/ GLM	1.00	1.00	1.00	1.00	1.00	0.50	0.50	1.00	1.00	0.99	1.00
1:1 $u_1 = 0, u_2 = 8$ w/ SVM	1.00	1.00	1.00	1.00	1.00	0.50	0.50	1.00	1.00	0.99	1.00
1:1 $u_1 = 0, u_2 = 8$ w/ SupRF	1.00	1.00	1.00	1.00	1.00	0.50	0.50	1.00	1.00	1.00	1.00
Effect size (Cohen's D) $\approx$ 2.2											
1:1 $u_1 = 0, u_2 = 3$ w/ GLM	0.86	0.86	0.86	0.86	0.86	0.50	0.50	0.86	0.86	0.84	0.88
1:1 $u_1 = 0, u_2 = 3$ w/ SVM	0.86	0.86	0.86	0.86	0.86	0.50	0.50	0.86	0.86	0.84	0.88
1:1 $u_1 = 0, u_2 = 3$ w/ SupRF	0.93	0.93	0.93	0.93	0.93	0.50	0.50	0.93	0.93	0.91	0.95
Effect size (Cohen's D) $\approx$ 0.7											
1:1 $u_1 = 0, u_2 = 1$ w/ GLM	0.65	0.64	0.64	0.65	0.65	0.50	0.51	0.64	0.64	0.61	0.67
1:1 $u_1 = 0, u_2 = 1$ w/ SVM	0.65	0.64	0.64	0.65	0.65	0.50	0.51	0.65	0.65	0.61	0.67
1:1 $u_1 = 0, u_2 = 1$ w/ SupRF	0.84	0.84	0.84	0.84	0.84	0.50	0.50	0.84	0.84	0.82	0.86
Effect size (Cohen's D) $\approx$ 2.2											
7:3 $u_1 = 0, u_2 = 3$ w/ GLM	0.77	0.93	0.82	0.77	0.80	0.30	0.28	0.85	0.88	0.86	0.90
7:3 $u_1 = 0, u_2 = 3$ w/ SVM	0.76	0.93	0.83	0.76	0.79	0.30	0.27	0.85	0.88	0.86	0.90
7:3 $u_1 = 0, u_2 = 3$ w/ SupRF	0.90	0.96	0.90	0.90	0.90	0.30	0.30	0.93	0.94	0.92	0.95
Effect size (Cohen's D) $\approx$ 2.4											
9:1 $u_1 = 0, u_2 = 3$ w/ GLM	0.56	0.98	0.79	0.56	0.65	0.10	0.07	0.77	0.94	0.92	0.95
9:1 $u_1 = 0, u_2 = 3$ w/ SVM	0.46	0.99	0.84	0.46	0.59	0.10	0.06	0.73	0.94	0.92	0.95
9:1 $u_1 = 0, u_2 = 3$ w/ SupRF	0.83	0.98	0.84	0.83	0.84	0.10	0.10	0.91	0.97	0.95	0.98
Effect size (Cohen's D) $\approx$ 2.2											
99:1 $u_1 = 0, u_2 = 3$ w/ GLM	0.18	1.00	0.72	0.18	0.37	0.01	0.00	0.59	0.99	0.98	1.00
99:1 $u_1 = 0, u_2 = 3$ w/ SVM	0.00	1.00	0.00	0.00	0.00	0.01	0.00	0.50	0.99	0.98	1.00
99:1 $u_1 = 0, u_2 = 3$ w/ SupRF	0.75	1.00	0.77	0.75	0.75	0.01	0.01	0.87	1.00	0.99	1.00

Table 1: Supervised classifiers (Generalized linear model, linear SVM [C=0.1, γ=0.5], and a random forest) were trained on simulated 1D data, where in a leave-one-out scheme, one of the 20 random samples generated is held as the test set and the other 19 are sampled (N=1000 examples). The simulations reported herein parallel those in Figure 7A1-5. The lower the effect size (Cohen's D), the worse the classification is. Additionally, if the displacement between the two classer are kept constant (i.e., similar effect size), reduction in the marginal frequency of the positive class negatively impacts the classification.

1.5 Transfer Learning, an overview

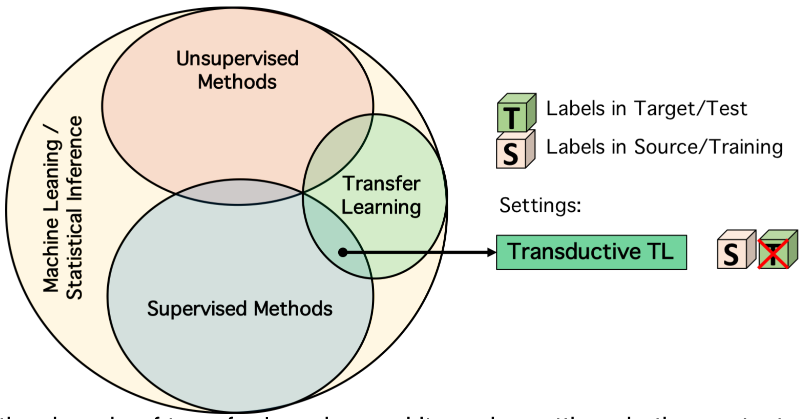


Figure 8: Relationship of the domain of transfer learning and its major settings in the context of supervised and unsupervised methods. We refer to [6] for a review of the definitions used herein.

Transfer Learning (TL) is a specialized application of machine learning distinct from general supervised (i.e., learning and searching for statistical patterns) and unsupervised (i.e., uncovering innate structure) methods. For the purposes of this dissertation, we assume training data is available to train on, which parallels human experts having access to literature and data to train on to make inference on new data. Thus in addition to supervised methods, transfer learning (TL) is another potential domain of machine learning. There are several types TL applications. However, some minor variations in defining them exist in the previously published literature. Specifically to our need, one branch is the transductive TL, where learning is achieved by training on the source data and the gained knowledge is generalized and transferred to the unlabeled target data. Unlike supervised and semi-supervised methods that assume the training and test input feature spaces i.e., X_{train} and X_{test} have been drawn from the same distribution, in TL, as it is in most applied circumstances, it is assumed that they are drawn from the two different distributions D^{source} and D^{target} respectively [7]. Another important distinguishing component pertains to the set of possible labels i.e., Y_{train} and Y_{test}; In transductive TL, they are assumed the same for the two distributions. A final component of transductive learning in literature is the utility of just X_{test} (without Y_{test}) during training; Cohen et al. discuss using the entire X_{test} [7] whereas Pan and Yang suggest using a fraction to obtain an estimate of the marginal probabilities in the target data [6].

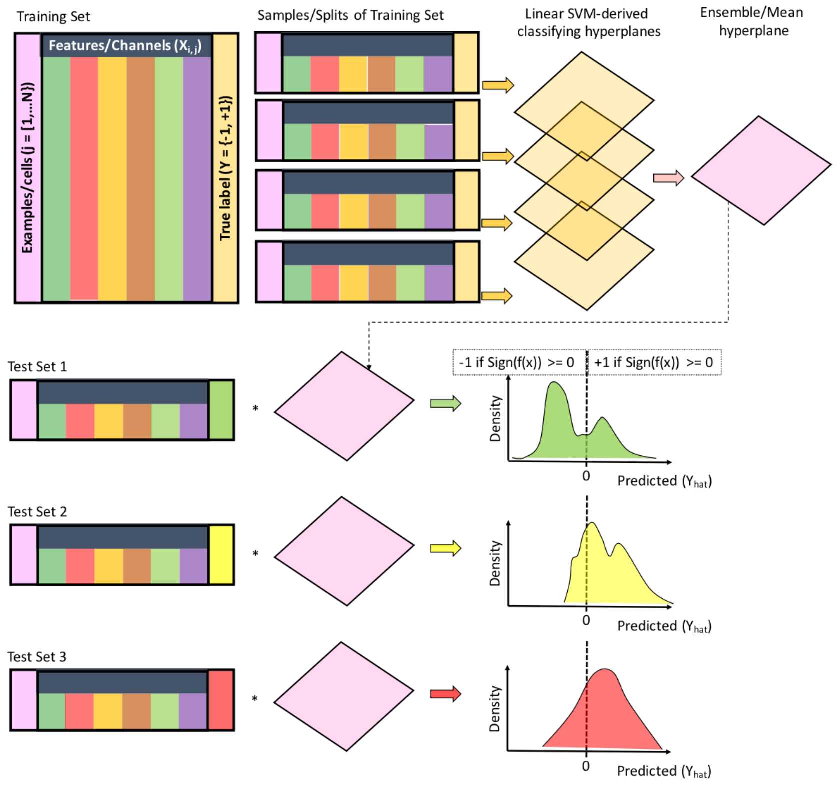

Training Set
Features/Channels ($X_{i,j}$)
Examples/cells (j = [1,...N])
True label (Y = {-1, +1})
Samples/Splits of Training Set
Linear SVM-derived classifying hyperplanes
Ensemble/Mean hyperplane
Test Set 1
Test Set 2
Test Set 3
-1 if Sign(f(x)) >= 0
+1 if Sign(f(x)) >= 0
Density
0
Predicted (Y_{hat})

Figure 9: The simple generalization of linear SVM hyperplanes from the training set (or subsets of it) to classify a new test set as described in [7]. In test set 1, we observe a mapping that resembles an optimal transfer where the correct minima is at 0, thus values above it are representative of those events predicted as positive. However, it is possible as in test set 2, the mapping is not optimal; the majority of the data is being classified positive, and the low-density region is evidence that potentially the linear classifying hyperplane can be shifted by updating its bias. Unfortunately in test set 3, we don't observe a low-density region, which can be explained by several factors. Perhaps the marginal frequency of the positive cases is different between the test and training sets or the effect size is altered (due to additional technical noise caused by a different experimental protocol).

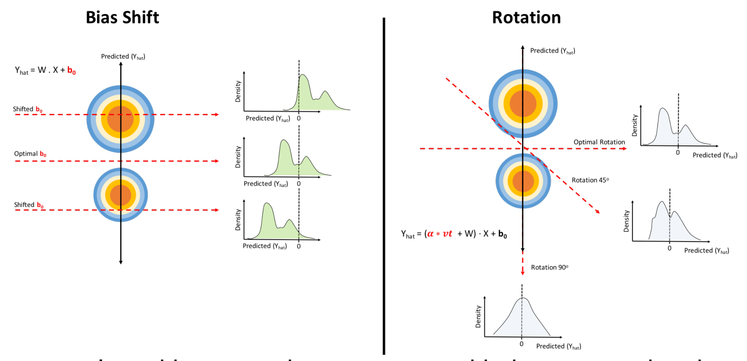


Figure 10: An illustration of a contour plot, with two modes or centers with the assumption that these two are the mappings of a binary classification. The classifying hyperplane Y-hat should pass through the center for the optimal classification. If a wrong bias is selected, the mapping can be shifted. On the right we observe the that rotation of the hyperplane can also have a significant impact on the expected density-curve shape of the mapped y-hat values.

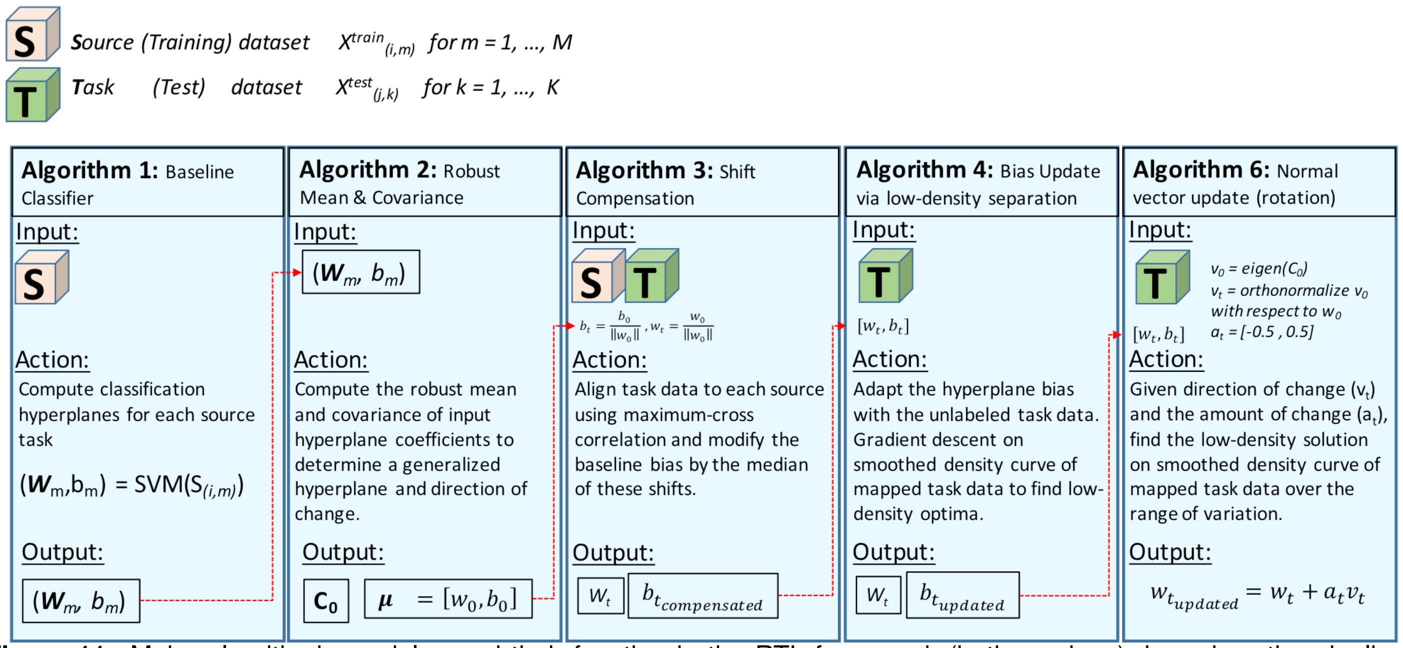
2 The RTL framework

The RTL framework a package that houses two transfer learning (TL) classification frameworks. As introduced earlier, a baseline version was initially implemented, based on the published algorithms in [52] because neither the code or dataset used was available. Based on our validation tests, we then developed extensions to several of the modules in the baseline version to improve classification robustness.

The various modules of the RTL framework utilize functions from library packages as well as newly written code. The new code includes functional parts of the algorithm, as well as connecting and figure-producing code. Although it is not generally necessary to validate the functionality of functions from major libraries, it is key to flag inputs and outputs for correctness; such as data structures, formats, size limitations, etc. The newly implemented functions, as well as the connecting code, needs to be validated i.e., the output at each critical point matches the expectation. Flagging and error capturing will be key to automating this process. Once the design was sketched out, previously distributed libraries were identified to be imported for building the RTL framework. The R language was chosen specifically because there are already a plethora of compatible libraries and packages for statistical modeling and analysis. Furthermore, as our intention is to use this framework for bioinformatics applications such as flow cytometric or scRNASeq analysis, there are also many libraries for preprocessing and exploratory analysis.

2.1 The Lee et al., 2011, automation of FC analysis

Lee et al. have proposed a transfer learning framework based on previous simpler form in [7] for the problem of classifying known immunophenotypes in multiple FC sample. These authors initially train linear-SVM classifiers on multiple gated FC data and obtain a generalization by obtaining a robust mean of the identified hyperplanes. Instead of direct inference using this naive classifier, they adapt the hyperplane to the test data by updating the hyperplane coefficients $< \boldsymbol{w_0}, b_0 >$ heuristically by identifying optimal low-density separation criterion. Figure 11 illustrates an overview of the learning and inference in the TL framework. One limitation of the original Lee et al. algorithm is that although their target population was clinically important, rare subsets were not the focus of the research, thus the lower boundary of population frequency has not been previously assessed. Finally, since Lee et al. 2011, did not publicly release an open-source version of their software nor dataset used, a direct comparison is not possible. Therefore, the testing and benchmarking conducted in this research pertains to the RTL framework and its current two versions.


S Source (Training) dataset $X^{train}_{(i,m)}$ for m = 1, ..., M
T Task (Test) dataset $X^{test}_{(j,k)}$ for k = 1, ..., K
Algorithm 1: Baseline Classifier
Input:
Action:
Compute classification hyperplanes for each source task
$(W_m, b_m) = SVM(S_{(i,m)})$
Output:
(W_m, b_m)
Algorithm 2: Robust Mean & Covariance
Input:
(W_m, b_m)
Action:
Compute the robust mean and covariance of input hyperplane coefficients to determine a generalized hyperplane and direction of change.
Output:
C_0
$\mu = [w_0, b_0]$
Algorithm 3: Shift Compensation
Input:
$b_t = \frac{b_0}{\|w_0\|}, w_t = \frac{w_0}{\|w_0\|}$
Action:
Align task data to each source using maximum-cross correlation and modify the baseline bias by the median of these shifts.
Output:
W_t
$b_{t_{compensated}}$
Algorithm 4: Bias Update via low-density separation
Input:
$[w_t, b_t]$
Action:
Adapt the hyperplane bias with the unlabeled task data. Gradient descent on smoothed density curve of mapped task data to find low-density optima.
Output:
W_t
$b_{t_{updated}}$
Algorithm 6: Normal vector update (rotation)
Input:
$v_0 = eigen(C_0)$
v_t = orthonormalize v_0 with respect to w_0
$[w_t, b_t]$
$a_t = [-0.5, 0.5]$
Action:
Given direction of change (v_t) and the amount of change (a_t), find the low-density solution on smoothed density curve of mapped task data over the range of variation.
Output:
$w_{t_{updated}} = w_t + a_t v_t$
Figure 11: Major algorithmic modules and their function in the RTL framework (both versions), based on the pipeline proposed by Lee et. al to generalize a linear classification boundary.

2.2 Algorithms of the baseline and extended modules of the RTL framework

2.2.1 Baseline Classifier (Algorithm 1)

Algorithm 1: Baseline Classifier (Algorithm 1)

```
input  : source (training) data $\{X_{train}\}_{m=1}^{M}$ for $m = 1, ..., M$;
         regularization parameters: $\{C_m\}_{m=1}^{M}$
output: $[\boldsymbol{W_m}, \boldsymbol{b_m}]$
1 for $m = 1$ to $M$ do
2     $[\boldsymbol{w_m}, b_m] = SVM(\{X_{train}\}_m, C_m)$
```

2.2.2 Robust Mean and Covariance (Algorithm 2)

Algorithm 2: Robust Mean and Covariance (Algorithm 2)

```
input  : $(\boldsymbol{W_m},\ \boldsymbol{b_m})$ for $m = 1, ..., M$;
output: $< \boldsymbol{\mu_{robust}}, \boldsymbol{C_{robust}} >$
1 Concatenate:
2 $\boldsymbol{u_m} \leftarrow [\boldsymbol{W_m},\ \boldsymbol{b_m}],\ \forall\ m$
3 Initialize:
4 $\boldsymbol{\mu} \leftarrow mean(\boldsymbol{u_m}),\ \boldsymbol{C} \leftarrow cov(\boldsymbol{u_m})$
5 RTL implementation: $< \boldsymbol{\mu_{robust}}, \boldsymbol{C_{robust}} > \leftarrow\ GSE :: HuberPairwise(\boldsymbol{u_m})$
```

2.2.3 Shift Compensation (Algorithm 3)

Algorithm 3: Shift Compensation (Algorithm 3)

```
input  : hyperplane $(\boldsymbol{w},\ b)$;
         source data $\{X_{train}\}_{m=1}^{M}$;
         target data $\{X_{test}\}_{k=1}^{K}$
output: $b_k$
1 $z_{,j} \leftarrow < \boldsymbol{w}, b >\ \cdot \{X_{test}\}_{,j}$
2 for $k = 1$ to $K$ do
3     for $m = 1$ to $M$ do
4         $z_{m,i} \leftarrow < \boldsymbol{w}, b >\ \cdot \boldsymbol{\{X_{train}\}_{m,i}}$
5         $e_m \leftarrow argmax_z KDE(z, z, j) \star KDE(z, zm, i)\ s.t.\ \star\ is\ maximum\ cross - correlation$
6     $b_k \leftarrow b_0 - median(e_m)$
```

2.2.4 Bias update (Algorithm 4)

Algorithm 4: Bias update (Algorithm 4)

```
1  Baseline implementation, with robustness:———————————-
   input  : hyperplane $(\boldsymbol{w},\ b)$;
            target data $\{X_{test}\}_{k=1}^{K}$;
   output: $b_k^{new}$
2  Compute: $z_{k,j} \leftarrow < \boldsymbol{w}, b > \ \cdot \{X_{test}\}_{k,j}$
3  Build Grid: $s_{k,j} \leftarrow sort(z_{k,j})$
4  $N_t \ = \ length(s_{k,j})$
5  for $j = 1$ to $N_t$ do
6      $c_{k,j} \leftarrow \sum_i \amalg \{ \frac{|z_{all} - s_{k,j}|}{\|\boldsymbol{w}\|} < Alg4_{margin} \}$
7  $h \leftarrow KernelBandwidth_{Silverman}(\{(s_{k,j}, c_{k,j})\})$
8  Smooth:
9  $\widehat{p}(z_k) \leftarrow \sum_j c_{k,j} k_h(z, s_{k,j})$
10 $z_k^* \leftarrow GradientDecent(\widehat{p}(z_k), 0)$
11 $b_k^n ew \leftarrow b_k - z_k^*$
12 Extended implementation:———————————-
   input  : hyperplane $(\boldsymbol{w},\ b)$;
            target data $\{X_{test}\}_{k=1}^{K}$;
   output: $b^{new}$
13 Compute: $z_{k,j} \leftarrow < \boldsymbol{w}, b > \ \cdot \{X_{test}\}_{k,j}$ $h_k \leftarrow KernelBandwidth_{Venables-Ripley}(\{(z_{k,j})\})$
14 $h_{k,s} \leftarrow h * scales \ scales \in (0, 1]$
15 $\hat{z_{k,s}} \leftarrow KernelSmoothEstimate_{Gau}((z_{k,j}), h_{k,s})$
16 $\hat{z_{k,s}\prime} \leftarrow Diff(\hat{z_{k,s}}) \ \forall \ s$
17 $z - \hat{inv}_{k,s}\prime\prime \leftarrow -1 * Diff(\hat{z_{k,s}\prime}) \ \forall \ s$
18 $z_k^* \leftarrow mean(GradientDescent(\hat{z_{k,s}}, NearPeak) \ \forall \ s)$
19 $z_k^*\prime \leftarrow mean(GradientDescent(\hat{z_{k,s}\prime}, NearPeak) \ \forall \ s)$
20 $z_k^*\prime\prime \leftarrow mean(GradientDescent(z - \hat{inv}_{k,s}\prime\prime, NearPeak) \ \forall \ s)$
21 $z_k^* < -mean(z_k^*, z_k^*\prime, z_k^*\prime\prime)$
22 $b_k^{new} \leftarrow b - z_k^*$
```

2.2.5 Kernel bandwidth estimation (Algorithm 5)

Algorithm 5: Kernel bandwidth estimation (Algorithm 5) [12, 13]

```
1  *Based on Silverman (1986):*————————————-
   **input**  : grid points and counts $\{(s_k, c_k)\}$
   **output:** $h$
2  $N \leftarrow \sum_k c_k$
3  $\bar{s} \leftarrow \frac{1}{N} \sum_k s_k c_k$
4  $\hat{\sigma} \leftarrow \sqrt{\frac{1}{N-1} \sum_k c_k (s_k - \bar{s})^2}$
5  $h \leftarrow 0.9 \hat{\sigma} N^{-1/5}$
6  *Based on Venables and Ripley (2002):*————————————-
   **input**  : counts $\{(c_k)\}$
   **output:** $h$
7  $r \leftarrow quntiles(c_k, (0.25, 0.75))$
8  $N \leftarrow length(c_k)$
9  $h \leftarrow (r_{0.75} - r_{0.25})/1.34$
10 $\hat{\sigma} \leftarrow var(c_k)$
11 $h \leftarrow 4 * min(\hat{\sigma}, h) * N^{-1/5}$
```

2.2.6 Normal vector update (Algorithm 6)

Algorithm 6: Normal vector update (Algorithm 6)

```
1  Baseline implementation:——————————-
   input  : hyperplane $(\boldsymbol{w},\ b)$
            direction of change $v_t$
            target data $\{X_{test}\}_{k=1}^{K}$
   output: $\boldsymbol{w_{new}}$
2  for $a_{rot} = -0.5$ to $0.5$ step 0.01 do
3      $\boldsymbol{w^{new}} \leftarrow \boldsymbol{w} + a_k \boldsymbol{v_t}$
4      $c_{k,} \leftarrow \sum_i \amalg\{\left|\frac{<\boldsymbol{w^{new}},\boldsymbol{\{X_{test}\}_{k,j}}>+b}{\|\boldsymbol{w^{new}}\|}\right| < Alg6_{margin}\}$
5  $h_k \leftarrow KernelBandwidth(\{(a_{rot}, c_{k,})\})$
6  Smooth: $\widehat{g}(a) \leftarrow \sum_j c_{k,} k_h(a, a_{rot})$
7  $a_t \leftarrow GradientDecent(\widehat{g}(a), 0)$
8  $\boldsymbol{w_{new}} \leftarrow \boldsymbol{w} + a_t \boldsymbol{v_t}$
9  Extended implementation:——————————-
   input  : hyperplane $(\boldsymbol{w},\ b)$
            direction of change $v_t$
            target data $\{X_{test}\}_{k=1}^{K}$
   output: $\boldsymbol{w_{new}}$
10 for $a_k = -0.5$ to $0.5$ step .01 do
11     $\boldsymbol{w^{new}} \leftarrow \boldsymbol{w} + a_k \boldsymbol{v_t}$
12     $c_{k,} \leftarrow \sum_i \amalg\{< \boldsymbol{w^{new}}, \boldsymbol{\{X_{test}\}_{k,j}} > +b < 0\}$
13 $h_{k,} \leftarrow KernelBandwidth_{Venables-Ripley}(\{(c_{k,})\})$
14 $\hat{c_{k,}} \leftarrow KernelSmoothEstimate_{Gau}((c_{k,}), h)$
15 $a_{k,t} \leftarrow GradientDescent(\hat{c_{k,}}, 0)$
16 $\boldsymbol{w_{new}} \leftarrow \boldsymbol{w} + a_{k,t} \cdot \boldsymbol{v_t}$
```

2.2.7 Combined set of algorithms for running multiple epochs (Algorithm 7)

Algorithm 7: Combined set of algorithms for running multiple epochs (Algorithm 7)

```
    input  : source data $\{X_{train}\}_{m=1}^{M}$;
             regularization parameters: $\{C_m\}_{m=1}^{M}$;
             target data $\{X_{test}\}$
    output: $[\boldsymbol{w_t}, b_t]$
1   for $m = 1$ to $M$ do
2       $[\boldsymbol{w_m}\ , b_m]\ =\ SVM(X_{train}\}_m, C_m)$
3   Initialize:
4   $[\boldsymbol{W_0}, b_0], \boldsymbol{C_0} \leftarrow Algorithm2(\{(\boldsymbol{W_m}, b_m)\}_m)$
5   $v_0 \leftarrow eig(\boldsymbol{C_0})$
6   Normalize:
7   $\boldsymbol{w_t} \leftarrow \frac{w_0}{\|w_0\|}$
8   $b_t \leftarrow \frac{b_0}{\|w_0\|}$
9   $v_t \leftarrow Orthonormalize\ v_0\ with\ respect\ to\ w_0$
10  Shift Compensation:
11  $b_t \leftarrow Alg3(\boldsymbol{w_t}, b_t, \{X_{train}\}_m\}, \{X_{test}\})$
12  Bias Update:
13  $b_t \leftarrow Alg4(\boldsymbol{w_t}, b_t, \{X_{test}\})$
14  Update Normal Vector:
15  $\boldsymbol{w_t} \leftarrow Alg6(\boldsymbol{w_t}, \boldsymbol{b_t}, \boldsymbol{v_t}, \{X_{test}\})$
16  Final hyperplane's figures and statistics:
17  $\hat{y_{final}} \leftarrow sign(\boldsymbol{w_t} \cdot \{X_{test}\} + b_t)$
```

2.2.8 Argmax Cross-Correlation

Algorithm 8: Argmax Cross-Correlation

```
    input  : $UniVarSeries_a$, $UniVarSeries_b$, $MaxLag$
    output: $\Delta Shift_{lag}$
1   $AllLags \leftarrow CrossCorrelation(KDE(UniVarSeries_a),\ KDE(UniVarSeries_b),\ MaxLag)\ s.t.\ AllLags =< cors,\ lags >$
2   $Lag_{max_{|cor|}} \leftarrow Lag[max(abs(cors))]$
3   $\Delta Shift_{lag} \leftarrow UniVarSeries_a[Lag_{max_{|cor|}} + k] - UniVarSeries_b[k]$ for k positions in either series
```

2.2.9 Gram-Schmidt Orthonormalization

Algorithm 9: Gram-Schmidt Orthonormalization of $\boldsymbol{V_a}$ on $\boldsymbol{V_b}$

```
    input  : $\boldsymbol{Vector_a}[1:d]$, $\boldsymbol{Vector_b}[1:d]$
    output: $\boldsymbol{W}$, Orthonormalization of $\boldsymbol{V_a}$ on $\boldsymbol{V_b}$
1   $\boldsymbol{X[i_{1,2}, d]} \leftarrow < i_1 : \boldsymbol{Vector_a}[1:d],\ i_2 : \boldsymbol{Vector_a}[1:d] >$
2   $\boldsymbol{W[i_{1,2}, 1]} \leftarrow \boldsymbol{X[i_{1,2}, 1]}$
3   for $k = 2$ to $d\ s.t.\ d = length(vector_a) = length(vector_b)$ do
4       $g_w \leftarrow repeat(0, i)$
5       for $j = 1$ to $k - 1$ do
6           $g_{k,j} \leftarrow \boldsymbol{W}^T[i_{1,2}, j] \cdot \boldsymbol{X}[i_{1,2}, k] / (\boldsymbol{W}^T[i_{1,2}, j] \cdot W[i_{1,2}, j])$
7           $g_w \leftarrow g_w + g_{k,j} \cdot \boldsymbol{W}[i_{1,2}, j]$
8       $\boldsymbol{W}[i_{1,2}, k] \leftarrow \boldsymbol{X}[i_{1,2}, k] - g_w$
9   $\boldsymbol{W} \leftarrow (\frac{1}{\sqrt{diag(crossproduct(\boldsymbol{W}))}} \cdot \boldsymbol{W}^T)^T$ where, $crossproduct()$ function returns the scalar squared sum and $diag()$
      returns the diagonal of the matrix. This is equivalent to obtaining the spectral norm of the matrix.
```

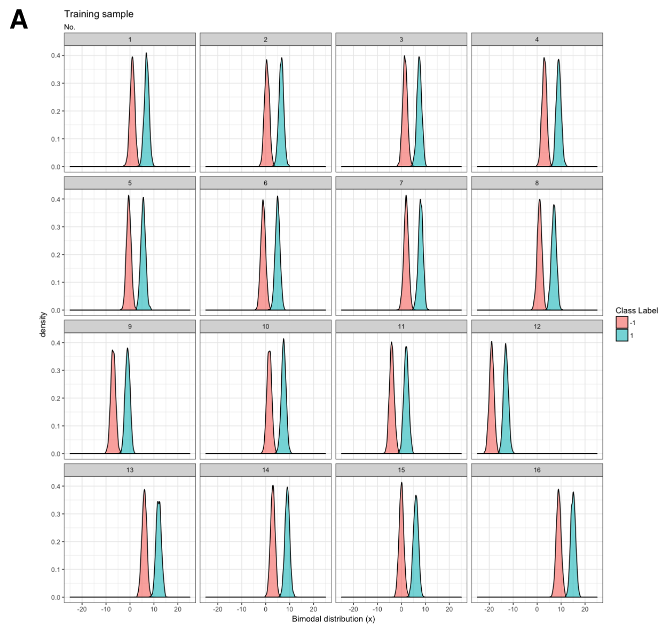
2.3 A 1D simulated dataset to validate the RTL framework

The training set includes 16 bimodal samples that were designed to be representative of variation seen in an FC experiment, but only with 1 bimodal feature/dimension. To create a bimodal sample, two Gaussian normal samples at difference centers (i.e., μ_1 and μ_2; $\sigma_1 = \sigma_2 = 1$) were simulated and combined. The marginal frequency of the positive class across the training samples does not change, rather only the centers of the two modes changed; this simulates minor domain shifts and effect size observed in a single channel of FC data across multiple samples (Figure 12). The goal for this training set is to return a set of classification hyperplanes, one per training sample to define a boundary for the classification hyperplanes. In practice, a formal assessment of the number of training samples can be done to optimize the framework for a specific application. A large set of training may not be needed and in fact, significantly slows down the classification. Table 2 summarizes the μ_1 and μ_2 used to sample a normal distributions such that $N(\mu, \sigma = 1)$; the computed effect sizes for this set is summarized in Table 2.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Neg	1.00	1.00	2.00	3.00	-0.00	-1.00	2.00	1.00	-7.00	1.00	-4.00	-19.00	6.00	3.00	-0.00	9.00
Pos	7.00	7.00	7.00	9.00	6.00	5.00	8.00	7.00	-1.00	8.00	2.00	-13.00	12.00	9.00	6.00	15.00

Table 2: Training set containing 16 samples, each with 5000 examples and a negative:positive class ratio of 6:4; shown are the bimodal centers, i.e., μ_1 and μ_2, used to sample Gaussian distributions i.e., $N(\mu, \sigma = 1)$.

The test set includes 40 bimodal simulated samples, specifically designed to evaluate the effects of changing the marginal frequency and the effect sizes in the test samples; created similar to training sets, but with a total of 2000 examples/cells. The first 10 samples, we validate the reproducibility of each algorithm in the framework as each of the 10 samples is a separate but similar random samples; also similar to the training sets, this first 10 test samples have the same negative:positive ratio of 6:4 (i.e., $N_1 = 1200$ and $N_2 = 800$) and the average effect size (Cohen's D) is approximately 6 ($\mu_1 = 1$ and $\mu_2 = 6$). Next, samples 11-20 starting at $\mu_1 = 1$ and $\mu_2 = 6$, we reduced effect size by changing μ_2; each $\mu_1 - \mu_2$ pair is represented by two samples to evaluate the robustness of the call (i.e., 11=12, 13=14, ...). For sample 21-30 we kept the effect size unchanged($\mu_1 = 0$ and $\mu_2 = 6$) but altered the marginal frequency of the two class labels from %20 to %5. Finally, for samples 31-40 we combined both confounding factors, where on average the effect size is 2 and changing marginal frequency respective to the samples 21-30.



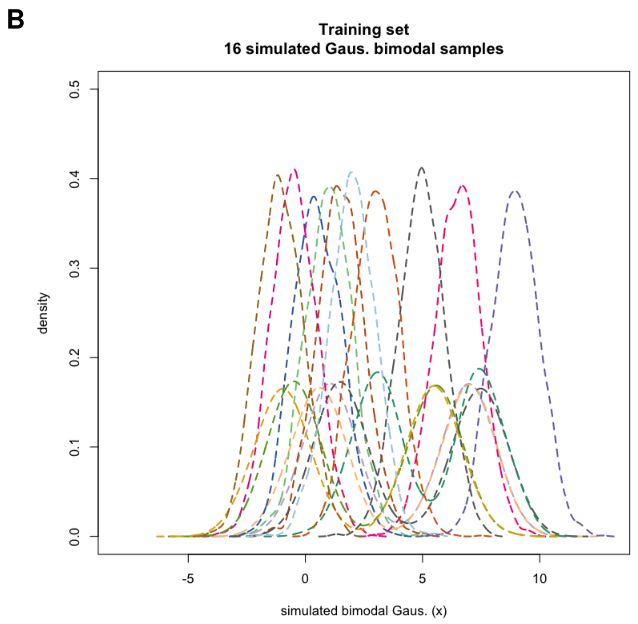


Figure 12: A. Each class label per sample is plotted as a density graph to highlight the variance in effect size between the two classes; the marginal frequency differences are not visible in this plot. **B.** Overlay of the density plots, to demonstrate the marginal frequency differences as well as the domain shifts for each sample.

Data Source	μ_1	μ_2	N_1	N_2
Test samples (1-10)	1	6	1600	400
Test samples (11-20)	1	2,3,4,5,6	1200	800
Test samples (21-24)	1	6	100, 200, 300, 400	1900, 1800, 1700, 1600
Test samples (25-26)	1	6	1000	1000
Test samples (27-30)	1	6	1600, 1700, 1800, 1900	400, 300, 200, 100
Test samples (31-34)	1	3	100, 200, 300, 400	1900, 1800, 1700, 1600
Test samples (35-36)	1	3	1000	1000
Test samples (37-40)	1	3	1600, 1700, 1800, 1900	400, 300, 200, 100

Table 3: A summary of the parameters used (μ_1, μ_2, N_1, and N_2) to create the 40 sample simulated test set using Gaussian distributions i.e., $N(\mu, \sigma = 1)$.

Effect Size (Cohen's D)	sample1	sample2	sample3	sample4	sample5	sample6	sample7	sample8	sample9	sample10
	6.04	6.04	6.04	5.95	6.03	5.83	6.20	6.08	6.13	5.84
	sample11	sample12	sample13	sample14	sample15	sample16	sample17	sample18	sample19	sample20
	5.07	4.87	4.05	3.93	3.02	3.08	2.02	1.96	1.12	0.94
	sample21	sample22	sample23	sample24	sample25	sample26	sample27	sample28	sample29	sample30
	4.96	5.19	5.14	5.09	4.97	4.91	5.02	5.09	4.83	5.31
	sample31	sample32	sample33	sample34	sample35	sample36	sample37	sample38	sample39	sample40
	2.17	1.92	1.96	2.03	2.01	2.00	2.16	1.88	2.06	2.04

Table 4: A summary of the computed effect size (Cohen's D) on the simulated test data (40 samples).

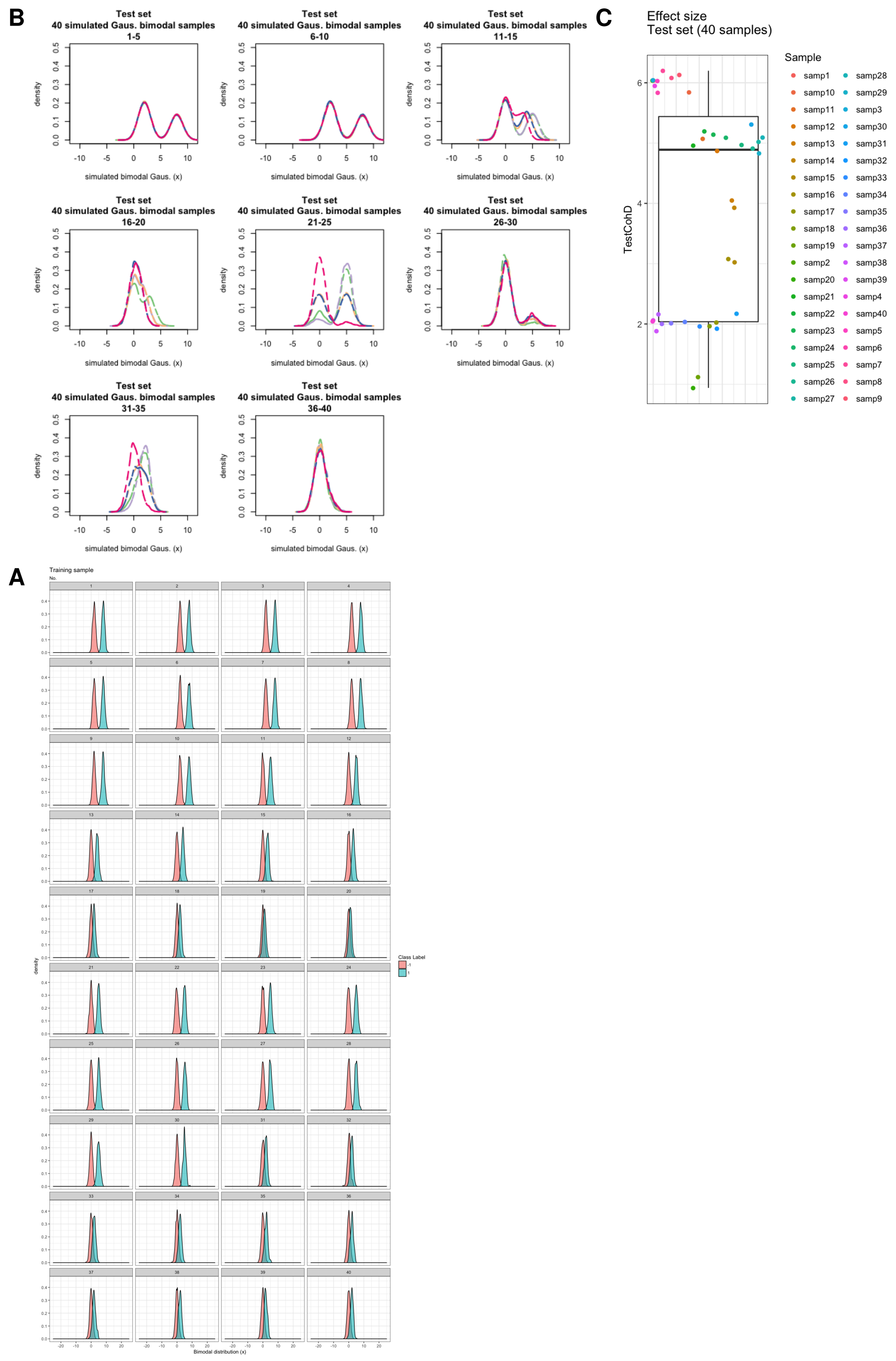


Figure 13: Test set (40 simulated samples): **A.** Each class label per sample is plotted as a density graph to highlight the differences in effect size (caused by the different means of the sampled Gaussian distributions) between the two classes; the marginal frequency difference is not visible in this plot. **B.** Overlay of the density plots, without considering the class label to demonstrate the marginal frequency differences as well as the shifts in the domain (x) of each sample.

2.3.1 A walk-though of the RTL framework using the 1D bimodal simulated data

***Algorithm 1**, computes the baseline hyperplanes* by taking each of the training samples $X^{train}_{(i,m)}$ (such that each sample ($m \in [1, 2, .., M]$) has i examples or simply X_{train}), as well as corresponding regularization parameters to train m linear SVM classification models. From each model, the hyperplane parameters ($\boldsymbol{W_m}$ and b_m) are obtained and passed to Algorithm 2. In our simulated case example, the first training sample and its corresponding SVM-derived classifying line is shown in Figure 14A. Since there are 16 training samples (m=16), we visually demonstrate in Figure 14B, the consequence of directly transferring any one of the resulting hyperplanes on the first test case.

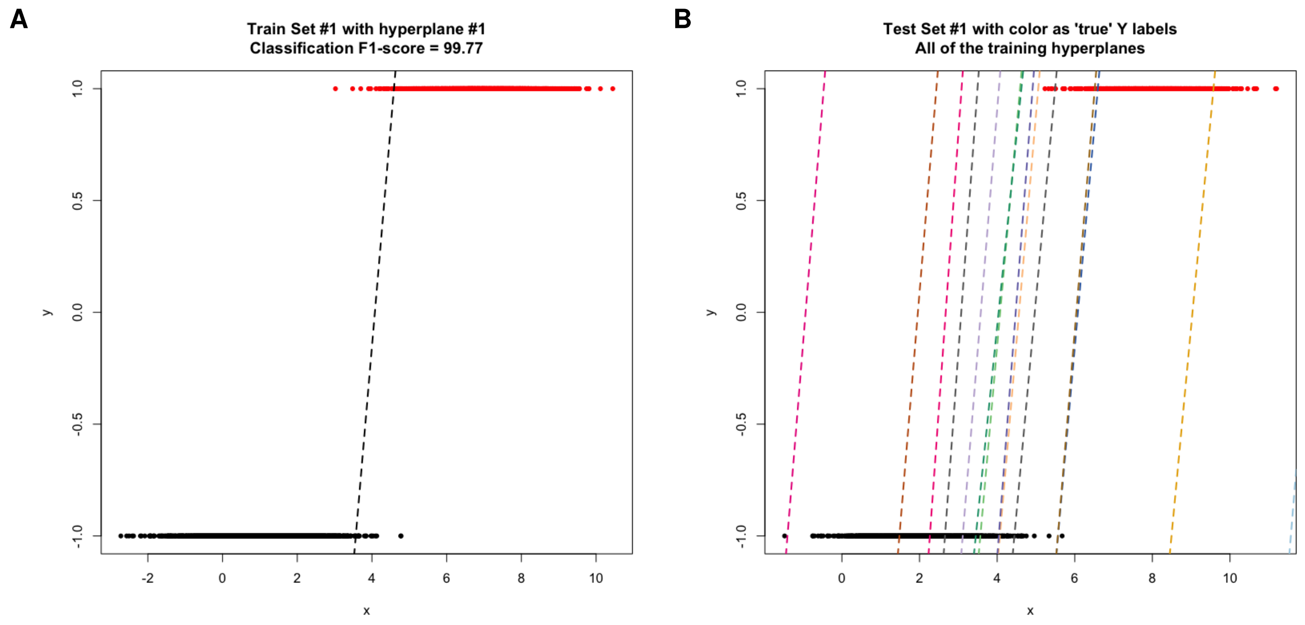


Figure 14: **A.** Demonstration of the trained classification boundary from training sample 1, plotted directly on it self, showing its optimal classification. **B.** The 16 trained classification boundaries shown on test set 1.

	sample_1	sample_2	sample_3	sample_4	sample_5	sample_6	sample_7	sample_8	sample_9	sample_10	sample_11	sample_12	sample_13	sample_14	sample_15	sample_16
W_m	-1.96	-2.15	-2.03	-1.95	-2.45	-2.10	-1.91	-1.70	-2.00	-2.30	-2.14	-2.02	-1.85	-2.10	-2.39	-2.15
$b-m$	-7.98	-7.68	-9.24	-11.85	-6.54	-4.11	-9.47	-6.88	7.72	-10.29	2.00	32.29	-16.71	-12.70	-7.34	-25.84

Table 5: Output of Algorithm 1, on the 1D simulated training set. From Algorithm 2, the simple mean hyperplane generalizing the training <W_m,$b-m$> is <-2.07, -5.91> and the robust mean is <-2.06, -7.83>.

***Algorithm 2**, computes the robust mean and covariance* of input hyperplanes obtained in Algorithm 1; i.e., the baseline < $\boldsymbol{W_0}$, b_0> hyperplane. Computing the robust mean as oppose to the simple mean, ensures outliers in training datasets do not heavily bias the overall position of the hyperplane. In the Lee et al paper, the method is described as computing the Mahalanobis distances and using the Huber-loss weight function to remove atypical variation in the set of hyperplanes in computing the robust mean, which defines the distance of m^{th} observation from the mean of the observations; commonly used to detect multi-dimensional outliers. This distance measure is used to weight the m^{th} observation such that large distances are down-weighted [53]. The etiology and application of several robust methods are reviewed in [54] which serves as a resource for key proofs and explanations about the appropriateness of procedures and parameters. In the RTL framework, we utilize the $HuberPairwise()$ function from the 'Generalized S-Estimator' (GSE) package [55] to estimate the robust mean $< \boldsymbol{w_0}, b_0 >$ and covariance $\boldsymbol{C_0}$; the tuning constant c0 = 1.345 is used for the Huber-pairwise estimation function.

Several additional variables are calculated at this stage for use in downstream algorithms. First, the magnitude (simple Euclidean) of the robust mean hyperplane $< \boldsymbol{w_0} >$ i.e., $||\boldsymbol{w}||$ was computed and used to normalize the hyperplane as in algorithm 7 to obtain $< \boldsymbol{w_t}, b_t >$. Next, we compute the direction of change as $v_0 = eig(\boldsymbol{C_0})$, and orthonormalize it with respect to $\boldsymbol{w_0}$. The first principle eigenvector is then chosen as the value of v_t; this will be used to rotate the normal vector around the first principle component. ***The orthonormalization*** *approach used by the RTL framework is the Gram-Schmidt orthonormalization method* as described in Algorithm 9. Simply put, the goal is to find a unit vector that is orthogonal to the input data. In the context of the RTL framework, the input data is the set of hyperplanes from Algorithm 1 which is decomposed (using the R's base spectral decomposition) to obtain its eigenvalues [4]; for further reading we refer to [56]for a brief and detailed explanation and derivation of eigenvalues and eigenvectors. Unfortunately, the exact algorithm used by Lee et al. is not known; it is only referred to in the workshop manuscript as 'orthonormalization' [5]. There are several different algorithms that perform this computation; pseudocode and details found in [57]. Regardless of the method, it is simple to test orthogonality since orthogonal vectors should always have their inner product equal to zero. i.e., $\boldsymbol{w}$ is orthogonal to $\boldsymbol{v}\ i.i.f.\ \boldsymbol{w} \cdot \boldsymbol{v} = 0$; a flag can always check this to be true.

Briefly, we compare the orthonormalization used in the RTL framework (a form of the modified) Gram-Schmidt algorithm with three other commonly utilized calculation. For simplicity, we combine five 2-dimensional vectors as a 2x5 matrix. The robust mean of the vectors is shown in black in Figure 15 and the orthonormal vectors are shown in various colored lines. We observe that the both the modified Gram-Schmidt algorithms from the RTL framework and the pracma package provide identical results. Although the relative position of the other two methods are different, we observe that they are all parallel, which is of most interest in the context of the RTL framework. The limitation of the pracma modified Gram-Schmidt is illustrated in the special case where the input matrix does not have full rank and is non-invertible i.e., a singular matrix [6]. Thus in practice, the modified Gram-Schmidt algorithm in the RTL framework is superior for automation of machine learning applications.

	1	2
1	1.00	2.00
2	0.90	1.50
3	0.80	1.00
4	0.95	0.50
5	0.98	0.00

Table 6: Five 2-dimensional vectors combined as a 2x5 matrix.

[4]eigenvalues are scalar sets associated to the system of equations (i.e., hyperplanes) usually represented as rows in a matrix of equations

[5]In a section not specifically defining the method, it is hinted the 'Gram-Schmidt process' might have been used by Lee et al.; described in the context of potential future direction of this algorithm

[6]for example this is a non-invertible matrix $<< -1, 2/3 >, < 3/2, -1 >>$.

A

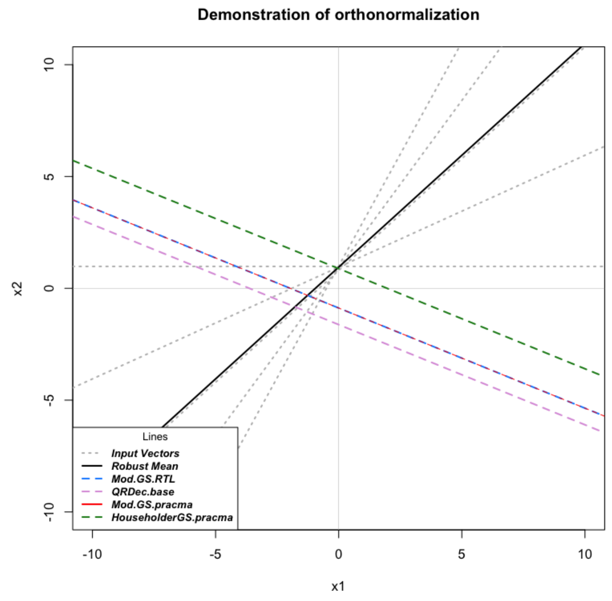


Figure 15: Visualization of five 2-dimensional vectors from Table 6. The robust mean is shown as the black line. The blue, plum, red, and green colored lines are the computed orthonormal vectors by the RTL framework's modified Gram-Schmidt, R-base QR decomposition [8], the Modified Gram-Schmidt in the pracma package [9], and the Householder method also from the pracma package.

Algorithm 3 *compensates for shift in range* by comparing the Gaussian-kernel smoothed projection of each task z_k sample k, $\{X_{test}\}_{k=1}^{K}$, with the generalized classification hyperplane to the projection of the source/training samples $\{X_{train}\}_{m=1}^{M}$ using $[\boldsymbol{w_m}, b_m]$ as in line 1 of algorithm 3. The shift between the test and training projections is then computed by identifying maximum cross-correlation and the median of the shifts is used to update the classifying hyperplane's bias i.e., $b_{k_{adapted}}$ as the output. This alignment is done because with a previously unseen test case, the robust mean hyperplane most likely will not provide the 'optimal' classification result. In other words, the mapping of $\boldsymbol{X} \rightarrow \boldsymbol{Y}$ for the test (target) case(s) need to be shifted compared to the training cases, where the robust mean hyperplane is derived from. Therefore, in algorithm 3, the $ArgMaxCross - Correlation$ algorithm is used. However, in the Lee et al. workshop manuscript, the exact algorithm and settings used is not specified. Thus, for the RTL framework, we formally describe the maximum cross-correlation (algorithm 8), that returns the $\Delta Shift_{lag}$ between the target test case(s) and the training examples which utilizes the $ccf()$ function from R's 'Stats' library to compute the cross-correlations and obtain the maximum absolute correlation.

To compute the shifts between the two vectors, the Gaussian kernel density estimates between the test case(s) and the training cases are first computed. It should be noted that as described in algorithm 7, the normalized robust mean hyperplane i.e., $< \boldsymbol{w_t}, b_t >$ is utilized and then updated in this algorithm. The $MaxLag$ parameter used in algorithm 8 defines how large the maximum lag between the two univariate series can be. The lag defines the positional shift i.e., where the position in the two vectors V_a or V_b is $V_a[i + shift]$ and $V_b[i]$ respectively. For example, if the values in each of the series is equally separated, such as in $(position, value)$ :: $k_a = [(1,1),(2,2),...(\Re,\Re)]$, the change in the lag will correspond equally with the shift change. However, this is not necessarily is expected to be the case with the mappings of $\boldsymbol{X} \rightarrow \boldsymbol{Y}$ and thus to compute the $\Delta Shift_k^{lag}$, we subtract the values with respect to the position lag. The value of $MaxLag$ parameter used in the original Lee et al. workshop manuscript is unknown, thus for RTL it is possible to experimentally identify an optimal value, however as a default it is computed as equal to half of the longer of the two vectors. Finally, for a robust shift, for each test k, the median of all computed shifts is returned as the final result;$\Delta Shift_k^{lag}$.

As a case example, Test set 1 is used to visualize the bias-shift process when this test case is compared to the 16 training examples (Figure 16). Overall, this algorithm performs as expected, i.e., returns the appropriate algorithmically defined output. However, we have implemented code that instead of maximum cross-correlation, the median-median of the mapped $\hat{y}$ can also robustly identify the shift; a significantly less complex computation.

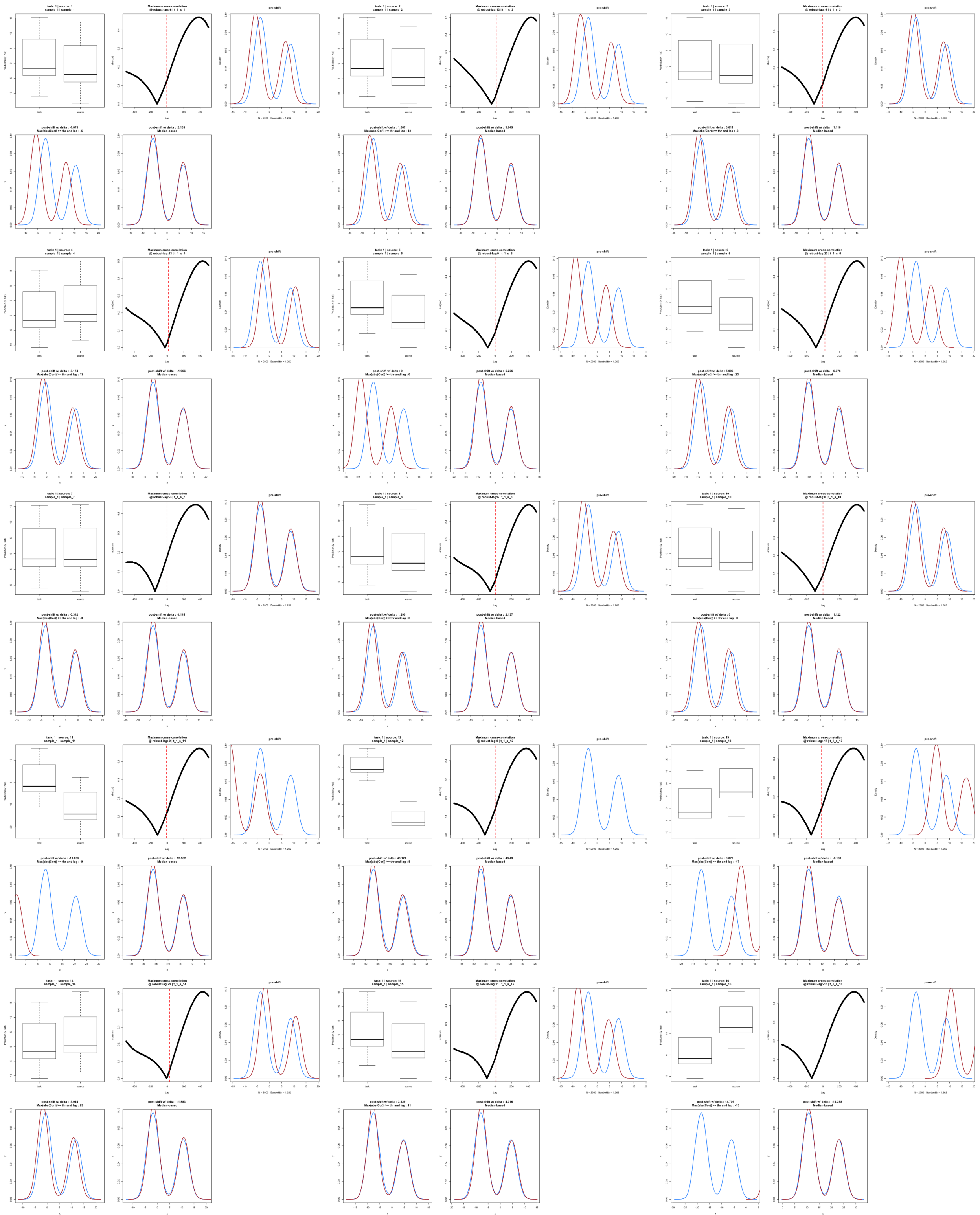


Figure 16: Algorithm 3 figures, demonstrating a case example in computing $\Delta Shift_{lag}$ for test set 1 compared to training sets 1 to 16 (starting top left moving right). Each figure set starts with a comparison of the distribution of the training and test mappings using boxplots. Next, lag is plotted against absolute(cross-correlation)) for the two mappings. The next sub-figure is the overlapped density plots prior to shift correction. Finally, in the next row of sub-figures we observe the density curves of the same mappings post-shift, first using median-max-absolute cross-correlation and also an optional approach using the median of the median of the mappings.

***Algorithm 4**, updates the hyperplane bias by identifying an an area of low-density* using the input the hyperplane $[\boldsymbol{w_t}, b_{t_{adapted}}]$ for each test k $(\{X_{test}\}_k)$. Briefly, this is done by identifying an optimal minima with a gradient descent algorithm near the robust mean (i.e., 'baseline'). Lee et al. define the margin $(\Delta = 1)$ as the range to count data-points near the normalized hyperplane $[\boldsymbol{w_t}, b_{t_{adapted}}]$ across the grid of biases s_j. Prior to the gradient descent, however, the counts are smoothed using an estimated bandwidth for Gaussian kernel smoothing computed as described in algorithm 5. The RTL framework utilizes the $ksmooth()$ function from the Stats package [8]; a highly utilized and validated package for various calculations in R. The $ksmooth()$ function implements the Nadaraya-Watson kernel regression smoother, which is a non-parametric linear estimator. However, the selection of the bandwidth is critical as smoothing with an inappropriate bandwidth, returns many 'NA' estimates because the binning can become very small. The production of 'NA' due to the small bandwidth is not unique to this function. The 2D-KDE estimator from the $KernSmooth$ package [58] also behaves similarly when tested. To avoid the potential risk of over-smoothing and losing the key optimas (maxima and minima) the bandwidth iteratively is changed, determined by quantifying the frequency of 'NA' values in the estimate. Still, finding the correct bandwidth is a difficult balance to automate: small bandwidths are likely produce sparse results and higher values over smooth and key topological landmarks are lost.

Depending on the application, the 'minima' may be difficult to identify on the smoothed counts; even across a range of bandwidths. To remedy this limitation, we can decrease the margin i.e., $\Delta < 1$; which is used in Algorithm 4 to count the number of events that fall within it. In the RTL framework, any value can be passed as the margin parameter Δ. During validation with the simulated data and preliminary analysis with flow cytometry data, we found the optimal Δ to be as low as 0.1, depending on the effect size and marginal frequency of the class labels. Globally, however, as long as the landmarks i.e., several key optimal points i.e., minima and maxima are not smoothed over the actual value of Δ is not highly critical.

The gradient descent algorithm used in the RTL framework is a stochastic average gradient descent (SAGD) Method from the gradDescent package [59]. To facilitate the automation of the framework, this algorithm is housed in a customized function that computes several additional variables and flags. However, the implemented custom gradient decent function adds to the complexity of the overall pipeline; especially if flags are caused which result in additional computations. As a solution to reduce the complexity, it is possible to narrow the search-space for the gradient descent algorithm using key landmarks to guide the gradient descent. For example, the peak position can be meaningful depending on the context of the classification. Since the goal here is finding the optimal intercept position in a binary classification, we expect to be able to find at minimum between two peaks. With this logic, the customized wrapping function first identifies landmarks (if possible) and create windows to search for possible local minimas and return the most optimal minima based on a pre-specified rule. For example, in the 'near zero' mode, the minima closest to 0 is returned as needed when Algorithm 6 calls the customized gradient descent function.

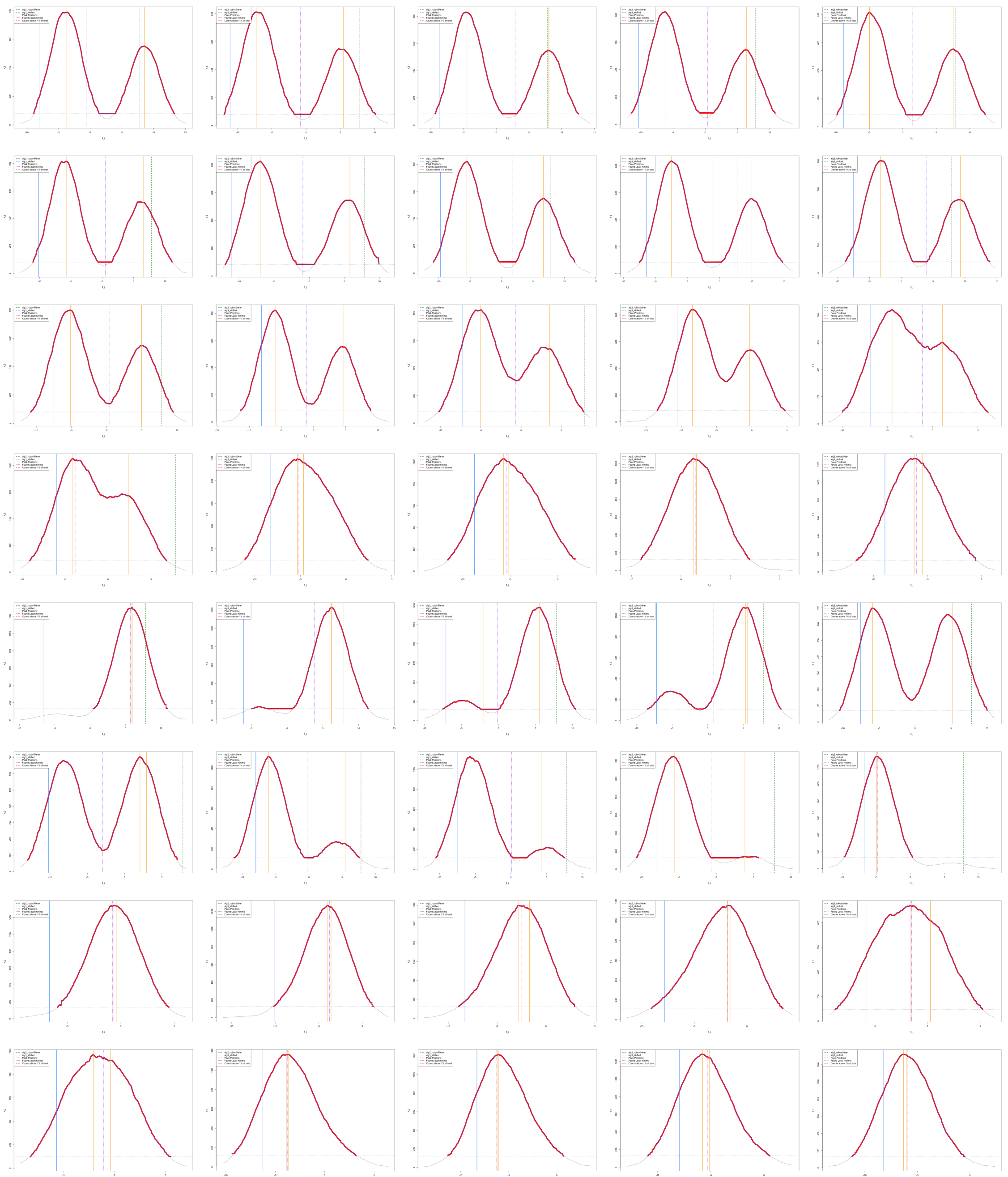


Figure 17: Updating the bias using the test samples (40) from the baseline implementation of Algorithm 4 given the margin $\Delta < 0.1$. Shown are the mapped $\hat{\mathbf{y}}$ for each test sample by the robust mean hyperplane derived in Algorithm 2; lack of bi-modality is correlated to a reduced effect size.

Algorithm 5 *estimates the kernel bandwidth* to be used for a Gaussian smoothing function. A non-parametric and practical approach as described in [12] for computing the smoothed density-curve estimate used in algorithms 4 and 6.

Algorithm 6 *finds an optimal direction for updating the normal vector* (i.e., the classification hyperplane is perpendicular ($\perp$) to), effectively rotating the classifier hyperplane based on a low-density criteria. Briefly, the inputs are each test k, $\{X_{test}\}_k$, the hyperplane $[\boldsymbol{w_t}, b_{adapted}]$, and the direction of change v_t. This is done by $\boldsymbol{w^{new}} \leftarrow \boldsymbol{w} + a_t\boldsymbol{v_t}$ such that a_t determines the range which the search is conducted in i.e., $a_t \in [-0.5, 0.5]$. Similar to algorithm 4, gradient descent is used to obtain a minima solution near the hyperplane. In other words, the goal of this algorithm is to rotate the hyperplane i.e., change $\boldsymbol{w_t}$ across a range to find a low-density solution. Although our case example demonstration has 1D samples, the benefit of such an approach is actually in the context of higher dimensions; the features collectively are mapped to $\hat{\boldsymbol{y}}$ such that the signed value results in '+1' (positive) and '-1' (negative) classification. The final hyperplanes $[\boldsymbol{w^{new}}, b_{adapted}]_k$ are then used to produce figures and classification statistics.

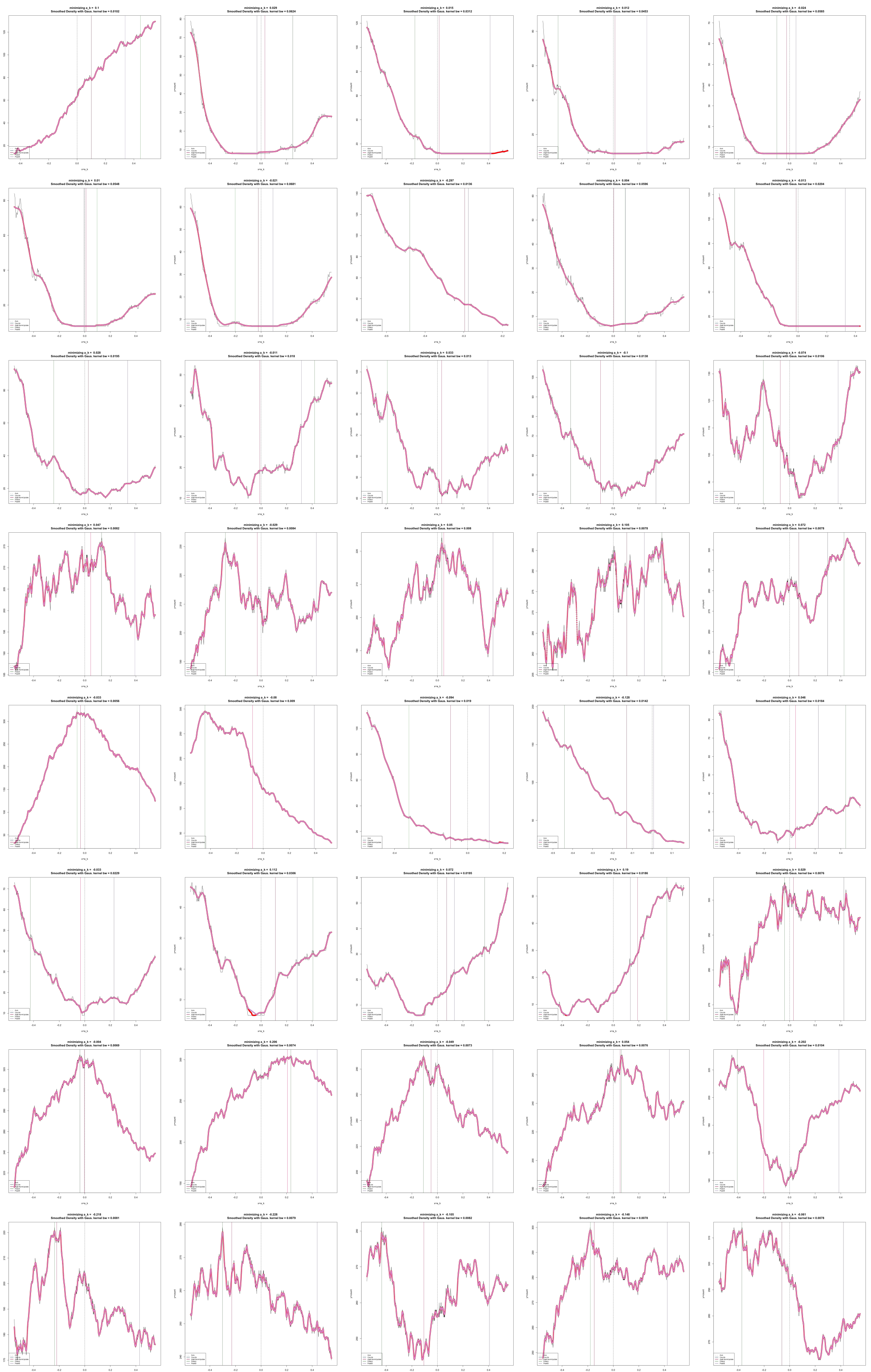


Figure 18: Finding the low-density optima to update the normal vector on test samples (40) in the baseline implementation of Algorithm 6. The x-axis is the rotation factor, from -0.5 to 0.5; since very low counts are removed to improved the identification of the appropriate minima, not all x-axis show this range, rather the range of the filtered segment.

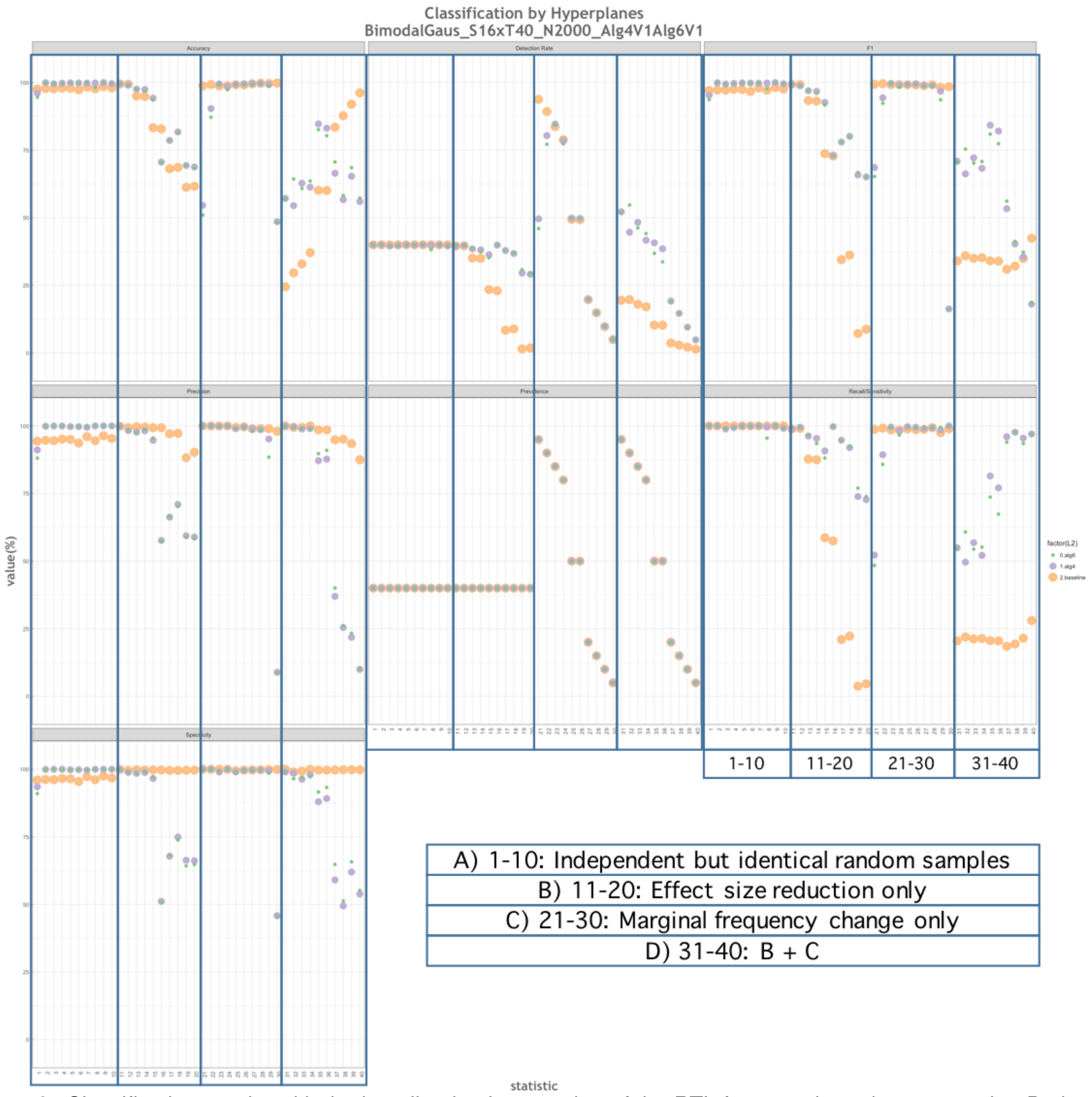
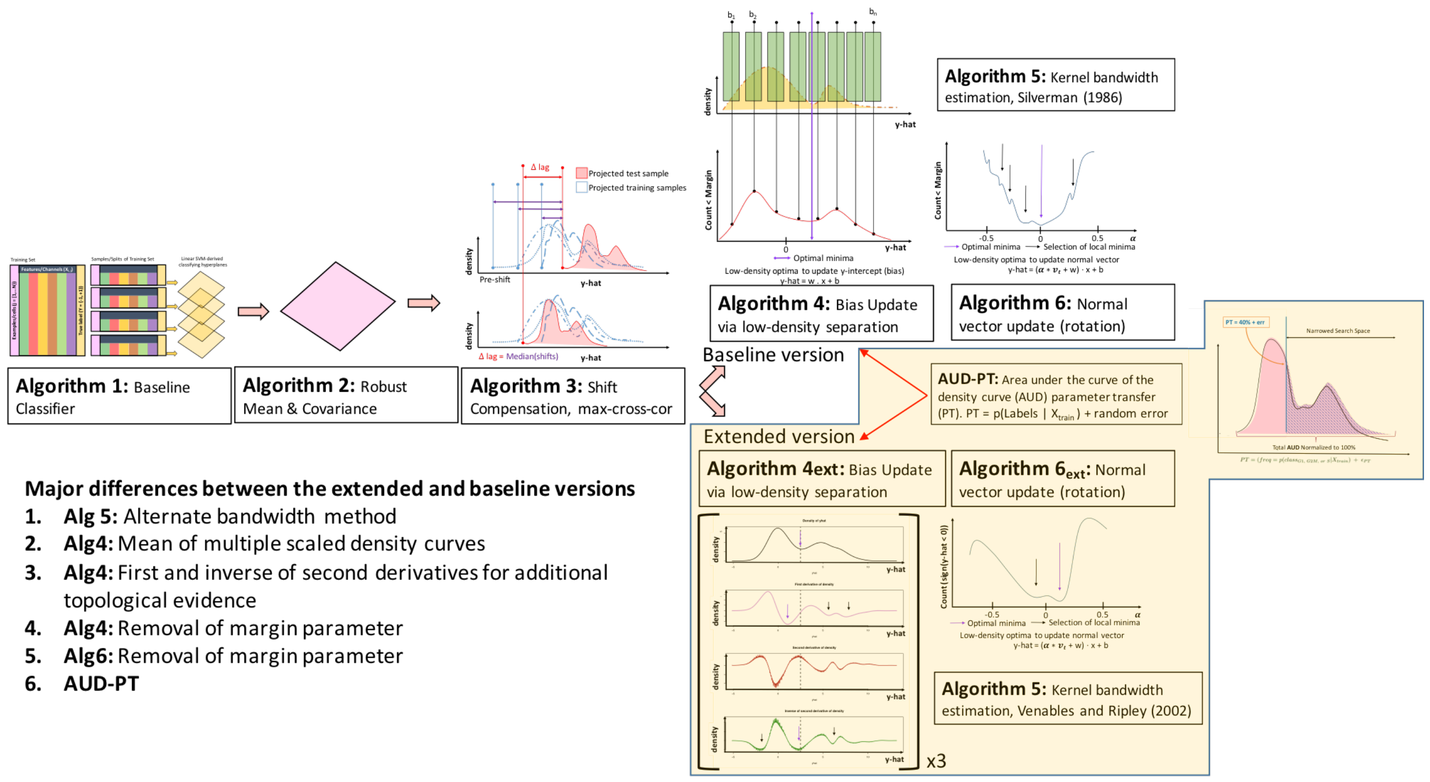


Figure 19: Classification results with the baseline implementation of the RTL framework on the 40-sample 1D simulated test set. Each sample demonstrate 3 results with 3 classification hyperplanes: the baseline (tan) is the robust mean hyperplane from Algorithm 2, the bias-updated hyperplane from Algorithm 4 (violet) and the final rotated hyperplane from Algorithm 6 (green). The 40 test samples represent four groups, A) independent but identical random samples, B) samples with their effect size reduced, c) samples with distinct changes to the marginal frequency of the positive/negative cases, and D) new samples derived with parameters combining B and C. See Table 4 for the respective effect sizes and Table 3 for the test set creation parameters.

2.3.2 Extending Algorithms 4, 5, and 6



Algorithm 1: Baseline Classifier
Algorithm 2: Robust Mean & Covariance
Algorithm 3: Shift Compensation, max-cross-cor
Δ lag
Projected test sample
Projected training samples
Pre-shift
Δ lag = Median(shifts)
density
y-hat
Count < Margin
Optimal minima
Low-density optima to update y-intercept (bias)
y-hat = w . x + b
Algorithm 4: Bias Update via low-density separation
Algorithm 5: Kernel bandwidth estimation, Silverman (1986)
Optimal minima
Selection of local minima
Low-density optima to update normal vector
y-hat = (α + v_t + w) · x + b
Algorithm 6: Normal vector update (rotation)
Baseline version
Extended version
AUD-PT: Area under the curve of the density curve (AUD) parameter transfer (PT). PT = p(Labels | X_train) + random error
PT = 40% + err
Narrowed Search Space
Total AUD Normalized to 100%
Algorithm 4ext: Bias Update via low-density separation
Algorithm 6ext: Normal vector update (rotation)
Count (sign(y-hat < 0))
Algorithm 5: Kernel bandwidth estimation, Venables and Ripley (2002)
x3

Major differences between the extended and baseline versions

1. **Alg 5:** Alternate bandwidth method
2. **Alg4:** Mean of multiple scaled density curves
3. **Alg4:** First and inverse of second derivatives for additional topological evidence
4. **Alg4:** Removal of margin parameter
5. **Alg6:** Removal of margin parameter
6. **AUD-PT**

Figure 20: A visualization of the major algorithmic modules and their functions in the RTL framework which generalize a linear classification boundary comparing the baseline and extended versions.

In testing our implemented RTL framework, we have identified potential limitations that negatively impact the overall classification. Both algorithms 4 (bias update) and 6 (normal vector update), which utilize gradient descent to identify an optimal minima defined as a region of low-density in the mapped X_{test}, suffer from bias introduced by the smoothness of the estimated density curve. This bias is the result of the existence of multiple local minimas from two main components: first, there are a low number of cells to split across each test K-fold. The second factor is the type of kernel and its bandwidth used to compute and smooth the density curve. To overcome this bias, we have implemented an alternative approach to algorithms 4, 5, and 6, described below.

The original method to estimate a kernel bandwidth for a Gaussian kernel [12] can be problematic in our testing as it commonly yields multiple local minimas. Although there are many other techniques to obtain an optimal bandwidth, we used a commonly utilized technique with a bandwidth estimate that is suitable for scaling the final smoothed estimate [13]. This capacity is important since to robustly identify the minima associated with the correct intercept for the separating hyperplane, in the extended Algorithm 4, the minima is identified as a mean location from several scaled density estimates. This approach allows us to use a set of the kernel-smoothed estimates for repeating downstream computations to obtain a robust final call. For additional evidence, the first and the inverse of the second derivatives of the density curve estimates are computed for each scaled bandwidth used. Because in our application, each classification is separating 2 classes, we use the first derivative to identify a major flip (inflection) in the direction of the curve, which identifies the local minima associated with the gap between the classes. Similarly, the inverse of the second derivative can be used to identify the peaks and valleys of the original density function; a major minima in the inverse of the second derivative is expected to be in the same place as the peak of the larger class in the original density curve. A limitation of these changes is that the additional information gained to improve the robustness of the call, comes at a significant cost to complexity; for each K-fold test set, computing at least 5 bandwidth scales, each with 3 gradient descent attempts (i.e., $y_{hat}, y'_{hat}, y''_{hat}$). On a positive note, a major gained advantage of the extended Algorithm 4 is also that we no longer require defining a margin; removing it as a confounding variable from the framework. This elimination also removes the need to initially find the optimal values for them, a minor reduction to the added computing complexity.

Next, as an alternate approach to Algorithm 6 (normal vector update) for improving its robustness, we made several changes. First, instead of counting how many of the normalized mappings fall within a predefined margin as the classification hyperplane is rotated, we count directly how many of the signed mappings of X_{test} (i.e., $sign(\hat{y})$) are less than 0 (i.e., classified as the negative case). These rotations are derived from multiplying the original direction change v_t (from Algorithm 2) by a pre-set sequence of numbers (s.t. $\alpha \in [-0.5, 0.5]$). The expected result is a valley between two peaks; the peaks pertain to rotations that result from a higher number of false negatives. Therefore, finding a minima near zero is the optimal solution with this approach, i.e., the smallest rotation with the least false negatives. Additionally, the larger of the two peaks defines the peak classification associated with the negative case label; the closer an α is chosen to this peak, the higher false negatives are expected; therefore, it is possible as an alternative to the minima near zero, other optimal points to be chosen as augmentations to the classification hyperplane. Overall, not only does this approach eliminates the definition of a margin as needed for the baseline algorithm 6, but also the computed counts provide a less rigid density curve for the gradient descent algorithm.

Finally, we introduce and test a new feature, 'parameter transfer' (PT), to transfer apriori knowledge learned from the training set regarding the frequency of the positive class label to improve the overall classification results on the unlabeled samples (Equation 33); this is especially important when focusing on rare cellular subsets (RCS). The idea is derived from the notion that a classifying hyperplane will map the positive cases with higher values, which is the main point of updating the bias of the hyperplane based on minima as in Algorithm 4; the positive and negative classes in the best case scenario should cluster with two modes. In the context of RCS, the true minima is difficult to identify as either it can be over-smoothed or diluted by other local minima. We, therefore, introduced AUD-based focusing via Parameter Transfer (PT). First, from the training set, we estimate the marginal frequency ($PT = p(class_{G1,G2M,S}|X_{train}) + \epsilon$). During the test set classification in Algorithm 4, once the density curve for the mapped test set is computed, its AUD is used in combination to the transferred PT to filter the majority of the negative cases in the test set; reducing the space which the gradient decent next searches for the optimal minima. As we assumed earlier that the marginal frequencies in the test sample differ from those in the training, we add a minor random Gaussian error the PT as in equation 34.

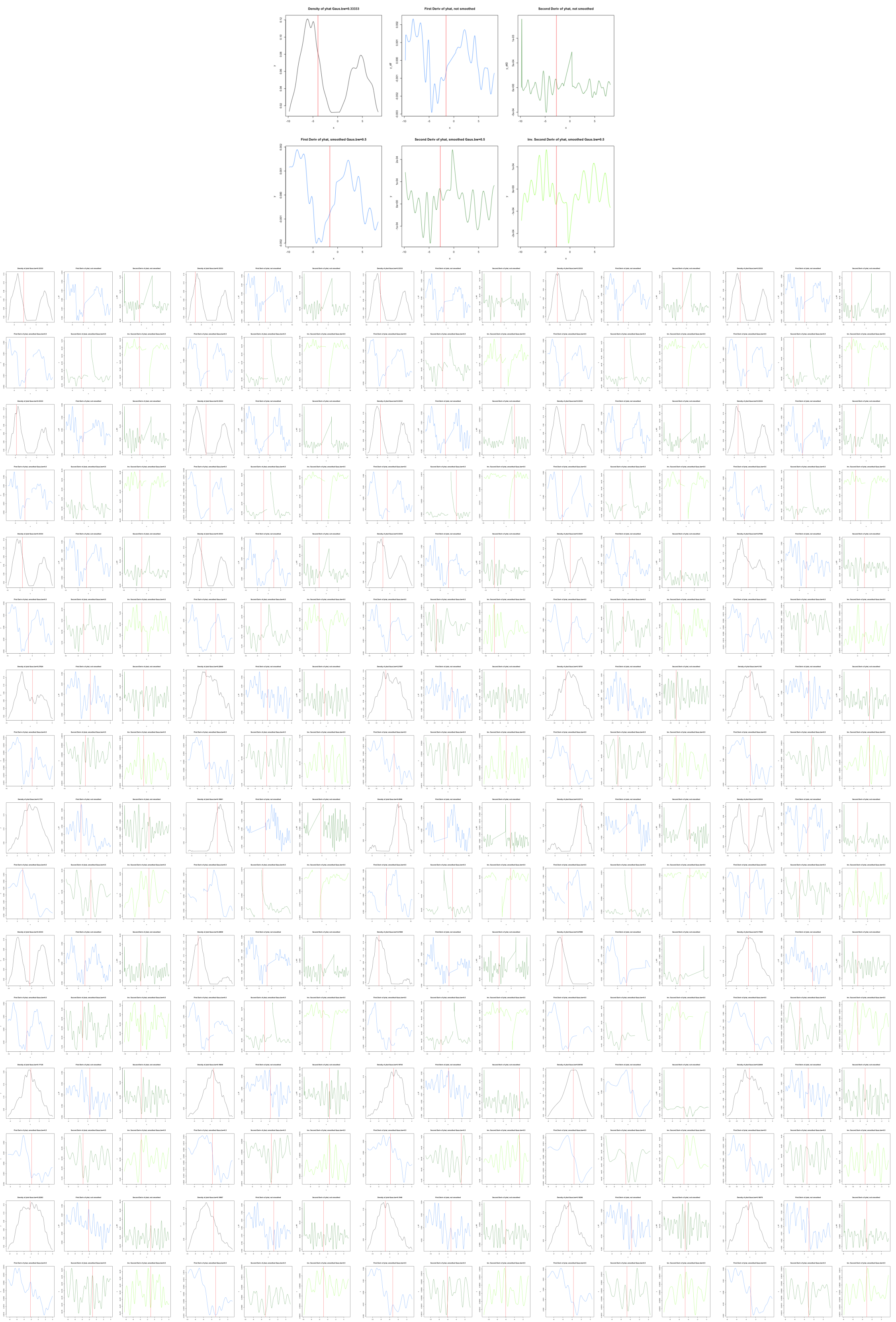


Figure 21: Updated bias on the test samples (40) using the extended implementation of Algorithm 4. Each figure set from each test sample, has 6 total figures. On the top, from left to right, the mapped and smoothed $(\hat{y})$, followed by its first and second derivatives are shown. On the bottom, the first, second, and inverse of the second derivatives are plotted and the red line indicates the optima identified respectively.As a representative, one of the figure sets is expanded on the top.

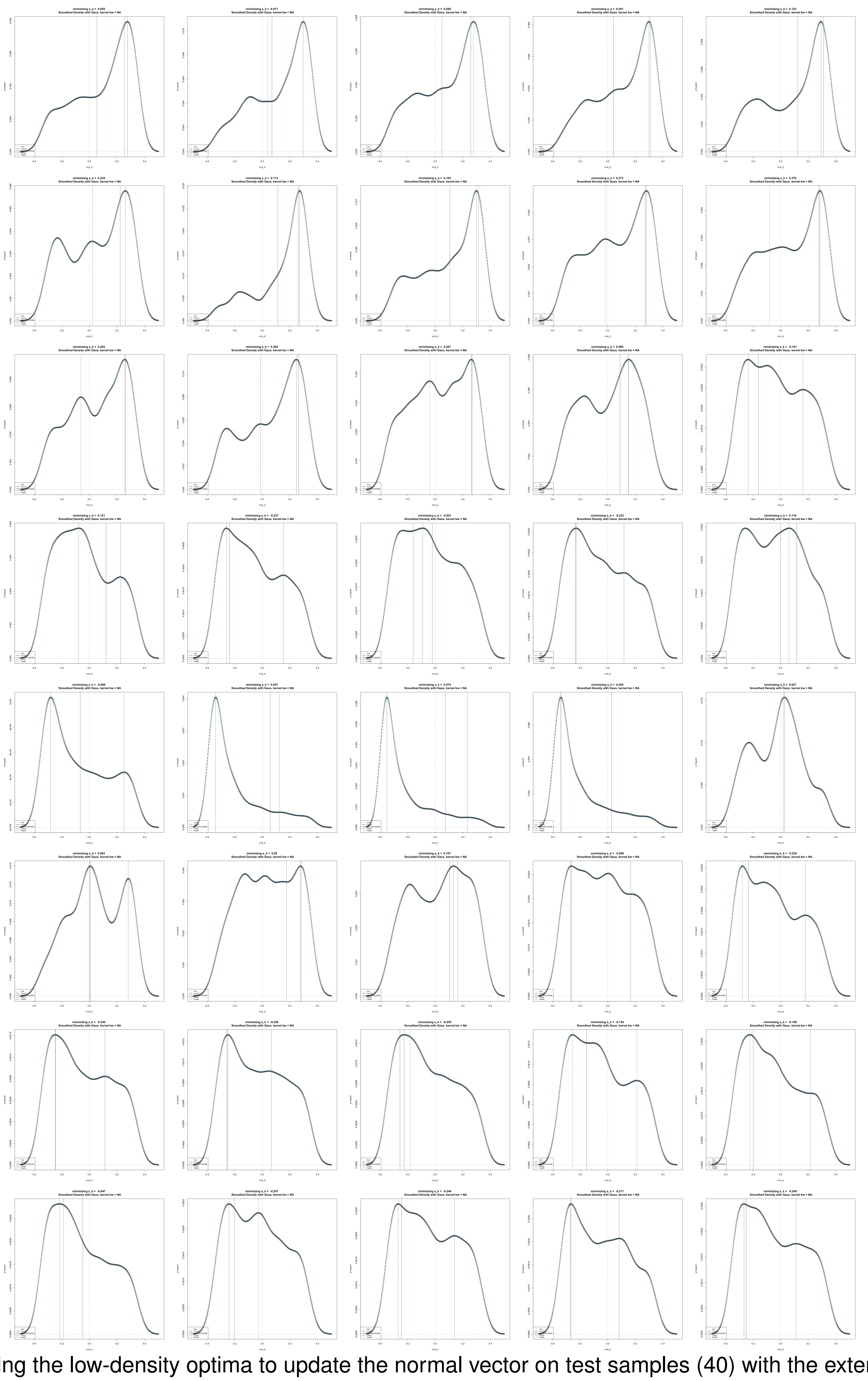


Figure 22: Finding the low-density optima to update the normal vector on test samples (40) with the extended implementation of Algorithm 6. The x-axis is the rotation factor, from -0.5 to 0.5; since very low counts are removed to improved the identification of the appropriate minima, not all x-axis show this range, rather the range of the filtered segment.

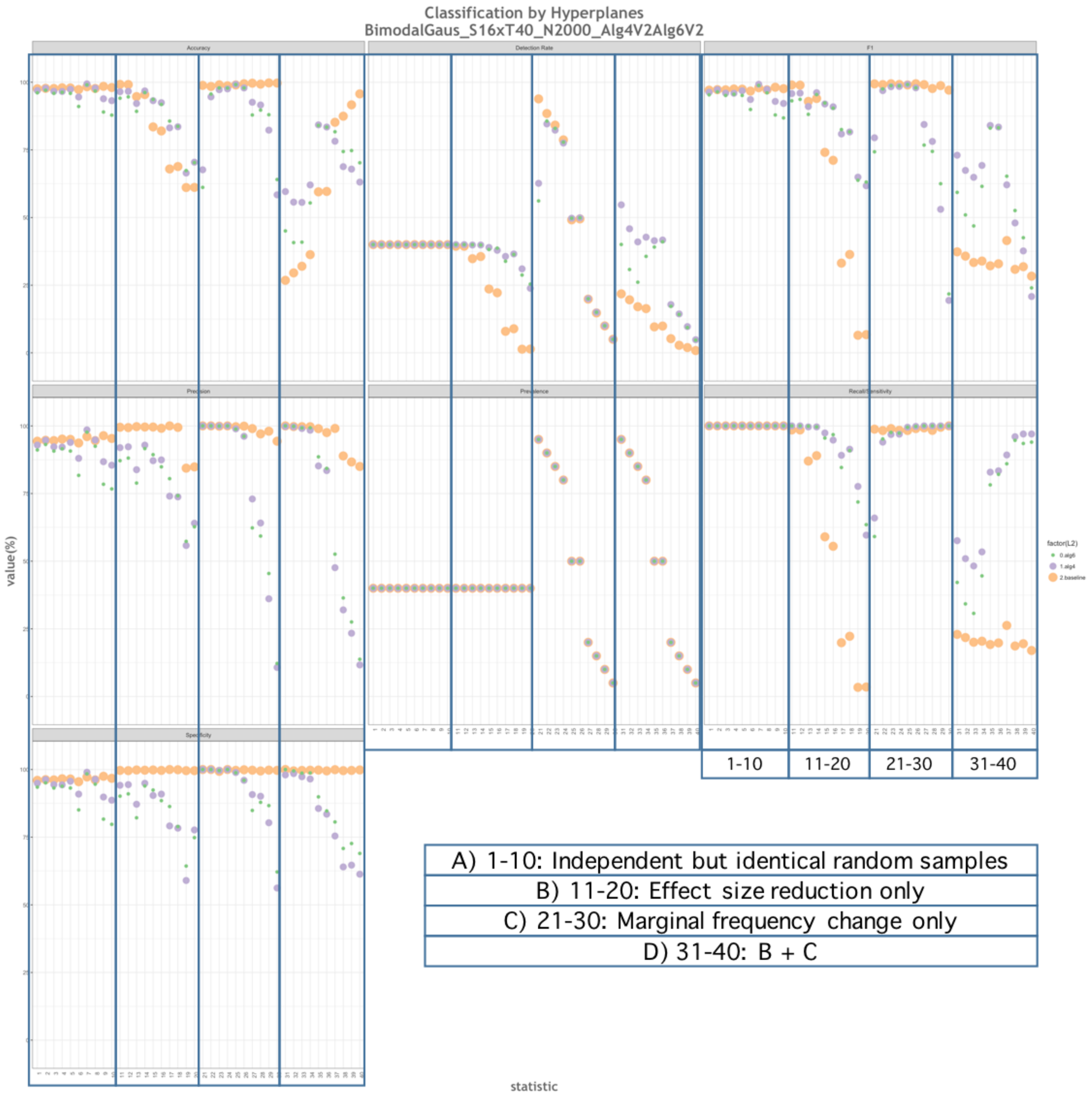


Figure 23: Classification results with the extended implementation of the RTL framework on the 40-sample 1D simulated test set. Each sample demonstrate 3 results with 3 classification hyperplanes: the baseline (tan) is the robust mean hyperplane from Algorithm 2, the bias-updated hyperplane from Algorithm 4 (violet) and the final rotated hyperplane from Algorithm 6 (green). The 40 test samples represent four groups, A) independent but identical random samples, B) samples with their effect size reduced, c) samples with distinct changes to the marginal frequency of the positive/negative cases, and D) new samples derived with parameters combining B and C. See Table 4 for the respective effect sizes and Table 3 for the test set creation parameters.

2.3.3 Discussion

To validate the RTL framework without losing intuition due to high-dimensional complexity, we simulated varying 1D training and test sets as previously described. After testing each new module and function independently to validate its performance relative to the expected outcome, we performed classification with the entire framework. We found that the main limitation of the RTL framework, regardless of which version we tested, is the effect size between the positive and negative cases; in other words, this factor has the most drastic impact on classification outcome. Altering the marginal frequency has a smaller impact on the classification performance. Both versions of the RTL framework performed similarly in this simulated evaluation. Since we have now gained confidence in the working segments and the overall functionality of the RTL framework, we next move on to non-simulated datasets.

3 Classification of cellular subsets or signatures in FC

Flow cytometry (FC) has been a powerful tool in quantifying cellular subsets. The earliest reports in enumerating rare subsets, specifically fetal red blood cells in the maternal blood (0.0001-0.00001 frequency) were done by Cupp et al. in the early 1980s [14]. In this infancy, the methods depended on standard morphologic techniques for discriminating between a few different cell type. Improvements in electronics and the FC technology as well as motivational challenges in the detection of minimal residual disease (MRD), in the context of hematological cancers, seeded the beginning stages of molecular profiling at the scope of single cells. Since measurements of MRD can be a robust prognostic of treatment outcome, the detection goal at the dawn of the twenty-first century in the context of MRD in acute leukemia was one leukemic cell among 10,000 normal bone marrow cells [60]. Nowadays, this goal has already been achieved by modern multi-parameter flow cytometry (mFC) [3]. In a different context, Nilsson et al. demonstrate the quantification of rare hematopoeitic stem cell with mFC in [61].

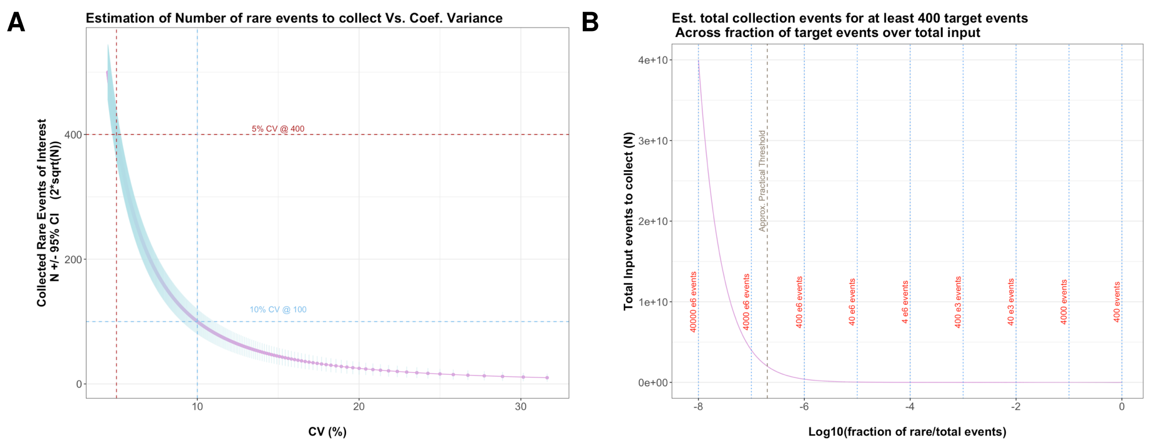


Figure 24: FC experiment estimation of number of events to collected based on formulations in [10]. **A.** Controlling coefficient of variation, by collecting N rare cells of interest. For example, if 5%CV is desired, this means at least 400 cells of the rare immunophenotype needs to be collected. $CV = SD/n_{EventsCollected} * 100$ and $SD = sqrt(n_{EventsCollected})$ **B.** How many total cells to collect (y-axis), given a priori the marginal frequency of a rare cell of interest shown as Log10(marginal frequency) on the x-axis.

Hedley and Keeney review the technical issues with rare cellular subsets (RCS) i.e., rare-event analysis in FC application in [10]. Briefly, they categorize and provide guidelines to overcome the technical issues in A) sample preparation, B) antibody and fluorochrome selection, and data acquisition. Furthermore, they provide a statistical framework to calculate the database/sample size needed, given a desired confidence and statistical power, for an RCS at a defined frequency (Figure 24. Nilsson et al. also highlight the importance of biological controls as well as gate setting controls in the design of mFC experiments; recommending using internal reference populations to improve the classification of RCS [61].

3.1 Manual Vs. Automated FC analysis

Classification of individual cells in FC analysis has been traditionally done by a manual process called 'gating'. A limitation of this approach is gating rare cellular subsets (RCS) because the 'true' state is difficult to discern. In other words, the domain expert(s) makes the call as to include a specific event within a gate. That is why manual gates risk classifying events as false positives or negatives since their classification can be subjective and non-reproducible. Although the hierarchy of gates used to identify the immunophenotypes (IP), overall improves the precision and specificity of their classification, each of the 2D scatter plots used to apply each of the gates can introduce error that accumulates and heavily confounds gating RCS. Finally, in practice we cannot assume that all of the populations in each of the 2D scatter plots are perfectly separable by a line (the most common boundary of manual gates); cellular debris, covariance between the features, autofluorescence, signal overlap, etc are known to introduce false events, spikes, and non-linearly separable boundaries in FC analysis. That said, manual gating is the current gold standard especially in the context of clinical trials where the critical quantifier is well-defined a-priori [62]. The problem is that it is difficult to scale manual gating, especially when RCS are the target; reproducibility is a key.

Computationally, two major branches of statistical learning are utilized for identifying subsets in high-dimensional data. Supervised methods can learn from curated manually gated datasets and automate the inference of specific IPs in new FC data [30]. On the other hand, unsupervised methods, identify clusters based on expression similarity; requiring the specification of a distance measure as well as a clustering method. Moreover, clustering requires the number of clusters to be given or estimated; which is not always known. Finally, in clustering, the identified clusters may not necessarily reflect biology. Unfortunately, the majority of such approaches are affected by unwanted bias and variance [7] rooted in confounding factors. For this reason, high-throughput experiments such as FC (and scRNASeq) require multi-scope standardization (i.e., experiment and analysis) with appropriate power-analyses [31]. Without such efforts, reproducibility of results are at risk due to a plethora of confounding factors which dilute the biological signal of interest.

3.1.1 Computational FC analysis

In recent years, several algorithms have been developed to address the computational needs for scaling automated, robust, and reproducible single-cell FC analysis; several are summarized in table 7. Overall, these tools are used to visualize and identify patterns in FC data. This means, there are overlaps in their inherent assumptions, limitations, and theory. For example, in many biological samples, imputing a hierarchy is important to unraveling the cellular subsets present; unlike cancer cells that may have stochastic state transitions [63] and thus no real hierarchy, immune cells are innately in a linear hierarchy. To visualize and leverage such hierarchy for downstream analysis, SPADE was developed. Initially, the input cytometry data are down-sampled to equally represent all cell types; requiring *a priori* manual gating to identify these cell types. Next, clustering identifies similar immunophenotypes (IP) on the down-sampled data. These clusters are used to compute a minimum spanning tree (MST) [8]. Finally, each of the original cells (and also any new datasets) are mapped to each of these clusters to visualize the magnitude of the IP in the sample in a quantitatively computed tree. Another clustering based tool, Citrus, delivers a statistical framework, going beyond visualization, to identify diverging signatures amongst clusters of similar cell types in conjuncture to an endpoint (or cohorts) of interest. In the next category of, i.e., dimensionality reduction, there are methods which project the samples from their input space (with possible covariates) to a constrained space. For example, in principle component analysis (PCA), an orthogonal projection, in a space where the new variables (PCs) are linearly uncorrelated. Generally, the first few PCs are identified as explanatory of the variance in the input data. Alternatively, t-SNE is non-linear dimensionality reduction method that explicitly maps the data two a 2D or 3D space, mainly for visualization of high-dimensional datasets. As the name suggests, the class of Stochastic Nonlinear Embedding (SNE) algorithms have limited reproducibility; large data with large computing resources are generally needed to be able to optimally utilize them. Finally, cell trajectory algorithms try to find order between the input cells base on their phenotypic identity.

It would be beneficial to conduct an exhaustive validation analysis of currently used tools with a standardized RCS-focused dataset. We expect that there will be an impact due to class-balance ratio, total sample size, and feature-selection [64, 65] on the performance key unsupervised and supervised methods such as k-nearest neighbor (k-NN), linear discriminant analysis (LDA), [66] random forests (RF), and [67] and support vector machines (SVM) [68].

[7]such as experimental and technical (batch, reagents, platforms, etc.), as well as biological (race, age, gender, disease, etc.)
[8]a tree connecting all clusters with minimum total edge length

General Methodological Domain	Algorithm Name	General Descriptions	Ref
Clustering			
	SPADE	Hierarchical clustering using minimum spanning tree (MST), visualizing marker progression, using uniform density across populations	[69]
	FLOW-MAP	Similar to SPADE, however uses forced-directed graph structure instead of MST	[70]
	Citrus	Signature discovery with multi-sample comparison, given clinical endpoints	[71]
	PhenoGraph	Social network algorithms clustering cells by phenotypic similarity	[72]
	Scaffold map	Force-directed graph using landmark populations	[73]
Dimensionality Reduction			
	PCA	Linear dimensionality reduction by low-dimensional embedding to best preserves their variance in HD space	[74]
	MDS	Linear multidimensional scaling by embedding to preserves the inter-point distances (equal to PCA if Euclidean distances)	[75]
	t-SNE	Stochastic Non-linear Embedding, preserving the relationships between points in HD into LD	[76]
Cell Trajectory			
	ISOMAP	Uses MDS and focuses on distances on only neighboring points	[75]
	Diffusion Map	Utilizing the assumption that the HD data lie on a LD-manifold of the dimensions (a property of repeat sampled data from the same process).	[77]
	Wanderlust	Graph-based trajectory detection converting to one-dimensional developmental trajectory	[78]

Table 7: Selected literature of automated FC-analytical methods. Although not comprehensive, this table summarizes the recent and major algorithms and pipelines.

3.2 The R computing statistical platform and importing FCS and FlowJo manual gates

There are several methods to analyze/gate FCS files. FlowJo is a software produced by TreeStar inc. that is commonly used to gate FCS data. There are also Python and R [8] platform packages that also can be used to import and analyze FCS files in a more flexible, scalable, and open-source environment. The latter has had significant proliferation amongst bioinformatics researchers in recent years; especially true for FC and other single-cell platforms [79].

3.3 Normalization, scaling, and transformation of FC data

Flow cytometry (fluorescent or mass) data, commonly stored as sets of '.fcs' [9] files, house the expression matrix with rows of cells and columns of measurements/features. Due to the various technical and experimental causes, the range of each channel is not the same across samples. For example, in fluorescent flow, each feature is a measure luminescence from the photochrome-labeled antibodies whereas, in mass cytometry, each feature is a measure of time of flight (TOF), as metals with various masses are bound to antibodies of the experiment. Naturally, for within platform analysis and interoperability across platforms, appropriate normalization, scaling, and transformations are needed. Common transformations are the Logicle (biexponential) [80], Box-Cox [81], and the generalized hyperbolic arcsine [82] transformations. Using simulated data, we demonstrate the consequence of such transformations. The Logicle (biexponential) transformation is shown in Equation 5; given the default attributes a = 0.5, b=1, c=0.5, d=1, f=0, w=0, then, it is simplified to Equation 6. Next, the adjusted arcsinh function (Equation 7 where the default values for the parameters are a = 0, b=1, and c=0. Finally the Box-Cox transformation (Equation 8 is shown and defined with the condition that parameter $\lambda > 0$. For a technical analysis of normalization on a large FC dataset, we refer to the recent work in [62]; unless stated, for our analysis, we use the asinh() transformation.

$$biexp(x) = a * exp(b * (x - w)) - c * exp(-d * (x - w)) + f \tag{5}$$

$$biexp(x) = (exp(x) - exp(-x))/2 \tag{6}$$

$$asinh(x) = asinh(a + b * x) + c \tag{7}$$

$$BoxCox(x) = (sign(x) * abs(x)^{\lambda} - 1)/lambda \; if \; \lambda > 0 \tag{8}$$

[9]a compressed MD5 structured file containing the expression matrix and pertaining metadata

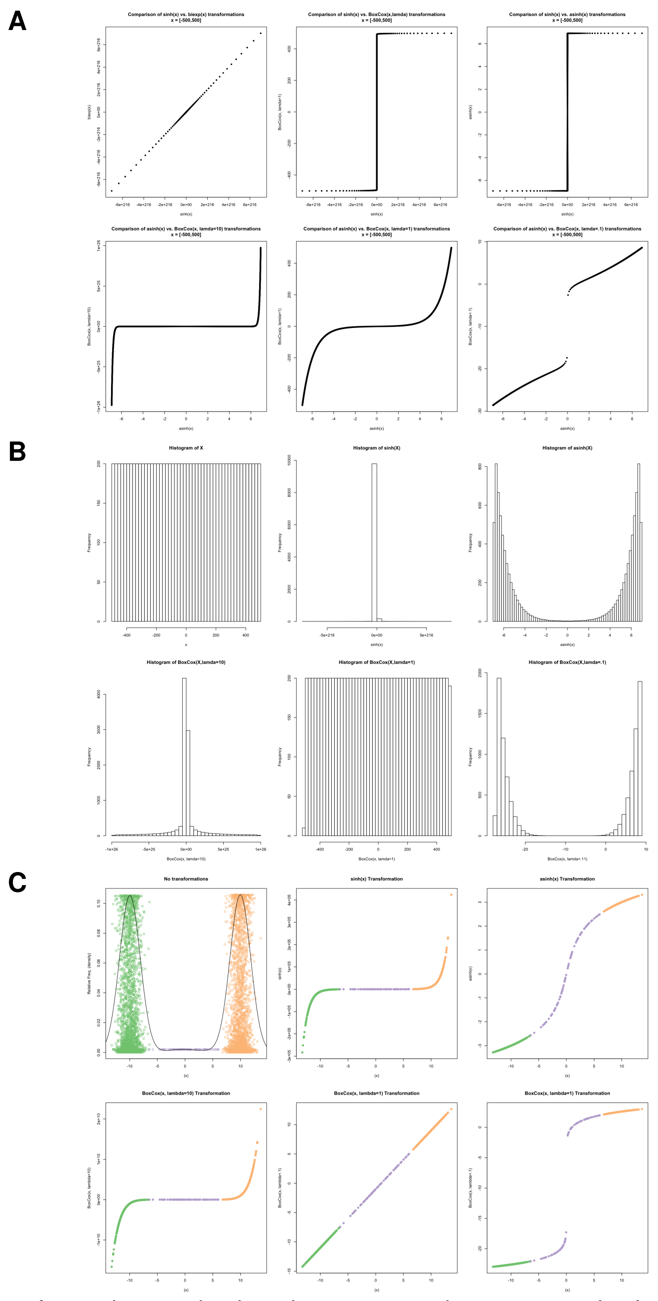


Figure 25: A. Comparison of transformations paired to demonstrate the range and relationships, where $x \in [-500, 500]$. **B.**The transformation consequence on the distribution is shown using histograms. **C.** Using a simulated trimodal sample (3 normal distributions where $\mu_1 = -10, \mu_2 = 0, \mu_3 = 10$) the consequence of the of the transformations are shown relative to the 3 labels.

3.4 The FlowCAP challenges

FlowCAP is the project seeded by a consortium of researchers aimed at providing standard and open datasets for algorithm development; enabling direct comparison of algorithmic performance towards specific tasks. FlowCAP I and II focused on developing computational tools to mimic manual gaiting as well as automated methods. Overall, they have provided several tools which can, in fact, achieve robustly automated gating for the majority of the cell populations. However, as there are limitations with FC analysis (i.e., antibody affinity, emission spectra overlap and autofluorescence) the classification quality also varies across cell types (i.e., monocytes vs. t cells), cohorts (e.g., infants vs. adults), and are heavily impacted by debris contamination. Although these approaches are useful for specific tasks, pre-processing is mandatory such as normalization to remove unwanted technical noise across datasets. No classification method is perfect; multiple iterations and validation by multiple experts are needed. Finally, an emergent result from the earlier FlowCAP challenges was that rare cellular subsets (RCS) are currently the research bottleneck (manual or automated).

3.4.1 FlowCAP-III Challenge 1 data

In response to the need for developing and evaluating computational tools that are robust and reproducible, especially in the context of RCS, FlowCAP-III includes a computational challenge focused on two RCS. A classification challenge evaluating inferred predictions against manual gating using the $F1_{score}$ on two rare immunophenotypes. Unfortunately, this dataset [10] has not been fully released yet as a challenge set [83]. Briefly, there were 405 FCS files, where about half (202) of the manual gating results are given for training where the labels (i.e., Y) were 0, 1, and 2 ("ungated", cell population 1, and cell population 2 respectively). The task was to infer these populations in the next half (403) files (i.e., $\hat{Y}$) and compare the classification results to the manual gating; although in the context of the challenge, the participants were blind to the true test set labels. Qui has demonstrated via a faithful down-sampling methodology [84] that a linear ensemble SVMs classifier can be trained to identify rare cellular subsets (RCS); The down-sampling is used to increase the influence of rare subsets on the algorithm. They also highlight the utility of the multivariate distance metric, the Hellinger divergence to quantify similarity across samples, which is claimed to account for the batch effects. This method can be investigated to be incorporated in our extended TL-FC framework to assess improvements of the TL predictions. They achieved an overall F-measure of 0.64 on the FlowCAP-III dataset. Unfortunately, for our benchmarking and evaluation needs, it is not possible to directly compare to this results.

3.5 HVTN080 immunogenicity and safety trial data

To develop and evaluate computational methods for flow cytometry (FC) a large well-curated dataset is needed. For our analysis herein we focus on a publicly available dataset [11] in the context of HIV vaccine safety and immunogenicity [85]. This study [12] is similar to the FlowCAP-III data, with available manual gating, which enables evaluating the classification learning and inference. Specifically, this dataset is produced experimentally using an intracellular cytokine staining (ICS) and phenotyping assay. Furthermore, both of the populations in FlowCAP-III Challenge 1 are present in the HVTN080 data.

3.5.1 Intracellular cytokine staining (ICS) assay in the context of HIV

Briefly, HIV is retrovirus composed of two RNA strands housed by a protein capsid [86]. The RNA code contains many open reading frames (ORFs) for structural and regulatory viral proteins, specifically the HIV genome is known to have 9 genes coding for 15 proteins [87, 88]. Three of these proteins, the structural polyproteins, are processed to smaller proteins: interior proteins i.e., group-specific antigen (Gag), viral enzymes (Pol) such as the reverse transcriptase, and the envelope (env). Peptides from such proteins are used as the stimuli experimentally; such as Intracellular cytokine staining (ICS) experiments. ICS is a common tool in the immunologist's toolbox, which enables the quantification of accumulated cytokine in cellular vesicles, over a period of time, in various conditions. The accumulation of the vesicles is usually because of a fungus-derived molecule, Brefeldin A that blocks protein transport from the endoplasmic reticulum (ER) to the Golgi apparatus.

3.5.2 Exploratory Data Analysis on the HVTN080 data

The HVTN080 FC data is from a t cell stimulation assay with 4 stimulations conditions (Negative-Control, Gag, Pol1, and Pol2), where responding cells are identified if they produce select cytokines in response to the specific stimuli. Once the full dataset is acquired, we find that there are a total of 470 FCS files corresponding to 48 healthy human subjects assayed at Day 0 and at day 40 post-vaccine. In addition to the FCS files, the manual gates are imported via the provided FlowJo

[10] The FlowCAP-III challenge 1 dataset pertains to the External Quality Assurance Program Oversight Laboratory (EQAPOL) project Proficiency Test.

[11] `https://flowrepository.org/id/FR-FCM-ZZ7U`

[12] "Safety of and Immune Response to the PENNVAX-B DNA Vaccine With and Without IL-12 in HIV-uninfected Adults." ClinicalTrials.gov Identifier: NCT00991354. https://clinicaltrials.gov/ct2/show/NCT00991354

exported XML files which reveals the files are split to 15 batches. Furthermore, there are several compensation files that represent a subset of the total 470 FCS files; reducing the set to 435 usable FCS files. The channels available are FITC, PerCP Cy55 Blue-A, Pacific Blue, AmCyan, APC-A, Alexa 680, APC Cy7, PE-Green, PE-TxRD, PE Cy5-A, PE Cy55-A which correspond to the targets "TNFa", "CD8", "CD57", "Live/Vivid", "Perforin", "GranzymeB", "CD4", "IL2", "CD3", "NA", "NA", and "IFNg"; these targets are cell-surface markers as well as several intracellular targets. The gating hierarchy reveals that there are 148 defined gates, but since our goal is the classification of one of these specific immunophenotype (IP) vs. the rest, not all gates are needed.

	Day 0	Day 5	Total
GAG-1-PTEG	48	46	94
negctrl	96	92	188
POL-1-PTEG	48	46	94
POL-2-PTEG	48	46	94
Total	240	230	470

Table 8: Full HVTN080 data, FCS count across the 2 time-points.

	Neg.CNTRL	HIV.GAG1	HIV.POL1	HIV.POL2	Total
Batch0880	16	8	8	8	40
Batch0881	16	8	8	8	40
Batch0882	12	6	6	6	30
Batch0883	12	6	6	6	30
Batch0884	12	6	6	6	30
Batch0939	6	3	3	3	15
Batch0940	8	4	4	4	20
Batch0941	10	5	5	5	25
Batch1049	14	7	7	7	35
Batch1050	8	4	4	4	20
Batch1053	16	8	8	8	40
Batch1054M	16	8	8	8	40
Batch1055M	16	8	8	8	40
Batch1056M	10	5	5	5	25
Batch1078	2	1	1	1	5
Total	174	87	87	87	435

Table 9: HVTN080 FCS count per Batch; 15 total batches, 435 FCS files; those with available manual gating that are not compensation controls.

Specifically, from the manual gating hierarchy we extract the "root" node cells i.e., all cellular events and debris. The IP of interests is the CD4 t cells, gated as "/S/Lv/L/3+/4+" i.e., singlet, live, lymphocytes, that are CD3 and CD4 positive; Table 10 summarizes the distribution of cells across the entire dataset. There is also another gated IP of interest, the stimulated IL-2 producing cytotoxic t cells, "/S/Lv/L/3+/8+/IL2+". However, this IP, is not well represented across all the samples; the majority of the data has less than 5 single positive events with many being 0. This is somewhat expected as the negative controls should have 0 stimulated t cells. To conduct classification with this IP, requires hand curation of the training and test sets. Therefore, to evaluate classification performance compared to manual gating, the CD4 t cell IP is better because: A) sufficient representative positive examples are present in all samples. B) there is biological variation across the subjects/donors. C) The experimental conditions are not expected to confound the marginal frequency of the IP [13] and thus can be considered as replicates. D) CD4 helper t cells are clinically important in the context of HIV pathogenesis. Therefore, we narrow our focus and reduce the input feature space to include the most relevant (per gating) features: FSCA, SSCA, APCCy7(CD4), PE-Green (IL-2), PerCPCy55Blue (CD8), PE-TxRD(CD3), and AmCyan (Live/Vivid). Specific to the CD4 t cells, the features used are the SSCA, AmCyan (Live/Vivid) , TxRD(CD3), and APCCy7(CD4).

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Freq. CD4 t cells of total/root	0.17	0.30	0.34	0.34	0.39	0.51
Freq. IL2+CD8 t cells of total/root	0.0000000	0.0000045	0.0000108	0.0000185	0.0000212	0.0006040
No. of CD4 t cells	31929	62024	74591	76688	90729	139330
No. of IL2+CD8 t cells	0	1	2	4	5	189
No. of total/root cells	117539	190746	220104	226366	258533	414124

Table 10: Distribution of two main immunophenotypes across the usable 435 FCS files in the HVTN080 dataset.

Prior to downstream classification analysis, to visualize the variance and topology of the entire dataset, we combined all of the 3000-event sampled FCS files (435 total); as our focus is the CD4 t cell immunophenotype, this dataset has the input feature space of SSCA, AmCyan (Live/Vivid) , TxRD(CD3), and APCCy7(CD4) thus a total of four principle components are computed. As summarized in Table 11, PC1 contains the majority of the variance proportionally. When we visualize the projection (Figure 26), i.e., the loadings of the PCA, where the co-linearity (linear covariance) amongst the input features

[13] In specific conditions, the expression of a marker might change (without change to the marginal frequency). For example, in the positive control condition with the 'SEB' super antigen, downregulation of CD3 and CD8 markers are sometimes observed.

has been removed, we observe two distinct clusters across PC1 and PC2. Furthermore, when we plotted the known confounders (batch, experimental conditions, and timepoint), specific patters identifying clusters of these confounders were not found, indicating that the majority of variance observed in these principle components is not due to these confounders. Finally, we identify the CD4 t cells clustered mostly together, indicating the homogeneity/similarity of their signature.

	PC1	PC2	PC3	PC4
Standard deviation	42620.8927	986.6542	507.2750	428.5234
Proportion of Variance	0.9992	0.0005	0.0001	0.0001
Cumulative Proportion	0.9992	0.9998	0.9999	1.0000

Table 11: PCA analysis with HVTN080 dataset, combining 3000 random examples from each 435 FCS samples, specific to the CD4 t cells, with the input feature space SSCA, AmCyan (Live/Vivid) , TxRD(CD3), and APCCy7(CD4).

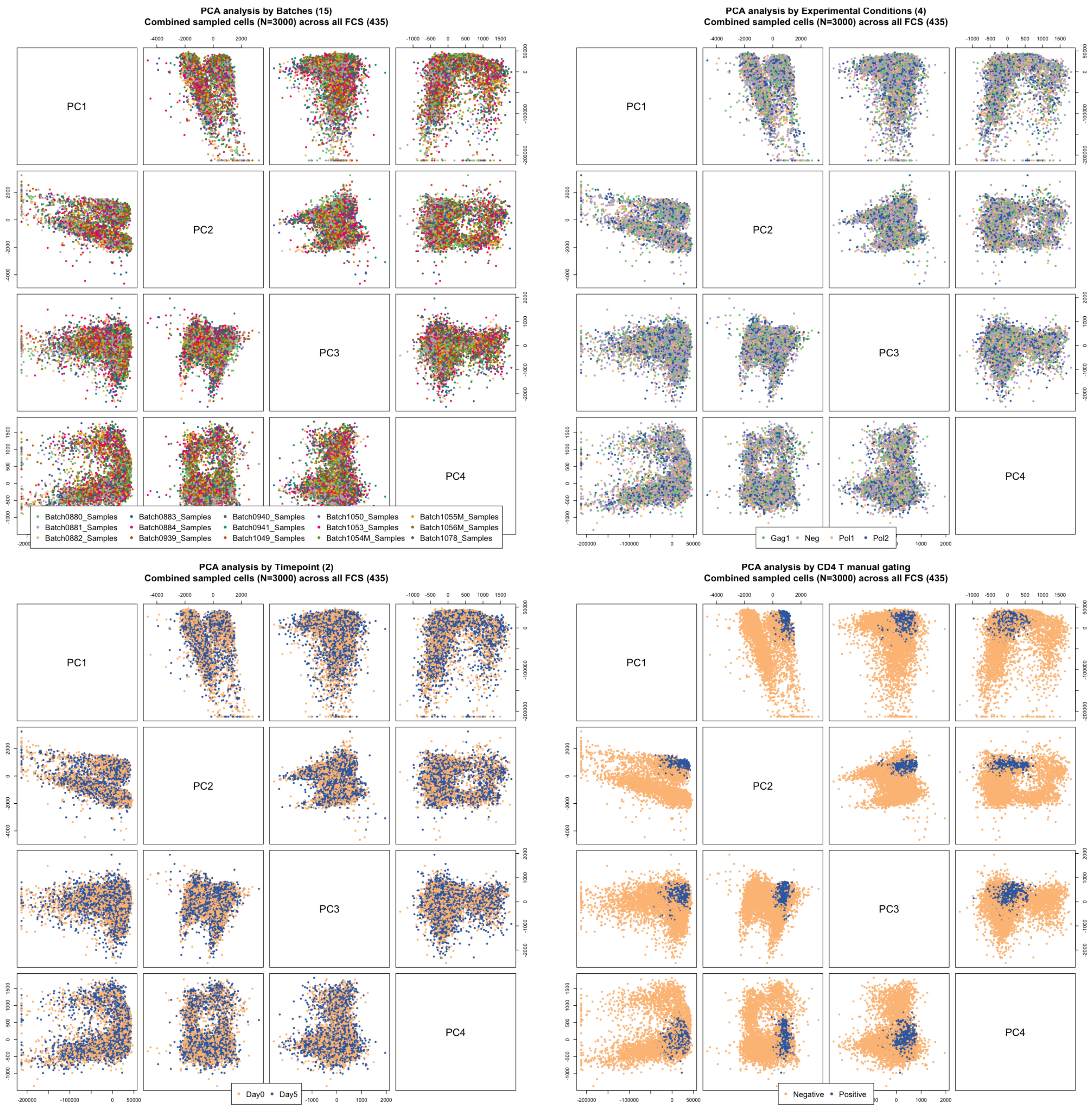
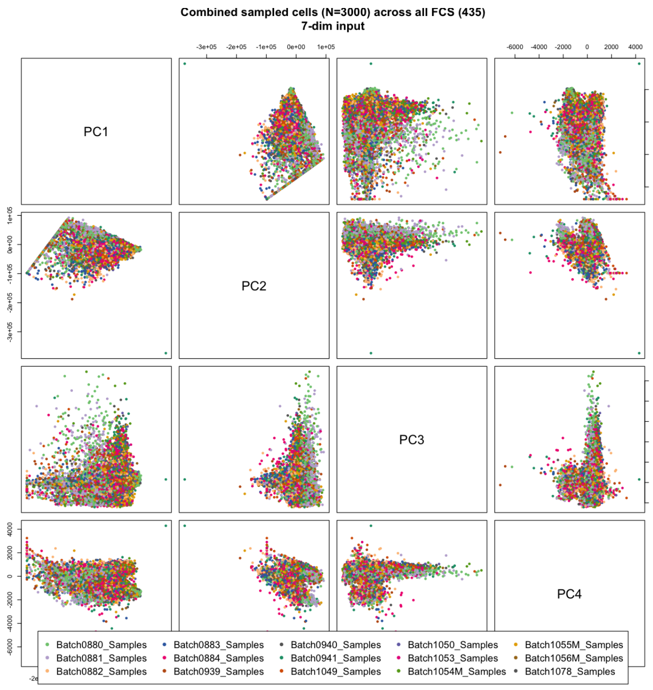
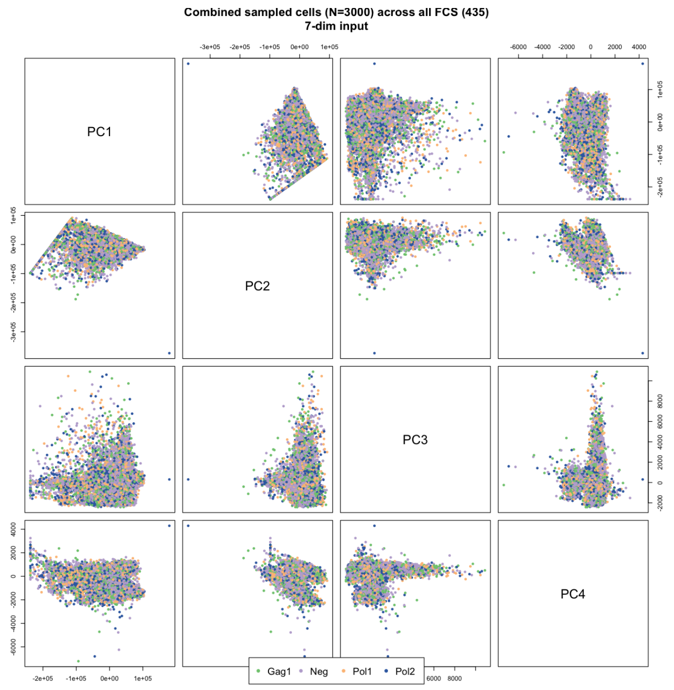
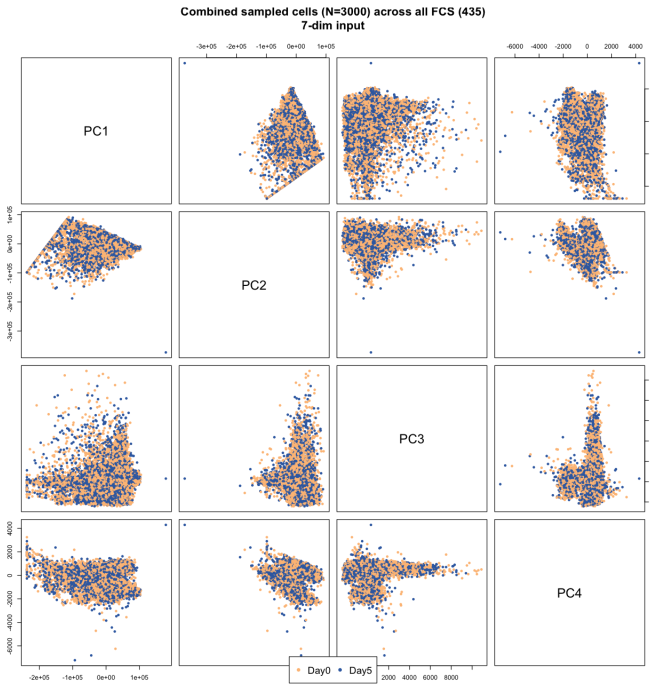


Figure 26: PCA with HVTN080 dataset, combining 3000 random examples from each 435 FCS samples, specific to the CD4 t cells, with the input feature space SSCA, AmCyan (Live/Vivid) , TxRD(CD3), and APCCy7(CD4). Plotted are the loadings from the four principle components. From top left, clockwise, the colors represent the batches, the experimental conditions, the actual gated CD4 t cells vs. rest (root gate), and finally the vaccination timepoints. In these figures, a representative random sample (N=6000) is plotted.







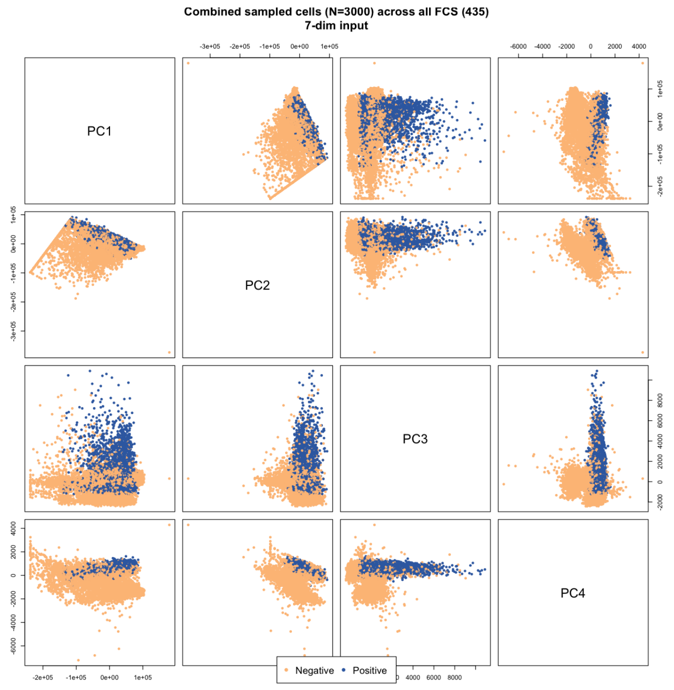


Figure 27: PCA with HVTN080 dataset, combining 3000 random examples from each 435 FCS samples, with additional input features than Figure 26; i.e., FSCA, SSCA, APCCy7(CD4), PE-Green (IL-2), PerCPCy55Blue (CD8), PE-TxRD(CD3), and AmCyan (Live/Vivid). In these figures, a representative random sample (N=6000) is plotted.

3.5.3 Training and Testing sets from with the HVTN080 data

To create the training and test sets, we start with the 435 actual experimental files with manual gating. To reduce the processing speed and to create samples with an equal number of events, each of the 435 samples was first shrunk by randomly selecting 100,000 cells from the root of the gating hierarchy; all cell subsets including debris. Subsequently, in the final stage of creating the training/test sets, we reduce each sample to 5000 random cells, with the option to be sampled at each epoch to avoid sample-bias via multiple iterations. Although there are several ways to create training and test sets, we have approached it in two schemes as shown in Figure 28. In *Scheme 1*, we assume agnosticism to all confounders (known and unknown) and create a completely random 1:1 split of the dataset. Specifically, we first randomly split this dataset to 220 training and 215 test samples. Next, From each, we randomly selected 200 of the samples to create 10 balanced training and test sets, each housing 20 random FCS files. This avoids training on the entire 200+ samples which not only reduces the training time but also reduces the likelihood that a set of outliers in the samples would heavily impact the inference. Furthermore, this strategy over several epochs allows for an iterative process where all the FCS files are eventually part of the classification dataset; reducing the effects of sample-bias. *Scheme 2* is designed to improve the inference on any set of test samples by combining the predictions from multiple sets. The combination can be done in two ways: First, each training set's generalized hyperplane obtained (i.e., Algorithm 2's robust mean hyperplane) can again be averaged then transferred to the test set, followed up by updating its bias and normal vector in downstream algorithms. Alternatively, each training set's hyperplane independently is transferred and updated in Algorithms 4 and 6, and the final calls are used to obtain each of the cells probability of being labeled either positive or negative. Below we conducted classification within the RTL framework using both scheme 1 and scheme 2; in the latter, we conducted the cell-level probability approach.

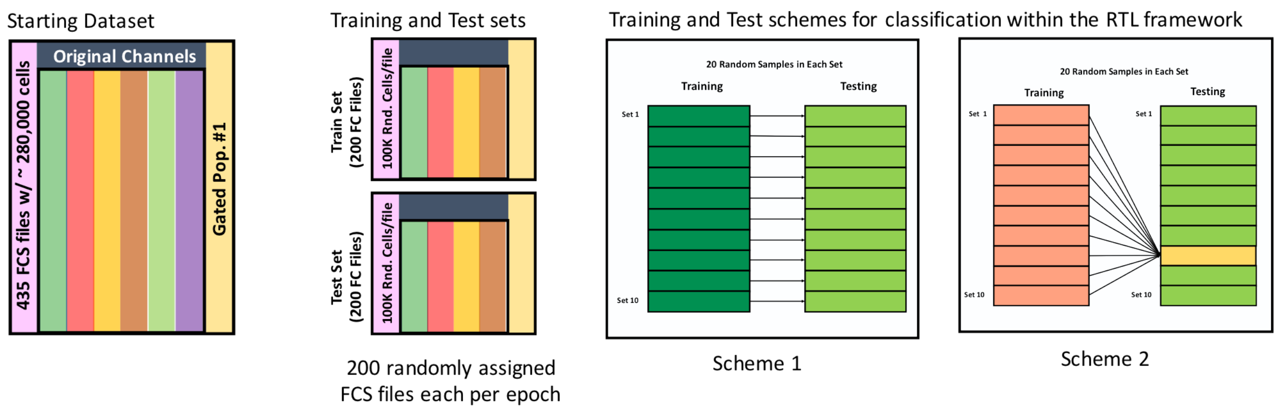


Figure 28: Creating training and testing sets from the acquired HVTN080 435-FCS files. Here the **Starting Dataset** which is actually 435 different FCS files, is summarized as a single expression matrix; **The Training and Test Sets** are similarly visualized. The example shown is specific for the gated CD4+ t cells, where the relevant features are the SSCA, AmCyan (Live/Vivid), TxRD(CD3), and APCCy7(CD4). For each epoch, the 220 FCS files are assigned to the training set and the remaining 215 to the test set. From each, 200 FCS files are randomly selected to create the final training and test sets. Next, in the two training and testing schemes are shown, each test set contains 20 random FCS files without replacement. In Scheme 1, the training from each of the FCS files per set is generalized and transferred to the test set of FCS files. Alternatively, for a more exhaustive approach, the generalized training from all of the 10 training sets is used on each test set to compute a final classification.

3.6 Classifying CD4 t cells from HVTN080 within the RTL Framework

In our initial approach, we employ the training-test set **'scheme 1'** (Figure 28) which starts with the 435 total FCS. From these files, 200 random non-repeated samples are assigned to the training set and 200 to the test set. Each of the 200-sample sets is further split into 10 sets of 20 random non-repeating samples. The goal of designing such training and test sets is so that their creation is completely agnostic to any potential confounders from the meta-data such as batch, subject, timepoint, etc. This approach at its core demonstrates the capacity to generalize classification of an immunophenotype from any set of training samples to any set of test samples. The limitation of this approach is that if one of the training sets yields a hyperplane that is poorly generalizable, the corresponding test set samples will not be properly classified, and also difficult to adapt to the test set within the transfer learning framework. Consequently in our alternate approach, **'scheme 2'** (Figure 28), instead of a 1 training set to 1 test set, we use the all of the 10 training sets per each test set (i.e., 10 to 1). To demonstrate this approach we created a single 50-sample test set and the training sets was created similar to scheme 1 (i.e., 10sets x 20 random fcs each). A final classification call is made by averaging across all the calls from each of the training sets per cell; since the calls are +1 (CD4 t cell) or -1 (not/rest) if the average is 0, it means that an inference was not possible on that individual cell, thus it was computed as an 'NA' for the computation of the confusion matrix. Alternatively, a probability-based approach and a threshold (e.g., 50%) can be used to determine the final call.

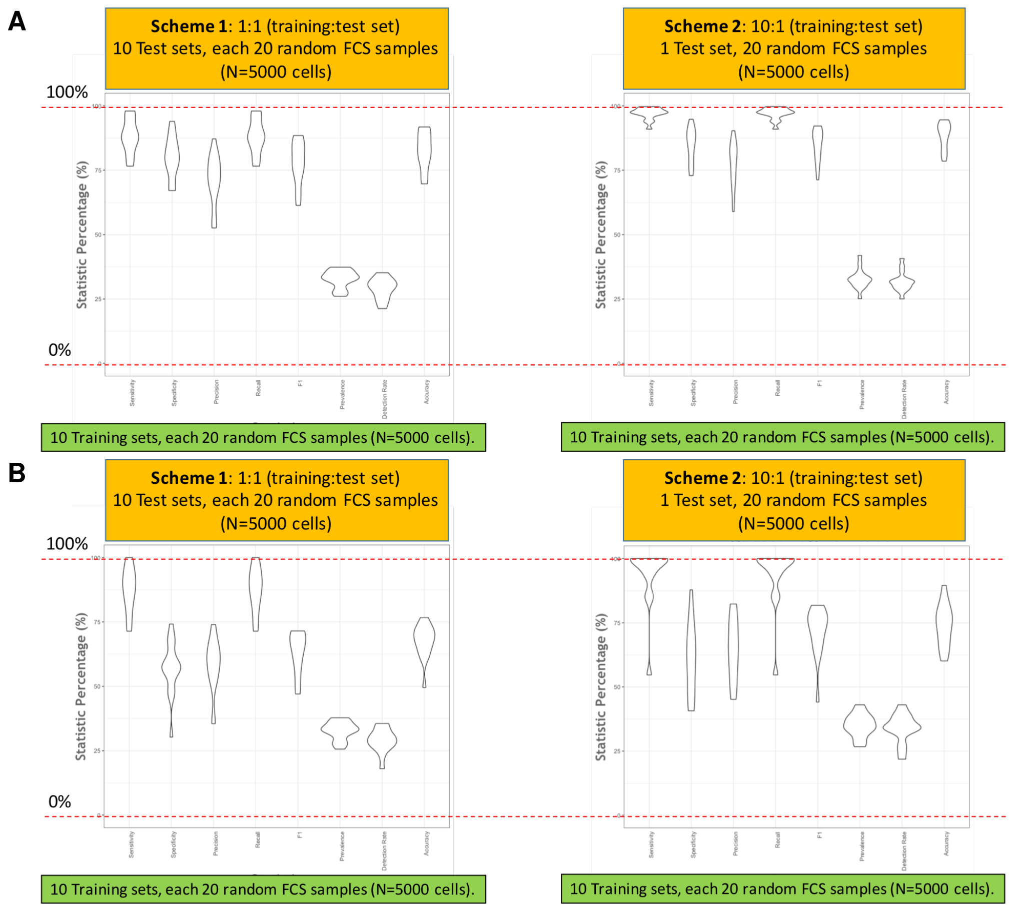


Figure 29: A summary of $F1_{score}$ results for both train/test data sampling schemes and both versions of the RTL framework. Violin plots of major classification statistics classifying CD4 t cells from the HVTN080 dataset **A.** Baseline version results. **B.** Extended version results.

3.6.1 Scheme 1, the baseline version

Training and inference are done one set at a time and the results are separately visualized in Figure 30. We observe that on average, the baseline transferred hyperplane, i.e., the robust mean hyperplane from the training set, performs fairly well on the test sets (Table 12). Across sample batches, timepoints, and experimental conditions we did not observe major differences with regards to classification in the test set. However, we do observe subject specific variation with regards to specificity and precision. This is in contrast to the very high recall observe, across the subject indicating that most of the CD4 t cells were identified.

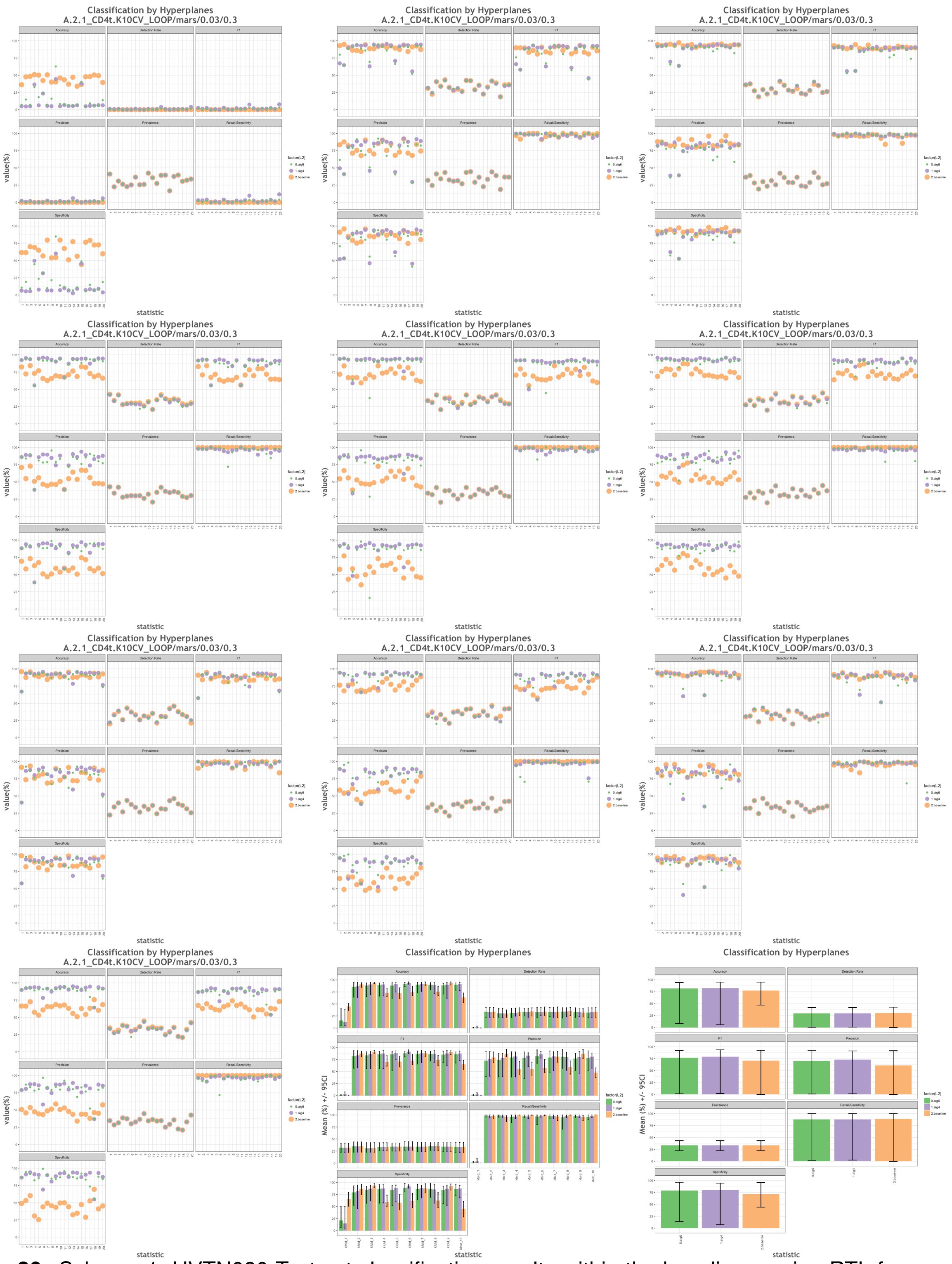


Figure 30: Scheme 1, HVTN080 Test set classification results within the baseline version RTL framework, tuned for FC data analysis. From top left going across, the results from test sets 1-10 are shown followed by a per set aggregate and cumulative bar graphs with 95%CI. For each, we demonstrate Accuracy, Detection rate, $F1_{score}$, Precision, Prevalence, Recall/Sensitivity, and Specificity. The Baseline (tan) result is the direct transfer of the robust means hyperplane from Algorithm 2. The Alg4 (plum) result is from the baseline hyperplane with the updated bias after Algorithm 4. Finally, Alg6 (green) result is after rotating the hyperplane in Algorithm 6.

	Sensitivity	Specificity	Precision	Recall	F1	Prevalence	Detection Rate	Accuracy
Average	88.23	79.99	71.28	88.23	77.90	32.77	29.09	82.61
By Batch								
Batch0880_Samples	97.36	82.13	75.41	97.36	83.55	36.23	35.19	86.76
Batch0881_Samples	80.23	75.07	69.06	80.23	74.04	37.37	29.87	76.99
Batch0882_Samples	89.48	75.99	64.80	89.48	73.41	32.67	29.82	80.34
Batch0883_Samples	90.03	84.25	76.93	90.03	82.53	32.46	29.62	86.41
Batch0884_Samples	95.36	83.22	76.45	95.36	84.02	33.73	32.20	87.31
Batch0939_Samples	85.93	78.92	70.24	85.93	76.62	35.58	31.83	81.66
Batch0940_Samples	86.82	93.96	87.18	86.82	86.00	29.60	25.62	91.77
Batch0941_Samples	98.00	87.15	80.70	98.00	88.49	35.19	34.51	91.05
Batch1049_Samples	86.88	68.15	53.23	86.88	64.43	26.04	22.17	72.41
Batch1050_Samples	97.70	88.00	80.50	97.70	88.17	33.15	32.41	91.30
Batch1053_Samples	87.19	78.88	69.21	87.19	76.66	34.60	30.17	81.77
Batch1054M_Samples	82.34	78.00	73.17	82.34	77.14	37.37	30.93	79.71
Batch1055M_Samples	77.37	67.08	52.60	77.37	61.33	27.31	21.31	69.71
Batch1056M_Samples	76.62	67.28	58.83	76.62	65.97	32.64	25.23	70.21
Batch1078_Samples	92.12	91.73	80.91	92.12	86.13	27.66	25.47	91.82
By Subject								
080-1	95.85	88.11	86.61	95.85	91.00	44.52	42.68	91.56
080-10	98.53	87.06	82.61	98.53	89.82	38.88	38.29	91.46
080-11	98.16	90.38	84.93	98.16	91.02	36.12	35.46	93.16
080-12	50.00	36.72	19.42	50.00	27.98	24.41	11.68	39.60
080-13	79.89	69.71	64.58	79.89	71.42	36.97	29.42	73.46
080-14	81.07	76.14	67.99	81.07	73.85	31.27	25.22	77.68
080-15	99.03	87.82	80.82	99.03	88.94	34.30	33.97	91.64
080-16	92.12	91.73	80.91	92.12	86.13	27.66	25.47	91.82
080-17	97.49	87.39	79.31	97.49	87.15	31.71	30.91	90.59
080-18	97.55	88.39	82.42	97.55	89.22	34.95	34.11	91.65
080-19	66.67	46.25	22.30	66.67	33.29	26.34	17.67	51.71
080-2	98.45	69.82	55.46	98.45	69.21	25.69	25.21	76.61
080-20	98.63	85.03	83.56	98.63	90.46	43.33	42.74	90.96
080-21	98.78	86.91	84.83	98.78	91.27	42.45	41.93	91.96
080-22	74.33	68.45	57.29	74.33	64.56	27.22	20.17	70.00
080-23	94.01	92.52	83.60	94.01	87.80	27.66	25.97	92.88
080-24	88.90	92.56	84.56	88.90	85.68	28.82	25.59	91.47
080-25	99.51	77.80	62.26	99.51	76.48	26.56	26.43	83.57
080-26	98.42	86.40	71.01	98.42	82.48	25.26	24.86	89.43
080-27	71.33	63.17	41.08	71.33	51.82	24.43	17.39	65.07
080-28	98.36	90.58	84.04	98.36	90.62	33.60	33.06	93.20
080-29	50.92	45.53	39.19	50.92	44.28	37.51	19.59	47.68
080-3	96.41	88.11	82.29	96.41	88.78	36.37	35.07	91.13
080-30	79.52	68.44	46.64	79.52	56.95	22.60	19.12	71.21
080-31	98.81	85.65	79.61	98.81	88.15	36.11	35.68	90.39
080-32	98.41	88.31	79.23	98.41	87.69	30.61	30.14	91.45
080-33	96.34	88.38	80.84	96.34	87.89	33.65	32.41	91.06
080-34	67.27	57.80	54.24	67.27	60.03	40.10	27.27	61.64
080-35	83.72	91.04	85.19	83.72	82.98	32.17	27.03	88.92
080-36	74.15	61.18	48.82	74.15	56.45	29.32	22.43	65.37
080-37	96.15	87.00	80.34	96.15	87.29	34.62	33.28	90.19
080-38	97.02	86.82	78.73	97.02	86.84	32.93	31.97	90.23
080-39	82.38	74.62	67.22	82.38	74.03	36.48	29.56	77.34
080-4	98.35	88.66	86.18	98.35	91.83	41.59	40.89	92.69
080-40	84.48	75.17	71.38	84.48	77.19	41.50	34.94	78.96
080-41	99.80	83.02	71.21	99.80	83.02	29.39	29.33	87.93
080-42	99.76	83.53	83.47	99.76	90.89	45.46	45.35	90.91
080-43	87.36	97.03	94.24	87.36	90.55	35.79	31.18	93.47
080-44	89.91	91.96	88.44	89.91	88.52	38.05	34.26	91.25
080-45	97.00	91.17	84.76	97.00	90.15	31.59	30.66	93.07
080-46	66.41	62.47	52.01	66.41	58.31	28.87	18.51	63.49
080-47	74.64	69.76	62.33	74.64	67.93	35.00	25.95	71.45
080-48	71.12	59.26	46.05	71.12	54.67	26.67	19.00	62.07
080-5	73.87	68.14	63.92	73.87	68.43	41.16	30.81	70.54
080-6	73.64	70.00	64.19	73.64	68.58	38.23	27.48	71.28
080-7	96.68	90.37	86.33	96.68	91.11	38.13	36.87	92.79
080-8	100.00	56.74	37.17	100.00	54.19	20.37	20.37	65.56
080-9	75.61	79.49	71.66	75.61	72.95	32.80	24.76	78.35
By Timepoint								
Day 0	88.38	77.82	69.14	88.38	76.42	32.45	28.72	80.97
Day 5	85.74	78.97	70.39	85.74	76.46	33.96	29.43	81.30
By Experimental Condition								
GAG-1-PTEG	87.26	81.68	71.99	87.26	78.24	33.63	29.94	83.77
negctrl	90.30	80.49	71.98	90.30	79.30	33.38	30.24	83.59
POL-1-PTEG	78.95	69.63	59.87	78.95	67.06	32.33	25.49	72.42
POL-2-PTEG	89.56	79.60	73.00	89.56	78.61	32.91	29.38	82.45

Table 12: Scheme 1, HVTN080 Test set classification results within the the baseline version RTL framework, tuned for FC data analysis.

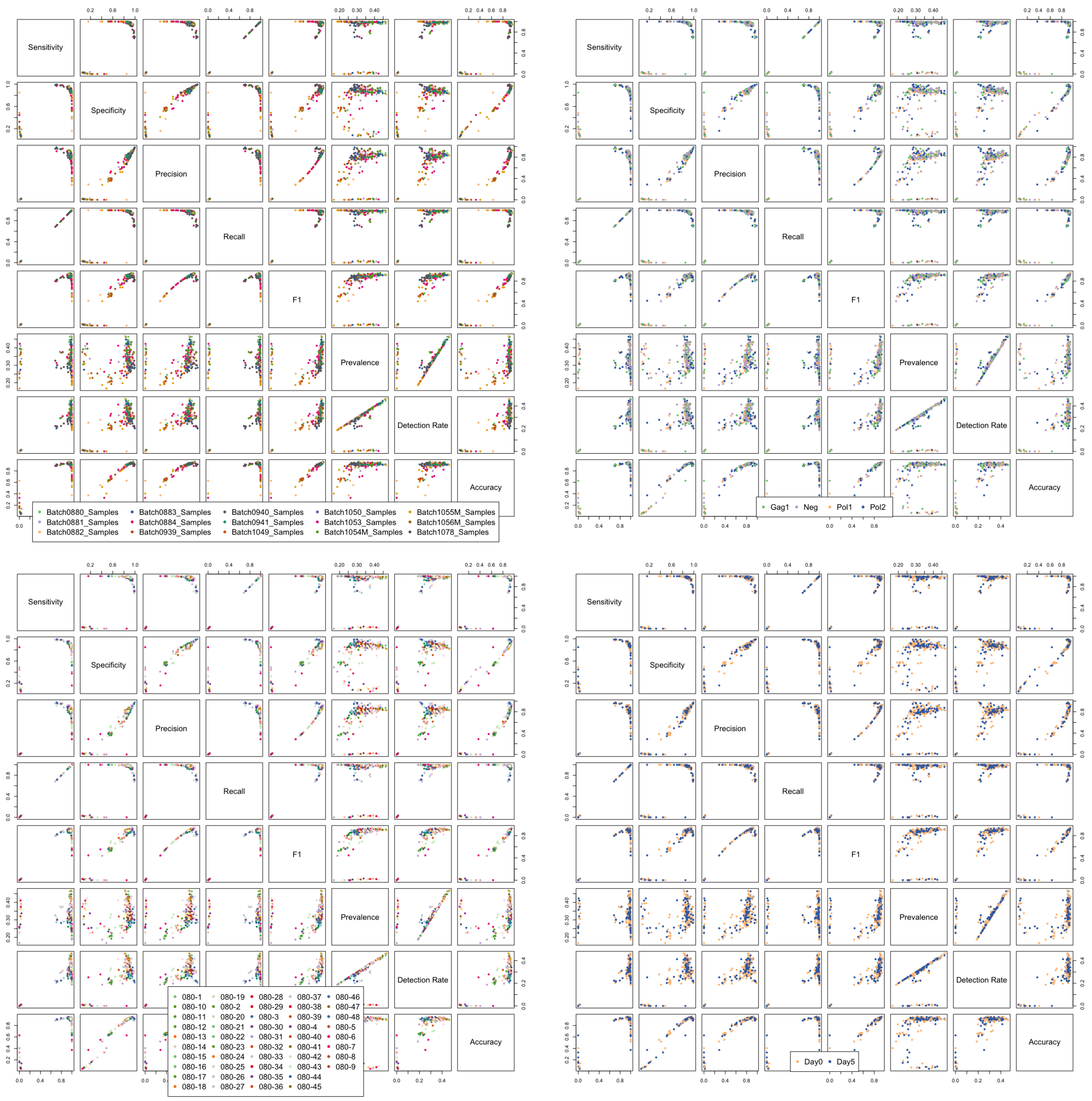


Figure 31: Aggregation of the HVTN080 classification statistics with the RTL framework to demonstrate their intrinsic relationships in the context of batch, condition, individual/subject, and timepoint.

3.6.2 Scheme 1, the extended version

Although the original intent of the extended version of the RTL framework was classification in transcriptomics data, where generally significantly lower number of examples (cells) per sample exist with many more quantified features, it is possible to evaluate its performance compared to the baseline version; in the transcriptomics section we show that the baseline version demonstrates a minor improvement in generalizability across datasets which potentially may also apply for FC data. Figure 32 and Table 13 summarize the findings. Compared to the results of scheme 1 with the baseline version of the RTL framework, we observe significantly lower specificity, precision and accuracy. A potential reason for this is potentially the different approach taken by Algorithm 6; In Figure 32, the final summary bar plot on the bottom right shows that on average, Algorithm 6's output reduced the accuracy of the call compared Algorithm 4 (i.e., without rotation). That said, scheme 1 is limited by its 1 training set (20 random fcs samples) to 1 test set (20 random fcs samples), which is why scheme 2 is done, as it is a better assessment of overall performance.

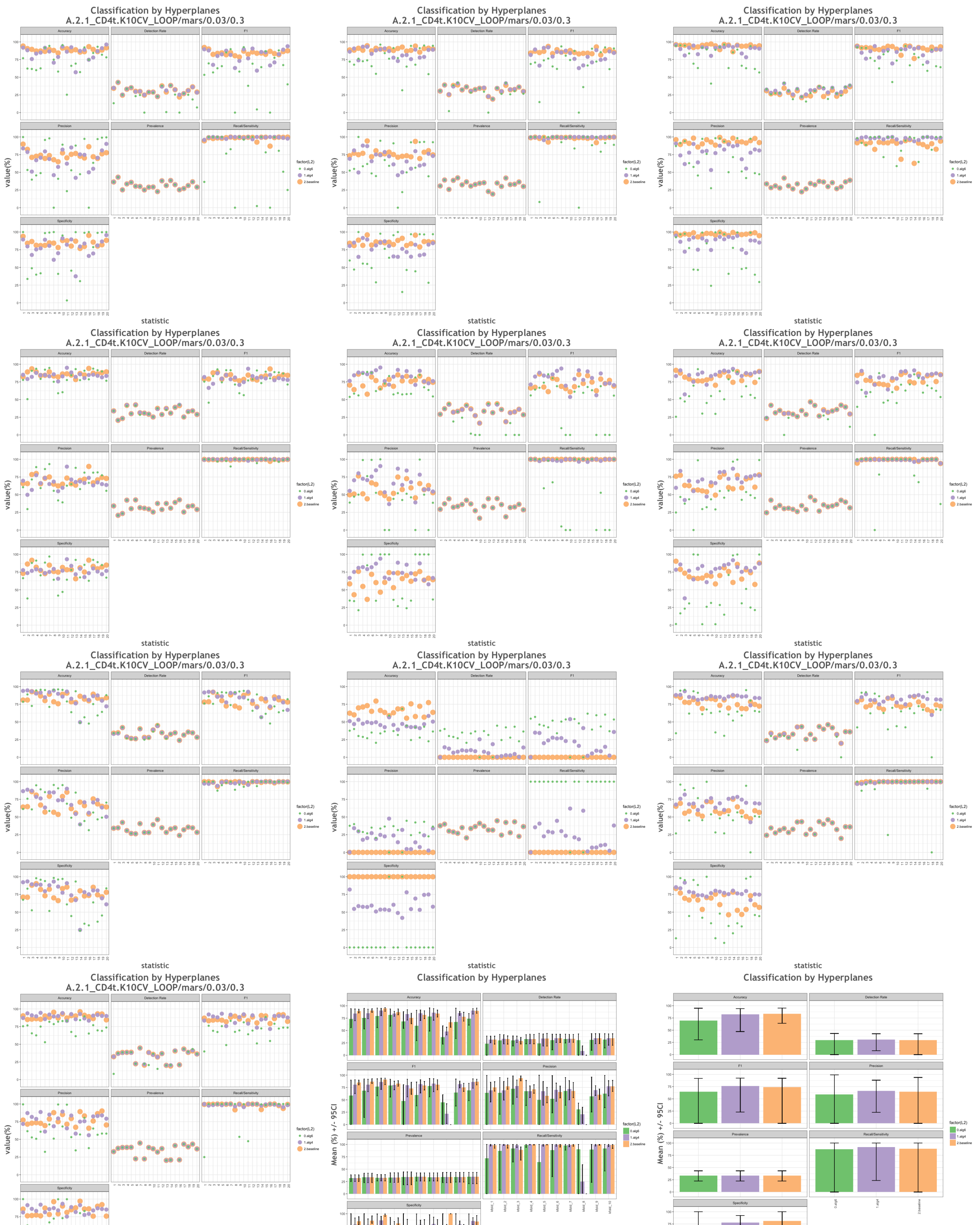


Figure 32: Scheme 1, HVTN080 Test set classification results within the extended version RTL framework, tuned for FC data analysis. From top left going across, the results from test sets 1-10 are shown followed by a per set aggregate and a cumulative bar graphs with 95%CI. For each, we demonstrate Accuracy, Detection rate, $F1_{score}$, Precision, Prevelance, Recall/Sensitivity, and Specificity. The Baseline (tan) result is the direct transfer of the robust means hyperplane from Algorithm 2. The Alg4 (plum) result is from the baseline hyperplane with the updated bias after Algorithm 4. Finally, Alg6 (green) result is after rotating the hyperplane in Algorithm 6.

	Sensitivity	Specificity	Precision	Recall	F1	Prevalence	Detection Rate	Accuracy
Average	87.48	57.02	58.17	87.48	63.13	32.79	29.00	67.52
By Batch								
Batch0880_Samples	92.55	50.24	59.62	92.55	67.42	36.58	34.07	66.38
Batch0881_Samples	81.14	68.20	73.96	81.14	69.51	37.79	30.62	73.59
Batch0882_Samples	97.18	56.17	60.51	97.18	71.47	33.09	32.19	69.93
Batch0883_Samples	93.49	57.40	62.05	93.49	68.61	31.96	30.23	70.22
Batch0884_Samples	87.21	46.52	51.46	87.21	56.70	33.73	29.60	60.43
Batch0939_Samples	71.40	56.03	36.03	71.40	47.03	34.84	26.02	62.85
Batch0940_Samples	88.41	58.05	56.02	88.41	61.81	29.59	26.30	67.17
Batch0941_Samples	99.65	47.07	53.71	99.65	68.61	35.72	35.57	65.50
Batch1049_Samples	71.93	61.05	53.23	71.93	48.18	25.64	17.99	62.26
Batch1050_Samples	82.05	65.52	65.13	82.05	63.88	32.55	26.97	71.59
Batch1053_Samples	91.50	58.88	65.30	91.50	71.15	34.47	31.34	70.34
Batch1054M_Samples	92.92	56.47	62.62	92.92	70.75	37.68	35.53	71.57
Batch1055M_Samples	76.76	74.15	65.98	76.76	59.29	27.70	22.72	76.70
Batch1056M_Samples	85.98	69.15	71.41	85.98	70.26	32.90	28.11	74.78
Batch1078_Samples	100.00	30.32	35.51	100.00	52.32	27.66	27.66	49.52
By Subject								
080-1	99.98	35.54	56.08	99.98	71.61	44.52	44.52	64.15
080-10	62.30	72.19	74.42	62.30	52.64	38.88	22.70	65.70
080-11	79.55	80.31	79.99	79.55	74.85	36.12	29.01	80.69
080-12	99.91	50.44	49.84	99.91	63.65	24.41	24.39	62.91
080-13	81.01	46.91	59.45	81.01	54.36	37.52	30.51	59.88
080-14	68.48	91.71	84.60	68.48	66.69	31.27	21.35	84.38
080-15	96.95	50.72	58.53	96.95	69.75	33.48	32.58	66.43
080-16	100.00	30.32	35.51	100.00	52.32	27.66	27.66	49.52
080-17	85.53	77.16	75.79	85.53	75.17	31.72	27.08	79.73
080-18	97.59	66.00	67.00	97.59	77.53	34.70	33.90	77.28
080-19	100.00	24.40	33.75	100.00	50.18	26.51	26.51	44.76
080-2	87.26	60.78	58.89	87.26	61.09	25.69	21.71	66.17
080-20	99.22	44.75	61.49	99.22	74.70	43.33	43.00	68.54
080-21	100.00	23.85	50.35	100.00	66.80	42.88	42.88	56.64
080-22	83.36	73.11	70.07	83.36	64.57	26.91	22.58	75.64
080-23	82.24	60.46	57.01	82.24	57.17	27.50	22.65	66.60
080-24	97.24	63.89	59.31	97.24	71.09	28.92	28.11	73.53
080-25	65.15	99.46	97.79	65.15	78.00	26.56	17.30	90.34
080-26	50.00	73.44	19.56	50.00	28.12	25.26	12.72	67.66
080-27	82.78	53.44	48.74	82.78	50.46	23.99	20.03	60.26
080-28	41.98	99.82	99.23	41.98	56.91	32.40	13.57	81.05
080-29	100.00	51.25	55.45	100.00	71.30	37.51	37.51	69.61
080-3	78.33	59.19	69.27	78.33	63.24	36.37	28.50	66.19
080-30	39.99	75.85	47.03	39.99	24.77	22.60	11.56	71.80
080-31	63.91	99.14	97.90	63.91	76.47	36.11	23.02	86.36
080-32	88.13	94.40	90.43	88.13	88.42	30.89	27.22	92.52
080-33	85.21	47.10	48.68	85.21	54.63	33.35	28.53	59.93
080-34	99.19	58.19	63.94	99.19	76.57	40.10	39.75	74.23
080-35	94.79	56.46	58.08	94.79	68.92	33.08	31.38	69.20
080-36	60.46	50.66	38.25	60.46	31.52	29.32	18.14	54.12
080-37	95.93	58.93	59.96	95.93	71.54	34.62	33.18	71.65
080-38	100.00	19.89	39.69	100.00	56.56	33.66	33.66	46.87
080-39	99.93	42.32	53.14	99.93	68.26	36.48	36.45	63.34
080-4	100.00	46.93	58.22	100.00	73.32	41.82	41.82	68.98
080-40	100.00	25.03	49.66	100.00	65.97	41.54	41.54	56.05
080-41	69.23	88.79	84.31	69.23	67.01	29.39	20.38	83.04
080-42	100.00	50.61	63.83	100.00	77.68	45.46	45.46	73.18
080-43	97.25	79.30	74.25	97.25	83.27	35.79	34.76	85.45
080-44	100.00	33.68	49.06	100.00	65.46	37.84	37.84	58.72
080-45	97.44	80.54	74.51	97.44	82.78	31.59	30.76	85.73
080-46	97.86	81.79	74.23	97.86	83.20	29.50	28.90	86.76
080-47	99.62	59.01	61.47	99.62	74.64	34.93	34.80	73.41
080-48	78.51	68.68	57.52	78.51	54.05	26.13	19.60	69.05
080-5	99.73	47.08	59.19	99.73	73.66	41.38	41.27	69.00
080-6	65.06	84.08	81.04	65.06	59.96	38.23	24.32	76.00
080-7	74.95	56.03	41.30	74.95	52.83	38.13	29.02	64.62
080-8	100.00	27.22	27.41	100.00	42.59	20.37	20.37	41.97
080-9	93.86	56.33	63.96	93.86	71.33	33.02	30.91	68.36
By Timepoint								
Day 0	84.92	59.80	61.30	84.92	63.07	32.50	28.18	69.04
Day 5	89.08	58.20	60.17	89.08	65.89	33.93	30.35	68.99
By Experimental Condition								
GAG-1-PTEG	87.54	56.70	57.14	87.54	63.02	33.98	29.83	67.04
negctrl	85.75	59.23	61.85	85.75	64.25	33.32	29.12	69.09
POL-1-PTEG	86.01	62.58	63.12	86.01	64.66	32.30	28.02	70.62
POL-2-PTEG	88.90	57.39	59.72	88.90	65.40	32.96	29.80	69.02

Table 13: Scheme1, HVTN080 Test set classification results within the extended version RTL framework, tuned for FC data analysis.

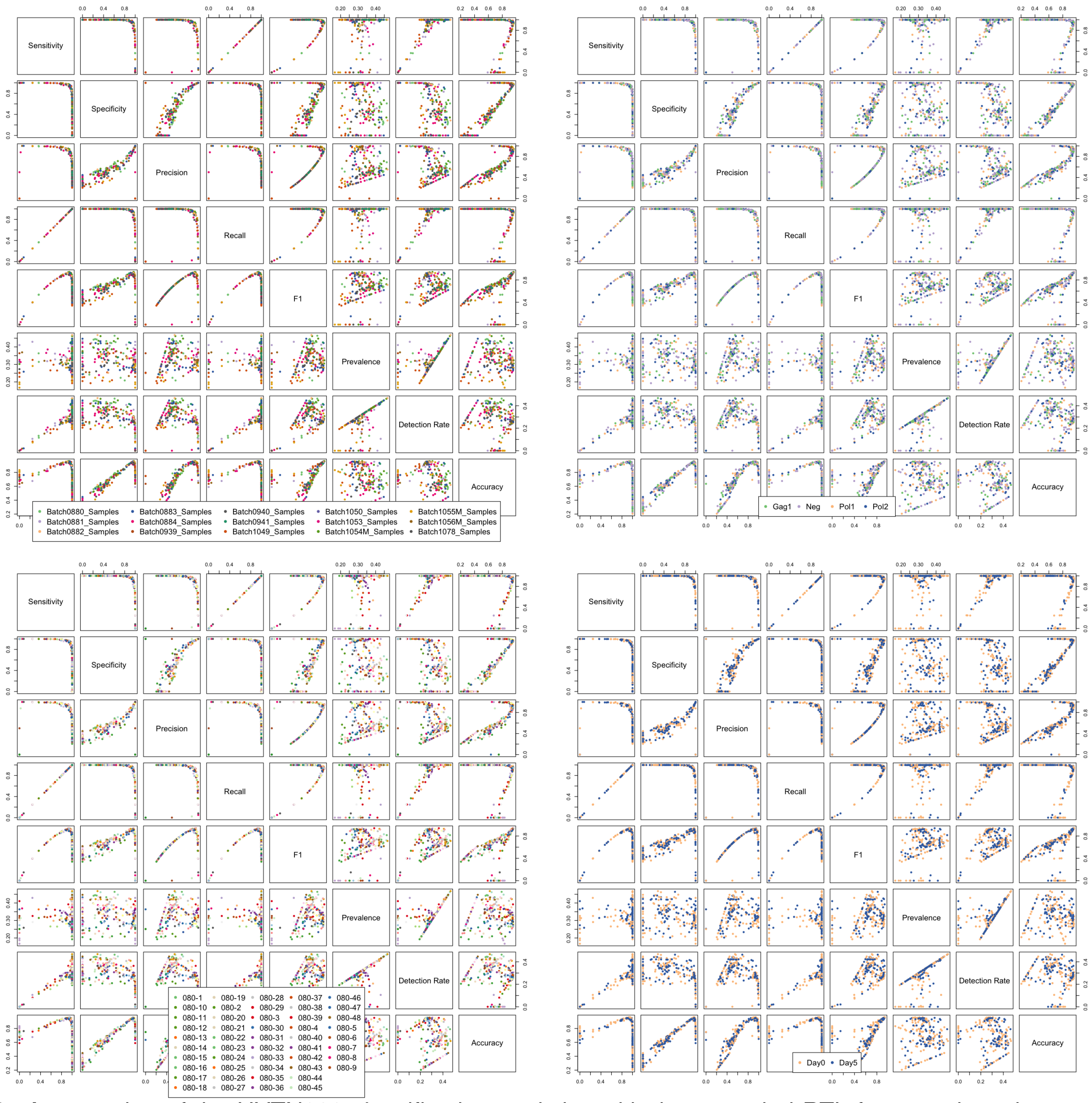


Figure 33: Aggregation of the HVTN080 classification statistics with the extended RTL framework to demonstrate their intrinsic relationships in the context of batch, condition, individual/subject, and timepoint.

3.6.3 Scheme 2, the baseline version

As described earlier, in scheme 2, a random 50 fcs samples are the test set, and the training set is created similar to scheme 1 with 10 sets of 20 random fcs files each. This design is done to evaluate the overall performance of the baseline and extended versions of RTL framework in the context of FC data without the potential limitations of sample-bias in scheme 1. Figure 34 and Table 14 summarize the findings. As expected, an ensemble classification outperform the simple 1 training set to 1 test set context; we observe a significant boost to classification statistics compared to scheme 1 because of the additional evidence each training set provides.

	Sensitivity	Specificity	Precision	Recall	F1	Prevalence	Detection Rate	Accuracy
Average	96.93	84.62	77.34	96.93	84.74	32.77	31.74	88.29
By Experimental Condition								
GAG-1-PTEG	94.08	89.10	82.49	94.08	86.87	34.94	32.80	90.57
negctrl	97.35	85.45	78.07	97.35	85.65	32.78	31.93	89.13
POL-1-PTEG	98.42	78.04	69.43	98.42	79.21	30.45	29.95	83.55
POL-2-PTEG	96.06	83.47	78.66	96.06	85.21	34.72	33.39	87.70

Table 14: HVTN080 Test set classification results with training-test set scheme 2, within the baseline version RTL framework, tuned for FC data analysis.

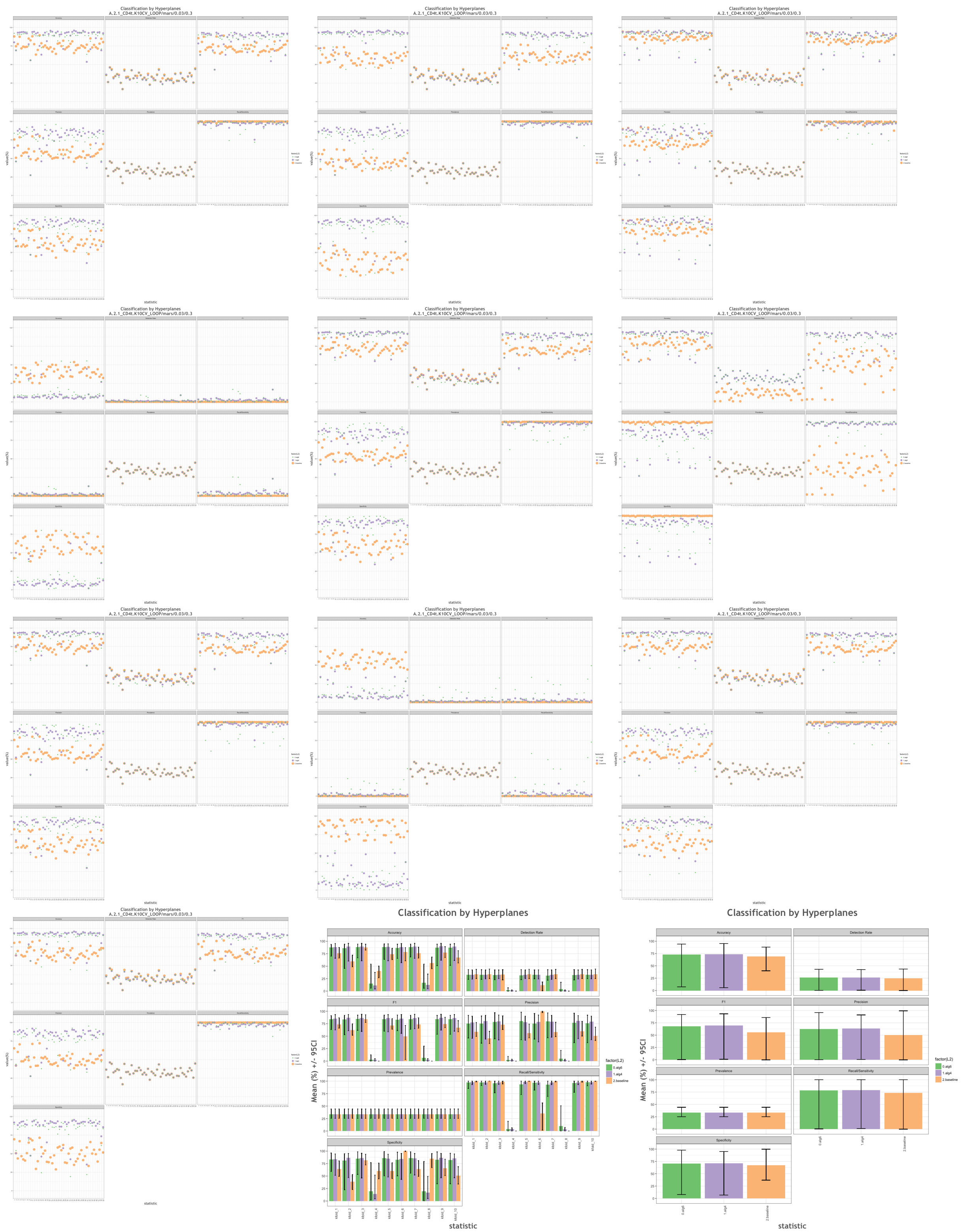


Figure 34: HVTN080 Test set classification results with training-test set scheme 2, within the the baseline version RTL framework, tuned for FC data analysis. From top left going across, the results from test sets 1-10 are shown followed by a per set aggregate and a cumulative bar graphs with 95%CI. For each, we demonstrate Accuracy, Detection rate, $F1_{score}$, Precision, Prevelance, Recall/Sensitivity, and Specificity. The Baseline (tan) result is the direct transfer of the robust means hyperplane from Algorithm 2. The Alg4 (plum) result is from the baseline hyperplane with the updated bias after Algorithm 4. Finally, Alg6 (green) result is after rotating the hyperplane in Algorithm 6.

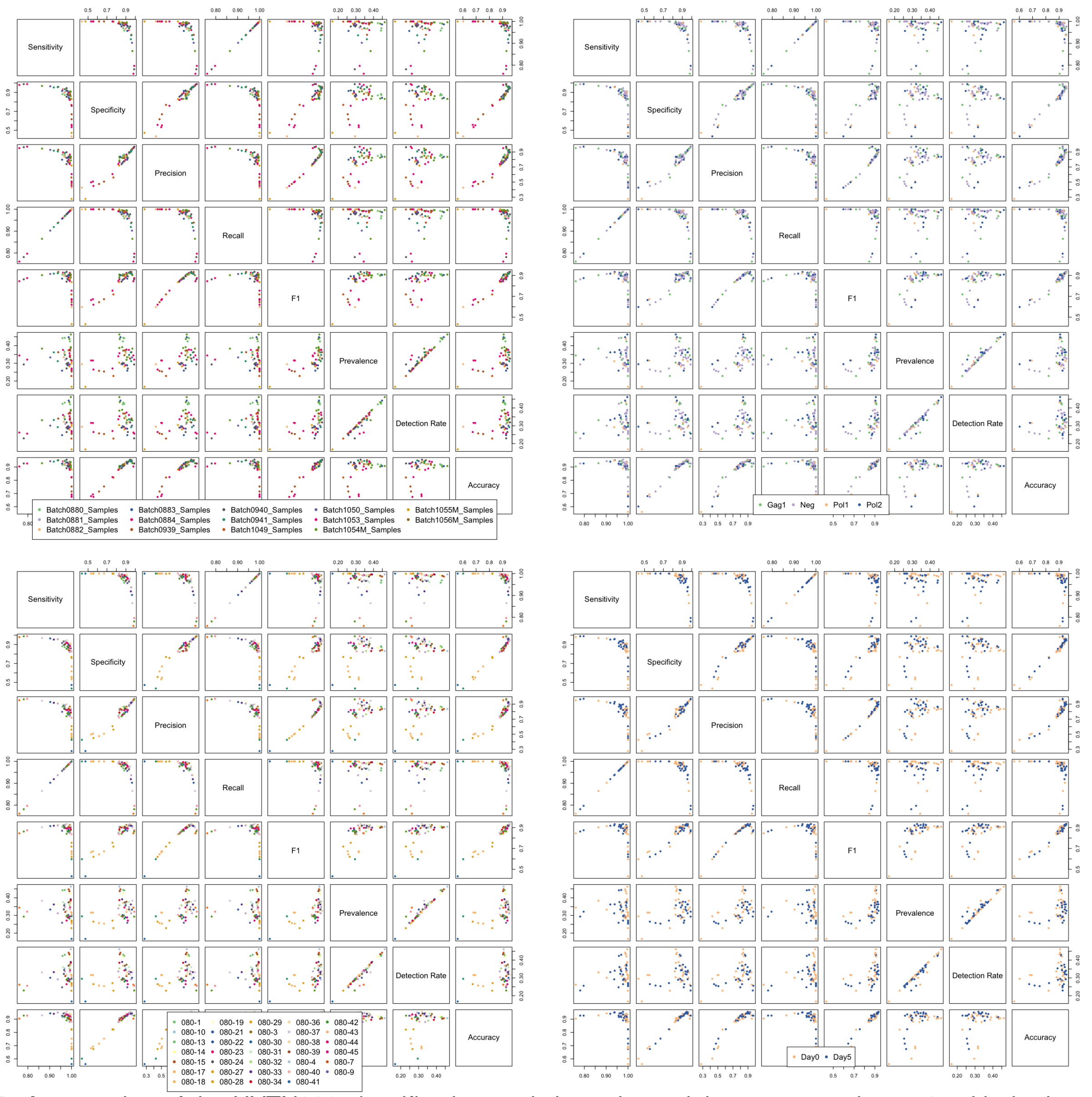


Figure 35: Aggregation of the HVTN080 classification statistics using training-test set scheme 2, with the baseline RTL framework to demonstrate their intrinsic relationships in the context of batch, condition, individual/subject, and timepoint.

3.6.4 Scheme 2, the extended version

In this train/test set design scheme, using the extended version we found similar classification to the baseline version of the RTL framework classifying FC samples, but with higher variance, especially with regards to sensitivity and recall. As before, the etiology of this higher variance could be Algorithm 6, as we observe on average lower accuracy compared to the un-rotated Algorithm 4 in the bottom right summary bar plot of Figure 36.

	Sensitivity	Specificity	Precision	Recall	F1	Prevalence	Detection Rate	Accuracy
Average	93.45	60.61	64.49	93.45	71.40	34.97	33.88	73.96
By Experimental Condition								
GAG-1-PTEG	95.47	58.86	62.70	95.47	72.12	32.68	31.96	72.57
negctrl	93.83	57.62	62.22	93.83	69.69	34.87	33.81	71.96
POL-1-PTEG	92.29	65.90	67.23	92.29	72.62	35.83	34.41	76.63
POL-2-PTEG	99.16	66.57	69.20	99.16	80.60	38.46	38.19	79.84

Table 15: HVTN080 Test set classification results with training-test set scheme 2, within the extended version RTL framework, tuned for FC data analysis.

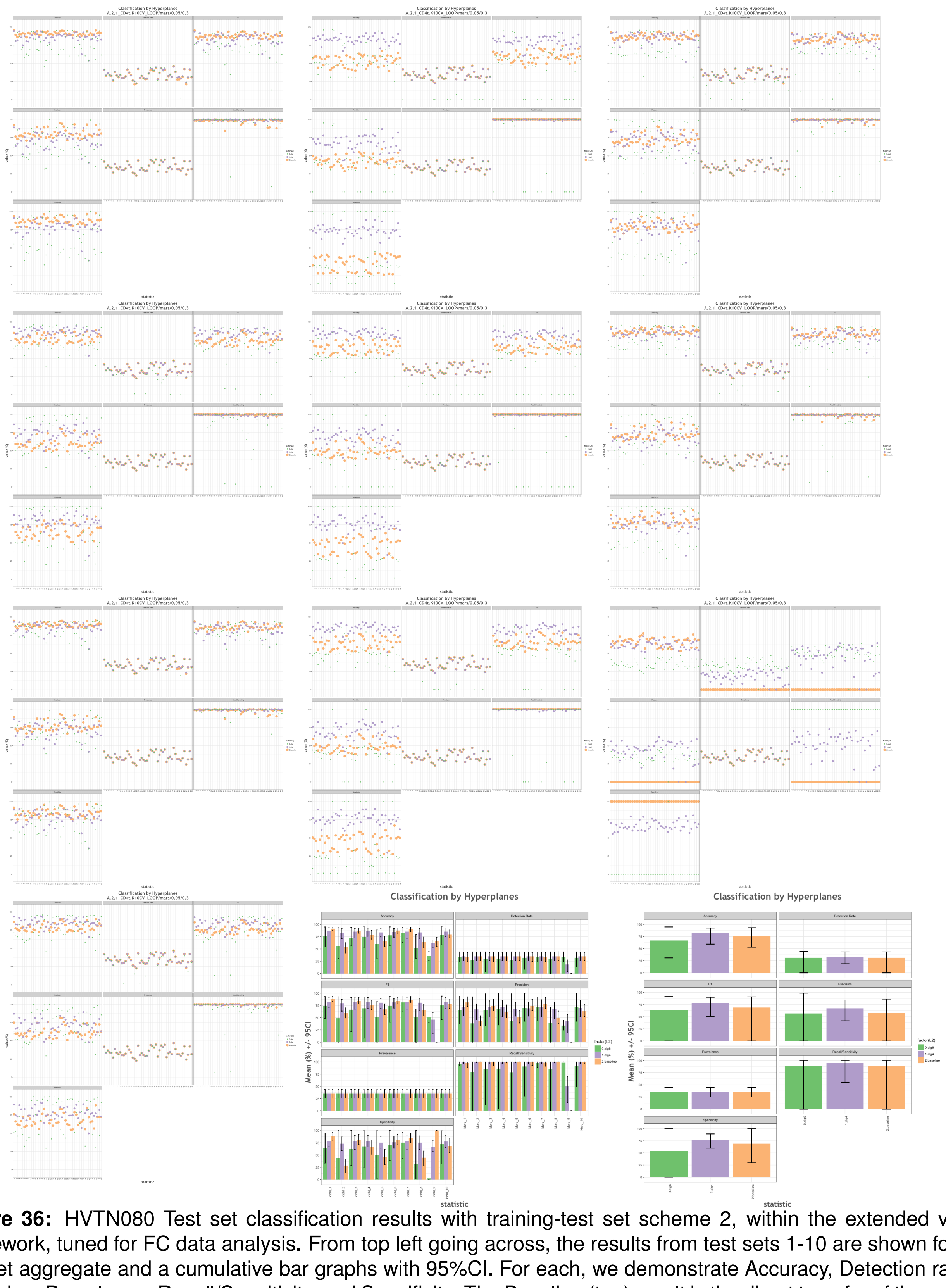


Figure 36: HVTN080 Test set classification results with training-test set scheme 2, within the extended version RTL framework, tuned for FC data analysis. From top left going across, the results from test sets 1-10 are shown followed by a per set aggregate and a cumulative bar graphs with 95%CI. For each, we demonstrate Accuracy, Detection rate, $F1_{score}$, Precision, Prevelance, Recall/Sensitivity, and Specificity. The Baseline (tan) result is the direct transfer of the robust means hyperplane from Algorithm 2. The Alg4 (plum) result is from the baseline hyperplane with the updated bias after Algorithm 4. Finally, Alg6 (green) result is after rotating the hyperplane in Algorithm 6.

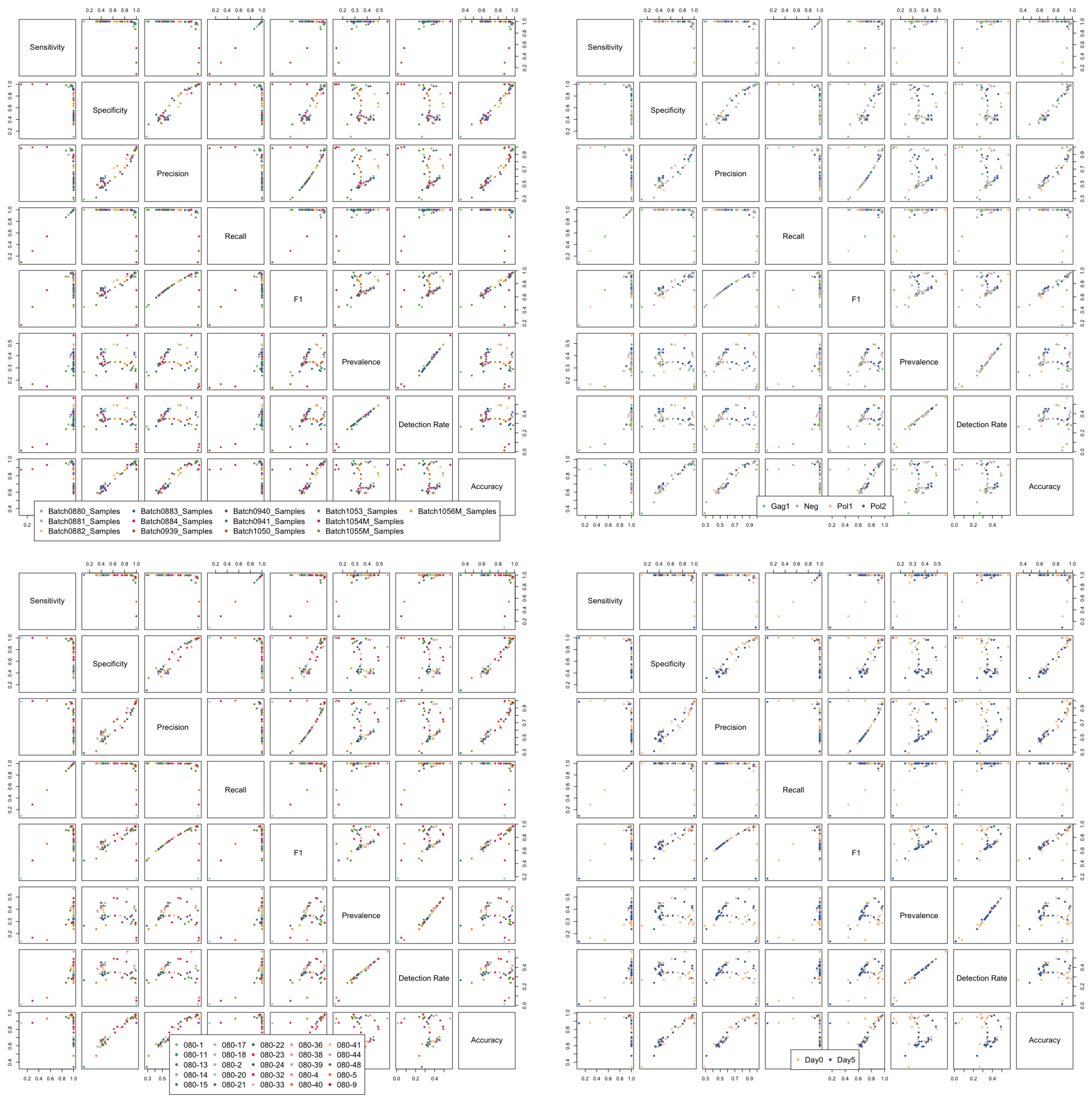


Figure 37: Aggregation of the HVTN080 classification statistics using training-test set scheme 2, with the extended RTL framework to demonstrate their intrinsic relationships in the context of batch, condition, individual/subject, and timepoint.

3.6.5 Discussion

To evaluate classification of a clinically valuable immunophenotype compared to manual gating, we utilized the publicly available HVTN080 dataset. We designed two train/test set schemes to evaluate the performance of the RTL framework within this FC context. Initially, in scheme 1, the entire HVTN080 (435 FCS files) was randomly sampled to create 10 sets of 20 random FCS files for training and testing samples (400 FCS used per epoch for symmetry and reduction of sample-bias). Summarized in Figure 29, in our 1 to 1 approach (scheme 1), where each training and test set pair was used for learning and inference, we found the baseline version of the RTL framework outperforms the extended version. However, in scheme 2, an ensemble approach was used; the test set was reduced to 50 random samples (instead of the 200 compared to scheme 1, to reduce the computation time) but the training set size was kept similar to scheme 1. Thus 10 sets of 20 FCS each, separately were used to learn and make inference on the same 50 sample test set within the RTL framework. A final call was made by averaging the calls (-1 or +1) where the average of 0 means no call ('NA'). As expected, the ensemble approach was able to achieve much higher classification statistics (scheme 1 vs scheme 2). Comparing the two version of the RTL framework, we found that in the context of FC the baseline version outperforms the extended. We believe the potential etiology of this to be the extended version's Algorithm 6, where on average the rotation found reduces the classification accuracy compared to the un-rotated hyperplane of Algorithm 4 as in Figures 32 and 36. However, this could also be rooted in the potential over-smoothing of the density curves in Algorithm 4 to find the optimal minima as the classification hyperplane's bias update. Regardless, from our scheme 2 results, the consequence is higher classification variance especially with regards to sensitivity and recall.

4 Classification of cellular subsets/signatures in scRNASeq

In this section we focus on classifying cells quantified transcriptionally by next-generation sequencing technologies (scRNASeq).

4.1 Acquisition and pre-processing of scRNASeq data

To evaluate the RTL framework, we utilized two publicly available single-cell RNA-seq datasets where the cell-cycle (G1, S, and G2M) states of each cell is known *a priori*; therefore, the classification goal of interest to be achieved within the RTL framework is to infer the cell cycle states transcriptionally and compare it to the known labels. These datasets have been previously used for similar benchmarking applications with 6 classifiers assessed by Scialdone et al. [89]. Specifically, we benchmark the RTL framework on the 182 cell EMTAB2805 data, filtered by their published 40 gene set, used by their top performing classifiers. Next, we evaluate classification generalizability with the RTL framework by training on EMTAB2805 and predicting on a similar but separate dataset (GSE42268). We can thus use the $F_1 - Score$ and the $macro - averaged\ F_1 - Score$ [14] to compare how well the RTL framework performed across all cell-cycle phases. Next, we test the effect of several normalization methods on the classification performance within the RTL framework. Finally, by performing re-sampling, new training and test samples are created where the marginal frequency of each cell cycle label from EMTAB2805 adjusted to represent a desired rare cell subsets (RCS) down to the defined frequency of 1% of the total. These experiments were run by conducting K-fold cross-validation, where each round a random set of cells (without replacement) are withheld for testing; a practical approach when data are scarce [50].

4.1.1 Data sets

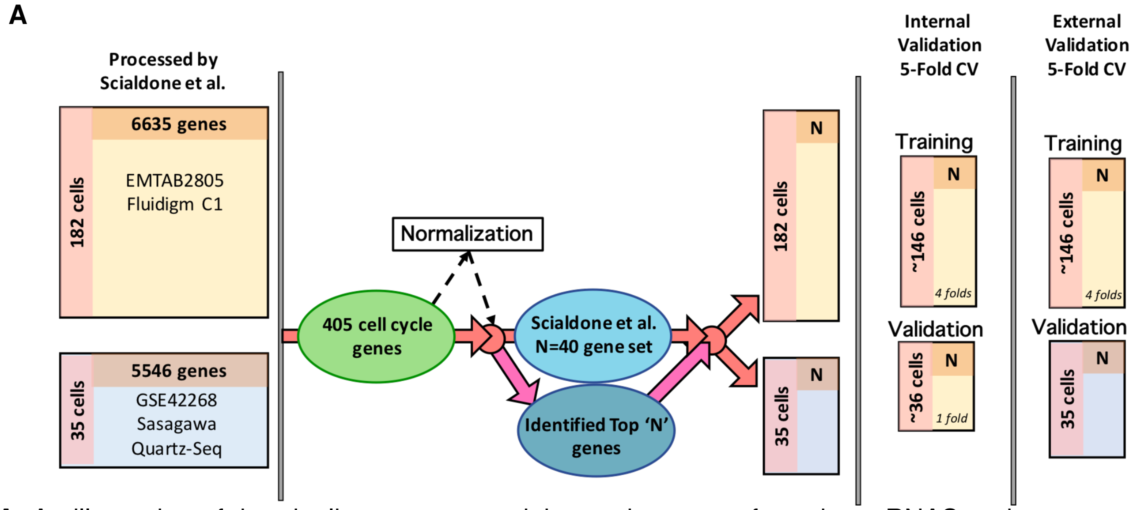


Figure 38: A. An illustration of the pipeline to create training and test sets from the scRNASeq datasets pre-processed by Scialdone et al. Once the datasets were reduced by the 405 cell cycle genes, the corresponding normalization (when specified) is done. A final reduction of the genes is done using either the Scialdone 40-gene set or the top 'N' genes identified as described in a previous section. For the internal validation, we conducted 5-fold cross-validation where the training is done on a combination of 4 folds and inference is done on the left-out fold, which is rotated such that each fold eventually has been used as the validation fold. For the external validation, we conducted 5-fold cross validation where the training set is iterated as a combination of 4 of the folds and the inference each time is on the entire 35 cells from the GSE42268 dataset.

[14]To make an independent assessment regardless of the number of cells in each cycle in the testing dataset, the average of precision and recall over all phases will be done before computing the harmonic mean as suggested in [89]

A) First, we utilize the gene-expression matrix of EMTAB2805 [90] as processed by Scialdone et al. [89]. This pre-processed EMTAB2805 dataset contains 182 staged moused embryonic stem cells (mESCs), described in [90]; the marginal distributions are summarized in Table 17. Briefly, cultured Rex1-GFP-expressing mESCs (Rex1-GFP mESCs) were stained with Hoechst[15]. FACS identification of three different cell-cycle states was done prior to sequencing. The sequencing was done using the Fluidigm C1 protocol[16]. The processed dataset available as a supplemental to [89] has already passed several quality control checks, removing low-quality cells. For example, cells with an aberrantly low total sum of counts of all genes and spike-ins [17], i.e, library size, are removed as they are suspect of issues with the capturing of mRNA to cDNA or the cDNA amplification process. Furthermore, as part of their processing, they normalized this dataset by two size factors, i.e., the technical size factors estimated using the ERCC spike-in genes as well as the endogenous size factors using the endogenous genes. We refer to [91] for a detailed description of this process and its assumptions and limitations. Finally, Scialdone et al. identified 6635 genes with variation above the technical background level (FDR < 0.1).

B) The second dataset used was the Sasagawa Quartz-Seq (GSE42268) [92], the transcriptional profiles of 35 mESC cells, summarized in Table 20. This dataset alone is not ideal for training machine learning algorithms as it is small and technical noise cannot be removed as it lacks the use of spike-in controls. However, normalization to remove technical noise is still possible by using the log-linear fit between the expression mean and the squared coefficient of variation between the cells. Scialdone et al. provide this normalized data, which identifies 5546 highly variable genes. Furthermore, compared to the EMTAB2805, there are two major differences: First, the culture condition, specifically the growth media of the mESC cells are different. Second, the sequencing protocols are different. Both of these factors can have consequential effects on the transcriptional profiles of the cells. This is apparent in our principal component analysis on the log-transformed gene counts (Figure 39) where the two datasets cluster separately. When we performed rank-based (Figure 40) or quantile normalization (41), we observe that cell cycle is explanatory of the main variance observed in the first two principal components. Using the log-transformed gene counts compared to the other normalization methods utilized presents an opportunity to evaluate the hypothesis that a TL classification framework overcomes the need for task-specific normalization prior to analysis. This hypothesis will be supported if the normalization of the data reduces or has no effect on the overall classification compared to the log-transformed counts, especially in the context of classifier generalizability.

4.1.2 Rank-based, Quantile, and log Median absolute value normalizations:

Broadly, normalization is a procedure to set a mutual reference between samples [93]. In other words, normalization is done to correct for systematic variance between two or more samples [94, 95]. In complex transcriptomics experiments, there are several scopes of unwanted variation [96]. For example, correcting for differences in sequencing depth as well as other technical sources of variation. Recently, standard procedures are developed for scRNASeq that aid this process such as using ERCC spike-in controls [44]. However, initially in the context of bulk transcriptomics with microarray and sequencing technologies, it was shown that normalization can itself affect reproducibility of the results [97, 98, 99]. Specific to the scRNASeq datasets used herein, Scialdone et al. also demonstrate that the selection of the normalization technique had a significant effect on their downstream classification [18]; highlighting the critical role of normalization for scRNASeq, especially in the context of classification generalizability. The rank-based normalization is done by Scialdone et al. is task-specific normalization in the context of the classification task; it is done because despite all efforts there still may be residual unwanted differences in the distribution of expression matrices between two samples.

As alternative approaches to the methods herein, as well as expanding on the potential landscape of pertinent solutions to conducting analysis across multiple datasets we highlight that Bacher et al., have recently introduced SCnorm [100] for "accurate and efficient normalization" of scRNASeq data. For a detailed recent review of challenges and opportunities in normalizing scRNASeq data we refer to [101]. Finally Wang et al. [102] provide a framework that improves generalizability of clustering (unsupervised methods) across scRNASeq dataset.

[15]Invitrogen, Hoechst 33342; optimized for the Rex1-GFP mESC

[16]C1 Single-Cell Auto Prep System (Fluidigm; 100-7000)

[17]number of non-zero counts

[18]Compared to rank-based normalization, when they trained their predictors on the total read count normalized transcriptomics data, the performance of the respective classifiers decreased

Specifically in this dissertation, for benchmarking purposes, similar to Scialdone et al's approach, we utilized the task-specific rank-based normalization. This will enable the direct comparison of our classifier with those in [89]. As a second normalization technique, we conducted quantile normalization, commonly utilized in transcriptomics analyses. Quantile normalization ensures that the distribution of the log-transformed counts to be the same in all the datasets used. Finally, as a non-ranked based normalization method, we conducted log Median absolute value (logMAV) normalization, such that the expression of the genes (columns) for each cell (rows) is normalized to the median of that cell's expression.

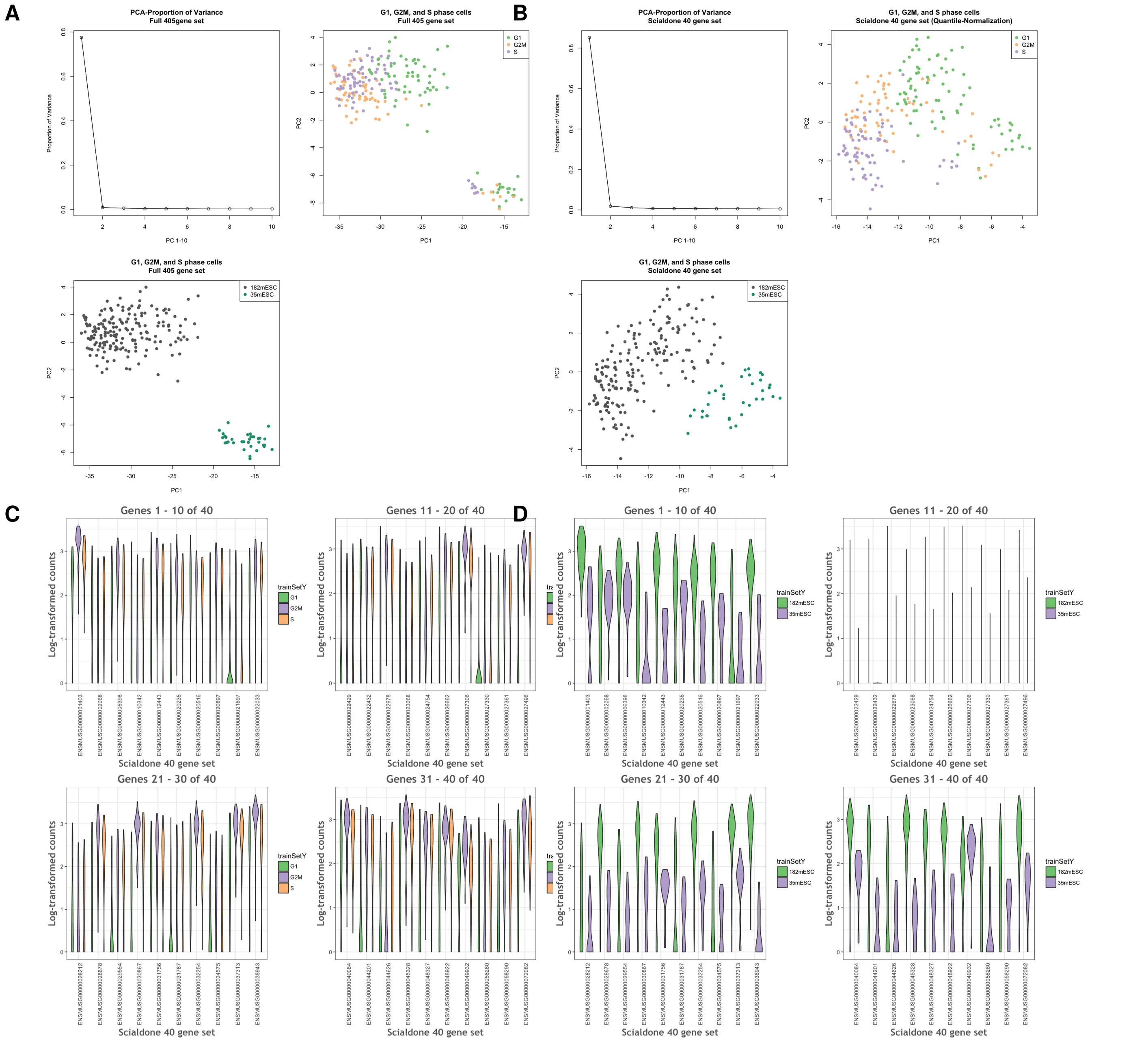


Figure 39: Log-transformed, combination of 182 cells from EMTAB2805 with 35 from GSE42268 processed by Scialdone et al. **A** Using the starting point 405 cell cycle genes, PCA is done on the combined data and the resulting projections on the first two principle components are shown relative to the cell cycle states and the origin of cells. **B** A second similar PCA, but on the Scialdone 40 gene set. Below them are violin plots of the Scialdone 40 gene set relative to cell cycle **C** and origin **D**.

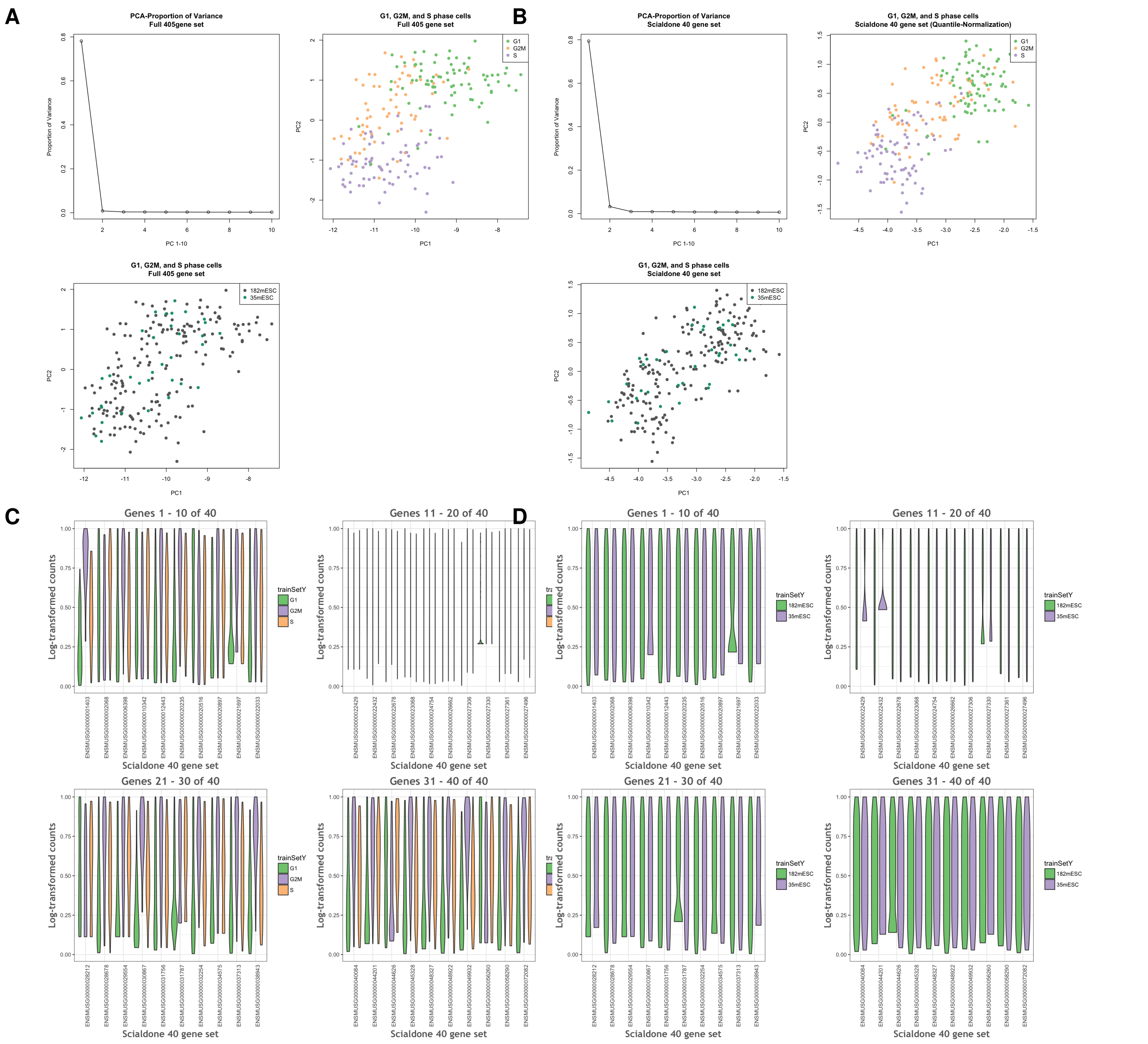


Figure 40: Rank-normalized, combination of 182 cells from EMTAB2805 with 35 from GSE42268 processed by Scialdone et al. **A** Using the starting point 405 cell cycle genes, PCA is done on the combined data and the resulting projections on the first two principle components are shown relative to the cell cycle states and the origin of cells. **B** A second similar PCA, but on the Scialdone 40 gene set. Below them are violin plots of the Scialdone 40 gene set relative to cell cycle **C** and origin **D**.

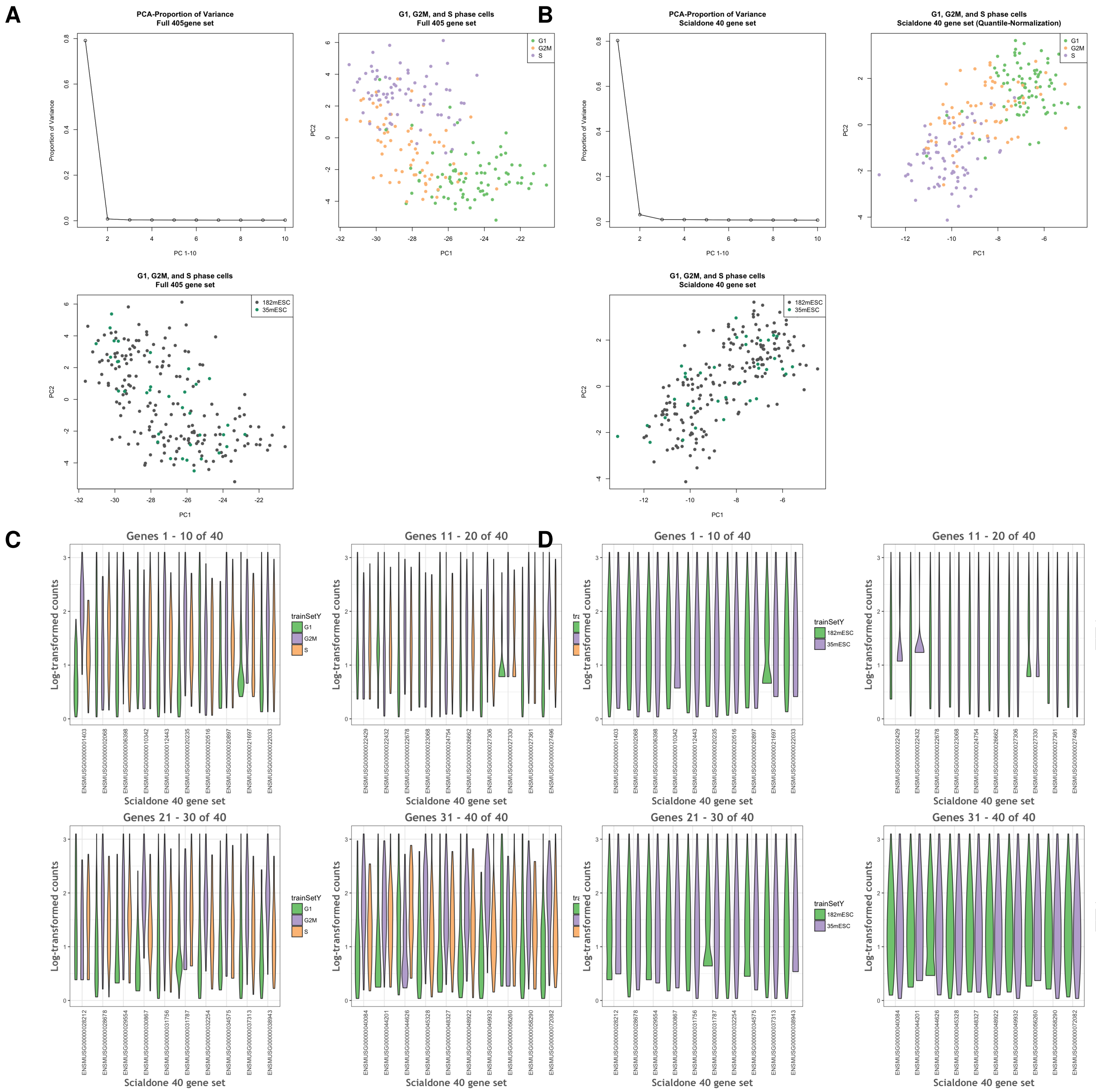


Figure 41: Quantile-normalized, combination of 182 cells from EMTAB2805 with 35 from GSE42268 processed by Scialdone et al. **A** Using the starting point 405 cell cycle genes, PCA is done on the combined data and the resulting projections on the first two principle components are shown relative to the cell cycle states and the origin of cells. **B** A second similar PCA, but on the Scialdone 40 gene set. Below them are violin plots of the Scialdone 40 gene set relative to cell cycle **C** and origin **D**.

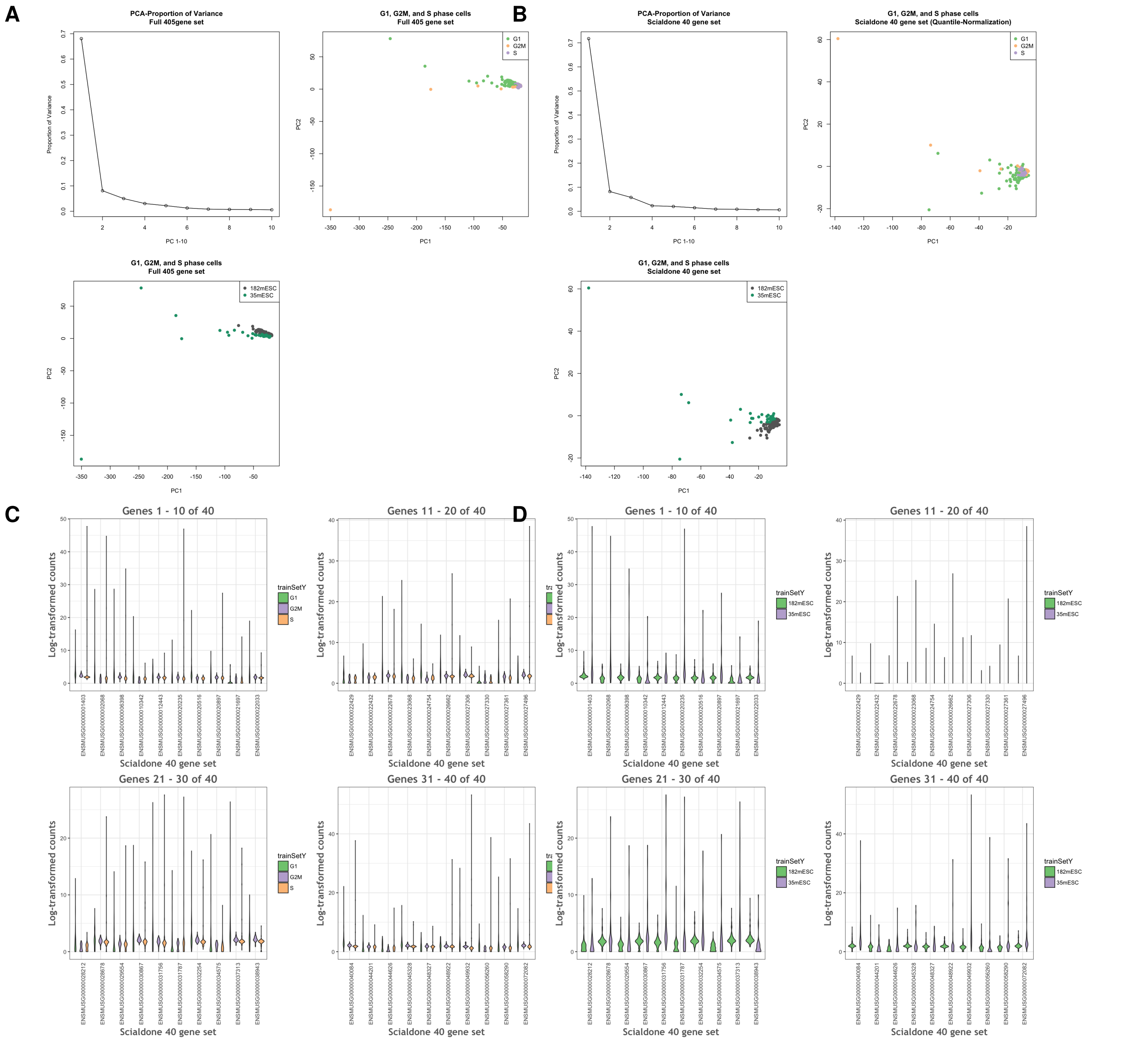


Figure 42: Median absolute value normalized, combination of 182 cells from EMTAB2805 with 35 from GSE42268 processed by Scialdone et al. **A** Using the starting point 405 cell cycle genes, PCA is done on the combined data and the resulting projections on the first two principle components are shown relative to the cell cycle states and the origin of cells. **B** A second similar PCA, but on the Scialdone 40 gene set. Below them are violin plots of the Scialdone 40 gene set relative to cell cycle **C** and origin **D**.

4.1.3 Feature (gene) importance/selection:

1:8	9:16	17:24	25:32	33:40
Plk1	aurka	cdc25c	tmem2	fzr1
nde1	ckap2l	kif23	aurkb	cdc20
tex14	h2afx	liph	cenpn	cdc25b
tacc3	bub1b	depdc1b	ccnf	kif11
cenpe	lrif1	prc1	kif2c	nusap1
smc1b	katnb1	rps6kb1	mad1l1	espl1
pbk	nus1	ube2c	cdca2	gabpb1
ccne1	ccne2	sephs1	papd7	dmc1

Table 16: Scialdone et al.'s identified set of top 40 cell cycle genes; ranked by the loadings on first principle component.

In processing the 182-cell mESCs dataset, Scialdone et al. have identified 6635 variant-genes above technical background (FDR<0.1). By intersecting this gene set with a curated cell cycle gene set from the Gene Ontology as well as the CycleBase databases, they found a smaller set of genes, the 'informative cell-cycle markers'. This set is further filtered by overlapping with it the informative cell-cycle markers found in the separate, but similar dataset (GSE42268) to a achieve a final set of 405 genes. We utilize their published cell-cycle 405-gene set as our starting point. Specifically, this gene set is used as the input for the normalization methods (Figure 38). To equally compare the RTL framework with the top performing classifiers of Scialdone et al.'s, we utilize a smaller 40-gene set, a subset of the 405 cell cycle genes (Table 16) which they describe in [89].

Although initially, we utilized the Scialdone et al.'s 40-gene set to benchmark and evaluate our implemented methods to the top predictors demonstrated by Scialdone et al., we asked, if a different and more specific set of genes can be identified per class label to boost the classification performance within the RTL framework. For our feature selection approach, we initially conducted ten repeated training/test set splits, for each conducting 10-fold cross-validation classification with a linear SVM classifier (equivalent to the RTL framework's Algorithm 1). As described in [103], the importance of each predictor was determined by computing the area under the curve (AUC) of receiver operating characteristic (ROC) curve. This approach was done with the $varIMP()$ function from the $Caret$ library [103] which enabled the identification of the top 50 predictors for each cell cycle label. This reduction from the 405 cell cycle genes significantly reduces the complexity of our downstream processes and it is supported by previous findings in [90, 89] which separately identify smaller optimal subsets of the 405 cell cycle genes for the task of computational classification of cell cycle labels. Next, to further refine our identified top 50 gene lists and obtain the final top 'N' gene sets, we define a threshold as the lowest mean classification sum squared-error (SSE) by iterating classification with linear SVM models (50 epochs); for each cell cycle label, each epoch classified the top 3 to 50 genes with random set of cells ($2/3^{rd}$ of the total, 20 per phase). Altogether, this procedure was repeated for each normalization scheme described herein. Briefly, we find that the normalization utilized confounds the sets of genes (Figure 45). In fact, compared to the Scialdone 40 gene set, the top 'N' gene sets can be dissimilar (Figure 44), but it is important to note a potential limitation of this approach is that using such specific gene set, derived from a single source, may reduce generalizability as the classification could be over-fitted to the training set. But as our results in the next section demonstrate, there is no reduction, in fact, there is an increase in the external validation classification statistics.

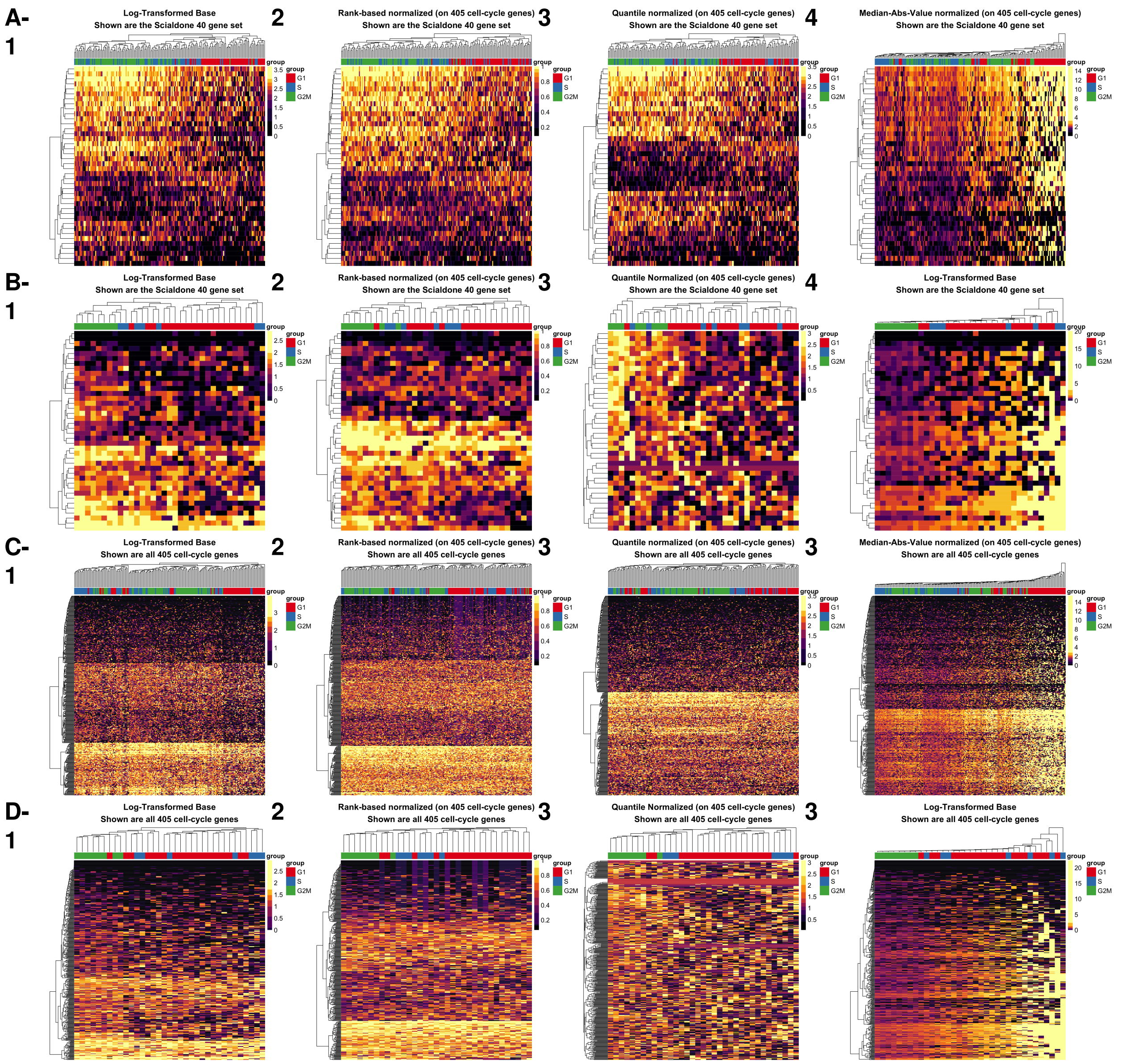


Figure 43: Unsupervised hierarchical pair-wise clustering with Euclidean distances and the complete linkage agglomeration method in the context of the known cell cycle states.**1-4** Log-transformed counts, rank-based normalized, quantile normalized, and log median absolute value (logMAV) normalization respectively such that the respective normalization is conducted on the 182 x 405 cell cycle gene set. **A. and C.** Heat map of the 182 mES cells from EMTAB2805 and **B. and D.** with 35 mES cells from GSE42268 such that A and B represent the 40 gene set and C and D represent the 405 cell cycle gene set.

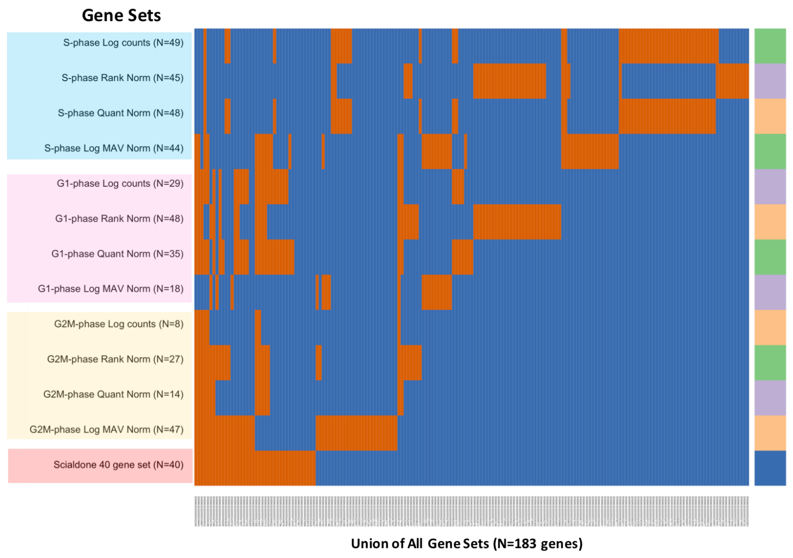


Figure 44: Comparing the Top 'N' genes identified by lowest classification sum-squared error. A boolean (0/1) heatmap to highlight gene association to specified set. Unsupervised clustering on columns demonstrates hierarchical Euclidean distance. For counts see Table 26.

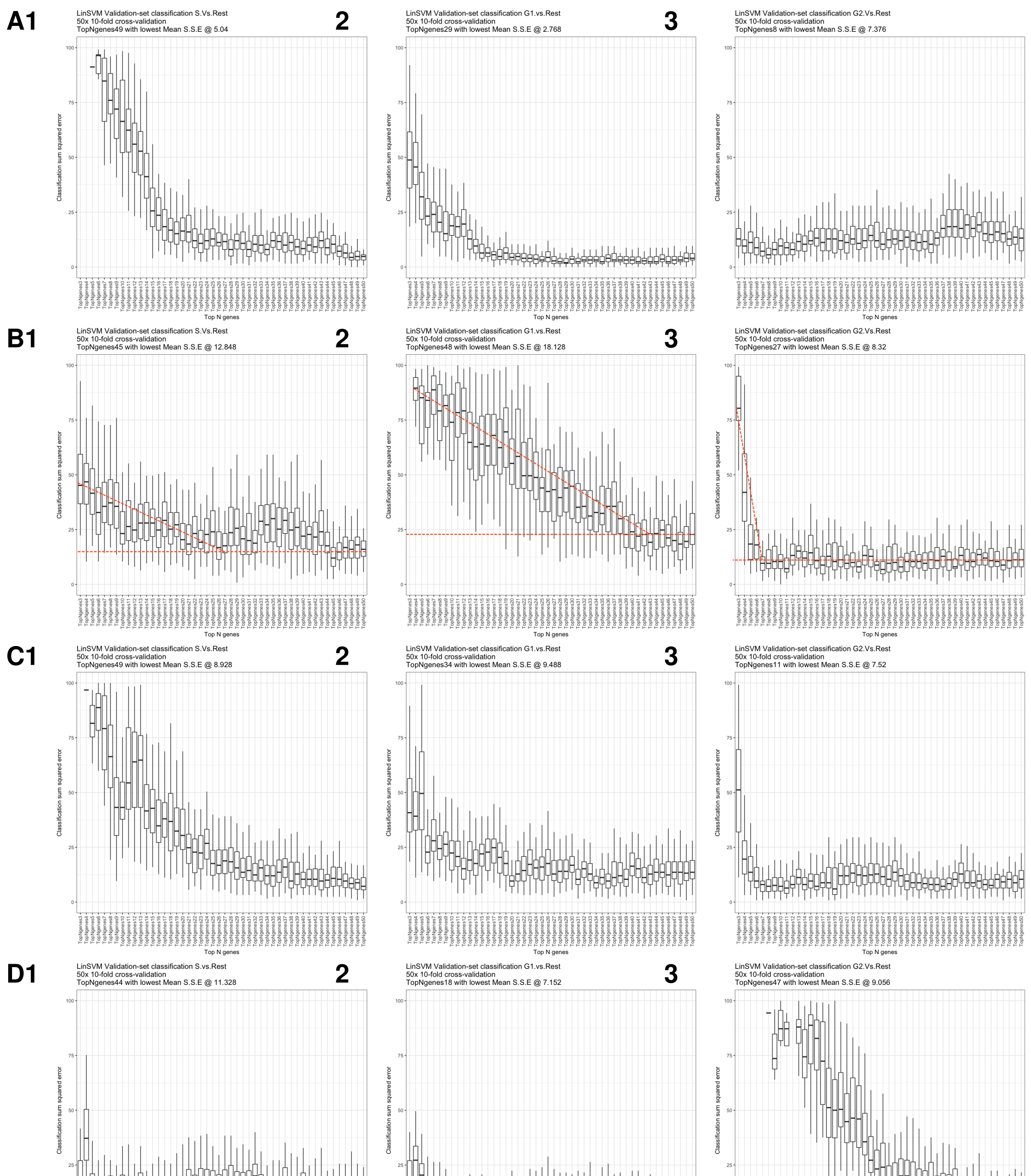


Figure 45: Identification of top 'N' genes for each class-label, determined by minimum mean classification summed squared-error (SSE) with Algorithm 1's classifier, conducting fifty (50), 10-fold cross-validated runs on the EMTAB2805 data. The ranking from 1-50 is first conducted as described previously by examining the ROC-AUC with a linear SVM. Then for each run, starting with top 3 genes as the minimum number of genes to the 50th ranked gene, SSE is computed. **A** Log-transformed baseline. **B** Rank-normalized. **C** Quantile-normalized.**D** Median absolute value (MAV) normalized. Alternatively, an 'elbow' can be identified instead of the minimum SSE to identify top N genes per class, as shown heuristically on the first row of graphs.

4.1.4 High-dimensionality and the need for multiple trained hyperplanes:

To be able to run high-dimensional data such as the 182 cell by 405 gene matrix in K-fold cross-validated runs of the RTL framework, a modified approach to the original algorithm is required. This is because in Algorithm 2, to compute the robust mean and covariance matrix of the trained hyperplanes from Algorithm 1, we require twice as many trained hyperplanes than the number of features under consideration. This requirement is directly associated with the current implementation that utilizes the $HuberPairwise$ function from the GSE package for this task. As a standard approach to K-fold cross-validation in the context of the RTL framework, in Algorithm 1, each training K-fold is used to obtain a single hyperplane (i.e., a total of 9 hyperplanes if K=10). Next, Algorithm 2 finds the robust mean and covariance of the training hyperplanes. To overcome this limitation, we have implemented a sampling strategy yielding twice as many samples as input-features; each sample contains a random and non-duplicated 60% of all examples; producing slightly variant hyperplanes per sample during training. Naturally, as the number features increases, so does the number of samples needed and the likelihood that two identical samples are retrieved; problematic for downstream calculations. Another possible approach is up/over-sampling i.e., re-sampling with replacement. However, since the gene sets used herein have a maximum of 50 features, our primary approach (random samples without replacement) is appropriate with lower computational complexity.

4.2 Classification results of cell cycle state in scRNASeq data with the RTL framework

To validate and benchmark the RTL framework we conducted an equivalent classification in comparison to the top performing algorithms reported in [89]; requiring parallel pre-processing to reach the final 40 gene set used (Table 16). The results reported are split based on several investigative criteria: As described previously, we first developed and adapted a baseline implementation of the RTL framework based on a previously published algorithm for classifying immunophenotypes in flow-cytometry, to classify transcriptional signatures in scRNASeq. Next, we implemented an overhaul of 3 of the modules (algorithms 4, 5, and 6) in an extended version (Alg4-Alg5-Alg6 V2.0) to improve the overall robustness in transfering the classification inference to the test set to make an adapted inference. Finally, we have introduced a new feature, the AUD-based parameter transfer (PT, equation 33), specifically to improve the classification in the context of rare cellular subsets (RCS). The final call is made by leveraging the stochastic nature of the RTL framework's internal algorithms (such as the gradient descent) by conducting repeated epochs (N=25) of 5-fold cross-validation to obtain the probability of each cell's association to the three class labels; the highest probability defines the final call, but to break ties, as oppose to a no-call, the priority scheme G1>G2M>S was used. This priority scheme is partially based on the PCA analysis of the training set (Figure 39) where we observed that the G1 cells (green) are clearly the most distinct cluster, and mostly overlap with the G2M cells. Similarly, the G2M and the S-phase cells mostly overlap; depending on the normalization, they may even represent a single cluster. Furthermore, previous publication on the EMTAB2805 data suggests that the S phase cells are the most difficult to classify. Combined the G1>G2M>S scheme ensures that when a 'S-phase' call is made, it is with high precision.

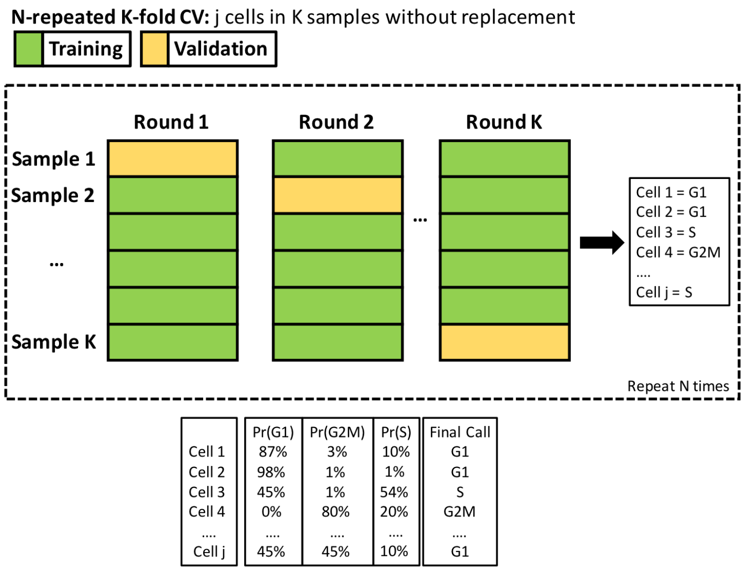


Figure 46: General scheme of conducting N (25) repeated K (5) fold cross-validation runs to compute each cell's probability to be classified to each of the class labels. In the external validation described below, the validation set shown is replaced with the test set of interest.

4.2.1 Internal Validation of the RTL framework with EMTAB2805:

Cell cycle	N	Freq%
S	58	31.9%
G1	59	32.4%
G2M	65	35.7%
Total	182	100.0%

Table 17: EMTAB2805 dataset

The baseline implementation of algorithms 4 and 6 require defining respective margins. Visualized as a boundary around a hyperplane, the margins in the specified algorithms are used as means to count mapped events that fall within in them; larger margins, effectively smooth over the minima that needs to be identified by a gradient descent algorithm and the smaller the margin we observe higher likelihood of multiple local minimas. Before we initiated classification within the RTL framework with the baseline version, it was necessary to find the optimal parameterization for these two key parameters. For each margin, we evaluated the classification results with two criteria: First, the classification sum squared-error (Equation 32) adapted for K-fold cross-validation with Equation 31 as in [50] was used. Second, a qualitative evaluation of the resulting density curves of the mappings from each algorithm i.e., $\hat{y}$ to visually verify the existence of a minima in the density curve which should be found by a gradient descent algorithm. We identified the optimal values can change based on the normalization procedure; for the rank-based normalization ($Alg4_{margin} \approx 0.1$ and $Alg6_{margin} \approx 0.4$) versus the other settings including the log-transformed counts ($Alg4_{margin} \approx 0.5$ and $Alg6_{margin} \approx 0.7$). This was done by conducting 5-fold cross-validation across a pair-wise matrix of pre-set values ($Alg4_{margin}\ and\ Alg6_{margin} \in (0, 1]\ step\ 0.1$); these values obtained on the training set are used for all downstream respective analyses when utilizing the baseline RTL implementation.

Finding the optimal K for cross-validation. Using the rank normalized EMTAB2805 182 cells by 405 gens dataset, filtered by the reduced 40-gene set, we computed the test set classification results obtained by K-fold cross-validation with K=3, 5, and 10. We chose 'K=5' as the optimal value for downstream classification attempts since the 5-folds hold a reasonably size representative sample to avoid concerns pertaining over-fitting. Additionally, the high accuracy and $F1_{score}$ achieved (Table 18) supports this selection. We also found that in this context, the baseline version performs with slightly higher precision, specificity, and recall.

Evaluating the effects of normalization on classification with the RTL framework. To observe the normalization effects on classification, we report the results of similar runs to the 5-fold CV runs with the rank-based normalization as well as two other methods compared to the log-transformed counts. This evaluation is done with the baseline and extended implementations of the RTL framework reported in Table 19 combined with Table 18-B and E. Initially, we observe the highest average $F1_{score}$ pertains too the log-median-absolute-value normalized samples using the baseline implementation ($F1_{score} = 0.91$). In the equivalent run with the extended implementation, lower specificity, precision, recall, and accuracy are found. As a case example, we examine the etiology of in Figure 47 for the G1 cells. The main difference we observe is that in updating the bias, the extended Algorithm 4 has found the optimal minima approximately at 0.1, whereas the baseline version has found the optima at -0.5; this difference explains the difference in recall (36% vs. 83% respectively). The associated outputs of Algorithm 6, are as expect. A broad takeaway from these results is that task-specific normalization can confound the classification results with the RTL framework; thus the selection or lack of task-specific normalization prior to classification with the RTL framework is important to evaluate on a given training set.

	Specificity	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Results w/ RTL baseline implementation:								
A = Rank normalized, K=3								
Class: G1	0.76	0.66	1.00	0.80	0.32	0.32	0.49	0.88
Class: G2M	0.89	0.82	0.89	0.85	0.36	0.32	0.39	0.89
Class: S	0.98	0.86	0.33	0.48	0.32	0.10	0.12	0.65
Average	0.87	0.78	0.74	**0.71**	0.33	0.25	0.33	0.81
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.75	0.68	0.81
B = A w/ K=5								
Class: G1	0.78	0.69	1.00	0.81	0.32	0.32	0.47	0.89
Class: G2M	0.87	0.80	0.92	0.86	0.36	0.33	0.41	0.90
Class: S	0.98	0.90	0.33	0.48	0.32	0.10	0.12	0.66
Average	0.88	0.80	0.75	**0.72**	0.33	0.25	0.33	0.81
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.76	0.69	0.82
C = A w/ K=10								
Class: G1	0.85	0.76	1.00	0.86	0.32	0.32	0.43	0.92
Class: G2M	0.84	0.77	0.98	0.86	0.36	0.35	0.46	0.91
Class: S	1.00	1.00	0.36	0.53	0.32	0.12	0.12	0.68
Average	0.89	0.84	0.78	**0.75**	0.33	0.26	0.33	0.84
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.79	0.72	0.85
Results w/ RTL Alg4-Alg5-Alg6 V2.0 extended version:								
D = A								
Class: G1	0.90	0.80	0.83	0.82	0.32	0.27	0.34	0.87
Class: G2M	0.74	0.67	0.97	0.79	0.36	0.35	0.52	0.85
Class: S	0.94	0.70	0.33	0.45	0.32	0.10	0.15	0.63
Average	0.86	0.73	0.71	**0.69**	0.33	0.24	0.33	0.78
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.72	0.65	0.78
E = B								
Class: G1	0.89	0.80	0.88	0.84	0.32	0.29	0.36	0.89
Class: G2M	0.69	0.64	0.98	0.78	0.36	0.35	0.55	0.84
Class: S	0.98	0.82	0.24	0.37	0.32	0.08	0.09	0.61
Average	0.85	0.75	0.70	**0.66**	0.33	0.24	0.33	0.78
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.71	0.64	0.78
F = C								
Class: G1	0.85	0.74	0.92	0.82	0.32	0.30	0.40	0.88
Class: G2M	0.70	0.65	0.98	0.78	0.36	0.35	0.54	0.84
Class: S	0.98	0.70	0.12	0.21	0.32	0.04	0.05	0.55
Average	0.84	0.70	0.67	**0.60**	0.33	0.23	0.33	0.76
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.69	0.61	0.75

Table 18: Test set classification results to determine optimal 'K=5' for K-fold cross-validation in the RTL framework. Cells are from EMTAB2805 filtered by the 405 cell cycle genes, rank-normalized, and further filtered by the Scialdone et al.'s 40 genes. Multiple (25) cross-validated runs per class label are used to determine the probability of each cell's association to each label; the highest probability was chosen as the final call. **A-C** Baseline RTL implementation and **D-F** classification with the RTL framework with the extended Alg4, Alg5, and Alg6 modules i.e., Alg4-Alg5-Alg6 V2.0.

	Specificity	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Results w/ RTL baseline implementation, K=5:								
A = Log counts, w/o PT								
Class: G1	0.91	0.84	0.97	0.90	0.32	0.31	0.37	0.94
Class: G2M	0.89	0.82	0.94	0.88	0.36	0.34	0.41	0.91
Class: S	0.96	0.88	0.60	0.71	0.32	0.19	0.22	0.78
Average	0.92	0.85	0.84	**0.83**	0.33	0.28	0.33	0.88
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.84	0.78	0.89
B = A w/ log median absolute value normalized								
Class: G1	0.98	0.96	0.83	0.89	0.32	0.27	0.28	0.91
Class: G2M	0.95	0.91	0.95	0.93	0.36	0.34	0.37	0.95
Class: S	0.93	0.86	0.93	0.89	0.32	0.30	0.35	0.93
Average	0.95	0.91	0.91	**0.91**	0.33	0.30	0.33	0.93
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.91	0.85	0.94
C = A w/ quantile normalized								
Class: G1	0.89	0.80	0.97	0.88	0.32	0.31	0.39	0.93
Class: G2M	0.82	0.75	0.97	0.85	0.36	0.35	0.46	0.89
Class: S	0.98	0.89	0.41	0.56	0.32	0.13	0.15	0.69
Average	0.89	0.81	0.78	**0.76**	0.33	0.26	0.33	0.84
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.79	0.72	0.85
Results w/ RTL Alg4-Alg5-Alg6 V2.0 extended version, K=5:								
D = A								
Class: G1	0.93	0.86	0.97	0.91	0.32	0.31	0.36	0.95
Class: G2M	0.78	0.71	0.97	0.82	0.36	0.35	0.49	0.87
Class: S	0.98	0.89	0.41	0.56	0.32	0.13	0.15	0.69
Average	0.89	0.82	0.78	**0.76**	0.33	0.26	0.33	0.84
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.79	0.72	0.85
E = B								
Class: G1	1.00	1.00	0.36	0.53	0.32	0.12	0.12	0.68
Class: G2M	0.86	0.77	0.85	0.81	0.36	0.30	0.39	0.85
Class: S	0.73	0.62	0.97	0.76	0.32	0.31	0.49	0.85
Average	0.86	0.80	0.72	**0.70**	0.33	0.24	0.33	0.79
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.73	0.65	0.79
F = C								
Class: G1	0.87	0.78	0.98	0.87	0.32	0.32	0.41	0.93
Class: G2M	0.81	0.74	0.95	0.83	0.36	0.34	0.46	0.88
Class: S	0.98	0.88	0.36	0.51	0.32	0.12	0.13	0.67
Average	0.89	0.80	0.77	**0.74**	0.33	0.26	0.33	0.83
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.77	0.71	0.83

Table 19: Test set classification results to evaluate the effects of normalization on classification with the RTL framework. Cells are from EMTAB2805 filtered by the 405 cell cycle genes, normalized as stated, and further filtered by the Scialdone et al.'s 40 genes. Multiple (25) 5-fold cross-validated runs per class label are used to determine the probability of each cell's association to each label; the highest probability was chosen as the final call. **A-C** Baseline RTL implementation and **D-F** classification with the RTL framework with the extended Alg4, Alg5, and Alg6 modules i.e., Alg4-Alg5-Alg6 V2.0.

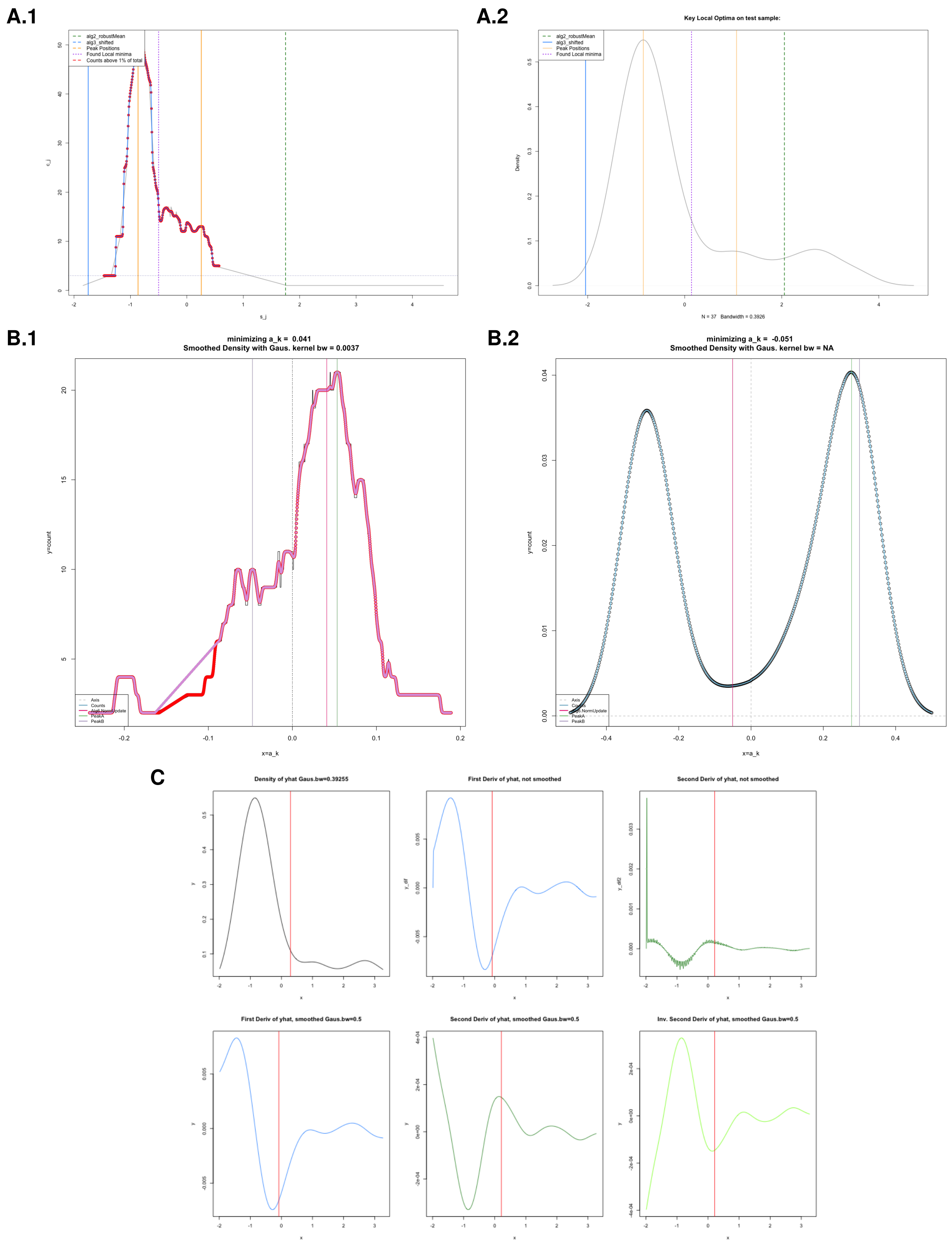


Figure 47: Case example forensics to compare the baseline and extended versions of the RTL framework. **A.1** The mapped results from the baseline version of Algorithm 4 and **A.2** the extended version. **B.1** and **B.2** are the respective outputs from Algorithm 6. **C** Illustration of the specific approach of the extended Algorithm 4 bias update where the density-curve, its first and inverse of second derivatives are used to make a robust final call.

4.2.2 External Validation with GSE42268:

Cell cycle	N	Freq%
S	7	20.0%
G1	20	57.1%
G2M	8	22.9%
Total	35	100.0%

Table 20: GSE42268 dataset

Evaluating generalizability in the context of normalization. We trained on the 182 cell by 40 gene EMTAB2805 dataset and transferred the classifying hyperplane to the 35 mESC of GSE42268 filtered by the 40-gene set; running multiple (25), 5-fold cross-validated epochs. This was done with the previously identified optimal parameters for Alg4 and Alg 6 margins on the training set when the baseline implementation was tested. Overall, the extensions (Alg4-Alg5-Alg6 V2.0) demonstrated slightly higher accuracy and average $F1_{score}$. The best classification statistics were achieved in the context of quantile normalization and the log-transformed counts. Finally, unlike in the internal validation results, here logMAV normalization resulted in the bottom two classification attempts (Table 21-B and F).

	Specificity	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Results w/ RTL baseline implementation, w/o PT, K=5:								
A = Log counts								
Class: G1	0.93	0.93	0.70	0.80	0.57	0.40	0.43	0.82
Class: G2M	0.67	0.44	0.88	0.58	0.23	0.20	0.46	0.77
Class: S	0.93	0.50	0.29	0.36	0.20	0.06	0.11	0.61
Average	0.84	0.62	0.62	**0.58**	0.33	0.22	0.33	0.73
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.66	0.48	0.81
B = A w/ log median absolute value normalized								
Class: G1	0.80	0.70	0.35	0.47	0.57	0.20	0.29	0.57
Class: G2M	0.81	0.00	0.00	0.00	0.23	0.00	0.14	0.41
Class: S	0.39	0.15	0.43	0.22	0.20	0.09	0.57	0.41
Average	0.67	0.28	0.26	**0.23**	0.33	0.10	0.33	0.46
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.29	0.15	0.46
C = A w/ quantile normalized								
Class: G1	1.00	1.00	0.65	0.79	0.57	0.37	0.37	0.82
Class: G2M	0.59	0.42	1.00	0.59	0.23	0.23	0.54	0.80
Class: S	0.93	0.33	0.14	0.20	0.20	0.03	0.09	0.54
Average	0.84	0.58	0.60	**0.53**	0.33	0.21	0.33	0.72
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.63	0.45	0.79
D = A w/ rank normalized								
Class: G1	1.00	1.00	0.75	0.86	0.57	0.43	0.43	0.88
Class: G2M	0.59	0.42	1.00	0.59	0.23	0.23	0.54	0.80
Class: S	1.00	1.00	0.14	0.25	0.20	0.03	0.03	0.57
Average	0.86	0.81	0.63	**0.57**	0.33	0.23	0.33	0.75
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.69	0.51	0.83
Results w/ RTL Alg4-Alg5-Alg6 V2.0 extended version, w/o PT, K=5:								
E = A								
Class: G1	0.93	0.94	0.75	0.83	0.57	0.43	0.46	0.84
Class: G2M	0.70	0.47	0.88	0.61	0.23	0.20	0.43	0.79
Class: S	0.93	0.50	0.29	0.36	0.20	0.06	0.11	0.61
Average	0.86	0.63	0.64	**0.60**	0.33	0.23	0.33	0.75
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.69	0.51	0.83
F = B								
Class: G1	1.00	1.00	0.20	0.33	0.57	0.11	0.11	0.60
Class: G2M	0.78	0.00	0.00	0.00	0.23	0.00	0.17	0.39
Class: S	0.25	0.16	0.57	0.25	0.20	0.11	0.71	0.41
Average	0.68	0.39	0.26	**0.19**	0.33	0.08	0.33	0.47
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.23	0.10	0.40
G = C								
Class: G1	0.87	0.88	0.75	0.81	0.57	0.43	0.49	0.81
Class: G2M	0.70	0.47	0.88	0.61	0.23	0.20	0.43	0.79
Class: S	0.96	0.67	0.29	0.40	0.20	0.06	0.09	0.62
Average	0.84	0.67	0.64	**0.61**	0.33	0.23	0.33	0.74
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.69	0.51	0.83
H = D								
Class: G1	0.73	0.80	0.80	0.80	0.57	0.46	0.57	0.77
Class: G2M	0.81	0.58	0.88	0.70	0.23	0.20	0.34	0.84
Class: S	0.93	0.33	0.14	0.20	0.20	0.03	0.09	0.54
Average	0.83	0.57	0.61	**0.57**	0.33	0.23	0.33	0.72
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.69	0.51	0.83

Table 21: External validation test set classification results within the RTL framework. Cells for training are from EMTAB2805, the test set cells are from GSE42268 filtered by the Scialdone et al.'s 40 genes. Multiple (25) 5-fold cross-validated runs per class label are used to determine the probability of each cell's association to each label; the highest probability was chosen as the final call. **A-D** Baseline RTL implementation and **E-H** classification with the RTL framework with the extended Alg4, Alg5, and Alg6 modules i.e., Alg4-Alg5-Alg6 V2.0.

4.2.3 Classification of rare cellular subsets (RCS):

In silico RCS evaluation with EMTAB2805. Because of the small sample size, down-sampling the positive class to represent a marginal frequency of 1% or lower was not feasible, thus we up-sampled the negative class (Figure 48). Using the 40-gene set for parallel benchmarking to our prior results, we focused on the classification of rare cellular subsets (RCS) by conducting multiple (25) K-fold cross-validation runs with the RTL framework on our *In silico* re-sampled dataset. Because of the up-sampling, the calls made on the multiple examples of the same cell(s) within each sample were averaged within an epoch. The accumulated result over multiple epochs (25) were used to obtain the probability of each cell being labeled as the positive class for each of the respective cell cycle labels. In other words, because each cell cycle label represents a specific signature, unlike in the non-RCS context, the final call is not made by combining the probabilities to determine the

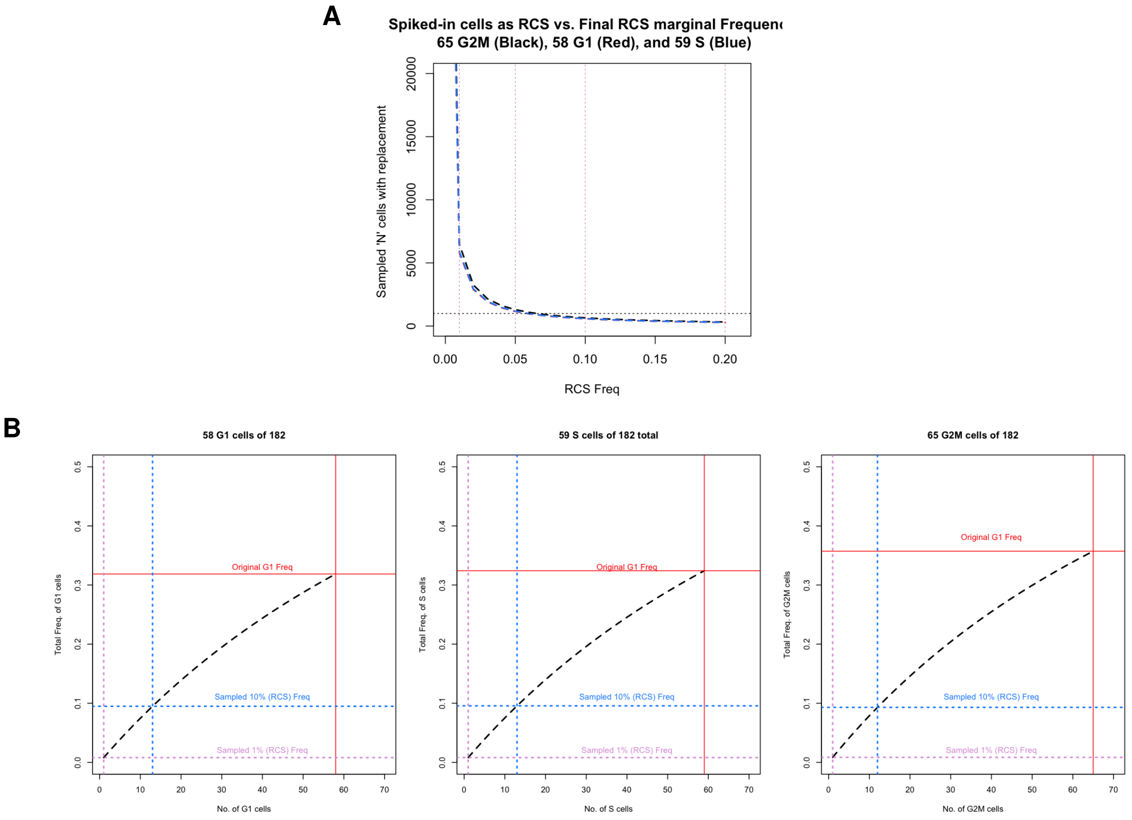
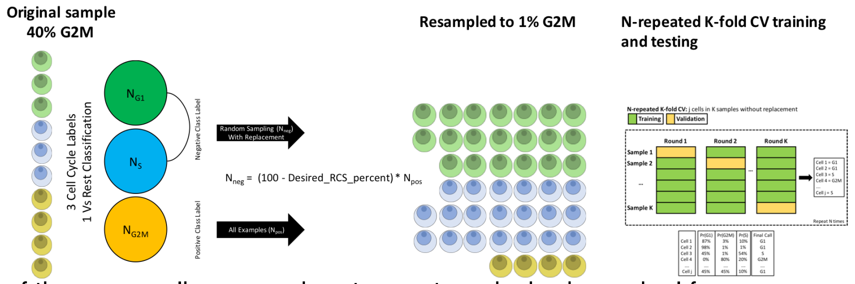


Figure 48: Total cells in the final RCS in silico sampling methods**A.** Negative subset sampled with replacement and combined with entire positive class (i.e., the negative subset is up-sampled). **B.** The positive class is down-sampled per class label to define rare cellular subsets (RCS); defined thresholds (e.g. 20-1% of total cells) illustrate feasibility at various marginal frequencies.

highest probability of a given cell to a specific label. Rather, individually for each label, the probability of being associated with that label is determined by being called as such in more than 50% of the K-fold cross-validation runs. As before, we first compare the baseline and extended versions of the RTL framework (without parameter transfer i.e., AUD-PT) as reported in Table 22. Briefly, we observe that the baseline implementation performs well, even with RCS down to 1% of the total. Without AUD-PT, the extended version has poor recall, but better precision than the baseline implementation suggesting the extensions yield bias in this regard.



	Pr(G1)	Pr(G2M)	Pr(S)	Final Call
Cell 1	87%	3%	10%	G1
Cell 2	98%	1%	1%	G1
Cell 3	45%	1%	54%	S
Cell 4	0%	80%	20%	G2M
....	....	....	....	....
Cell j	45%	45%	10%	G1

Figure 49: A visualization of the re-sampling procedure to create a desired marginal frequency per class label.

	Specificity	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Results w/ RTL baseline implementation, w/o PT, K=3:								
A (1%) = rank normalized								
Class: G1	0.99	0.47	1.00	0.64	0.01	0.01	0.02	0.99
Class: G2M	0.99	0.63	0.98	0.77	0.01	0.01	0.02	0.99
Class: S	0.99	0.42	0.72	0.53	0.01	0.01	0.02	0.86
Average	0.99	0.51	0.90	**0.65**	0.01	0.01	0.02	0.95
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.99	0.99	0.99
B (5%) = rank normalized								
Class: G1	0.95	0.47	1.00	0.64	0.04	0.04	0.09	0.97
Class: G2M	0.97	0.60	0.98	0.75	0.05	0.05	0.08	0.98
Class: S	0.96	0.44	0.71	0.54	0.04	0.03	0.07	0.83
Average	0.96	0.50	0.90	**0.64**	0.04	0.04	0.08	0.93
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.96	0.94	0.97
C (10%) = rank normalized								
Class: G1	0.90	0.48	1.00	0.64	0.08	0.08	0.17	0.95
Class: G2M	0.93	0.59	0.98	0.74	0.09	0.09	0.15	0.96
Class: S	0.93	0.52	0.79	0.63	0.08	0.06	0.13	0.86
Average	0.92	0.53	0.93	**0.67**	0.09	0.08	0.15	0.92
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.92	0.90	0.94
Results w/ RTL Alg4-Alg5-Alg6 V2.0 extended version, w/o PT, K=3:								
D (1%) = A								
Class: G1	1.00	0.64	0.98	0.78	0.01	0.01	0.01	0.99
Class: G2M	0.99	0.50	1.00	0.67	0.01	0.01	0.02	1.00
Class: S	0.99	0.46	0.76	0.57	0.01	0.01	0.01	0.88
Average	0.99	0.54	0.91	**0.67**	0.01	0.01	0.02	0.95
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.99	0.99	0.99
E (5%) = B								
Class: G1	0.99	0.82	0.86	0.84	0.04	0.04	0.05	0.93
Class: G2M	0.95	0.52	0.98	0.68	0.05	0.05	0.09	0.97
Class: S	0.96	0.44	0.76	0.56	0.04	0.03	0.07	0.86
Average	0.97	0.59	0.87	**0.69**	0.04	0.04	0.07	0.92
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.96	0.95	0.97
F (10%) = C								
Class: G1	1.00	1.00	0.25	0.41	0.08	0.02	0.02	0.63
Class: G2M	1.00	0.96	0.75	0.84	0.09	0.07	0.07	0.88
Class: S	1.00	0.90	0.16	0.26	0.08	0.01	0.01	0.58
Average	1.00	0.95	0.39	**0.50**	0.09	0.03	0.04	0.69
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.95	0.93	0.96

Table 22: Classifying rare cellular subsets (RCS) using EMTAB2805 without density-based parameter transfer (PT). Cells are from EMTAB2805 filtered by the 405 cell cycle genes, normalized as stated, and further filtered by the Scialdone et al.'s 40 genes. Separately, multiple (25) 5-fold cross-validated runs per class label are used to determine the probability of each cell's association to each label in the context RCS. That is, the negative class is re-sampled without replacement as described earlier, to produce the specified marginal frequency of the positive class (i.e., 1, 5, and 10% of total). Classification is reported as 1-vs-rest scheme.

	Specificity	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Results w/ RTL baseline implementation, w PT, K=3:								
A (1%) = rank normalized								
Class: G1	1.00	0.68	0.98	0.81	0.01	0.01	0.01	0.99
Class: G2M	1.00	0.77	0.95	0.85	0.01	0.01	0.01	0.98
Class: S	1.00	0.56	0.53	0.55	0.01	0.00	0.01	0.77
Average	1.00	0.67	0.82	**0.73**	0.01	0.01	0.01	0.91
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.99	0.99	1.00
B (5%) = rank normalized								
Class: G1	0.97	0.61	1.00	0.76	0.04	0.04	0.07	0.99
Class: G2M	0.97	0.62	0.98	0.76	0.05	0.05	0.08	0.98
Class: S	0.98	0.64	0.60	0.62	0.04	0.03	0.04	0.79
Average	0.98	0.62	0.86	**0.71**	0.04	0.04	0.06	0.92
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.97	0.96	0.98
C (10%) = rank normalized								
Class: G1	0.96	0.69	1.00	0.81	0.08	0.08	0.12	0.98
Class: G2M	0.93	0.60	0.98	0.74	0.09	0.09	0.15	0.96
Class: S	0.93	0.48	0.69	0.56	0.08	0.06	0.12	0.81
Average	0.94	0.59	0.89	**0.71**	0.09	0.08	0.13	0.92
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.94	0.92	0.95
Results w/ RTL Alg4-Alg5-Alg6 V2.0 extended version, w PT, K=3:								
D (1%) = A								
Class: G1	1.00	1.00	0.39	0.56	0.01	0.00	0.00	0.69
Class: G2M	1.00	0.88	0.91	0.89	0.01	0.01	0.01	0.95
Class: S	1.00	0.91	0.17	0.29	0.01	0.00	0.00	0.59
Average	1.00	0.93	0.49	**0.58**	0.01	0.00	0.01	0.74
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.99	0.99	1.00
E (5%) = B								
Class: G1	1.00	1.00	0.27	0.43	0.04	0.01	0.01	0.64
Class: G2M	1.00	0.94	0.72	0.82	0.05	0.03	0.04	0.86
Class: S	1.00	1.00	0.14	0.24	0.04	0.01	0.01	0.57
Average	1.00	0.98	0.38	**0.50**	0.04	0.02	0.02	0.69
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.97	0.96	0.98
F (10%) = C								
Class: G1	1.00	1.00	0.15	0.26	0.08	0.01	0.01	0.58
Class: G2M	1.00	1.00	0.51	0.67	0.09	0.05	0.05	0.75
Class: S	1.00	0.94	0.26	0.41	0.08	0.02	0.02	0.63
Average	1.00	0.98	0.31	**0.45**	0.09	0.03	0.03	0.65
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.94	0.92	0.96

Table 23: Classifying rare cellular subsets (RCS) using EMTAB2805 with density-based parameter transfer (PT) such that PT = (freq=$p(class_{G1,\ G2M,\ or\ S}|X_{train})\ +\ \epsilon$). Cells are from EMTAB2805 filtered by the 405 cell cycle genes, normalized as stated, and further filtered by the Scialdone et al.'s 40 genes. Separately, multiple (25) 3-fold cross-validated runs per class label are used to determine the probability of each cell's association to each label in the context RCS. That is, the negative class is re-sampled without replacement as described earlier, to produce the specified marginal frequency of the positive class (i.e., 1, 5, and 10% of total). Classification is reported as 1-vs-rest scheme.

4.2.4 AUD-based parameter transfer (PT) in a non-RCS setting

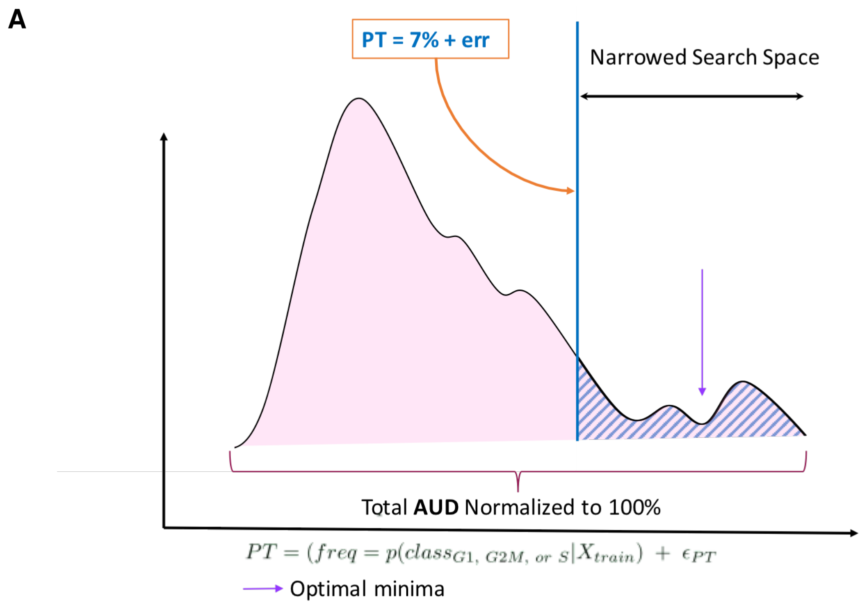

A
PT = 7% + err
Narrowed Search Space
Total AUD Normalized to 100%
$PT = (freq = p(class_{G1,\ G2M,\ or\ S}|X_{train}) + \epsilon_{PT}$
Optimal minima

Figure 50: An illustration of the AUD-based parameter transfer (PT). From the training set the marginal frequency of the positive class is obtained with an added random error to account for the differences between the testing and training sets. The reduction of the search space aids the gradient decent algorithm in finding the optimal low-density minima.

we tested the marginal frequency of the positive class learned during training for either RTL versions would improve the overall classification (Table 24). AUD-PT (PT for short) was originally conceived and implemented to overcome the limitations associated with classifying RCS, discussed in an earlier section. When this optimization was applied to the classification in this non-RCS setting, no major changes were observed to the overall accuracy or the average $F1_{score}$, implying that the original classification calls were optimal relative to the underlying data. But in the RCS mode, as expected, we observe an overall boost to precision, recall, and thus the $F1_{score}$. In fact, the best performance for each frequency tested i.e., 1, 5, and 10% was the baseline implementation with AUD-PT.

	Specificity	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Results w/ RTL baseline implementation, w/ PT, K=5:								
A = rank normalized								
Class: G1	0.82	0.73	1.00	0.84	0.32	0.32	0.45	0.91
Class: G2M	0.86	0.79	0.95	0.87	0.36	0.34	0.43	0.91
Class: S	1.00	1.00	0.40	0.57	0.32	0.13	0.13	0.70
Average	0.89	0.84	0.78	0.76	0.33	0.26	0.33	0.84
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.79	0.72	0.85
Results w/ RTL extended version, w/ PT, K=5:								
A = rank normalized								
Class: G1	0.93	0.84	0.71	0.77	0.32	0.23	0.27	0.82
Class: G2M	0.72	0.66	0.98	0.79	0.36	0.35	0.53	0.85
Class: S	0.87	0.54	0.33	0.41	0.32	0.10	0.19	0.60
Average	0.84	0.68	0.67	0.66	0.33	0.23	0.33	0.76
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.69	0.61	0.75

Table 24: Parallel classification to the internal validation using EMTAB2805, but with density-based parameter transfer (PT) such that PT = (freq=$p(class_{G1,\ G2M,\ or\ S}|X_{train}) + \epsilon$). Only rank-based normalization shown for equivalent comparison.

	Specificity	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Accuracy
Results w/ RTL baseline implementation, w/ PT, K=5:								
A = rank normalized								
Class: G1	0.93	0.93	0.70	0.80	0.57	0.40	0.43	0.82
Class: G2M	0.70	0.50	1.00	0.67	0.23	0.23	0.46	0.85
Class: S	0.93	0.50	0.29	0.36	0.20	0.06	0.11	0.61
Average	0.86	0.64	0.66	0.61	0.33	0.23	0.33	0.76
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.69	0.51	0.83
Results w/ RTL extended version, w/ PT, K=5:								
A = rank normalized								
Class: G1	0.73	0.80	0.80	0.80	0.57	0.46	0.57	0.77
Class: G2M	0.89	0.67	0.75	0.71	0.23	0.17	0.26	0.82
Class: S	0.89	0.50	0.43	0.46	0.20	0.09	0.17	0.66
Average	0.84	0.66	0.66	0.66	0.33	0.24	0.33	0.75
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.71	0.54	0.85

Table 25: Parallel classification to the external validation using EMTAB2805 (training) and GSE42268 (testing), but with density-based parameter transfer (PT) such that PT = (freq=$p(class_{G1,\ G2M,\ or\ S}|X_{train})\ +\ \epsilon$). Only rank-based normalization shown for equivalent comparison.

4.2.5 Classification with an alternate set of genes/features:

Top 'N' identified gene sets	Length of gene set
1- S-phase Rank Norm	45.00
2- G1-phase Rank Norm	48.00
3- G2M-phase Rank Norm	27.00
4- S-phase Log counts	49.00
5- G1-phase Log counts	29.00
6- G2M-phase Log counts	8.00
7- S-phase Log MAV Norm	44.00
8- G1-phase Log MAV Norm	18.00
9- G2M-phase Log MAV Norm	47.00
10- S-phase Quant Norm	48.00
11- G1-phase Quant Norm	35.00
12- G2M-phase Quant Norm	14.00
13- Scialdone 40 gene set	40.00

Table 26: Gene sets (13) and their lengths

Ensemble Gene ID	Gene Symbol Name	N times Genes in sets (n/13)
ENSMUSG00000049932	H2afx	11.00
ENSMUSG00000038943	Prc1	9.00
ENSMUSG00000032254	Kif23	9.00
ENSMUSG00000030867	Plk1	8.00
ENSMUSG00000001403	Ube2c	8.00
ENSMUSG00000041431	Ccnb1	8.00
ENSMUSG00000027306	Nusap1	8.00
ENSMUSG00000006398	Cdc20	8.00
ENSMUSG00000027496	Aurka	7.00
ENSMUSG00000037313	Tacc3	7.00
ENSMUSG00000072082	Ccnf	7.00
ENSMUSG00000024754	Tmem2	6.00
ENSMUSG00000027469	Tpx2	6.00
ENSMUSG00000020235	Fzr1	6.00
ENSMUSG00000045328	Cenpe	6.00
ENSMUSG00000044201	Cdc25c	5.00
ENSMUSG00000048327	Ckap2l	5.00
ENSMUSG00000010342	Tex14	5.00
ENSMUSG00000044626	Liph	5.00
ENSMUSG00000031787	Katnb1	5.00
ENSMUSG00000021250	Fos	4.00
ENSMUSG00000022945	Chaf1b	4.00
ENSMUSG00000028583	Pdpn	4.00
ENSMUSG00000015839	Nfe2l2	4.00
ENSMUSG00000022432	Smc1b	4.00
ENSMUSG00000058290	Espl1	4.00
ENSMUSG00000001517	Foxm1	4.00
ENSMUSG00000064302	Clasp1	4.00
ENSMUSG00000027330	Cdc25b	4.00
ENSMUSG00000048922	Cdca2	4.00
ENSMUSG00000034575	Papd7	4.00
ENSMUSG00000021697	Depdc1b	4.00
ENSMUSG00000024087	Cyp1b1	3.00
ENSMUSG00000052087	Rgs14	3.00
ENSMUSG00000052560	Cpne8	3.00
ENSMUSG00000063065	Mapk3	3.00
ENSMUSG00000020745	Pafah1b1	3.00
ENSMUSG00000025077	Dclre1a	3.00
ENSMUSG00000031392	Irak1	3.00
ENSMUSG00000034154	Ino80	3.00
ENSMUSG00000063358	Mapk1	3.00
ENSMUSG00000078652	Psme3	3.00
ENSMUSG00000033713	Foxn3	3.00
ENSMUSG00000029554	Mad1l1	3.00
ENSMUSG00000028678	Kif2c	3.00
ENSMUSG00000031756	Cenpn	3.00
ENSMUSG00000022033	Pbk	3.00
ENSMUSG00000040084	Bub1b	3.00
ENSMUSG00000023150	Ivns1abp	3.00
ENSMUSG00000020516	Rps6kb1	3.00

Table 27: Genes in more than 2 gene sets (out of 13 as in Table 26)

	Specificity	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Results w/ RTL baseline implementation, w/ PT, K=5:								
A = Log counts								
Class: G1	0.98	0.95	0.97	0.96	0.32	0.31	0.33	0.97
Class: G2M	0.96	0.93	0.97	0.95	0.36	0.35	0.37	0.96
Class: S	0.98	0.94	0.88	0.91	0.32	0.28	0.30	0.93
Average	0.97	0.94	0.94	0.94	0.33	0.31	0.33	0.95
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.94	0.89	0.97
B = A w/ log median absolute value normalized								
Class: G1	1.00	1.00	0.88	0.94	0.32	0.29	0.29	0.94
Class: G2M	0.97	0.95	0.97	0.96	0.36	0.35	0.36	0.97
Class: S	0.93	0.86	0.95	0.90	0.32	0.30	0.35	0.94
Average	0.97	0.94	0.93	0.93	0.33	0.31	0.33	0.95
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.93	0.89	0.97
C = rank normalized								
Class: G1	0.91	0.83	0.93	0.88	0.32	0.30	0.36	0.92
Class: G2M	0.96	0.93	0.97	0.95	0.36	0.35	0.37	0.96
Class: S	0.97	0.92	0.76	0.83	0.32	0.24	0.26	0.86
Average	0.95	0.89	0.89	0.89	0.33	0.30	0.33	0.92
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.89	0.84	0.93
Results w/ RTL extended version, w/ PT, K=5:								
A = Log counts								
Class: G1	0.97	0.93	0.95	0.94	0.32	0.31	0.33	0.96
Class: G2M	0.70	0.64	0.97	0.77	0.36	0.35	0.54	0.84
Class: S	0.98	0.88	0.36	0.51	0.32	0.12	0.13	0.67
Average	0.88	0.82	0.76	0.74	0.33	0.26	0.33	0.82
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.77	0.70	0.83
B = A w/ log median absolute value normalized								
Class: G1	1.00	1.00	0.63	0.77	0.32	0.20	0.20	0.81
Class: G2M	0.99	0.98	0.69	0.81	0.36	0.25	0.25	0.84
Class: S	0.67	0.59	1.00	0.74	0.32	0.32	0.54	0.83
Average	0.89	0.85	0.77	0.77	0.33	0.26	0.33	0.83
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.77	0.70	0.83
C = rank normalized								
Class: G1	0.93	0.86	0.83	0.84	0.32	0.27	0.31	0.88
Class: G2M	0.79	0.73	0.98	0.84	0.36	0.35	0.48	0.89
Class: S	0.97	0.89	0.57	0.69	0.32	0.18	0.20	0.77
Average	0.90	0.83	0.79	0.79	0.33	0.27	0.33	0.85
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.80	0.74	0.86

Table 28: Test set classification results to evaluate classification using an alternate set of genes. Parallel with the internval validation test set classification results using cells from EMTAB2805, but filtered by top 'N' genes as previously described per normalization method. Multiple (25) 5-fold cross-validated runs per class label are used to determine the probability of each cell's association to each label; the highest probability was chosen as the final call. **A-D** Baseline RTL implementation and **E-H** classification with the RTL framework with the extended Alg4, Alg5, and Alg6 modules i.e., Alg4-Alg5-Alg6 V2.0.

	Specificity	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Results w/ RTL baseline implementation, w/ PT, K=5:								
A = Log counts								
Class: G1	0.93	0.94	0.75	0.83	0.57	0.43	0.46	0.84
Class: G2M	0.70	0.50	1.00	0.67	0.23	0.23	0.46	0.85
Class: S	0.96	0.67	0.29	0.40	0.20	0.06	0.09	0.62
Average	0.87	0.70	0.68	0.63	0.33	0.24	0.33	0.77
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.71	0.54	0.85
B = A w/ log median absolute value normalized								
Class: G1	0.87	0.67	0.20	0.31	0.57	0.11	0.17	0.53
Class: G2M	0.85	0.33	0.25	0.29	0.23	0.06	0.17	0.55
Class: S	0.32	0.17	0.57	0.27	0.20	0.11	0.66	0.45
Average	0.68	0.39	0.34	0.29	0.33	0.10	0.33	0.51
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.29	0.15	0.46
C = rank normalized								
Class: G1	0.93	0.93	0.70	0.80	0.57	0.40	0.43	0.82
Class: G2M	0.81	0.58	0.88	0.70	0.23	0.20	0.34	0.84
Class: S	0.82	0.38	0.43	0.40	0.20	0.09	0.23	0.62
Average	0.86	0.63	0.67	0.63	0.33	0.23	0.33	0.76
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.63	0.45	0.79
Results w/ RTL extended version, w/ PT, K=5:								
A = Log counts								
Class: G1	0.93	0.93	0.70	0.80	0.57	0.40	0.43	0.82
Class: G2M	0.78	0.57	1.00	0.73	0.23	0.23	0.40	0.89
Class: S	0.93	0.67	0.57	0.62	0.20	0.11	0.17	0.75
Average	0.88	0.72	0.76	0.71	0.33	0.25	0.33	0.82
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.74	0.57	0.88
B = A w/ log median absolute value normalized								
Class: G1	0.73	0.67	0.40	0.50	0.57	0.23	0.34	0.57
Class: G2M	0.89	0.70	0.88	0.78	0.23	0.20	0.29	0.88
Class: S	0.64	0.23	0.43	0.30	0.20	0.09	0.37	0.54
Average	0.76	0.53	0.57	0.53	0.33	0.17	0.33	0.66
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.51	0.34	0.69
C = rank normalized								
Class: G1	0.73	0.80	0.80	0.80	0.57	0.46	0.57	0.77
Class: G2M	0.89	0.67	0.75	0.71	0.23	0.17	0.26	0.82
Class: S	0.89	0.50	0.43	0.46	0.20	0.09	0.17	0.66
Average	0.84	0.66	0.66	0.66	0.33	0.24	0.33	0.75
						Accuracy	Accuracy Lower CI	Accuracy Upper CI
						0.69	0.51	0.83

Table 29: External validation test set classification results within the RTL framework. Cells for training are from EMTAB2805, the test set cells are from GSE42268 filtered by top 'N' genes as previously described per normalization method. Multiple (25) 5-fold cross-validated runs per class label are used to determine the probability of each cell's association to each label; the highest probability was chosen as the final call. **A-D** Baseline RTL implementation and **E-H** classification with the RTL framework with the extended Alg4, Alg5, and Alg6 modules i.e., Alg4-Alg5-Alg6 V2.0.

4.2.6 Statistical comparison between to the two RTL versions:

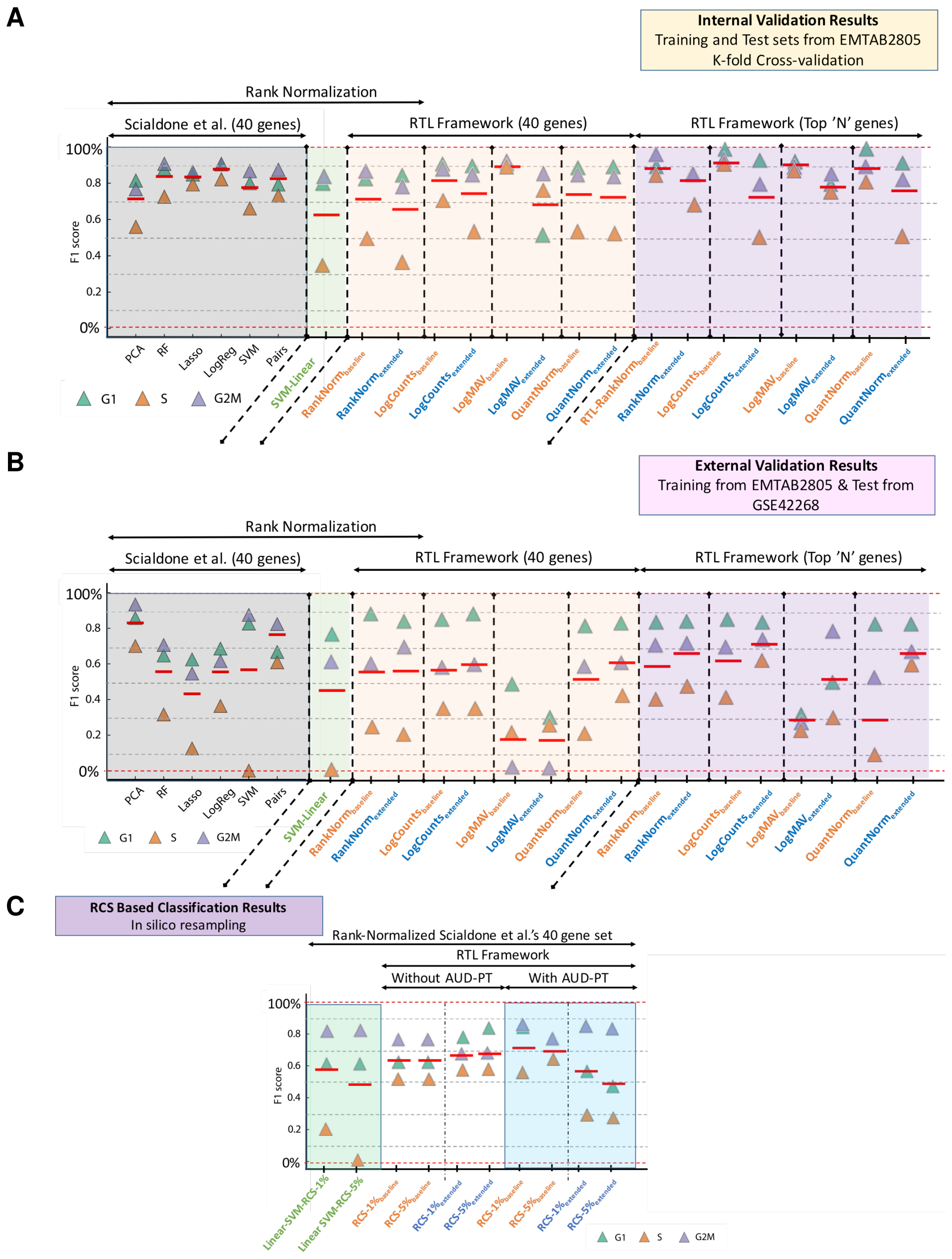


Figure 51: Benchmark comparison between the RTL framework and Scialdone et al. **A:** The internal validation results. **B:** The external validation results. **C:** The RCS classification results.

We report extensively the final classification statistics (accuracy, precision, recall, etc.), in the respective tables and figures in previous sections. As summary of our results compared to the published Scialdone et al. results, we have adopted their figure and imputed our results alongside it. It is difficult to assess significance between our results and theirs without acquiring their code and experimental parameterization. As a general approach, their mean result can be compared to the distribution of results we compute for our methods.

To quantitatively test the difference between any two models there are two potential ways to compute significance, aided by the fact that for each classification task, we have conducted multiple epochs. *Cell cycle allocation:* A) Use each epoch to compute a classification call and assess the distribution of each statistic of interest (e.g., $F1_{scores}$) between the models. B)

Sample the result of the previously ran epochs, randomly, multiple times, where each set is used to derive the probability of the examples/cells ub the test set. This latter approach is synergistic with how the final call derived in our summarized results in all the assessments, where the complete set of epochs are used to make a final call. In other words, because the former method cannot be used to derive probabilities for each cell being associated to each of the labels, the final classification call is different than the "N-repeated K-fold CV" approach we used to compute the final calls. Furthermore, the published results from Scialdone et al. is a multi-class combination of 1-vs-rest classification attempts. In fact for their 'Pairs' method, they found that their classifier performed better if the cell cycle allocation was done based on the G1 and G2M score where if a score of above 50% was not achieved for either, the cell was allocated to the S-phase. Combined, this support the use of the second approach, creating random samples of the already ran multiple epochs; this approach is feasible and avoids conducting a superfluous number of additional runs with different epoch sizes and computing their distribution. Two potential issues are first, how large should should the sampled sets be? and if $F1_{scores}$ is indeed an appropriate measure? For an equivalent comparison with Scialdone et al. with the "N-repeated K-fold CV" method used for classification with the RTL framework, the $F1_{scores}$ is the only option. In the context of RCS, the $F1_{scores}$ is a balance between precision and recall which is better than using either independently or other measure such as accuracy.

In the second approach discussed above, to obtain a range of possible classification calls using the already ran epochs (for each classification setting per cell cycle label), we sampled the random result of 2000 sets of 10-25 long calls [19]. From each set, a final classification call is made, as before by finding the probability of each cell being labeled to each of the cell cycle class labels. Another benefit of this approach is that it highlights the improvement in the final classification call when "N-repeated K-fold CV" is used as oppose to running a single run. For a statistical comparison of the $F1_{scores}$ distributions as the comparative quantifier between models, we employed the non-parametric Mann-Whitney-Wilcoxon two-sided rank-based test; the null hypothesis states that the $F1_{scores}$ have identical distributions; more specifically, the likelihood of a random sample taken from distribution A, to be greater or less than a randomly selected sample from distribution B. Using a generally accepted $\alpha = 0.05$, we reject the null hypothesis when p-values with smaller significance are computed i.e., higher p-values means there is no evidence that they are actually different.

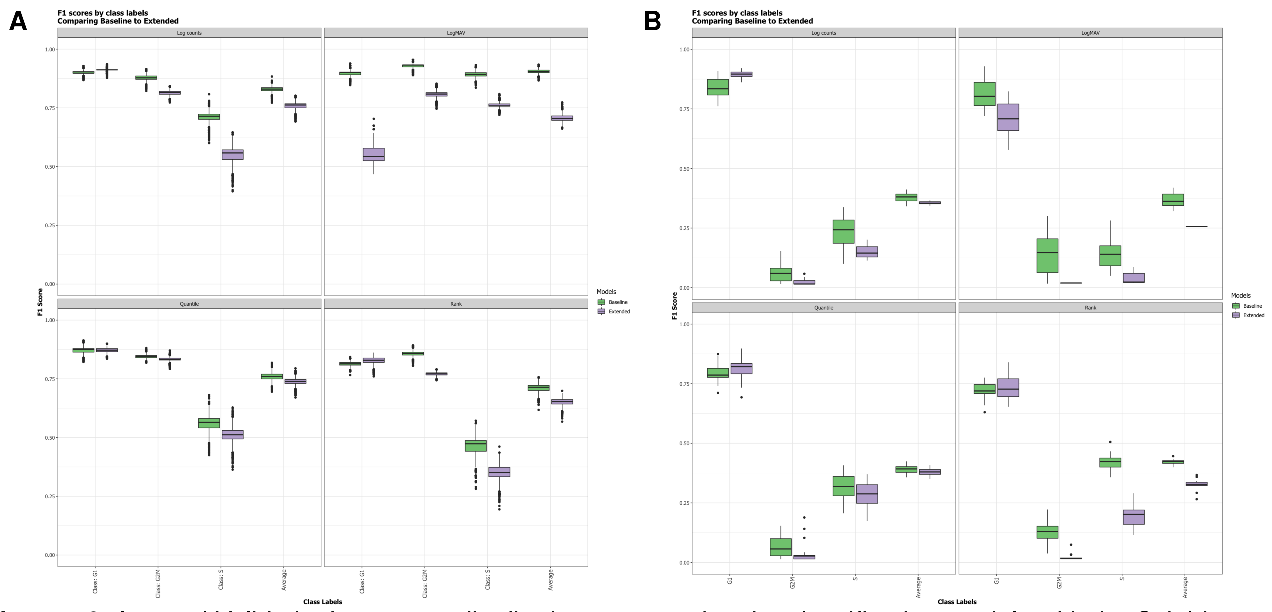


Figure 52: Internal Validation's $F1_{scores}$ distributions, comparing the classification models with the Scialdone 40 gene set. **A.** From the 25 epochs, 2000 random sets of 10-25 length are derived and the probabilistic final call is used to derive the classification statistics. **B.** From each epoch individually a final call is made per cell cycle label in a 1-vs rest.

[19]Smaller sets tested, e.g. 3-25 or 5-25, result in distributions that are skewed, suggesting a larger sets is needed; avoiding sampling bias

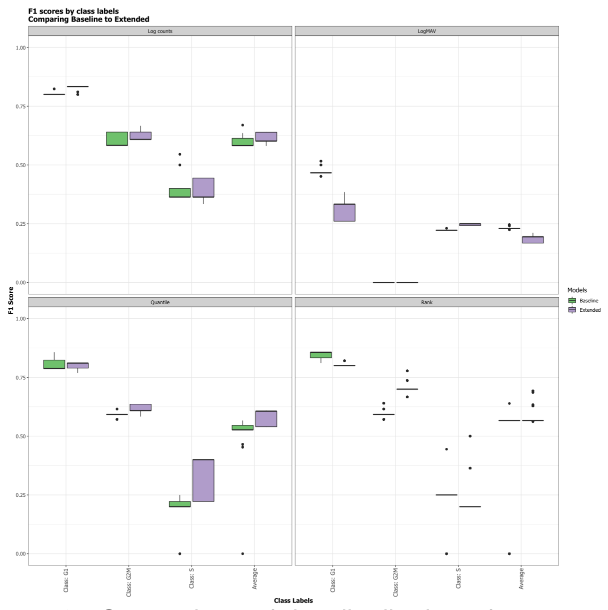
	logCount.Vs.LogMAV	logCount.Vs.QuantNorm	logCount.Vs.RankNorm	QuantNorm.Vs.RankNorm	QuantNorm.Vs.LogMAV	RankNorm.Vs.LogMAV
1 epoch per final call:						
Baseline version:						
G1	0.13258815	0.00090520	0.00000000	0.00000050	0.30829963	0.00000067
G2M	0.00041270	0.82335290	0.00006942	0.00010401	0.00044432	0.41506227
S	0.00004122	0.00009210	0.00000000	0.00000003	0.00000001	0.00000000
Average	0.06147057	0.01231150	0.00000000	0.00000005	0.00100128	0.00000001
Extended version:						
G1	0.00000000	0.00000001	0.00000000	0.00000252	0.00000067	0.21418479
G2M	0.62323598	0.73316097	0.29356047	0.89667844	0.49127770	0.06695384
S	0.00007018	0.00000000	0.00063250	0.00000106	0.00007292	0.00007292
Average	0.11157131	0.00000156	0.00009599	0.00000183	0.11334724	0.13211782
2000 sets of 10-25 epochs:						
Baseline version:						
G1	0.00214132	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
G2M	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
S	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Average	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Extended version:						
G1	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
G2M	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
S	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Average	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000

Table 30: Internal Validation's Significance p-values comparing the distributions of $F1_{scores}$ of specified classification models. Mann-Whitney-Wilcoxon two-sided rank-based test was used to obtain p-values for the final comparisons.

	LogMAV	LogCount	Quantile	Rank
1 epoch per final call:				
G1	0.00002539	0.00000263	0.04775766	0.52181148
G2M	0.00026421	0.00060789	0.02585825	0.00000059
S	0.00043130	0.00000086	0.04256153	0.00000000
Average	0.11069662	0.00001759	0.00745458	0.00000032
2000 sets of 10-25 epochs:				
G1	0.00000000	0.00000000	0.02720969	0.00000000
G2M	0.00000000	0.00000000	0.00000000	0.00000000
S	0.00000000	0.00000000	0.00000000	0.00000000
Average	0.00000000	0.00000000	0.00000000	0.00000000

Table 31: Internal Validation's Significance p-values comparing the distribution of $F1_{scores}$ between the baseline and extended versions of the RTL framework given the log-scaled counts or the normalization methods utilized. To compute the p-values, we utilized the non-parametric Mann-Whitney-Wilcoxon two-sided rank-based test.

A



B

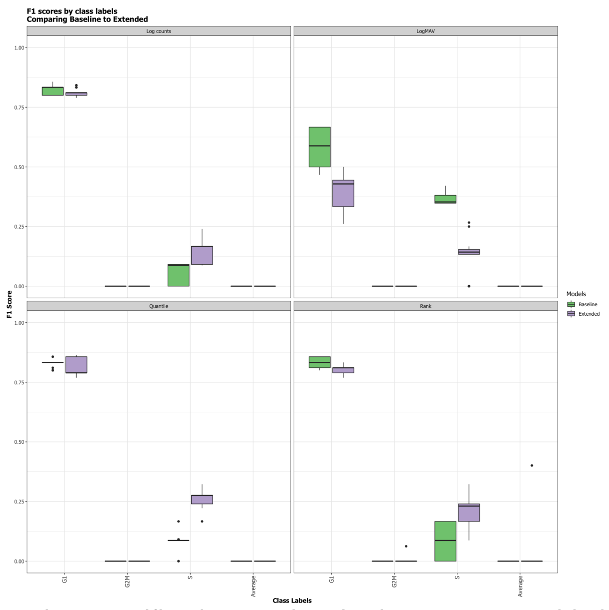


Figure 53: Comparison of the distribution of $F1_{scores}$ from the classification models in the external validation with the Scialdone 40 gene set for all the normalization methods tested. **A.** From the 25 epochs, 2000 sets of 5-25 length are derived and the probabilistic final call is used to derived the classification statistics. **B.** From each epoch individually a final call is made per cell cycle label in a 1-vs rest.

	logCount.Vs.LogMAV	logCount.Vs.QuantNorm	logCount.Vs.RankNorm	QuantNorm.Vs.RankNorm	QuantNorm.Vs.LogMAV	RankNorm.Vs.LogMAV
1 epoch per final call:						
Baseline version:						
G1	0.00000000	0.51004489	0.17896504	0.67461041	0.00000000	0.00000000
G2M	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
S	0.00000000	0.96606401	0.31419183	0.65971374	0.00000000	0.00000000
Average	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Extended version:						
G1	0.00000000	0.85840483	0.07441506	0.42408557	0.00000000	0.00000000
G2M	0.00000000	0.00000000	0.33705521	0.33705521	0.00000000	0.33705521
S	0.06353135	0.00000002	0.00119528	0.00014202	0.00000002	0.00002254
Average	0.00000000	0.00000000	0.33705521	0.33705521	0.00000000	0.33705521
2000 sets of 10-25 epochs:						
Baseline version:						
G1	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
G2M	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
S	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Average	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Extended version:						
G1	0.00000000	0.00000000	0.00000000	0.00126946	0.00000000	0.00000000
G2M	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
S	0.00000000	0.00000000	0.00000000	0.00000000	0.00021129	0.00000000
Average	0.00000000	0.00000000	0.00000000	0.00003214	0.00000000	0.00000000

Table 32: External Validation's Significance p-values comparing the distributions of $F1_{scores}$ of specified classification models. Mann-Whitney-Wilcoxon two-sided rank-based test was used to obtain p-values for the final comparisons.

	LogMAV	LogCount	Quantile	Rank
1 epoch per final call:				
G1	0.00000001	0.14046261	0.48969535	0.00006656
G2M	0.00000000	0.00000000	0.00000000	0.33705521
S	0.00000000	0.00000256	0.00000000	0.00000287
Average	0.00000000	0.00000000	0.00000000	0.33705521
2000 sets of 10-25 epochs:				
G1	0.00000000	0.00000000	0.00000000	0.00000000
G2M	0.00000000	0.00000000	0.00000000	0.00000000
S	0.00000000	0.00023838	0.00000000	0.00000000
Average	0.00000000	0.00000000	0.00000000	0.00000000

Table 33: External Validation's Significance p-values comparing the distribution of $F1_{scores}$ between the baseline and extended versions of the RTL framework given the log-scaled counts or the normalization methods utilized. To compute the p-values, we utilized the non-parametric Mann-Whitney-Wilcoxon two-sided rank-based test.

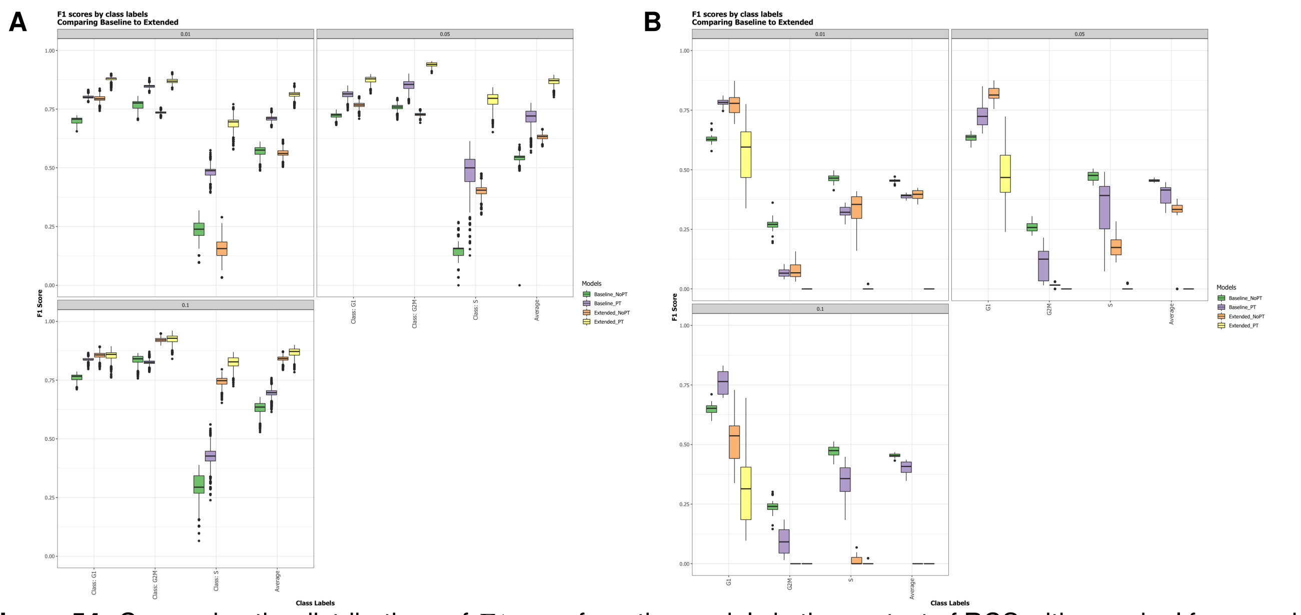


Figure 54: Comparing the distributions of $F1_{scores}$ from the models in the context of RCS with marginal frequencies tested (1%, 5%, and 10%). **A.** From the 25 epochs, 2000 sets of 10-25 length are derived and the probabilistic final call is used to derived the classification statistics. **B.** From each epoch individually a final call is made per cell cycle label in a 1-vs rest.

	RCS_freq_0.01_baseline	RCS_freq_0.01_extended	RCS_freq_0.05_baseline	RCS_freq_0.05_extended	RCS_freq_0.1_baseline	RCS_freq_0.1_extended
1 epoch per final call:						
G1	0.00000000	0.00000005	0.00000000	0.00000000	0.00000000	0.00019147
G2M	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
S	0.00000000	0.00598381	0.00000111	0.00019473	0.00000000	0.05469942
Average	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
2000 sets of 10-25 epochs:						
G1	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.74813879
G2M	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
S	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Average	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000

Table 34: Significance p-values comparing with and without AUD-PT; the distributions of $F1_{scores}$ from the models in the context RCS with marginal frequencies tested (1%, 5%, and 10%). We utilized the non-parametric Mann-Whitney-Wilcoxon two-sided rank-based test to obtain p-values for the final comparisons.

4.2.7 Supplemental Results

A representative example of directly comparing and visualizing the classification results obtained by two separate inferences by the RTL framework is shown in Figure 55. In these figures, if the results were identical, they should form on the diagonal line shown. A skew (above or below the diagonal) is indicative of higher probability given to those cells by that model. For example, in Figure 55A, the internal validation results of the baseline to the extended RTL versions on the log-scaled counts are comparatively shown. The baseline version (x-axis) results demonstrate a skew i.e., higher probability was given to the S and G2M cells when classifying G1 vs rests. Whereas in Figure 55B, when classifying G2M vs rest cells, the extended model infers higher probability to the S and some of the G1 phase cells. Comparing populations and population based measures (e.g., mean, sd, etc) can helpful in many contexts to benchmark and evaluate models, however, deeper forensics is sometimes needed. This is the intention behind Figure 55, which aids comparing the prediction differences on a cell by cell level, i.e., how did a pair of models do on the same cell?

To find the top cell cycle genes, that their expression was highly correlated with the predicted outcome (the class labels) i.e., which genes were most influential in the predictive outcome of each model, we computed the Spearman's rank-based correlation coefficient for all cell cycle genes in the set, relative to the predicted probability of each class label. In Figure 56D, the distribution of the correlation coefficients for each label is plotted in the context of log-scaled expression values. For each label, genes with $|cor| >= 0.5$ were filtered; some of tested classification results had 0 genes that match this criterion, but if higher than 9 were found, the top 9 were filtered for visualization purposes. Figure 56 visualizes the result of the baseline RTL framework on the log-scaled counts with the Scialdone et al. 40 gene set. For an equivalent comparison Figure 57 is also on the log-scaled counts, but with the Top'N' genes identified. This comparison is done to find highly correlative genes outside of the Scialdone et al's 40 genes, as part of the top 'N' genes we have identified (described in an earlier section). Finally, in figure 58, the baseline RTL framework on the rank-normalized data with the Scialdone et al. 40 gene set is visualized. This comparison is shown to highlight the effect of normalization on the classification at this cell-by-cell scope.

Besides differences in expression for each pertinent highly correlative gene, in parts E and F of each, we compare the mean and variance of the 405 cell cycle expression by plotting them against the predicted probability of each cell cycle label respectively. Interestingly, the G1 cells on average have lower expression compared to the G2M cells. The S phase cells show a gradient. Potentially, this is one evidence supporting the difficulty of classifying the S-phase cells due to their higher biological variance.

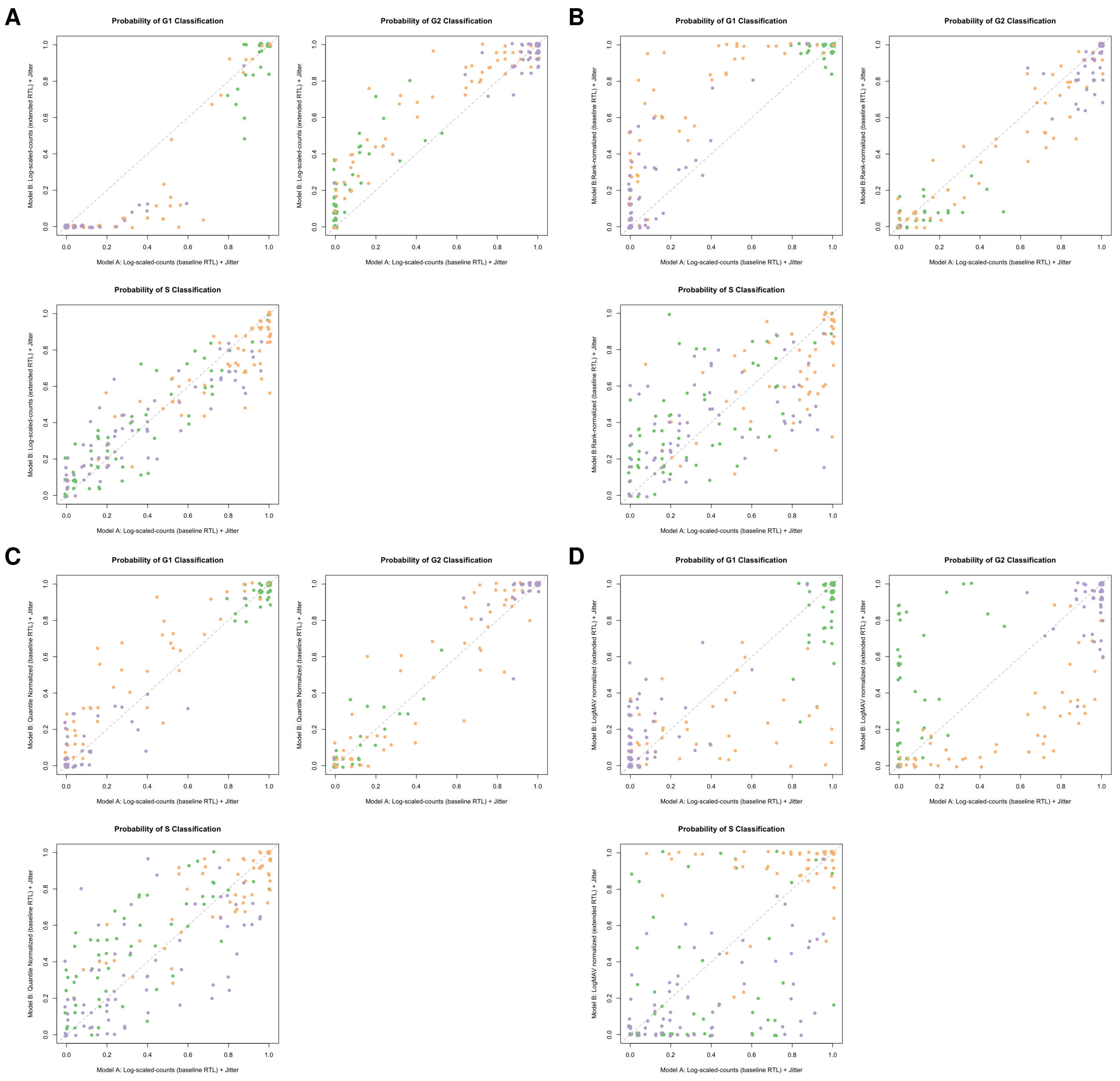


Figure 55: Direct comparison of pairs of predicted probability outcomes by different classification models conducted with the RTL framework. All utilize the Scialdone et al.'s 40-gene set. **A** Log-scaled counts baseline vs. log-scaled counts extended. **B** log-scaled counts baseline vs. rank-normalized baseline.**C** log-scaled counts baseline vs. quantile normalize baseline. **text** log-scaled counts vs. LogMAV normalized. The colors are the true label: green = G1, plum = G2M, and orange = S.

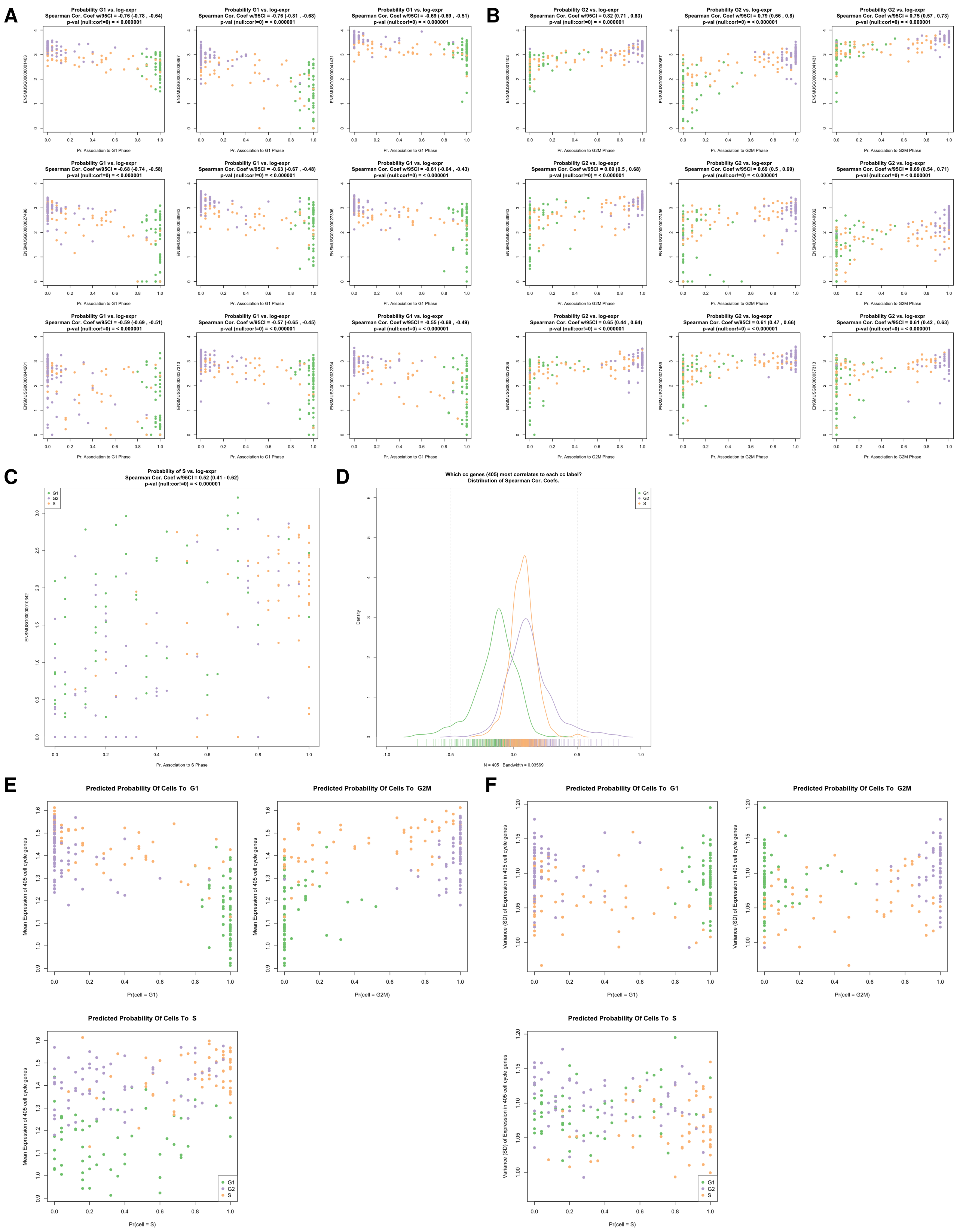


Figure 56: Top cell cycle genes that their *log-scaled expression* was highly correlated with the predicted outcome i.e., the class label. The classification model here is the baseline RTL framework using the log-scaled-counts with the **Scialdone et. al's 40 gene set**. The Spearman's rank-based correlation coefficient was computed for all genes in the set, relative to the predicted probability of each class label. **A** the top 9 correlates for the G1 label are shown. **B** the top 9 correlates for the G2M label are shown. **C** Relative to the S-phase prediction, only 1 gene had a $|cor| >= 0.5$. **D** The distribution of Spearman's Correlation Coefficient values computed. **E** Mean Expression of the 405 cell cycle genes relative to the predicted class label. **F** Standard Deviation of the expression of the 405 cell cycle genes relative to the predicted class label. The colors are the true label: green = G1, plum = G2M, and orange = S.

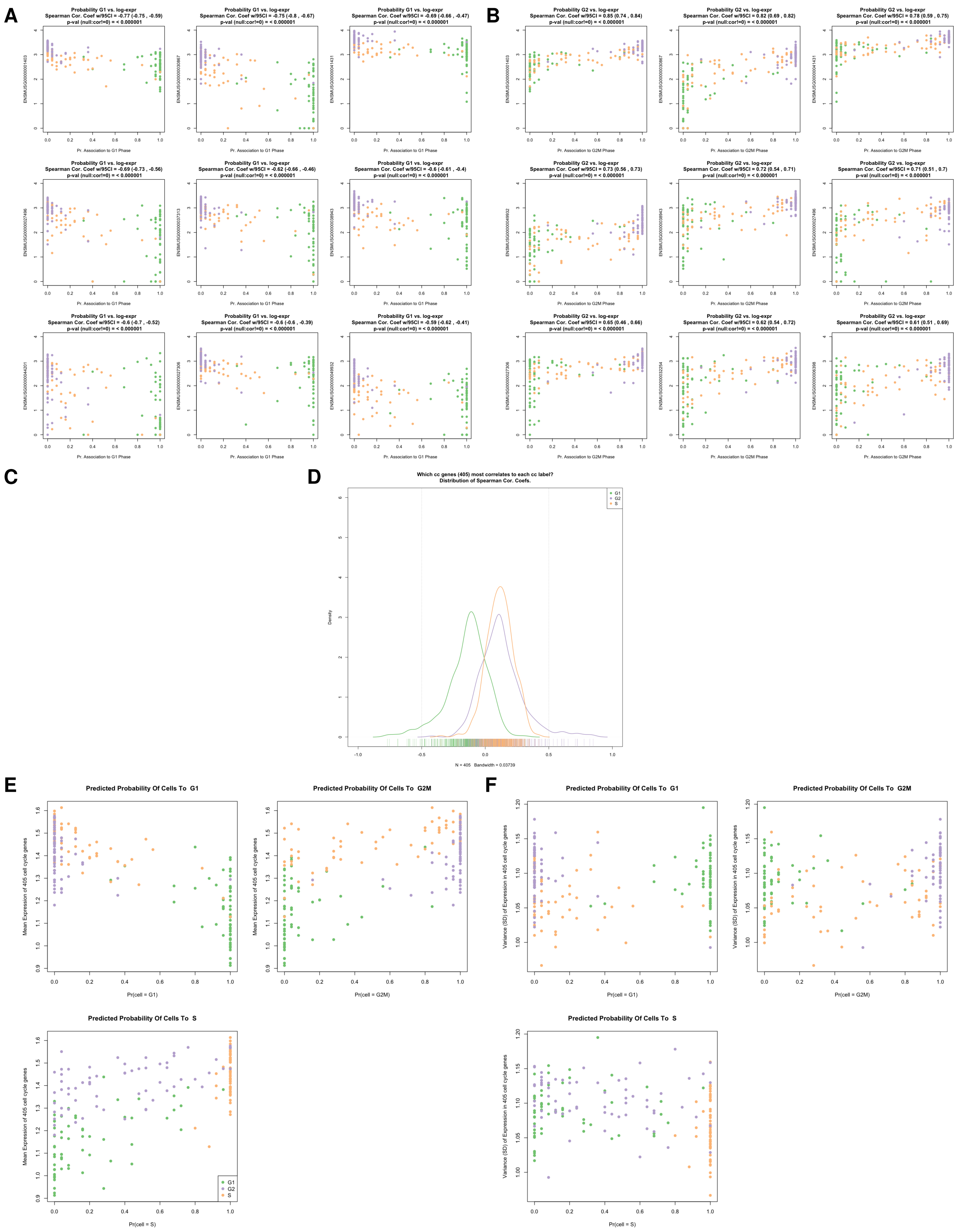


Figure 57: Top cell cycle genes that their *log-scaled expression* was highly correlated with the predicted outcome i.e., the class label. The classification model here is the baseline RTL framework using the log-scaled-counts with the respective **Top'N' gene set**. The Spearman's rank-based correlation coefficient was computed for all genes in the set, relative to the predicted probability of each class label. **A** the top 9 correlates for the G1 label are shown. **B** the top 9 correlates for the G2M label are shown. **C** Relative to the S-phase prediction, 0 genes had a $|cor| >= 0.5$. **D** The distribution of Spearman's Correlation Coefficient values computed. **E** Mean Expression of the 405 cell cycle genes relative to the predicted class label. **F** Standard Deviation of the expression of the 405 cell cycle genes relative to the predicted class label. The colors are the true label: green = G1, plum = G2M, and orange = S.

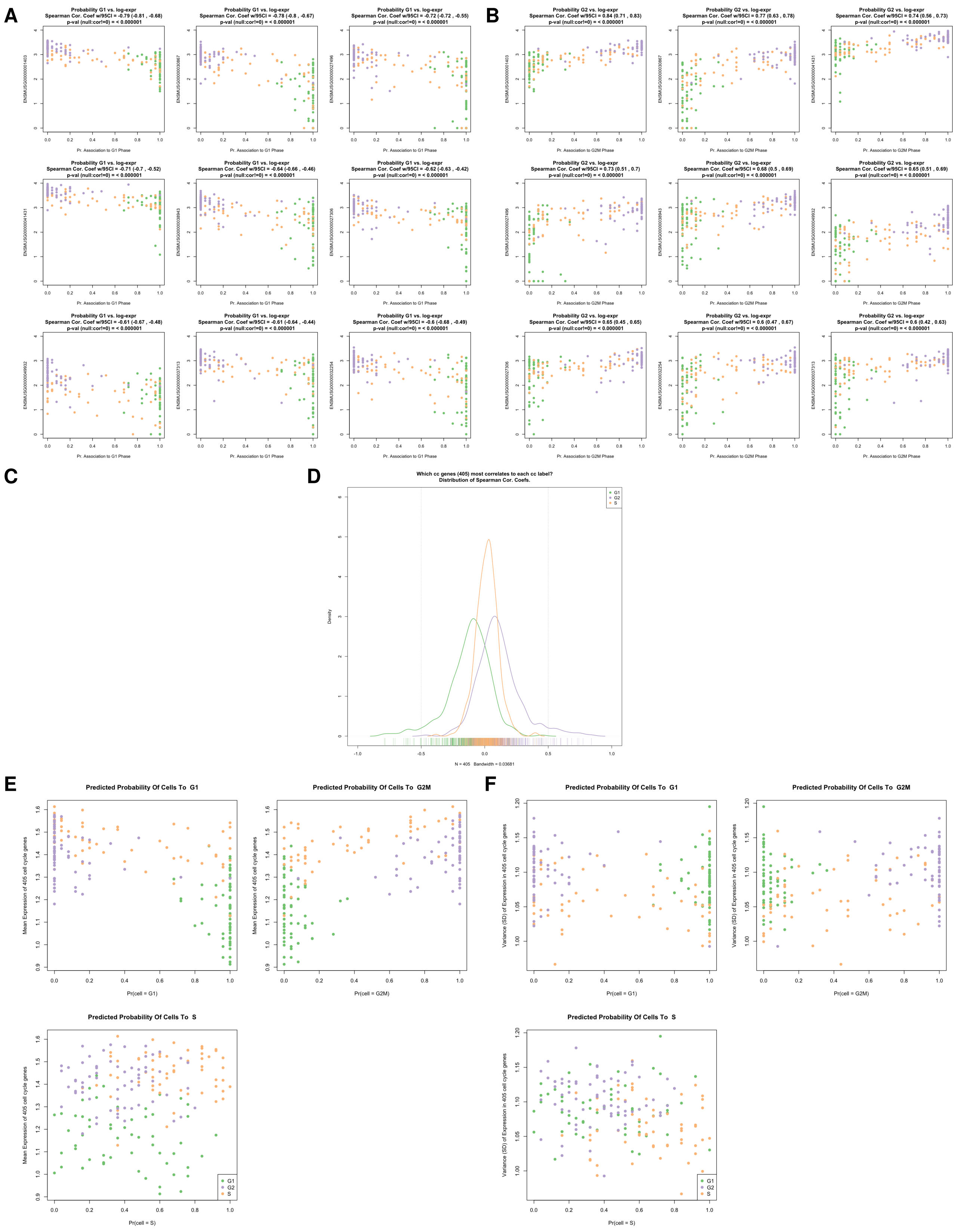


Figure 58: Top cell cycle genes that their *rank-normalized expression* was highly correlated with the predicted outcome i.e., the class label. The classification model here is the baseline RTL framework using the rank-normalized data with the **Scialdone et. al's 40 gene set**. The Spearman's rank-based correlation coefficient was computed for all genes in the set, relative to the predicted probability of each class label. **A** the top 9 correlates for the G1 label are shown. **B** the top 9 correlates for the G2M label are shown. **C** Relative to the S-phase prediction,0 genes had a $|cor| >= 0.5$. **D** The distribution of Spearman's Correlation Coefficient values computed. **E** Mean Expression of the 405 cell cycle genes relative to the predicted class label. **F** Standard Deviation of the expression of the 405 cell cycle genes relative to the predicted class label. The colors are the true label: green = G1, plum = G2M, and orange = S.

4.3 Discussion

Transcriptional analysis of single cells is nowadays a major scientific focus. At a high-level, being able to use the genomic and/or epigenomic information to categorize/subtype the cells and consequently study them in the context of a specific hypothesis is a major investigative need. When quantifying hundreds to thousands of features (genes/transcripts/markers) per cell, robust computational methods need to be developed and tested; a common task for machine learning (statistical) methods. For example, how to separate immune cells from tumor cells from a patient's primary biopsy, sampled as a scRNASeq experiment? Commonly unsupervised methods identify clusters, which then can then be interrogated and characterized as immune v.s. tumor cells. However, what if the question is to identify cells of a specific signature (perhaps pertaining to a specific phenotype) across multiple donor samples obtained from several scRNASeq datasets produced by multiple research centers? In this context, commonly, supervised methods are used to train on a gold standard and then to infer on the patient samples. An obvious problem is that the acquired data is confounded by technical sources of variation across the centers, batches, and instrumentation. In fact, supervised methods are prone to over-fitting to a specific dataset because of the noise introduced by such confounding factors. And the clusters produced by unsupervised methods may not be biologically meaningful. That is why usually specific normalization procedure(s) are chosen to remove unwanted technical and biological noise. Additional task-specific normalization may also be necessary to be able to perform a specific analysis (such as classification) across multiple datasets. However, such normalization methods can introduce bias and affect the reproducibility of the results. That is why asides from supervised unsupervised methods, we have investigated transfer learning as a potential approach to potentially overcome the need for task-specific normalization.

Transfer learning (TL), is a large sub-domain of machine learning in literature. There are several different applied branches of TL (reviewed in [7], [6], and [104]) and in this research we focus on the transductive TL branch, as we aim to classify transcriptomic signatures (similar classification task in analogous feature spaces) across scRNASeq datasets from diverging sources (i.e., different input feature distributions and marginal frequencies). The developed RTL framework is based on a previous application, classifying immunophenotypes in flow cytometry data. However, we have developed a major overhaul of three of the algorithms which improve the generalizability of the classification. Additionally, we have introduced the area under the density-curve (AUD) parameter transfer (PT) feature to improve the precision and specificity of calls in the context of rare cellular subsets (RCS). To evaluate the RTL framework (both versions with and without PT) we conducted parallel classification runs to the six supervised methods published by Scialdone et al. [89]. Specifically, using the $F1_{score}$, we found equivalent or higher results compared to Scialdone et al.'s top classifiers, using the same 40 genes; however, we showed the performance can be confounded by the normalization scheme we used.

The effects of the normalization scheme have a major impact on the input feature space. We computed the 2D principal component space on the first and second components to visualize these difference (Figure 39, 41, 40, and 42). First, in the log-transformed counts, we observe that the two datasets cluster separately, although the relative orientation of the distribution of the cell cycle labels across PC1 is similar. Quantile and rank based normalization result in similar distributions and the two datasets overlap in a single stretched cluster, where the clusters observed belong to the cell cycle labels; albeit, the clusters are not completely homogeneous. Finally, the logMAV normalization causes a major compression of the data points in PC-space, of both datasets; relative to PC1, the three cell cycle clusters are still apparent. Based on the PCA outcome, the rank and quantile normalization schemes are the most ideal for machine learning i.e., minimizing unwanted variance such that the projected PC1 and PC2 no longer show two clusters based on data origin as in the log-transformed counts. In fact, as mentioned earlier, the RTL framework can generalize the classification across two datasets from divergent sources directly on the log-transformed gene counts. Other domains of machine learning such as supervised and unsupervised methods, however, can be dramatically affected. For example, the consequence of these normalization methods is shown using hierarchical unsupervised clustering relative to the cell cycle labels as heatmaps in Figure 43. Next, we observe the effects of a supervised method in Figure 45. Specifically, using an optimally tuned linear SVM classifier, we identify the top 'N' genes for each cell cycle label and found that not only the number but the set of genes changes relative to the normalization (Figure 44). Because of the strong effects that such diverging normalization schemes cause, the selection of the right method is expected to demonstrate the enhanced separation of the classes. Thus perhaps an ensemble classification scheme with varied normalizations may also compute a final improved inference.

In the machine learning application demonstrated in this manuscript, in the context of internal validation with the Scialdone 40-gene set, the highest average $F1_{score}$ (0.91) was obtained with the logMAV normalization followed by the log-scaled counts (0.83) using the baseline version of the RTL framework. When using the top 'N' gene sets, the highest average

$F1_{score}$ (0.94) was achieved on the log-scaled counts with the baseline version (Table 28). However, in the context of generalizability (external validation), we found the highest average $F1_{score}$ was achieved on the rank normalized data (0.61) followed by the log-scaled counts (0.6) using the Scialdone 40-gene set with the extended version of the RTL framework. With the alternate top 'N' gene sets, the highest average $F1_{score}$ (0.71) was achieved on the log-transformed counts using the extended version. Combined, this illustrates that: A) the extended version outperforms the baseline in the context of generalizing across datasets. B) the task-specific normalization method chosen does affect the classification performance, however, within the RTL framework, it may prove to be superfluous as classification can be performed on the log-transformed counts. Another important consideration is the feature set used, which in our experiments generally improved the classification performance compared to the Scialdone 40-gene set. This finding combined with the effects we observe normalization has on the feature space and thus the classification statistics, suggests the limitation of our external validation is potentially confounded by the pre-processing and sequencing instrumentation differences between the two datasets. Additionally, the linear classifier used can confound the classification, but for our purpose, linearly classifying cell cycle labels proves to be sufficient, as we achieved near perfect average classification accuracies. However, it is possible to utilize the inner-product of kernels (such as Gaussian, quadratic, etc.) to overcome this limitation within the RTL framework. The caveat is the computational cost/complexity associated with estimating the kernels for the high-dimensional genomic feature spaces.

Overall, we find similar to Scialdone et al., that the S-phase cells are the most difficult to classify. The root of the low precision and recall, specific to the S-phase cells is associated with at least two criteria, as also described in [89]. First, the resolution of capturing each of the 3 class labels with flow cytometry is different, and lowest is associated with the S-phase 'phenotype'. Second, the transcriptional signature of the S-phase cells is the also the least sensitive; base on previous classification attempts by Scialdone et al. and the results showed herein. However, normalization can change the sensitivity with which each of the labels is classified. This is also evident in our external validation results (Table 21). This limitation can partially be explained by the overlapping clusters of the S-phase and G2M-phase cells in the principal component 2D space (Figure 39).

If a signature of interest belongs to a small number of cells relative to the total cells sampled, it can be considered a rare cellular subset (RCS); the actual frequency depends on the context, from stem cells to the heterogeneous sub-populations of cancer cells and even the dynamic repertoire of specific immune cells during an infection that expands from very small seeding populations. Experimentally, there may be several ways to enrich for such populations, but the caveat is the mechanical, chemical, and experimental conditions can confound the transcriptional profiles of these cells. We tested RCS marginal frequency of up to 1% of the total cells mainly due to the statistical considerations in the context of our sampling method. Outside of the biomedical literature, there are many published works on rare-event detection; generally, however, training and inference in such a context require unique considerations as most commonly used supervised methods don't perform well. The large class-imbalance in rare-event classification especially in high-dimensional data presents several key challenges. As the performance of a classifier generally depends on the distribution of the underlying data, the within-class variance and the mean difference between any two classes affect the classifier's performance. Another root cause of poor classification performance is the lack of data, especially for the RCS of interest. That is why in our 5-fold cross-validation runs, at each K, the entire positive class labeled cells are classified; with multiple epochs to obtain the probability of association to the class label. In our findings, where each cell type is, in fact, a separate signature which was sampled to produce the desired marginal frequency the classification statistics also depends on the distributions of the underlying data. The highest $F1_{score}$ in the context of RCS classification for both versions belongs to the G2M-phase cells. As we demonstrated herein, it is possible to classify rare events/RCS (from 10% to 1% of total events) within the RTL framework fairly well in this manner. However, for using AUD-PT to improve the classification, we adhere to the assumption that the marginal frequency of the positive class label is similar between the train and test sets.

We conclude that the publicly available, robust and reproducible RTL framework described herein facilitates the classification transcriptomic signatures across scRNASeq data sources. However, the existence/development of a training set is assumed. Exploratory analysis is needed to determine optimal normalization relative to the class labels; perhaps instead of a single method, using an Ensemble classification with a few different normalization methods may improve the final call. As the RTL framework is intended for broad and open-source application to enhance the discovery and screening in biomedicine, in addition to scRNASeq data, mass or fluorescent flow cytometry data can also be classified with the RTL framework, but this will require alternate pre-processing prior to classification. Finally, we believe the next developmental direction for the RTL framework needs three major considerations: first, as the datasets (FC and scRNASeq) are getting larger (more events from more samples with more feature measure per cell) moving to cloud-based (multi-CPU) environment is needed to scale the framework. Second, as described earlier, many classification tasks may not be linearly separable, thus the evaluation and testing of the proposed utilization of the inner-product of kernels also remain as future work. Finally, incorporate a tree-based or network based approach to classifying different cellular subsets within the same dataset.

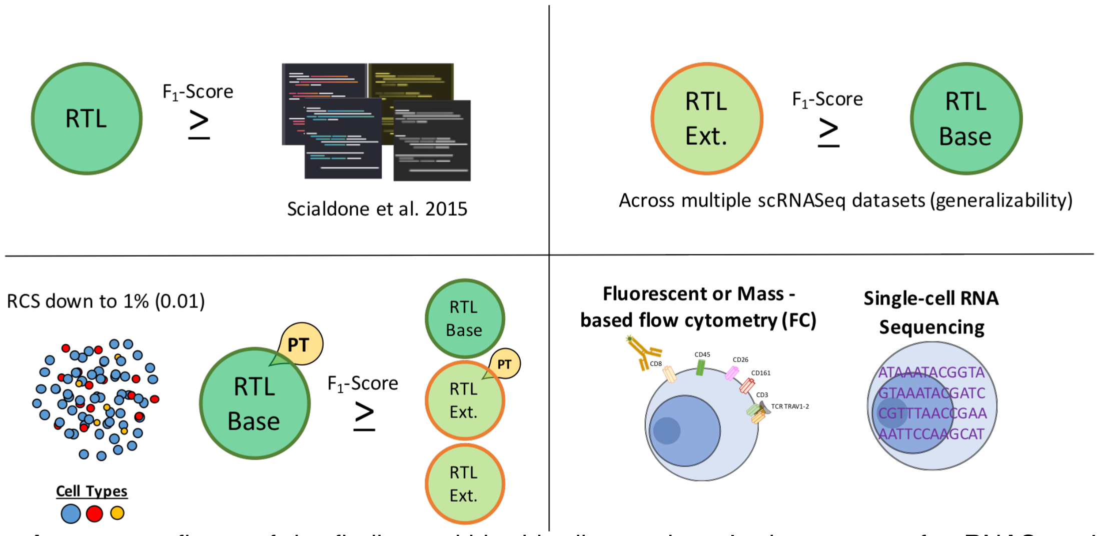


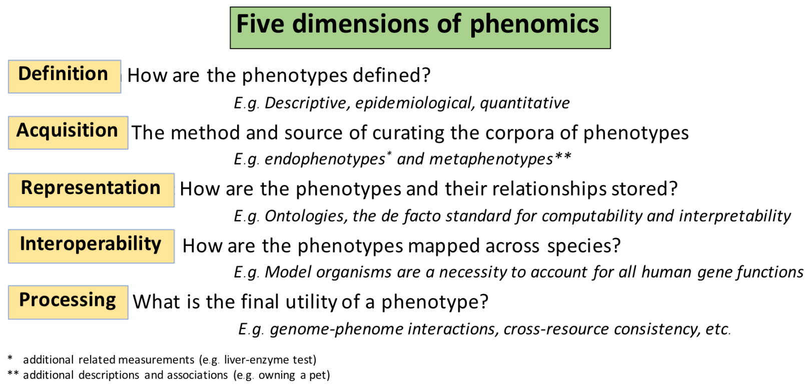
Figure 59: A summary figure of the findings within this dissertation. In the context of scRNASeq classification found Equivalent or higher classification (F1) compared to previous published results of Scialdone et al. (Internal validation K-fold CV runs); Normalization and feature selection are key. The extended version demonstrates improvement in generalizability across the scRNASeq data, relative to the same F1 measure. Rare cellular subsets (with *in silico* resampling) were classified successfully with frequencies as low as 0.01; again, the signature strength (effect size) is key. AUD-based parameter transfer (PT) improves the precision (and thus F1) of classifying RCS. Finally we have also evaluated the RTL framework successfully with FC data compared to manual gating.

5 Supporting material

5.1 Phenotyping: Phenotypes, Phenomes, and Phenomics

As an important, but a tangential supplement to our research as wel as its future directions, in this section we focus on connecting the concept of a critical (cellular) phenotype/signature to other -omics-based data to derive at potentially new emergent properties and relationships.

Briefly, to the quantitative biologist, a phenotype is a descriptive tool; like all tools, its value and utility is proportional to its validity and reliability (i.e., quality). The main goal of phenotyping is to be able to categorize and compare biology by evaluating the relationships between the genome, phenome, and the environment. Phenotyping has significant utility in the clinic and in evolutionary, developmental, and biomedical research. This high value has propelled the development of resources for phenotypic associations with genotypes, whole-genome/exome associations, rare diseases, Mendelian or complex disease, cancer, and pharmacology; for humans [105] and cross-species interoperation with model organisms [106].



Five dimensions of phenomics

Definition How are the phenotypes defined?
E.g. Descriptive, epidemiological, quantitative

Acquisition The method and source of curating the corpora of phenotypes
E.g. endophenotypes and metaphenotypes***

Representation How are the phenotypes and their relationships stored?
E.g. Ontologies, the de facto standard for computability and interpretability

Interoperability How are the phenotypes mapped across species?
E.g. Model organisms are a necessity to account for all human gene functions

Processing What is the final utility of a phenotype?
E.g. genome-phenome interactions, cross-resource consistency, etc.

* additional related measurements (e.g. liver-enzyme test)
** additional descriptions and associations (e.g. owning a pet)

Figure 60: Five major dimensions of phenomics adapted from [11]. Definition, Acquisition, Representation, Interoperability, and Processing.

To explore the world of phenotyping, Collier et al. have defined a four-dimensional framework to comprehensively cover phenotypic descriptions computationally: Representation, acquisition, processing, and interoperability [107]. In a later review, Oellrich et al. further evaluate the current status of phenotyping and associated methods with regards to these dimensions [108]. These authors define representation as a computable format to store and relate phenotypes. Acquisition, is defined as the action of capturing phenotypes. Interoperability is defined as concordance across all phenotypes, across tiers of complexity within an organism and across species. Finally, processing is defined as the analysis and the resulting knowledge discovery. However, to acquire, represent, interoperate, and process a phenotype, an a priori definition of that phenotype is implied. Definition of phenotypes is an integral and complex challenge in its own right. Therefore, we propose the definition of a phenotype to be considered the fifth dimension of phenomics.

5.1.1 Phenotypes' Definition

Brown et al. visualize the definition of a phenotype as a three-dimensional matrix of genetic context, environmental features, and multiple tests [109]. However, there is great heterogeneity in how phenotypes are defined and utilized in the literature (scope and aims). This is further confounded by experimental complexity which includes setting up proper controls that ultimately define the differentiating phenotype of interest. Phenotypes can have a range of expression (i.e., severity), which may only be observed in a specific window of time (temporality) or in a given environment. Some phenotypes can be defined by single traits. Yet some phenotypes, may have overlapping etiologies/pathways (e.g., obesity) [110] or be associated with several compound phenotypes (e.g. obstructed airflow [111]). Finally, mutations in the same gene can lead to multiple phenotypes, known as pleiotropy. Brown et al. advocate for the realization that human diseases are compound gatherings of observations; thus a disease label is a 'probabilistic inference' [109]. This modern view of evidence-based medicine means bioinformaticians can conduct cross-species association of phenotypic features [112] to enable the new discoveries in phenomics, such as novel biomarkers and targeted methods towards precision medicine and unraveling our understanding of functional genomics.

Sub-phenotypes are important both in research and the clinic [20]. One way to identify subphenotypes is cluster analysis. For example, we define a 2-dimensional matrix of the phenotype classes (or individuals) in the rows and the quantitative (or qualitative features) as columns. Several unsupervised algorithms, such as agglomerative hierarchical clustering, can uncover the structure of this matrix, however, the major limitation is the dependence on a similarity metric [114] and more importantly, the clusters are not guaranteed to return biologically meaningful clusters. When such unsupervised approaches are utilized, technical variance (e.g., batch), as well as covariates such as ethnicity, gender, and age, are major sources of variance; many times the identified discriminating signatures are in fact explained by such unwanted variance.

The **severity** of a phenotype's expression is a complex function of the interplay of the genetics and epigenetics of an organism, triggered by environmental stimuli [115], through a tightly regulated but still 'noisy' transcription process [116]. One method to define levels of severity is thresholding; used to transition between continuous and discrete phenotypic data [117]. In a clinical context, severity i.e., the burden of illness, is difficult to assess. As an example in the clinical realm, Horn and Horn propose using a composite index from the 'mode' of seven overlapping features, each composed of a four-level index [118]. One downside to such qualitative approaches is the loss of statistical power due to the binning. On the other hand, in quantitative methods, the selection of the control group significantly changes the interpretation. Consider that many phenotypes are defined on the bases of comparing healthy vs. disease groups. Such an experimental approach is limiting as it lacks precision in severity (i.e., small but significant vs. large differences) as well as lacking a systems-level perspective that explains the observed variance in the phenotypic expression of 'non-diseased' i.e., healthy vs. 'disease' traits.

Temporality in phenotypes can be considered in two types: phenotypes that are observed within a window (either due to a medical intervention or environmental perturbations) and phenotypes related to aging [119]. Both forms of temporality are intuitive and ubiquitous concepts in healthcare and research, thus their utility in practice with electronic health records and precision medicine with 'big data' challenges are under investigation. For example, Che et al. demonstrate the potential for using electronic health records to capture meaningful temporal patterns that describe phenotypes such as physiology [120]. The ability to capture temporal data for quantitative analysis is a key feature of utilizing of Electronic health records (EHRs) for phenome-wide research. In the context of immunophenotyping, at the scope of an organism/patient, clinicians measure the pathologically relevant phenotypes for utility in diagnosis, treatment, and prognosis. At the scope of the immune system, the dynamic changes that occur during an infection, for example, can be quantified by transcriptional changes (thus likely phenotypic changes), at various time points; dictating the function, expansion, and taxis of multiple cellular components [121].

The **environment** is a powerful force in the relationships originally proposed by Watson and Crick as 'the central dogma' of molecular biology and physiology. On the evolutionary scale, the environment is the driver of natural selection. On a cellular level, temporal environmental cues or stimuli can cause specific responses [122]. Furthermore, the environment can trigger permanent effects on the development of an organism [123] and [124]. Such developmental consequence can also be epigenetically imprinted [125] and transferred to the next generations [115].

Phenotypic **plasticity** is defined as the ability to change or modify phenotypic trait(s) in response to stimuli or cues; internal and external. Fienberg suggests that this plasticity is an important property, such that its disruption is a 'common theme' of disease, which at the cellular level is etiologically tied with the epigenome. Fienberg defines two etiological classes of such monogenic epigenetic diseases that reduce phenotypic plasticity: A) affected genes (i.e., epigenetic regulated such as imprinting) B) affected epigenome and machinery [125]. In multi-cellular organisms, the epigenome drives cellular differentiation, transcriptional and functional regulation; in fact, cells from the same tissues tend to cluster together in a recent analysis of 111 human epigenomes [126]. This not only highlights that the epigenome regulation can produce a

[20] For example, the diagnosis of asthma is too broad and fails to properly identify and treat a severe subtype of asthma, 'refractory/severe asthma'. This subtype specifically diagnoses about 10% of asthma patients; defined by at least one major and two minor criteria defined by the American thoracic society workshop consensus for the definition of refractory asthma [113]

plethora of phenotypes from the same ’code’, but also demonstrates the importance of epigenome studies across many tissues to unravel the pathological mechanisms of disease.

5.1.2 Phenotypes’ Acquisition

The primary source of phenotypic data is the archive of biomedical literature [127]. Manual curation has been the early gold standard; however, this method is costly and not efficiently scalable. On the other side of the spectrum, using computer and technology such as deep phenotyping (derived from deep [machine] learning) is an aggregation method that can explore the clinical data from the electronic health record; an indispensable criterion of precision medicine [128]. Such technological amendments will enable the automation needed to contribute to the acquisition of a comprehensive set of phenotypes (derived from the medical history, physical examination, quantitative clinical and laboratory tests, and various imaging/radiographic sources) [114]. However, phenotyping from the health record can be fraught with incompleteness, complexity, inaccuracy, and bias and thus quality evaluations such as sensitivity and specificity are needed to evaluate phenotypes acquired in this manner [129] as well as external experimental validation.

Tracy, in the context of clinical phenotyping proposes three goals for expanding the engagement of bioinformatics to improve the overall acquisition for the (near) future of deep phenotyping applications: The need for translational research (clinical outcomes), the need for higher statistical power in associations, and the need for incorporating the temporality of pathophysiology [130]. Such an expansion may seem superficially simple, however, it requires tightly designed and executed studies.

In the context of immunophenotyping, in recent years, the acquisition of the raw and analyzed phenotypic data has been facilitated by several key repositories that house FC and transcriptomics data. The limitation currently is that although there are several standards that the field uses, there are platform differences as well as inconsistent national/global standards in recording and providing metadata (related to the organism, experiment, etc). Furthermore, confounding factors such as biological factors (age, gender, medical history, etc) as well as technical variance need to be systematically and if possible automatically recorded for minimizing their effect in downstream processing. Without such standardization, it is inhibitive to apply âĂŞomics level bioinformatics.

5.1.3 Phenotypes’ Representation

Representation refers to how phenotypes are digitally saved and stored that optimizes their computability, and interoperability. Because phenotypic data are from diverse research domains and can be highly heterogeneous, there is a need to represent them in a standardized method; enabling computability and integration with other data types. Ontologies were formally introduced in the late 1990s [131], which were later adopted by biology within the context of gene ontologies (GO). This effort was aimed at developing and validating a structured vocabulary to describe molecular functions, cellular components, and biological processes [132]. With the success and adoption of GO, ontologies were used as a structured footing for biological research and phenotypes [133]. Initial attempts to capture detailed quantitative or qualitative data suggested using entity-quality (EQ) formalism. This had limitations regarding the stored information and their relationships, so in 2004, Gkoutos et al. proposed five classes of ontologies to describe mouse phenotypes: âĂIJorganism, entity, attribute, assay, and valueâĂİ [134]. Nowadays, the open biomedical ontologies (OBO) consortium supports hundreds of ontologies [135], such as the human phenotype ontology project [136], focused on human disease phenotypes. In the context of cellular phenotyping and immunophenotyping, the cell ontology (CO) is slowly being utilized and incorporated in software which is ubiquitously used by the community, however, there are limitations in naming conversion and completeness.

5.1.4 Phenotypes’ Interoperability

Interoperability is defined as the ability to effectively combine and interrogate phenotypes across species; especially useful when model organisms are utilized. There are several approaches to interoperability, such as construct validity, orthologue identification, or direct mapping using ontologies which are described in detail by Robinson et al. in the translational research context [137]. Briefly, to objectively discover phenotypic equivalences across spices, it is intuitive to identify orthologous genes (or determinant mutations of the genes in human health and disease) in a model species postulated upon the functional similarity of molecules is maintained by their sequence/structure similarity i.e., the orthologue conjecture. Although it has been useful in exploring several examples in biomedicine (e.g., the thousands of models between humans and mice [138]), the orthologue conjecture ignores the systems perspective, where cardinality of the relationships within a given species is much more complex than 1:1 [139] and [140].

The alternative to such assessments of construct validity in finding orthologues is identifying candidate gene sets through methods such as pathway analysis to determine specific functional pathway, that interoperates with another species. Such an approach is valid because even if the two species are evolutionarily distant relatives (e.g., yeast vs. humans), the assumption is that they are functional orthologues. However, it can easily be imagined that at the scope of pathways, or even higher such as systems and organisms, each composed of multiple functioning components, it is difficult to infer lower level (e.g., gene level) similarity in function or structure; especially because higher-level scopes can have redundancies in the function of the lower-level components, and thus disruptions of these components (e.g., an SNP mutation) may not have a higher-level effect. Also, not all genes may have an equal effect on the phenotype of interest.

Finally, ontologies provide a computational basis to directly map across species. By representing knowledge in a structured and controlled-vocabulary boundary, ontologies can be used to computationally interoperated across species. Robinson et al. demonstrate this in an example that interoperates 'aortic stenosis' from the Human Phenotype Ontology (HPO) to 'aortic valve stenosis' defined in the Mammalian Phenotype Ontology (HPO) through mapping 'constricted' using the Phenotype, Attribute, and Trait Ontology (PATO) and the anatomical term 'aortic valve' is mapped using Uberon, a combined cross-species ontology [137].

5.1.5 Phenotypes' Processing (End-utility)

Phenotyping enables clinicians and researchers to describe their observations and derive a hypotheses/diagnosis to test/treat. Therefore, in the clinic, phenotyping is a powerful diagnostic, therapeutic, and prognostic tool which also has empowered the research community to understand the etiology of phenotypes by mapping them on genotypes. For example, in the early 1960s, correlations between karyotype and phenotype, demonstrated an early link between genetic abnormality and disease manifestation of Turner's syndrome [141]. Since then, a rapid growth of technology has promoted research leading to thousands of genes implicated as the etiology of specific Mendelian and non-Mendelian complex traits [142]. Technological progress for such research and advances in the application of bioinformatics and computational biology has led the path towards improving clinical evaluations and genomic studies [114] and are paving the road towards precision medicine [143] and [144].

Tools such as the haplotype map of the human genome (HapMap) project [145], the international human phenome project [146], and the various genome-wide investigations and phenome-wide studies [147] broadly share the goal of associating the entire set of phenotypes to a genotype of interest [148]. Although currently, phenome-wide studies are in their early stages, the idea of mapping phenotypes or traits has shown significant promise. For example, Valdes et al. demonstrated the linkage of severe alcoholism to chromosome 19 [145]. Using expression quantitative trait loci (eQTL) analysis with short-term selected and bred mice for phenotypes related to alcohol consumption and withdrawal, Hitzemann et al., demonstrated common eQTL for these phenotypes [146]. More recently Hitzemann et al. have highlighted that RNA-seq can provide higher accuracy in expression-QTL analysis due to the higher dynamic range over microarrays [149] as an indicator of progress and current technology.

5.2 Linear Support Vector Machine (SVM)

Support vector machines (SVM) are powerful supervised machine learning/statistical models; in general for learning the structure of data through applications of regression or classification. In the more common supervised case, a training set that has the 'true' labels is needed. New test cases are then inferred on to obtain the relevant results. A linear SVM does this specifically by defining linear hyperplanes (a line in the 2D case) that define a discriminatory boundary. SVMs are in general effective at tasks in high-dimensional spaces and the use of Kernel methods makes them versatile in various types of decision boundaries. They are also known to have limitations in the context of complexity, thus very large datasets need higher computing resources, but once trained, the inference is highly efficient. Finally, if the training set is small, in fact fewer example (rows) than features (columns) are present, poor performance is also expected i.e., the curse of dimensionality.

5.2.1 A formal description of linear SMVs

• $\mathbf{Training}$: Learn the mapping from $\boldsymbol{X} \rightarrow \boldsymbol{Y}$ where x_i are the examples (i.e., rows, objects or cell) such that y_i are the class labels (e.g., ± 1); $x_i \in \boldsymbol{X}$ and $y_i \in \boldsymbol{Y}$. Since we have multiple samples $j \in \{1,..,K\}$ to train on, we have $\{x_{i,j}\}_{i,j=1}^{N,K}$ dataset with labels $\{y_{i,j}\}_{i,j=1}^{N,K}$. We can split this data for training and test sets as needed for learning and model evaluation.

• $\mathbf{Inference}$: given a new sample $x \in \boldsymbol{X}$ estimate $\hat{y}\ or f(\hat{x})$.

• $\mathbf{Result}$: The result of training j samples with i examples i.e., $x_{i,j}$, are a set of j hyperplanes with the general equation for K-features as in equation 9. If $\boldsymbol{w}_0 = 0$, then the hyperplanes $f(\boldsymbol{X})$ goes through the origin.

$$f(\boldsymbol{X_j}) = \boldsymbol{w}_{j,0} + \boldsymbol{w}_{j,1}\boldsymbol{X_{j,1}} + \cdots + \boldsymbol{w}_{j,K}\boldsymbol{X_{j,K}} = 0 \tag{9}$$

Here $\boldsymbol{w}$ is a vector of coefficients pertaining to the separating hyperplane; analogous to $y = mx + b$ where m is the slope coefficient of the line with the intercept b. The $sign()$ function limits and discretizes $f(\boldsymbol{X})$ to ± 1. Shown in equation 10, the discriminatory boundary $\boldsymbol{Y}$ or $f(\boldsymbol{X})$ is defined as a line (2D), plane (3D), or hyperplane ($\geq$4D). This boundary is geometrically perpendicular ($\perp$) to $\boldsymbol{w}$. When testing new cases, we estimate which side it belongs to as in equation 11.

$$f(\boldsymbol{w} \cdot \boldsymbol{X} + b) = 0 \tag{10}$$

$$\hat{f}(\boldsymbol{X}) = sign(\boldsymbol{w} \cdot \boldsymbol{X} + b) \tag{11}$$

Given any two points $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ that define the limits of the margin (i.e., support vectors) and a hyperplane $f(\boldsymbol{X})$, equation 12 shows the width of this margin is $2 * |a|$. If the data is scaled so that $|a| = 1$ (without changing the problem), the margin's width will equal to 2. This means that the distance (Euclidean) between $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ i.e., $\|(\boldsymbol{x_1} - \boldsymbol{x_2})\|$ can be derived from equations 13 and 14. The latter shows that in order to maximize the margin, we need to minimize $\|\boldsymbol{w}\|$. This geometric relationship between the hyperplane $\boldsymbol{w}$, $f(\boldsymbol{X})$ and example data-points is visualized in a 2D coordinate system in figure 61. The magnitude of the $\boldsymbol{w}$ i.e., $\|\boldsymbol{w}\|$ is algebraically the Euclidean distance $\sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}$.

$$\boldsymbol{w} \cdot \boldsymbol{X} + b = +a \cdots and \cdots \boldsymbol{w} \cdot \boldsymbol{X} + b = -a \tag{12}$$

$$\boldsymbol{w} \cdot (\boldsymbol{x_1} - \boldsymbol{x_2}) = 2 \tag{13}$$

$$\|\boldsymbol{x_1} - \boldsymbol{x_2}\| = \frac{2}{\|\boldsymbol{w}\|} \tag{14}$$

Once training with a an SVM is finished, a subset of the data-points from $\boldsymbol{x_{i,j}}$ define the margin and are called the support vectors. For each support vector $\boldsymbol{x_{SV_i}}$, a coefficient α_i is computed (i.e., slope or weight). A linear combination of these support vectors and α_is, results in the $\boldsymbol{w}$ vector as in equation 15, where $No.SVi$ is the number of support vectors. As well as the α_is, the bias b is also computed and returned.

$$\boldsymbol{w} = \sum_{i=1}^{No.SVi} \alpha_i \boldsymbol{x_{SVi}} \quad s.t.\ a_i \neq 0 \tag{15}$$

At their core, SVMs are solving a quadratic problem called the *constrained optimization problem*. A **minimization** $min_{\boldsymbol{w},b}(\frac{\|\boldsymbol{w}\|^2}{2})$ to find **optimal** $< \boldsymbol{w}, b >$ given the constraint $f_i(\boldsymbol{x} \cdot \ < \boldsymbol{w},\ b >) \geq 1$ as in equation 16. This problem

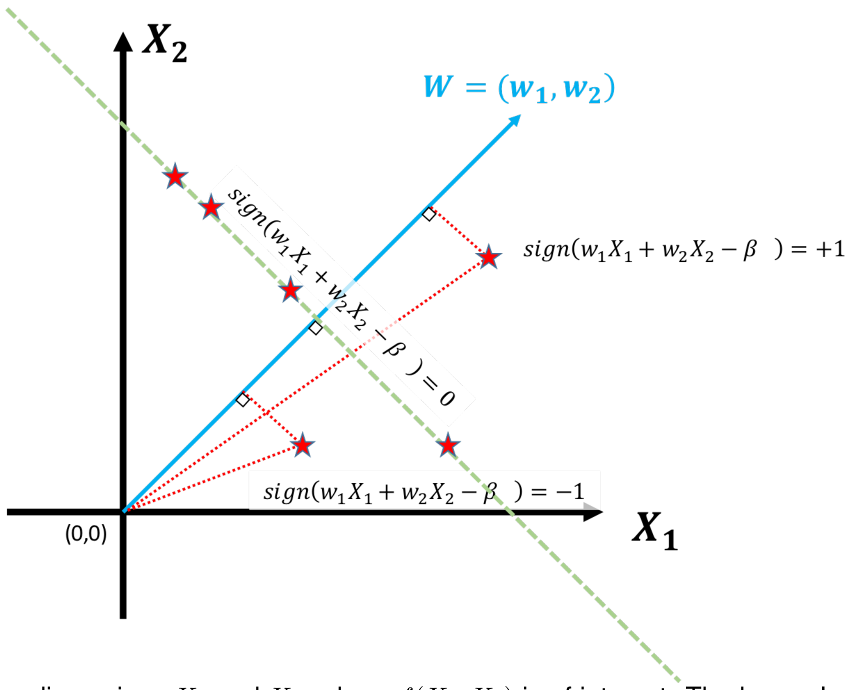

X_2
$W = (w_1, w_2)$
$sign(w_1X_1 + w_2X_2 - \beta\ ) = 0$
$sign(w_1X_1 + w_2X_2 - \beta\ ) = +1$
$sign(w_1X_1 + w_2X_2 - \beta\ ) = -1$
(0,0)
X_1

Figure 61: Given the two dimensions X_1 and X_2, where $f(X_1, X_2)$ is of interest. The hyperplane $\boldsymbol{W}$ and its relationship to the coordinates is visualized.

has been computationally optimized by rephrasing it as a *convex quadratic program*, which is implemented in the *e1071* package.

$$Find < \boldsymbol{w}, b >,\ min(\frac{\|\boldsymbol{w}\|^2}{2})\ s.t.\ f_i(\boldsymbol{x} \cdot\ < \boldsymbol{w}, b >) \geq 1\ where\ f_i \in \{-1, 1\} \tag{16}$$

To reduce the rigidity of the margin, we can penalized the objective function which allows some amount of error in the classification. For this utility, cost C is defined as the trade-off of margin-width and discordance. Generally, this is written as in equation 17 where the loss is the step-loss (or zero-one loss). The loss is shown with the indicator function $II()$ such that $II(\delta) = 1$ if the condition δ is "true", 0 otherwise. This minimization is considered a combinatorial optimization problem, which are computationally expensive. A $\mathbf{lossfunction}(\ell)$ quantifies the discordance between the prediction $\hat{y}$ and the truth y. Another commonly used loss function known as the "hinge-loss" ($\mathbb{H}$) is defined as $\ell(y, \hat{y}) = max(0, 1 - y(\hat{y}))$. By using the hinge-loss we are casting the problem with combinatorial complexity to quadratic or better because one of the properties of $\mathbb{H}$, one being its convex upper bound.

$$min_{\boldsymbol{w},b}(\frac{\|\boldsymbol{w}\|^2}{2} + C\sum_i II[f_i(\boldsymbol{x_i} \cdot\ < \boldsymbol{w}, b >) < 1]) \tag{17}$$

By introducing ξ_i as the error tolerance for each i^{th} example in the training we simplify the algebra as in equation 18. If $\xi_i = 0$ or $C = 0$, the margin would be have the same original rigid boundary; thus both are defined as to be greater or equal to 0 i.e. $\xi_i \geq 0$ and $C \geq 0$. However, as $C \to \infty$ or the penalization $\xi_i \to \infty$, the solution again gets closer to the rigid-boundary.

$$min_{\boldsymbol{w},b}(\frac{\|\boldsymbol{w}\|^2}{2} + C\sum_i \xi_i)\ \ s.t.\ f_i(\boldsymbol{x} \cdot\ < \boldsymbol{w}, b >) \geq 1 - \xi_i \tag{18}$$

To solve this optimization, its **Lagrangian dual function** needs to be derived. A simple general example, the Lagrangian is defined when given an optimization problem on $x \in \Re^n$ which we aim to minimize $f_0(x)$ while subject to

$f_i(x) \leq 0$ and $h_j(x) = 0$ where $i \in \{1, \cdots m\}$ and $j \in \{1, \cdots p\}$. The Lagrangian $L : \Re^n * \Re^m * \Re^p \rightarrow \Re$ is thus written as in equation 19. The Lagrange multipliers $\lambda\ and\ \nu$ are vectors defined in the Lagrange dual function as in equation 20. A takeaway from this dual function is that the domain of $g()$ is defined when $g > -\infty$ given sets of λ and ν. This means that the dual function consists of many functions albeit constrained, and it is concave in (λ, ν) enabling the identification the optimal the minimum or maximum values.

$$L(x, \lambda, \nu) := f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{p} \nu_i h_j(x) \tag{19}$$

$$g(\lambda, \nu) = inf_{x \in D} L(x, \lambda, \nu) \tag{20}$$

Specifically for the SVM, the Lagrangian for this problem is written as in equation 21. Minimizing with respect to the $\boldsymbol{w}$, b, and ξ means computing the partial derivatives with respect to w, b, and xi_i 22, 23, and 24 respectively. The latter highlights that for non-support vectors, $\alpha = 0$, for margin support vectors, $0 < \alpha < 0$, and finally for non-margin support vectors, $\alpha = C$. Firstly implication is that the non-margin support vectors can only be as influential with an upper range of C; reducing the severity of extreme outliers.

$$g(\lambda, \boldsymbol{w}, b, \xi, \alpha) = \frac{\|\boldsymbol{w}\|^2}{2} + C\sum_i \xi_i + \sum_i \alpha_i(1 - f_i(\boldsymbol{x} \cdot \ < \boldsymbol{w}, b >) - \xi_i) + \sum_i \lambda_i(-\xi_i) \ \ s.t.\ \alpha_i \geq 0,\ \lambda_i \geq 0 \tag{21}$$

$$\frac{\mathrm{d}L}{\mathrm{d}w} = w - \sum_i \alpha_i y_i x_i = 0 \ \therefore \ w = \sum_i \alpha_i y_i x_i \tag{22}$$

$$\frac{\mathrm{d}L}{\mathrm{d}b} = \sum_i \alpha_i y_i = 0 \tag{23}$$

$$\frac{\mathrm{d}L}{\mathrm{d}\xi_i} = C - \alpha_i - \lambda_i = 0 \ \therefore \ \alpha_i = C - \lambda_i;\ given\lambda \geq 0 \ \therefore \alpha_i \leq C \tag{24}$$

Similar to the general example demonstrated earlier, the Lagrangian dual function can now be derived as in equation 25 and simplified to equation 26. The goal here is to compute a solution for bmw by maximizing the dual function subject to the constraints $0 \leq \alpha_i \leq C$ and $\sum_i f_i\ alpha_i = 0$.

$$g(\alpha, \lambda) = \frac{\|\boldsymbol{w}\|^2}{2} + C\sum_i \xi_i + \sum_i \alpha_i(1 - f_i(\boldsymbol{x} \cdot \ < \boldsymbol{w}, b >) - \xi_i) + \sum_i \lambda_i(-\xi_i) \tag{25}$$

$$g(\alpha) = \sum_i \alpha_i - \frac{\sum_i^m \sum_j^m \alpha_i \alpha_j y_i y_j x_i x_j}{2} \ \ s.t. m \in \{1, \cdots, No.SVi\} \tag{26}$$

Finally, as another representation of the SVM classifier, we define the kernel-based SVM classifier, given the kernel $k(x, \cdot)$. To do this, we introduce the reproducing kernel Hilbert space (RKHS) as a mathematical approach to evaluate any two functions in a Hilbert space; this allows the leveraging of key properties of the Hilbert space. For example the RKHS corresponding to the Gaussian Kernel, as with other RKHs, essentially performs a smoothing task. The decision function with $k(x, \cdot)$ is shown in equation 27.

$$\boldsymbol{w} = \sum_i f_i \alpha_i k(x_i, \cdot) \tag{27}$$

5.3 General Equations Used

$$F_1 = 2 \cdot \frac{precision * recall}{precision + recall} \tag{28}$$

$$recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{29}$$

$$precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{30}$$

$$CV(\hat{f}, \alpha) = \frac{\sum_{i=1}^{N} L(y_i, \hat{f}^{-k(i)}(x_i, \alpha))}{N} \tag{31}$$

$$SSE = \sum_{1}^{n} (y_i - \hat{f}^{k(i)})^2 \quad n \in \{1, \cdots, N\} \tag{32}$$

$$PT = (freq = p(class_{G1,\ G2M,\ or\ S}|X_{train}) \ + \ \epsilon_{PT} \tag{33}$$

$$\epsilon_{PT} = mean(norm(1000, u = \mu_{p(class_{G1,\ G2M,\ or\ S}|X_{train})}, \sigma = 1)) * 0.3 \tag{34}$$

5.4 R Workflows

5.4.1 Code availability and open-source access

This project is produced in R with a culmination of open-source code, packages, and libraries described in detail in the next section. The entire RTL framework, was intended for public-release and open access which can be access here: `https://github.com/eisascience/RTL`

There are several requirements for installation, documented in the vignettes. The structure and flow of data is demonstrated via the 1-D simulated demo (code available in a section below and as a vignette within the RTL framework's package).

5.4.2 Software, packages, and libraries

- R version 3.4.3 (2017-11-30), `x86_64-apple-darwin15.6.0`
- Locale: `en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Running under: `OS X El Capitan 10.11.6`
- Matrix products: default
- BLAS: `/System/Library/Frameworks/Accelerate.framework/Versions/A...`
- BLAS: `/Frameworks/vecLib.framework/Versions/A/libBLAS.dylib`
- LAPACK: `/System/Library/Frameworks/Accelerate.framework...`
- LAPACK: `/Versions/A/Frameworks/vecLib.framework/Versions/A/libLAPACK.dylib`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: data.table 1.10.4-3, ggplot2 2.2.1.9000, knitr 1.20, rmarkdown 1.8, RTL 1.0.0, xtable 1.8-2

- Loaded via a namespace (and not attached): assertthat 0.2.0, backports 1.1.2, bindr 0.1, bindrcpp 0.2, boot 1.3-20, broom 0.4.3, caret 6.0-78, class 7.3-14, codetools 0.2-15, colorspace 1.3-2, compiler 3.4.3, CVST 0.2-1, ddalpha 1.3.1, DEoptimR 1.0-8, DescTools 0.99.23, digest 0.6.14, dimRed 0.1.0, doParallel 1.0.11, dplyr 0.7.4, DRR 0.0.2, e1071 1.6-8, evaluate 0.10.1, expm 0.999-2, foreach 1.4.4, foreign 0.8-69, glue 1.2.0, gower 0.1.2, gradDescent 2.0.1, grid 3.4.3, GSE 4.1, gtable 0.2.0, htmltools 0.3.6, ipred 0.9-6, iterators 1.0.9, kernlab 0.9-25, labeling 0.3, lattice 0.20-35, lava 1.5.1, lazyeval 0.2.1, lubridate 1.7.1, magrittr 1.5, manipulate 1.0.1, MASS 7.3-48, Matrix 1.2-12, matrixStats 0.52.2, mnormt 1.5-5, ModelMetrics 1.1.0, munsell 0.4.3, mvtnorm 1.0-6, nlme 3.1-131, nnet 7.3-12, parallel 3.4.3, parallelSVM 0.1-9, pillar 1.0.1, pkgconfig 2.0.1, plyr 1.8.4, prodlim 1.6.1, psych 1.7.8, purrr 0.2.4, R6 2.2.2, RColorBrewer 1.1-2, Rcpp 0.12.16, RcppRoll 0.2.2, recipes 0.1.1, reshape2 1.4.3, rlang 0.1.6.9003, robustbase 0.92-8, rpart 4.1-11, rprojroot 1.3-1, scales 0.5.0.9000, sfsmisc 1.1-1, sgd 1.1, splines 3.4.3, stats4 3.4.3, stringi 1.1.6, stringr 1.2.0, survival 2.41-3, tibble 1.4.1, tidyr 0.7.2, tidyselect 0.2.3, timeDate 3042.101, tools 3.4.3, withr 2.1.1.9000, zoo 1.8-0

5.4.3 A Demo Run With the RTL Framework Using A Simulated One-Dimensional Dataset

Since the code is done in R using rknitr package, the output when knitted in the pdf format, is used to import sections of it into this document; redundant output is removed to conserve the number of pages.

RTL Demo on 1D Simulated Data

@eisamahyari

3/8/2018

The RTL package includes everything necessary to run TL in R. See Installs.R for easy install of all imports.

```
library(knitr)
library(rmarkdown)
#render("./vignettes/RTL_1D_SimDemo.Rmd", html_document())
#render("./vignettes/RTL_1D_SimDemo.Rmd", pdf_document())
#render("./vignettes/RTL_1D_SimDemo.Rmd", latex_document())
#render("./vignettes/RTL_1D_SimDemo.Rmd", "all")

#load the packages needed
suppressPackageStartupMessages(library(RTL))
library(xtable)
library(ggplot2)
library(data.table)
```

```
## data.table 1.10.4.3

## **********
## This installation of data.table has not detected OpenMP support. It should still work but in single-
## **********

##   The fastest way to learn (by data.table authors): https://www.datacamp.com/courses/data-analysis-t

##   Documentation: ?data.table, example(data.table) and browseVignettes("data.table")

##   Release notes, videos and slides: http://r-datatable.com
```

```
getwd(); knitting = F
```

```
## [1] "/Volumes/Maggie/School/Projects/Eisa/R/RTL/RTL_public_release_v1.0/vignettes"
```

```
if(grepl("vignettes", getwd())) knitting = T
```

```
#Name the run
runID = "RTL.1DSim.FullRun.01"
strID = "BimodalGaus_S5xT10_N2000_Alg4V2Alg6V2"

CoreClassifier = "LinSVM" #LinSVM

RTLalg4v2BL = T # if T, use alg4 v2; default = F
RTLalg6v2BL = T # if T, use alg6 v2; default = F

DensityFocusBL = F #set to T to pass a density threshold to focus the GD
```

Check the directory for Figures, log files, and saved intermediate .rds files. They are saved during the run in seperate folders.

A 1D Simulation is available as a function in the RTL framework. With Ntest=50, we the design is we obtain 5 cohors with 10 samples in each as the test set; Loaded from .RDS provided. Up to 10 cohorts are availble, or simulate your own 1,2,.., N dim samples.

```r
getwd()
```

```
## [1] "/Volumes/Maggie/School/Projects/Eisa/R/RTL/RTL_public_release_v1.0/vignettes"
```

```r
#Load demo data
#simulated bimodal gaussian, i.e., two distributions, to classify one from the other
#5 training 1D sample and 'Ntest' tests. The training and tests vary only by shift.
#this dataset is chosen for simplicity in interpretation

TrainTest.ls <- Load1DSim(Ntest = 10)
#16 simulated training sets are avaiable, for speed we choose 5
TrainTest.ls$TrainSetXYls <- lapply(names(TrainTest.ls$TrainSetXYls)[1:5], function(xN){
  TrainTest.ls$TrainSetXYls[[xN]]
})
```

———-Some exploratory analysis of the datasets created. ———

The training set bimodal centers are shown:

```r
TrainCenters <- t(rbindlist(lapply(1:length(TrainTest.ls$TrainSetXYls), function(xi){
  as.data.frame(cbind(Neg=round(mean(TrainTest.ls$TrainSetXYls[[xi]]$X.train[which(TrainTest.ls$TrainSe
    Pos=round(mean(TrainTest.ls$TrainSetXYls[[xi]]$X.train[which(TrainTest.ls$TrainSetXYls[[xi]]$Y.trai
})))

colnames(TrainCenters) <- paste("TrainSamp", 1:ncol(TrainCenters), sep="")
rownames(TrainCenters) <- paste(rownames(TrainCenters), "Class", sep="")

print(xtable(TrainCenters), type="html")
```

TrainSamp1

TrainSamp2

TrainSamp3

TrainSamp4

TrainSamp5

NegClass

1.00

1.00

2.00

3.00

-0.00

PosClass

7.00

7.00

7.00

9.00

6.00

The Effect Size:

```r
EffectSizeSumm <- t(summary(unlist((lapply(1:length(TrainTest.ls$TrainSetXYls), function(sampI) {
  Cohens_d(
    x = TrainTest.ls$TrainSetXYls[[sampI]]$X.train[which(TrainTest.ls$TrainSetXYls[[sampI]]$Y.train==1)
    y = TrainTest.ls$TrainSetXYls[[sampI]]$X.train[which(TrainTest.ls$TrainSetXYls[[sampI]]$Y.train==-1
}))))

EffectSizeSumm <- rbind(EffectSizeSumm, t(summary(unlist((lapply(1:length(TrainTest.ls$TestSetXYls), fu
  Cohens_d(
    x = TrainTest.ls$TestSetXYls[[sampI]]$X.test[which(TrainTest.ls$TestSetXYls[[sampI]]$Y.test==1)],
    y = TrainTest.ls$TestSetXYls[[sampI]]$X.test[which(TrainTest.ls$TestSetXYls[[sampI]]$Y.test==-1)])
})))))

rownames(EffectSizeSumm) <- c("TrainSet", "TestSet")

print(xtable(EffectSizeSumm), type="html")
```

Min.

1st Qu.

Median

Mean

3rd Qu.

Max.

TrainSet

5.92

5.95

6.00

6.00

6.01

6.13

TestSet

5.83

5.97

6.04

6.02

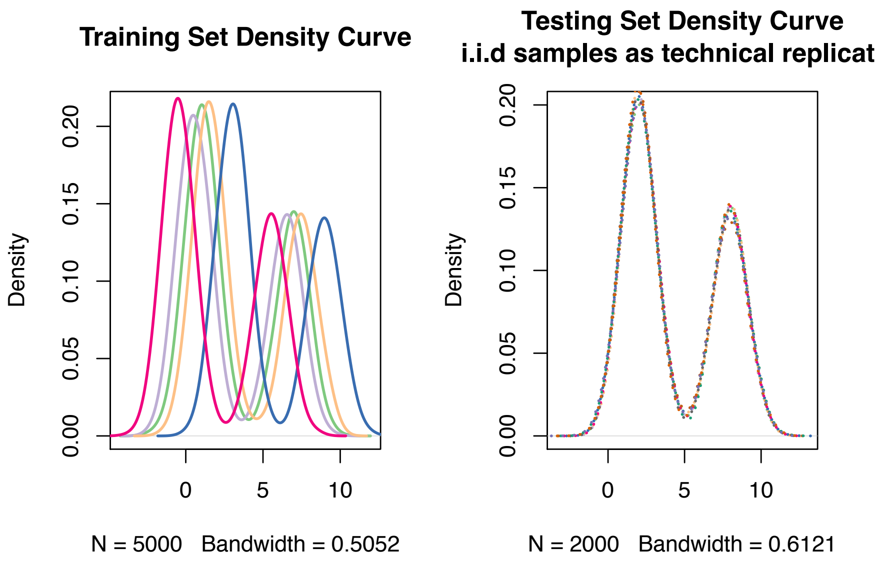
6.07

6.20

Lets visualize the 1D density curves these bimodal samples create.

```r
BaseFigDIR <- paste(figsdir,"TLViz/", runID, "/", strID, "/",sep="" )
if(!dir.exists(BaseFigDIR)) dir.create(BaseFigDIR, recursive = T)



par(mfrow=c(1,2))
```

```
plot(density(TrainTest.ls$TrainSetXYls$sample_1$X.train),
     main="Training Set Density Curve", col=col_vector[1], lwd=2)

for(counti in 2:length(TrainTest.ls$TrainSetXYls)){

  lines(density(TrainTest.ls$TrainSetXYls[[names(TrainTest.ls$TrainSetXYls)[counti]]]$X.train),
                lwd=2, col=col_vector[counti])
}

plot(density(TrainTest.ls$TestSetXYls$sample_1$X.test),
     main="Testing Set Density Curve\ni.i.d samples as technical replicates", col=col_vector[1], lwd=2,

for(counti in 2:length(TrainTest.ls$TestSetXYls)){

  lines(density(TrainTest.ls$TestSetXYls[[names(TrainTest.ls$TestSetXYls)[counti]]]$X.test),
                lwd=2, col=col_vector[counti], lty=3)
}
```



Algorithm 1, trains to find a classification boundary for each training; thus for each training example (here 16), we have a single boundary (total 16 here) in each feature space.

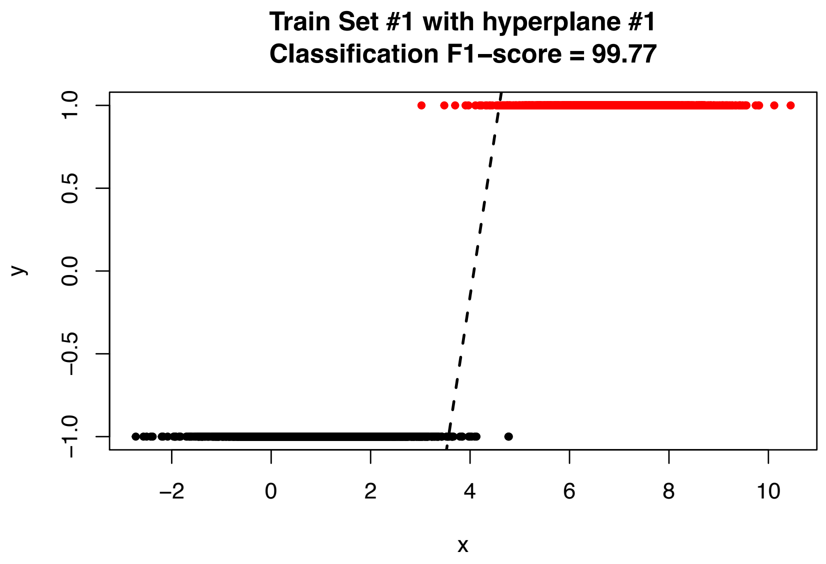
```
par(mfrow=c(1,1))

analysisID = "TLAlg1_FCmode_10KCV_g0.02_C0.5"

fileID = paste(savedir, runID, strID, analysisID, ".rds", sep="")

if(!file.exists(fileID)){
```

4

```
  alg1_res <- alg1_baselineClass(
    TrainXls = lapply(TrainTest.ls$TrainSetXYls, function(x){x$X.train}),
    TrainYls = lapply(TrainTest.ls$TrainSetXYls, function(x){x$Y.train}),
    TestXls = lapply(TrainTest.ls$TestSetXYls, function(x){x$X.test}),
    TestYls = lapply(TrainTest.ls$TestSetXYls, function(x){x$Y.test}),
    K_forCrossV   =  10,
    svmGamma      =  .02,
    svmCost       =  .5,
    prnt2scr      =   T,
    X_cols2Keep   =  NA,
    transX=F, sampleRed=F, doParalellSVM=F, datatyp="FC")


  saveRDS(alg1_res, fileID)
  PrimaryRun = T
} else {
  alg1_res <- readRDS(fileID)
  PrimaryRun = F
}

(alg1_res$baselineSVM)
```

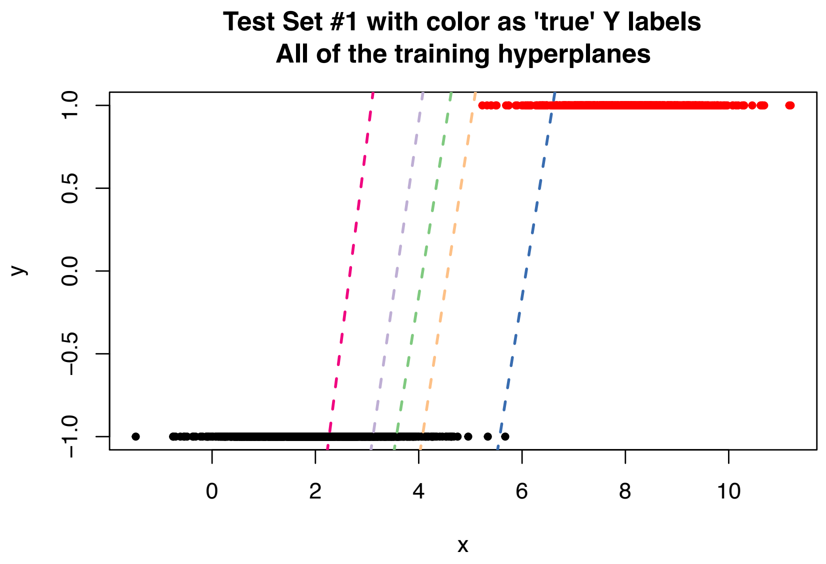
```
               Wm1      b.int
sample_1 -1.957020  -7.980612
sample_2 -2.147168  -7.684405
sample_3 -2.025060  -9.243727
sample_4 -1.948663 -11.850207
sample_5 -2.447174  -6.544719
```

```
par(mfrow=c(1,1))
plot(x=TrainTest.ls$TrainSetXYls$sample_1$X.train,
     y=TrainTest.ls$TrainSetXYls$sample_1$Y.train,
     col=factor(TrainTest.ls$TrainSetXYls$sample_1$Y.train),
     pch=20, xlab="x", ylab="y", main=paste("Train Set #1 with hyperplane #1\nClassification F1-score =
abline(a=alg1_res$baselineSVM[1,2], b=-alg1_res$baselineSVM[1,1], lwd=2,lty=2)
```



```
#--------------------------

plot(x=TrainTest.ls$TestSetXYls$sample_1$X.test,
     y=TrainTest.ls$TestSetXYls$sample_1$Y.test,
     col=factor(TrainTest.ls$TestSetXYls$sample_1$Y.test),
     pch=20, xlab="x", ylab="y", main="Test Set #1 with color as 'true' Y labels\nAll of the training hy
for(HypInt in 1:nrow(alg1_res$baselineSVM)){
  abline(a=alg1_res$baselineSVM[HypInt,2], b=-alg1_res$baselineSVM[HypInt,1], lwd=2,lty=2, col=col_vect

}
```

6



```
#--------------------------

#dot product to obtain yhat example on first Train set
yhat <- sign(-TrainTest.ls$TrainSetXYls$sample_1$X.train * alg1_res$baselineSVM[1,1] + alg1_res$baselin

#confusion matrix
table(pred=yhat, ref=TrainTest.ls$TrainSetXYls$sample_1$Y.train)
```

```
##     ref
## pred   -1    1
##   -1 2996    5
##   1     4 1995
```

```
#The found classification matches internal alg1 results
alg1_res$results.all$sample_1$train$table
```

```
##           Reference
## Prediction   -1    1
##         -1 2996    5
##         1     4 1995
```

```
alg1_res$results.all$sample_1$train$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.9975000            0.9986667            0.9979990
##       Neg Pred Value            Precision               Recall
##            0.9983339            0.9979990            0.9975000
##                   F1           Prevalence       Detection Rate
##            0.9977494            0.4000000            0.3990000
## Detection Prevalence    Balanced Accuracy
##            0.3998000            0.9980833
```

```
alg1_res$results.all$sample_1$train$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9982000      0.9962497      0.9965858      0.9991766      0.6000000
## AccuracyPValue  McnemarPValue
##      0.0000000      1.0000000
```

```
#Trainin Stats
summary(t(as.data.frame(lapply(alg1_res$results.all, function(resi){
  c(resi$train$byClass[c(2,5,6,7,8,9,11)], resi$train$overall[1])
}))))
```

```
##   Specificity       Precision          Recall              F1
##  Min.   :0.9983   Min.   :0.9975   Min.   :0.9975   Min.   :0.9977
##  1st Qu.:0.9987   1st Qu.:0.9980   1st Qu.:0.9980   1st Qu.:0.9978
##  Median :0.9990   Median :0.9985   Median :0.9980   Median :0.9982
##  Mean   :0.9989   Mean   :0.9984   Mean   :0.9983   Mean   :0.9983
##  3rd Qu.:0.9990   3rd Qu.:0.9985   3rd Qu.:0.9980   3rd Qu.:0.9982
##  Max.   :0.9997   Max.   :0.9995   Max.   :1.0000   Max.   :0.9998
##    Prevalence  Detection Rate   Balanced Accuracy    Accuracy
##  Min.   :0.4   Min.   :0.3990   Min.   :0.9981    Min.   :0.9982
##  1st Qu.:0.4   1st Qu.:0.3992   1st Qu.:0.9982    1st Qu.:0.9982
##  Median :0.4   Median :0.3992   Median :0.9985    Median :0.9986
##  Mean   :0.4   Mean   :0.3993   Mean   :0.9986    Mean   :0.9987
##  3rd Qu.:0.4   3rd Qu.:0.3992   3rd Qu.:0.9985    3rd Qu.:0.9986
##  Max.   :0.4   Max.   :0.4000   Max.   :0.9998    Max.   :0.9998
```

These classification boundaries are summarized to a single via robust mean in algorithm 2.

```
analysisID = "TLAlg2"
BaseFigDIR <- paste(figsdir,"TLAlg2/", runID, "/", strID, "/", analysisID,sep="" )
if(!dir.exists(BaseFigDIR)) dir.create(BaseFigDIR, recursive = T)


fileID = paste(savedir, runID, strID, analysisID, ".rds", sep="")

if(!file.exists(fileID)){
  #UVC01.nonRCS.CD3.unstimulated
  alg2_res <- alg2_rob_meanNCov(alg1_res$baselineSVM)
  saveRDS(alg2_res, fileID)
  PrimaryRun = T
} else {
  alg2_res <- readRDS(fileID)
  PrimaryRun = F
}

head(alg2_res)
```

$U_simple Wm1 b.int -2.105017 -8.660734

$U_robust Wm1 b.int -2.025060 -7.980612

$C_simple Wm1 b.int Wm1 0.04289774 -0.3024523 b.int -0.30245226 4.1014805

$C_robust Wm1 b.int Wm1 0.0128295 -0.1467514 b.int -0.1467514 3.5069820

$v_0 [1] 0.0128295

$w_euc_mag [1] 2.02506

```
print(xtable(t(alg1_res$baselineSVM)), type="html")
```

sample_1

sample_2

sample_3

sample_4

sample_5

Wm1

-1.96

-2.15

-2.03

-1.95

-2.45

b.int

-7.98

-7.68

-9.24

-11.85

-6.54

```
round(alg2_res$U_simple,2) #simple average
```

Wm1 b.int -2.11 -8.66

```
round(alg2_res$U_robust,2) #robust average
```

Wm1 b.int -2.03 -7.98

```
round(alg2_res$C_robust,2) #robust covariance
```

```
    Wm1 b.int
```

Wm1 0.01 -0.15 b.int -0.15 3.51

```
round(alg2_res$C_simple,2) #simple covariance
```

```
    Wm1 b.int
```

Wm1 0.04 -0.3 b.int -0.30 4.1

Each test set's mapping with the robust classifier is aligned to each trainin set's y_hat via maximum cross-correlation, and the median shift compared to the training samples is reported in algorithm 3.

```
analysisID = "TLAlg3_FCmode_ml0_AbsCor"
BaseFigDIR <- paste(figsdir,"TLAlg3/", runID, "/", strID, "/", analysisID,sep="" )
if(!dir.exists(BaseFigDIR)) dir.create(BaseFigDIR, recursive = T)

fileID = paste(savedir, runID, strID, analysisID, ".rds", sep="")

PermLogSaveDirV <<- paste(saveLnLdir, PermLogSaveName, "/","TLAlg3/", runID, "/", strID, "/", analysisI
```

```r
if(!dir.exists(PermLogSaveDirV)) dir.create(PermLogSaveDirV, recursive = T)


if(!file.exists(fileID)){

  alg3_res <- alg3_shiftComp(
    task_list = lapply(TrainTest.ls$TestSetXYls,
                       function(x){x$X.test}),
    source_list    =  lapply(TrainTest.ls$TrainSetXYls,
                             function(x){x$X.train}),
    alg2_result    =  alg2_res,
    print2screen   =  F,
    ImpFeats = "",
    save2file      =  T,
    maximumLag = 0,
    CoreClassifier=CoreClassifier,
    datatyp="FC",
    useAbsCor = T,
    medianMediansBL = F)



  saveRDS(alg3_res, fileID)
  PrimaryRun = T
} else {
  alg3_res <- readRDS(fileID)
  PrimaryRun = F
}
```

The robust classifier is adapted to each test case by updating its bias. This is done on the mapped y_hat of each test case to find a low-density optima initialized at 0.

```r
analysisID = "TLAlg4_FCmode"

if(RTLalg4v2BL) {
  analysisID <- paste(analysisID, "_Mna_V2", sep="")
} else {
  analysisID<- paste(analysisID, "_M1_V1", sep="")
}

BaseFigDIR <- paste(figsdir,"TLAlg4/", runID, "/", strID, "/", analysisID,sep="" )

if(!dir.exists(BaseFigDIR)) dir.create(BaseFigDIR, recursive = T)

fileID = paste(savedir, runID, strID, analysisID, ".rds", sep="")

if(!file.exists(fileID)){

  if(!RTLalg4v2BL) {
    alg4_res <- alg4_BiasUpdate(task_list = lapply(TrainTest.ls$TestSetXYls,
                       function(x){x$X.test}),
                            alg1_result = alg1_res,
                            alg2_result = alg2_res,
                            alg3_result = alg3_res,
                            goodColumns = "", alg4MinFx = "gd",
                            Marg = 1,
                            save2file =T, ADM=F,
                            useMedian = T, ZnormMappingBL=F, datatyp="FC",
                            RCSmodeBL = F,
                            CoreClassifier = CoreClassifier)

  }
  if(RTLalg4v2BL) {
    alg4_res <- alg4_BiasUpdateV2(task_list = lapply(TrainTest.ls$TestSetXYls,
                       function(x){x$X.test}),
                            alg1_result = alg1_res,
                            alg2_result = alg2_res,
                            alg3_result = alg3_res,
                            goodColumns = "", alg4MinFx = "gd",
                            Marg = 1,
                            save2file =T, ADM=F,
                            useMedian = T, ZnormMappingBL=F, datatyp="FC",
                            RCSmodeBL = F,
                            CoreClassifier = CoreClassifier)
  }


  saveRDS(alg4_res, fileID)
  PrimaryRun = T
} else {
  alg4_res <- readRDS(fileID)
  PrimaryRun = F
}

(alg4_res)
```

```
       b_alg2_norm b_alg3_norm b_alg_norm
Task1    -7.980612    8.006727   7.120307
Task2    -7.980612   11.401461   8.488168
Task3    -7.980612    8.655556   7.340460
Task4    -7.980612   10.711306   7.310723
Task5    -7.980612   10.187537   7.339616
Task6    -7.980612    9.887072   7.199037
Task7    -7.980612   11.376925   8.499601
Task8    -7.980612   10.006021   7.986975
Task9    -7.980612    9.826883   7.088367
Task10   -7.980612    7.461974   6.691979
```

```
###############
#As this is a Demo Run, the settings we chose run the Extended version (V2.0) of the RTL framework.
#for comparison of results, we also run the base line here side by side
# This is a hacky approach for this Vignette, usually only 1 is chose at a time, comparin

fileID = paste(savedir, runID, strID, analysisID, "_hackyBaseline.rds", sep="")

if(!file.exists(fileID)){

alg4_res_baselineV <- alg4_BiasUpdate(task_list = lapply(TrainTest.ls$TestSetXYls,
                       function(x){x$X.test}),
                            alg1_result = alg1_res,
                            alg2_result = alg2_res,
                            alg3_result = alg3_res,
                            goodColumns = "", alg4MinFx = "gd",
                            Marg = 1,
                            save2file =T, ADM=F,
                            useMedian = T, ZnormMappingBL=F, datatyp="FC",
                            RCSmodeBL = F,
                            CoreClassifier = CoreClassifier)
saveRDS(alg4_res_baselineV, fileID)
  #PrimaryRun = T
} else {
  alg4_res_baselineV <- readRDS(fileID)
  #PrimaryRun = F
}
```

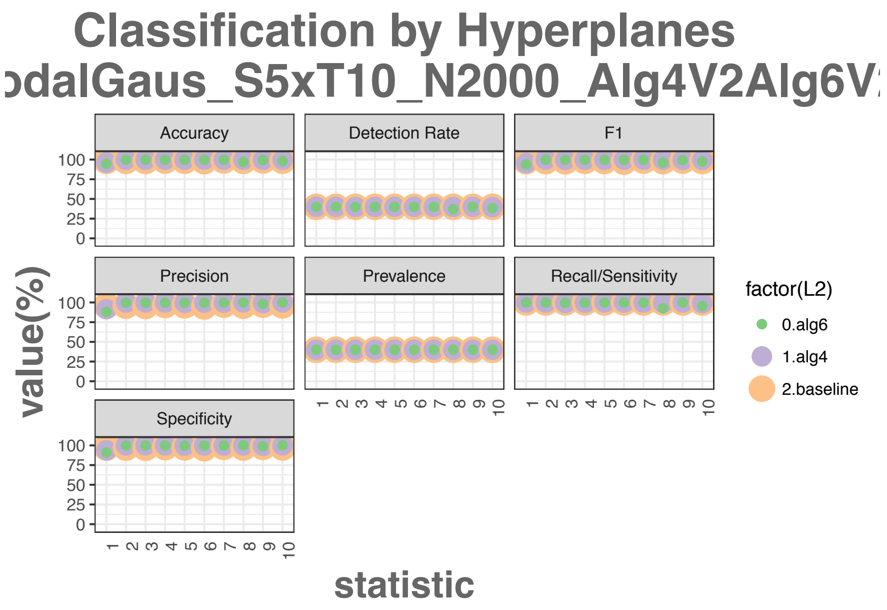
Each new classification boundary (50 of them as there are 50 test cases), is really the robust classifier with its bias updated in algorithm 4. Now we update the normal vector which has the effect of rotating the classifier; in the current implementation this rotation is only along the first principle component. In the next update, a robust approach to finding the rotation will be implemented. The rotation will also go beyond the first PC.

```
analysisID = "TLAlg6_FCmode"

if(RTLalg6v2BL) {
  analysisID <- paste(analysisID, "_Mna_V2", sep="")
} else {
  analysisID<- paste(analysisID, "_M0.2_V1", sep="")
}

BaseFigDIR <- paste(figsdir,"TLAlg6/", runID, "/", strID, "/", analysisID,sep="" )

if(!dir.exists(BaseFigDIR)) dir.create(BaseFigDIR, recursive = T)
fileID = paste(savedir, runID, strID, analysisID, ".rds", sep="")

if(!file.exists(fileID)){

    if(!RTLalg6v2BL) {
    alg6_res <- alg6_NormalVectorUpdate(task_list = lapply(TrainTest.ls$TestSetXYls, function(x){x$X.te
                                    alg1_result = alg1_res,
                                    alg2_result = alg2_res,
                                    alg3_result = alg3_res,
                                    alg4_result = alg4_res,
                                    X_feat_cols = "",
                                    save2file = T,
                                    Marg = .2, ADM=F, datatyp="FC",
                                    RCSmodeBL = F,
                                    CoreClassifier = CoreClassifier)
    }
  if(RTLalg6v2BL) {
    alg6_res <- alg6_NormalVectorUpdateV2(task_list = lapply(TrainTest.ls$TestSetXYls, function(x){x$X.
                                    alg1_result = alg1_res,
                                    alg2_result = alg2_res,
                                    alg3_result = alg3_res,
                                    alg4_result = alg4_res,
                                    X_feat_cols = "",
                                    save2file = T,
                                    Marg = .2, ADM=F, datatyp="FC",
                                    RCSmodeBL = F,
                                    CoreClassifier = CoreClassifier)

  }
  saveRDS(alg6_res, fileID)
  PrimaryRun = T
} else {
  alg6_res <- readRDS(fileID)
  PrimaryRun = F
}

###############
#As this is a Demo Run, the settings we chose run the Extended version (V2.0) of the RTL framework.
#for comparison of results, we also run the base line here side by side
# This is a hacky approach for this Vignette, usually only 1 is chose at a time, comparin

fileID = paste(savedir, runID, strID, analysisID, "_hackyBaseline.rds", sep="")

if(!file.exists(fileID)){

  alg6_res_baselineV <- alg6_NormalVectorUpdate(task_list = lapply(TrainTest.ls$TestSetXYls, function(x)
                                    alg1_result = alg1_res,
                                    alg2_result = alg2_res,
                                    alg3_result = alg3_res,
                                    alg4_result = alg4_res_baselineV,
                                    X_feat_cols = "",
                                    save2file = T,
                                    Marg = .2, ADM=F, datatyp="FC",
                                    RCSmodeBL = F,
                                    CoreClassifier = CoreClassifier)
saveRDS(alg6_res_baselineV, fileID)
  #PrimaryRun = T
} else {
  alg6_res_baselineV <- readRDS(fileID)
  #PrimaryRun = F
}
```

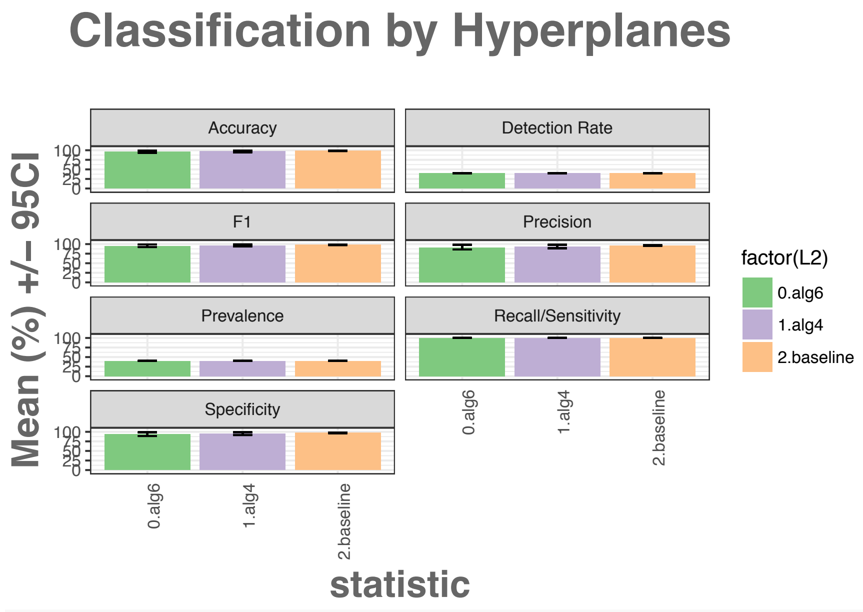
Select set of figures produced as evaluation of the classification and the framework. Check the main dir. for logs, latex tables, figures, . . . that also demonstrates the inner workings of the RTL framework, QA/QC, and reproducibility.

```
###Viz
    analysisID = "TLFinalViz"

    BaseFigDIR <- paste(figsdir,"RTLviz/", runID, "/", strID, "/", analysisID,sep="" )
    if(!dir.exists(BaseFigDIR)) dir.create(BaseFigDIR, recursive = T)
    fileID = paste(savedir, runID, strID, analysisID, ".rds", sep="")
    if(!file.exists(fileID)){

      Viz_res <- FinalViz(TrainTestSet.ls = TrainTest.ls$TestSetXYls,
        alg1_result = alg1_res,
        alg2_result = alg2_res,
        alg3_result = alg3_res,
        alg4_result = alg4_res,
        alg6_result = alg6_res,
        datatyp =  "FC",
        ADM = F)

      saveRDS(Viz_res, fileID)

    } else{

      Viz_res <- readRDS(fileID)

    }

    #######hacky version of the baseline versions from above
    analysisID = "TLFinalViz_hackyBaseline"

    BaseFigDIR <- paste(figsdir,"RTLviz/", runID, "/", strID, "/", analysisID,sep="" )
    if(!dir.exists(BaseFigDIR)) dir.create(BaseFigDIR, recursive = T)

    Viz_res_baselineV <- FinalViz(TrainTestSet.ls = TrainTest.ls$TestSetXYls,
        alg1_result = alg1_res,
        alg2_result = alg2_res,
        alg3_result = alg3_res,
        alg4_result = alg4_res_baselineV,
        alg6_result = alg6_res_baselineV,
        datatyp =  "FC",
        ADM = F)
```

```
[1] 1

[1] 2

[1] 3

[1] 4

[1] 5

[1] 6

[1] 7

[1] 8

[1] 9

[1] 10
```



```
  dtb.CSS <- as.data.table(Viz_res$comStats.sub)
  dtb.CSS[, Mean:=mean(value, na.rm = T), by=list(variable, L2)]
  dtb.CSS[, CI95:=quantile(value, c(.95), na.rm = T) , by=list(variable, L2)]
  dtb.CSS[, CI05:=quantile(value, c(.05), na.rm = T) , by=list(variable, L2)]

  dtb.CSS[, Meankfolds:=mean(value, na.rm = T), by=list(variable, L2)]
  dtb.CSS[, MeankfoldsLCI:=quantile(value, c(0.05), na.rm = T), by=list(variable, L2)]
  dtb.CSS[, MeankfoldsUCI:=quantile(value, c(0.95), na.rm = T), by=list(variable, L2)]



gg6 <- ggplot(dtb.CSS, aes(x=factor(L2), y=(Meankfolds*100), fill=factor(L2))) +
  geom_bar(position=position_dodge(), stat="identity") +
  geom_errorbar(aes(ymin=MeankfoldsLCI*100, ymax=MeankfoldsUCI*100),
                width=.2, position=position_dodge(.9)) +
  facet_wrap(~factor(variable), ncol=2) +
  theme_bw() +
  theme(plot.title = element_text( color="#666666", face="bold", size=25, hjust=0.5)) +
  theme(axis.title = element_text( color="#666666", face="bold", size=20)) + ylim(-5,105) +
  labs(title = paste("Classification by Hyperplanes\n", sep=""), y = "Mean (%) +/- 95CI", x = "statis
  theme(axis.ticks.x=element_blank(), axis.text.x = element_text(angle = 90, hjust = 1)) +
  scale_fill_manual(values = col_vector)
plot(gg6)
```



```
dtb.CSS
```

```
          variable     value       L1         L2 id      Mean      CI95
  1:      Accuracy 0.9830000 accuracy 2.baseline  1 0.9841500 0.9885000
  2:      Accuracy 0.9595000 accuracy     1.alg4  1 0.9694500 0.9921500
  3:      Accuracy 0.9505000 accuracy     0.alg6  1 0.9609000 0.9917000
  4:   Specificity 0.9716667     rest 2.baseline  1 0.9735833 0.9808333
  5:   Specificity 0.9325000     rest     1.alg4  1 0.9490833 0.9869167
 ---
206:    Prevalence 0.4000000     rest     1.alg4 10 0.4000000 0.4000000
207:    Prevalence 0.4000000     rest     0.alg6 10 0.4000000 0.4000000
208: Detection Rate 0.4000000     rest 2.baseline 10 0.4000000 0.4000000
209: Detection Rate 0.4000000     rest     1.alg4 10 0.4000000 0.4000000
210: Detection Rate 0.4000000     rest     0.alg6 10 0.4000000 0.4000000
          CI05 Meankfolds MeankfoldsLCI MeankfoldsUCI
  1: 0.9798500  0.9841500     0.9798500     0.9885000
  2: 0.9496500  0.9694500     0.9496500     0.9921500
  3: 0.9342000  0.9609000     0.9342000     0.9917000
  4: 0.9664167  0.9735833     0.9664167     0.9808333
  5: 0.9160833  0.9490833     0.9160833     0.9869167
 ---
206: 0.4000000  0.4000000     0.4000000     0.4000000
207: 0.4000000  0.4000000     0.4000000     0.4000000
208: 0.4000000  0.4000000     0.4000000     0.4000000
209: 0.4000000  0.4000000     0.4000000     0.4000000
210: 0.4000000  0.4000000     0.4000000     0.4000000
```

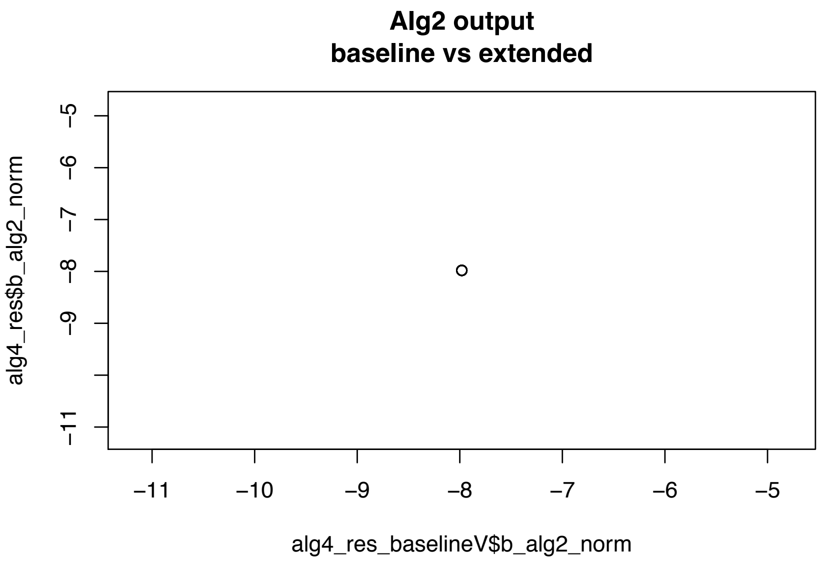
```
alg4_res_baselineV
```

```
       b_alg2_norm b_alg3_norm b_alg_norm
Task1    -7.980612    8.006727   7.227297
Task2    -7.980612    11.40146   9.837877
Task3    -7.980612    8.655556  11.089900
Task4    -7.980612    10.71131  10.180175
Task5    -7.980612    10.18754   9.282849
Task6    -7.980612    9.887072  10.524864
Task7    -7.980612    11.37692  10.243215
Task8    -7.980612    10.00602  11.435009
Task9    -7.980612    9.826883   8.955640
Task10   -7.980612    7.461974  11.284305
```

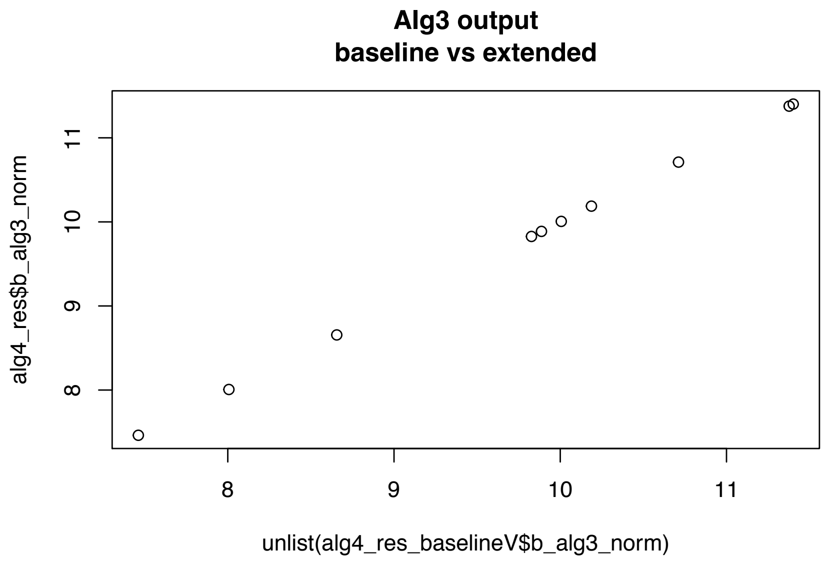
```
alg4_res
```

```
       b_alg2_norm b_alg3_norm b_alg_norm
Task1    -7.980612    8.006727   7.120307
Task2    -7.980612   11.401461   8.488168
Task3    -7.980612    8.655556   7.340460
Task4    -7.980612   10.711306   7.310723
Task5    -7.980612   10.187537   7.339616
Task6    -7.980612    9.887072   7.199037
Task7    -7.980612   11.376925   8.499601
Task8    -7.980612   10.006021   7.986975
Task9    -7.980612    9.826883   7.088367
Task10   -7.980612    7.461974   6.691979
```

```
plot(alg4_res_baselineV$b_alg2_norm, alg4_res$b_alg2_norm, main="Alg2 output\nbaseline vs extended")
```

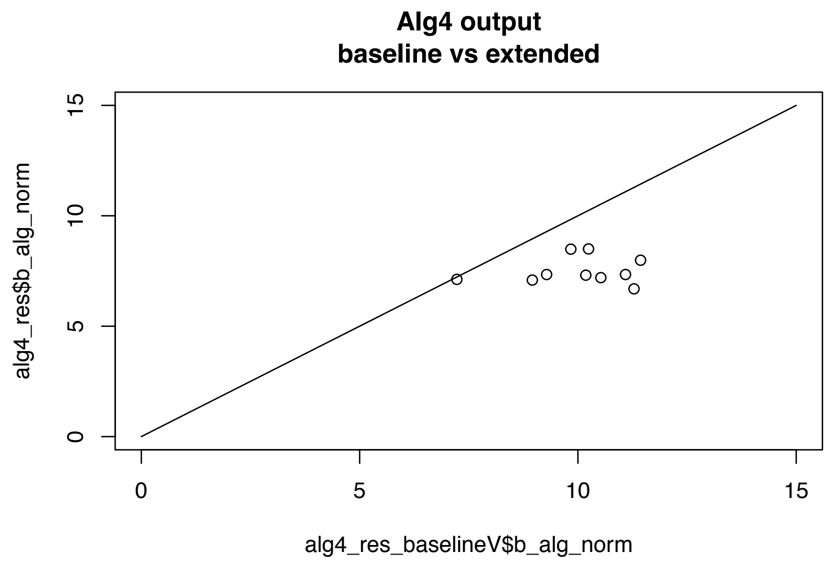


```r
plot(unlist(alg4_res_baselineV$b_alg3_norm), alg4_res$b_alg3_norm, main="Alg3 output\nbaseline vs exten
```



```r
plot(alg4_res_baselineV$b_alg_norm, alg4_res$b_alg_norm, main="Alg4 output\nbaseline vs extended",
     xlim=range(0,15),ylim=range(0,15))
lines(0:15, 0:15)
```

18



```
cbind(alg6_res$alg6_w_new, betas = alg4_res$b_alg_norm)
```

```
            Wm1    betas
 [1,] 2.073064 7.120307
 [2,] 2.067730 8.488168
 [3,] 2.079731 7.340460
 [4,] 2.066397 7.310723
 [5,] 2.161738 7.339616
 [6,] 2.113734 7.199037
 [7,] 2.033394 8.499601
 [8,] 1.992724 7.986975
 [9,] 2.148404 7.088367
[10,] 2.091066 6.691979
```

```
cbind(alg6_res_baselineV$alg6_w_new, betas = alg4_res_baselineV$b_alg_norm)
```

```
            Wm1     betas
 [1,] 2.132966  7.227297
 [2,] 2.036076  9.837877
 [3,] 2.153208 11.089900
 [4,] 2.072900 10.180175
 [5,] 1.961584  9.282849
 [6,] 2.088216 10.524864
 [7,] 2.025795 10.243215
 [8,] 1.706827 11.435009
 [9,] 2.115975  8.955640
[10,] 1.798291 11.284305
```

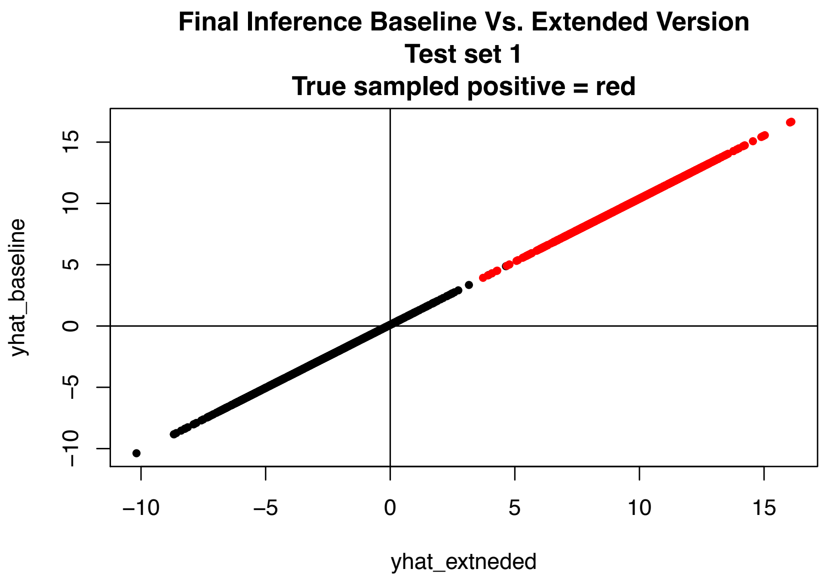
```
#Test set 1
yhat <- sign(TrainTest.ls$TestSetXYls$sample_1$X.test * alg6_res$alg6_w_new[1] - alg4_res$b_alg_norm[1]
table(truth=TrainTest.ls$TestSetXYls$sample_1$Y.test, pred=yhat)
```

```
     pred
truth   -1    1
   -1 1101   99
   1     0  800
```

```
yhat_baselineV <- sign(TrainTest.ls$TestSetXYls$sample_1$X.test * alg6_res_baselineV$alg6_w_new[1] - al

table(truth=TrainTest.ls$TestSetXYls$sample_1$Y.test, pred=yhat_baselineV)
```

```
     pred
truth   -1    1
   -1 1093  107
   1     0  800
```

```
predicRes <- cbind(yhat_extneded = TrainTest.ls$TestSetXYls$sample_1$X.test * alg6_res$alg6_w_new[1] - a
      yhat_baseline = TrainTest.ls$TestSetXYls$sample_1$X.test * alg6_res_baselineV$alg6_w_new[1] - alg

plot(predicRes, pch=20, col=factor(TrainTest.ls$TestSetXYls$sample_1$Y.test),
     main="Final Inference Baseline Vs. Extended Version\nTest set 1\nTrue sampled positive = red")
abline(h=0)
abline(v=0)
```



References

[1] M. Mooney, S. McWeeney, G. Canderan, and R. P. Sékaly. A systems framework for vaccine design. *Current Opinion in Immunology*, 25(5):551–555, 2013.

[2] A. A. Kolodziejczyk, J. K. Kim, V. Svensson, J. C. Marioni, and S. A. Teichmann. The technology and biology of single-cell RNA sequencing. *Molecular cell*, 58(4):610–20, may 2015. PMID: 26000846.

[3] F. Ravandi, J. L. Jorgensen, S. M. O'Brien, E. Jabbour, D. A. Thomas, G. Borthakur, R. Garris, X. Huang, G. Garcia-Manero, J. A. Burger, A. Ferrajoli, W. Wierda, T. Kadia, N. Jain, S. A. Wang, S. Konoplev, P. Kebriaei, R. E. Champlin, D. McCue, Z. Estrov, J. E. Cortes, and H. M. Kantarjian. Minimal residual disease assessed by multi-parameter flow cytometry is highly prognostic in adult patients with acute lymphoblastic leukaemia. *British Journal of Haematology*, 172(3):392–400, feb 2016.

[4] V. Proserpio and T. Lönnberg. Single-cell technologies are revolutionizing the approach to rare cells. *Immunology and Cell Biology*, 94(3):225–229, 2016. PMID: 26620630.

[5] A. Maratea, A. Petrosino, and M. Manzo. Adjusted F-measure and kernel scaling for imbalanced data learning. *Information Sciences*, 257:331–341, 2014.

[6] S. J. Pan and Q. Yang. A Survey on Transfer Learning. *Ieee Transactions on Knowledge and Data Engineering*, pages 1–15, 2009.

[7] a. Arnold, R. Nallapati, and W. Cohen. A Comparative Study of Methods for Transductive Transfer Learning. *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, 1:77–82, 2007.

[8] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017.

[9] H. W. Borchers. *pracma: Practical Numerical Math Functions*, 2017. R package version 2.1.1.

[10] B. D. Hedley and M. Keeney. Technical issues: Flow cytometry and rare event analysis. *International Journal of Laboratory Hematology*, 35(3):344–350, 2013. PMID: 23590661.

[11] N. Collier, A. Oellrich, and T. Groza. Toward knowledge support for analysis and interpretation of complex traits. 14(9):1, 2013.

[12] B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[13] W. Venables and B. Ripley. Random and mixed effects. In *Modern applied statistics with S*, pages 271–300. Springer, 2002.

[14] J. E. Cupp, J. F. Leary, E. Cernichiari, J. C. S. Wood, and R. A. Doherty. Rare-event analysis methods for detection of fetal red blood cells in maternal blood. *Cytometry*, 5(2):138–144, mar 1984.

[15] S. P. Perfetto, P. K. Chattopadhyay, and M. Roederer. Innovation: Seventeen-colour flow cytometry: unravelling the immune system. *Nature Reviews Immunology*, 4(8):648–655, aug 2004.

[16] R. R. Brinkman, N. Aghaeepour, G. Finak, R. Gottardo, T. Mosmann, and R. H. Scheuermann. State-of-the-Art in the Computational Analysis of Cytometry Data. *Cytometry Part A*, 87(7):591–593, 2015. PMID: 26111230.

[17] D. L. Bolton, K. McGinnis, G. Finak, P. Chattopadhyay, R. Gottardo, and M. Roederer. Combined single-cell quantitation of host and SIV genes and proteins ex vivo reveals host-pathogen interactions in individual cells. *PLOS Pathogens*, 13(6):e1006445, jun 2017.

[18] WHO | Tuberculosis (TB). *WHO*, 2017.

[19] WHO | HIV/AIDS. *WHO*, 2018.

[20] M. K. O'Shea and H. McShane. A review of clinical models for the evaluation of human TB vaccines. *Human Vaccines and Immunotherapeutics*, 12(5):1177–1187, 2016. PMID: 26810964.

[21] Y. Fukazawa, H. Park, M. J. Cameron, F. Lefebvre, R. Lum, N. Coombes, E. Mahyari, S. I. Hagen, J. Y. Bae, M. D. Reyes, T. Swanson, A. W. Legasse, A. Sylwester, S. G. Hansen, A. T. Smith, P. Stafova, R. Shoemaker, Y. Li, K. Oswald, M. K. Axthelm, A. McDermott, G. Ferrari, D. C. Montefiori, P. T. Edlefsen, M. Piatak, J. D. Lifson, R. P. Sékaly, and L. J. Picker. Lymph node T cell responses predict the efficacy of live attenuated SIV vaccines. *Nature medicine*, 18(11):1673–81, 2012.

[22] D. Gurdasani, L. Iles, D. G. Dillon, E. H. Young, A. D. Olson, V. Naranbhai, S. Fidler, E. Gkrania-Klotsas, F. A. Post, P. Kellam, K. Porter, and M. S. Sandhu. A systematic review of definitions of extreme phenotypes of HIV control and progression. *Aids*, 28(2):149–162, 2014. PMID: 24149086.

[23] M. C. Gold, S. Cerri, S. Smyk-Pearson, M. E. Cansler, T. M. Vogt, J. Delepine, E. Winata, G. M. Swarbrick, W. J. Chua, Y. Y. Yu, O. Lantz, M. S. Cook, M. D. Null, D. B. Jacoby, M. J. Harriff, D. A. Lewinsohn, T. H. Hansen, and D. M. Lewinsohn. Human mucosal associated invariant T cells detect bacterially infected cells. *PLoS Biology*, 8(6):1–14, 2010. PMID: 20613858.

[24] M. C. Gold, R. J. Napier, and D. M. Lewinsohn. MR1-restricted mucosal associated invariant T (MAIT) cells in the immune response to Mycobacterium tuberculosis. *Immunological Reviews*, 264(1):154–166, 2015. PMID: 25703558.

[25] M. Mooney and S. McWeeney. *Data integration and reproducibility for high-throughput transcriptomics*, volume 116. Elsevier Inc., 1 edition, 2014.

[26] A. K. Shalek and M. Benson. Single-cell analyses to tailor treatments. *Science translational medicine*, 9(408):eaan4730, sep 2017. PMID: 28931656.

[27] K. E. Saatman, A.-C. Duhaime, R. Bullock, A. I. Maas, A. Valadka, and G. T. Manley. Classification of Traumatic Brain Injury for Targeted Therapies. *Journal of Neurotrauma*, 25(7):719–738, 2008. PMID: 18627252.

[28] J. Spidlen, W. Moore, D. Parks, M. Goldberg, C. Bray, P. Bierre, P. Gorombey, B. Hyun, M. Hubbard, S. Lange, R. Lefebvre, R. Leif, D. Novo, L. Ostruszka, A. Treister, J. Wood, R. F. Murphy, M. Roederer, D. Sudar, R. Zigon, and R. R. Brinkman. Data file standard for flow cytometry, version FCS 3.1. *Cytometry Part A*, 77(1):97–100, 2010. PMID: 19937951.

[29] P. K. Chattopadhyay, C. M. Hogerkorp, and M. Roederer. A chromatic explosion: The development and future of multiparameter flow cytometry. *Immunology*, 125(4):441–449, 2008.

[30] M. Malek, M. J. Taghiyar, L. Chong, G. Finak, R. Gottardo, and R. R. Brinkman. flowDensity: reproducing manual gating of flow cytometry data by automated density-based cell population identification. 31(4):606–607, 2015.

[31] G. Finak, M. Langweiler, M. Jaimes, M. Malek, J. Taghiyar, Y. Korin, K. Raddassi, L. Devine, G. Obermoser, M. L. Pekalski, N. Pontikos, A. Diaz, S. Heck, F. Villanova, N. Terrazzini, F. Kern, Y. Qian, R. Stanton, K. Wang, A. Brandes, J. Ramey, N. Aghaeepour, T. Mosmann, R. H. Scheuermann, E. Reed, K. Palucka, V. Pascual, B. B. Blomberg, F. Nestle, R. B. Nussenblatt, R. R. Brinkman, R. Gottardo, H. Maecker, and J. P. McCoy. Standardizing Flow Cytometry Immunophenotyping Analysis from the Human ImmunoPhenotyping Consortium. *Scientific reports*, 6(August 2015):20686–20686, 2016.

[32] D. R. Bandura, V. I. Baranov, O. I. Ornatsky, A. Antonov, R. Kinach, X. Lou, S. Pavlov, S. Vorobiev, J. E. Dick, and S. D. Tanner. Mass Cytometry: A Novel Technique for Real-Time Single Cell Multi-Target Immunoassay Based on Inductively Coupled Plasma Time of Flight Mass Spectrometry. *Analytical Chemistry*, 81(16):6813–6822, 2009. PMID: 19601617.

[33] K. E. Diggins, P. Brent Ferrell, and J. M. Irish. Methods for discovery and characterization of cell subsets in high dimensional mass cytometry data. *Methods*, 82:55–63, 2015. PMID: 25979346.

[34] E. A. D. Amir, K. L. Davis, M. D. Tadmor, E. F. Simonds, J. H. Levine, S. C. Bendall, D. K. Shenfeld, S. Krishnaswamy, G. P. Nolan, and D. Pe'Er. ViSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. *Nature Biotechnology*, 31(6):545–552, 2013. PMID: 23685480.

[35] N. Samusik, Z. Good, M. H. Spitzer, K. L. Davis, and G. P. Nolan. Automated mapping of phenotype space with single-cell data. *Nature Methods*, 13(6):493–496, 2016. PMID: 27183440.

[36] B. Gaudilliere, G. K. Fragiadakis, R. V. Bruggner, M. Nicolau, R. Finck, M. Tingle, J. Silva, E. A. Ganio, C. G. Yeh, W. J. Maloney, J. I. Huddleston, S. B. Goodman, M. M. Davis, S. C. Bendall, W. J. Fantl, M. S. Angst, and G. P. Nolan. Clinical recovery from surgery correlates with single-cell immune signatures. *Science Translational Medicine*, 6(255):255ra131–255ra131, 2014. PMID: 25253674.

[37] R. Melchiotti, F. Gracio, S. Kordasti, A. K. Todd, and E. de Rinaldis. Cluster stability in the analysis of mass cytometry data. *Cytometry Part A*, 91(1):73–84, 2017.

[38] C. Perou, T. Sorlie, M. Eisen, M. van de Rijn, S. Jeffrey, C. Rees, J. Pollack, D. Ross, H. Johnsen, L. Akslen, O. Fluge, A. Pergamenschikov, C. Williams, S. Zhu, P. Lonning, A.-L. Borrensen-Dale, P. Brown, and D. Botstein. Molecular portraits of human breast tumors. *Nature*, 406(6797):747—-52, 2000.

[39] E. Shapiro, T. Biezuner, and S. Linnarsson. Single-cell sequencing-based technologies will revolutionize whole-organism science. *Nature Reviews Genetics*, 14(9):618–630, 2013. PMID: 23897237.

[40] O. Stegle, S. A. Teichmann, and J. C. Marioni. Computational and analytical challenges in single-cell transcriptomics. *Nature Reviews Genetics*, 16(3):133–145, 2015. PMID: 25628217.

[41] R. Cuevas-Diaz Duran, H. Wei, and J. Q. Wu. Single-cell RNA-sequencing of the brain. *Clinical and Translational Medicine*, 6(1):20, dec 2017.

[42] S. Anders, P. T. Pyl, and W. Huber. HTSeq–a Python framework to work with high-throughput sequencing data. *Bioinformatics*, 31(2):166–169, jan 2015.

[43] A. K. Shalek, R. Satija, X. Adiconis, R. S. Gertner, J. T. Gaublomme, R. Raychowdhury, S. Schwartz, N. Yosef, C. Malboeuf, D. Lu, J. J. Trombetta, D. Gennert, A. Gnirke, A. Goren, N. Hacohen, J. Z. Levin, H. Park, and A. Regev. Single-cell transcriptomics reveals bimodality in expression and splicing in immune cells. *Nature*, 498(7453):236–240, 2013. PMID: 23685454.

[44] L. Jiang, F. Schlesinger, C. A. Davis, Y. Zhang, R. Li, M. Salit, T. R. Gingeras, and B. Oliver. Synthetic spike-in standards for RNA-seq experiments. *Genome research*, 21(9):1543–51, sep 2011. PMID: 21816910.

[45] D. J. McCarthy, K. R. Campbell, A. T. Lun, and Q. F. Wills. Scater: Pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R. *Bioinformatics*, 33(8):1179–1186, 2017. PMID: 28088763.

[46] A. Cron, C. Gouttefangeas, J. Frelinger, L. Lin, S. K. Singh, C. M. Britten, M. J. Welters, S. H. van der Burg, M. West, and C. Chan. Hierarchical Modeling for Rare Event Detection and Cell Subset Alignment across Flow Cytometry Samples. *PLoS Computational Biology*, 9(7), 2013. PMID: 23874174.

[47] R. Blagus and L. Lusa. Class prediction for high-dimensional class-imbalanced data. *BMC Bioinformatics*, 11(1):523, oct 2010.

[48] W.-j. J. Lin and J. J. Chen. Class-imbalanced classifiers for high-dimensional data. *Briefings in Bioinformatics*, 14(1):13–26, 2013. PMID: 22408190.

[49] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20, jun 2004.

[50] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[51] B. Anchang, M. T. Do, X. Zhao, and S. K. Plevritis. CCAST: A Model-Based Gating Strategy to Isolate Homogeneous Subpopulations in a Heterogeneous Population of Single Cells. *PLoS Computational Biology*, 10(7):13–17, 2014. PMID: 25078380.

[52] G. Lee, L. Stoolman, and C. Scott. Transfer Learning for Auto-gating of Flow Cytometry Data. *JMLR(workshop)*, pages 155–166, 2012.

[53] N. A. Campbell. Robust Procedures in Multivariate Analysis I: Robust Covariance Estimation. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 29(3):231–237, 1980.

[54] J. S. Marron and M. P. Wand. Robust M-Estimators of Multivariate Location and Scatter. *The Annals of Statistics*, 4(1):51–67, 1976.

[55] A. Leung, M. Danilov, V. Yohai, and R. Zamar. *GSE: Robust Estimation in the Presence of Cellwise and Casewise Contamination and Missing Data*, 2016. R package version 4.1.

[56] E. W. Weisstein. Eigenvalue.

[57] G. Stewart. Matrix Algorithms, vol. 1. *SIAM, Philadelphia*, page 188, 1998.

[58] M. Wand. *KernSmooth: Functions for Kernel Smoothing Supporting Wand and Jones*, 2015. R package version 2.23-15.

[59] D. Handian, I. F. Nasrulloh, and L. S. Riza. *gradDescent: Gradient Descent for Regression Tasks*, 2017. R package version 2.0.1.

[60] D. Campana and E. Coustan-Smith. Detection of minimal residual disease in acute leukemia by flow cytometry. *Cytometry*, 38(4):139–152, aug 1999.

[61] A. Rundberg Nilsson, D. Bryder, and C. J. H. Pronk. Frequency determination of rare populations by flow cytometry: A hematopoietic stem cell perspective. *Cytometry Part A*, 83(8):721–727, 2013. PMID: 23839904.

[62] G. Finak, W. Jiang, K. Krouse, C. Wei, I. Sanz, D. Phippard, A. Asare, S. C. De Rosa, S. Self, and R. Gottardo. High-throughput flow cytometry data normalization for clinical trials. *Cytometry Part A*, 85(3):277–286, mar 2014.

[63] P. B. Gupta, C. M. Fillmore, G. Jiang, S. D. Shapira, K. Tao, C. Kuperwasser, and E. S. Lander. Stochastic state transitions give rise to phenotypic equilibrium in populations of cancer cells. *Cell*, 146(4):633–644, 2011. PMID: 21854987.

[64] W. J. Lin and J. J. Chen. Class-imbalanced classifiers for high-dimensional data. *Briefings in Bioinformatics*, 14(1):13–26, 2013.

[65] R. Blagus and L. Lusa. Class prediction for high-dimensional class-imbalanced data. *BMC Bioinformatics*, 11(1):523–523, 2010.

[66] S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data. *Journal of the American Statistical Association*, 97(457):77–87, 2002.

[67] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[68] V. Vapnik. *Statistical learning theory*. Wiley, New York, 1 edition, 1998.

[69] S. C. Bendall, E. F. Simonds, P. Qiu, E. a. D. Amir, P. O. Krutzik, R. Finck, R. V. Bruggner, R. Melamed, A. Trejo, O. I. Ornatsky, R. S. Balderas, S. K. Plevritis, K. Sachs, D. Pe'er, S. D. Tanner, G. P. Nolan, A. D. Amir el, P. O. Krutzik, R. Finck, R. V. Bruggner, R. Melamed, A. Trejo, O. I. Ornatsky, R. S. Balderas, S. K. Plevritis, K. Sachs, D. Pe'er, S. D. Tanner, and G. P. Nolan. Single-Cell Mass Cytometry of Differential Immune and Drug Responses Across a Human Hematopoietic Continuum. *Science*, 332(6030):687–696, 2011.

[70] E. R. Zunder, E. Lujan, Y. Goltsev, M. Wernig, and G. P. Nolan. A continuous molecular roadmap to iPSC reprogramming through progression analysis of single-cell mass cytometry. *Cell Stem Cell*, 16(3):323–337, 2015.

[71] B. Gaudillière, G. K. Fragiadakis, R. V. Bruggner, M. Nicolau, R. Finck, M. Tingle, J. Silva, E. A. Ganio, C. G. Yeh, W. J. Maloney, J. I. Huddleston, S. B. Goodman, M. M. Davis, S. C. Bendall, W. J. Fantl, M. S. Angst, and G. P. Nolan. Clinical recovery from surgery correlates with single-cell immune signatures. *Science translational medicine*, 6(255):255ra131–255ra131, 2014.

[72] J. H. Levine, E. F. Simonds, S. C. Bendall, K. L. Davis, E. A. D. Amir, M. D. Tadmor, O. Litvin, H. G. Fienberg, A. Jager, E. R. Zunder, R. Finck, A. L. Gedman, I. Radtke, J. R. Downing, D. Pe'er, and G. P. Nolan. Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis. *Cell*, 162(1):184–197, 2015.

[73] M. H. Spitzer, P. F. Gherardini, G. K. Fragiadakis, N. Bhattacharya, R. T. Yuan, A. N. Hotson, R. Finck, Y. Carmi, E. R. Zunder, W. J. Fantl, S. C. Bendall, E. G. Engleman, and G. P. Nolan. IMMUNOLOGY. An interactive reference framework for modeling a dynamic immune system. *Science (New York, N.Y.)*, 349(6244):1259425–1259425, 2015.

[74] E. Newell, N. Sigal, S. Bendall, G. Nolan, and M. Davis. Cytometry by Time-of-Flight Shows Combinatorial Cytokine Expression and Virus-Specific Cell Niches within a Continuum of CD8+ T Cell Phenotypes. *Immunity*, 36(1):142–152, 2012.

[75] M. Wong, J. Chen, S. Narayanan, W. Lin, R. Anicete, H. Kiaang, M. DeăLafaille, M. Poidinger, and E. Newell. Mapping the Diversity of Follicular Helper T Cells in Human Blood and Tonsils Using High-Dimensional Mass Cytometry Analysis. *Cell Reports*, 11(11):1822–1833, 2015.

[76] L. J. P. van der Maaten and G. E. Hinton. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85–85, 2008.

[77] L. Haghverdi, F. Buettner, and F. J. Theis. Diffusion maps for high-dimensional single-cell analysis of differentiation data. *Bioinformatics*, 31(18):2989–2998, 2014.

[78] S. C. Bendall, K. L. Davis, E. A. D. Amir, M. D. Tadmor, E. F. Simonds, T. J. Chen, D. K. Shenfeld, G. P. Nolan, and D. Pe'Er. Single-cell trajectory detection uncovers progression and regulatory coordination in human b cell development. *Cell*, 157(3):714–725, 2014.

[79] G. Finak and M. Jiang. *flowWorkspace: Infrastructure for representing and interacting with the gated cytometry*, 2011. R package version 3.26.3.

[80] D. R. Parks, M. Roederer, and W. A. Moore. A new "Logicle" display method avoids deceptive effects of logarithmic scaling for low signals and compensated data. *Cytometry Part A*, 69A(6):541–551, jun 2006.

[81] K. Lo, R. R. Brinkman, and R. Gottardo. Automated gating of flow cytometry data via robust model-based clustering. *Cytometry Part A*, 73A(4):321–332, apr 2008.

[82] G. Finak, J.-M. Perez, A. Weng, and R. Gottardo. Optimizing transformations for automated, high throughput analysis of flow cytometry data. *BMC Bioinformatics*, 11(1):546, 2010. PMID: 21050468.

[83] FlowCAP - Flow Cytometry: Critical Assessment of Population Identification Methods.

[84] P. Qiu. Computational prediction of manually gated rare cells in flow cytometry data. *Cytometry. Part A : the journal of the International Society for Analytical Cytology*, 87(7):594–602, 2015. PMID: 25755118.

[85] S. A. Kalams, S. D. Parker, M. Elizaga, B. Metch, S. Edupuganti, J. Hural, S. De Rosa, D. K. Carter, K. Rybczyk, I. Frank, J. Fuchs, B. Koblin, D. H. Kim, P. Joseph, M. C. Keefer, L. R. Baden, J. Eldridge, J. Boyer, A. Sherwat, M. Cardinali, M. Allen, M. Pensiero, C. Butler, A. S. Khan, J. Yan, N. Y. Sardesai, J. G. Kublin, and D. B. Weiner. Safety and Comparative Immunogenicity of an HIV-1 DNA Vaccine in Combination with Plasmid Interleukin 12 and Impact of Intramuscular Electroporation for Delivery. *The Journal of Infectious Diseases*, 208(5):818–829, sep 2013.

[86] K. Lu, X. Heng, and M. F. Summers. Structural determinants and mechanism of HIV-1 genome packaging. *Journal of molecular biology*, 410(4):609–33, jul 2011. PMID: 21762803.

[87] G. Li and E. De Clercq. HIV Genome-Wide Protein Associations: a Review of 30 Years of Research. *Microbiology and molecular biology reviews : MMBR*, 80(3):679–731, sep 2016. PMID: 27357278.

[88] G. Li, S. Piampongsant, N. R. Faria, A. Voet, A.-C. Pineda-Peña, R. Khouri, P. Lemey, A.-M. Vandamme, and K. Theys. An integrated map of HIV genome-wide variation from a population perspective. *Retrovirology*, 12(1):18, dec 2015.

[89] A. Scialdone, K. N. Natarajan, L. R. Saraiva, V. Proserpio, S. A. Teichmann, O. Stegle, J. C. Marioni, and F. Buettner. Computational assignment of cell-cycle stage from single-cell transcriptome data. *Methods*, 85:54–61, 2015. PMID: 26142758.

[90] F. Buettner, K. N. Natarajan, F. P. Casale, V. Proserpio, A. Scialdone, F. J. Theis, S. a. Teichmann, J. C. Marioni, and O. Stegle. Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. *Nature biotechnology*, 33(2):155–60, 2015. PMID: 25599176.

[91] P. Brennecke, S. Anders, J. K. Kim, A. A. Kołodziejczyk, X. Zhang, V. Proserpio, B. Baying, V. Benes, S. A. Teichmann, J. C. Marioni, and M. G. Heisler. Accounting for technical noise in single-cell RNA-seq experiments. *Nature Methods*, 10(11):1093–1098, 2013. PMID: 24056876.

[92] Y. Sasagawa, I. Nikaido, T. Hayashi, H. Danno, K. D. Uno, T. Imai, and H. R. Ueda. Quartz-Seq: a highly reproducible and sensitive single-cell RNA sequencing method, reveals non-genetic gene-expression heterogeneity. *Genome Biology*, 14(4):3097, 2013. PMID: 23594475.

[93] A. Conesa, P. Madrigal, S. Tarazona, D. Gomez-Cabrero, A. Cervera, A. McPherson, M. W. Szcześniak, D. J. Gaffney, L. L. Elo, X. Zhang, and A. Mortazavi. A survey of best practices for RNA-seq data analysis. *Genome Biology*, 17(1):13, dec 2016.

[94] R. A. Irizarry, B. Hobbs, F. Collin, Y. D. Beazer?Barclay, K. J. Antonellis, U. Scherf, and T. P. Speed. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, 4(2):249–264, apr 2003.

[95] J. A. Gagnon-Bartsch and T. P. Speed. Using control genes to correct for unwanted variation in microarray data. *Biostatistics*, 13(3):539–552, jul 2012.

[96] D. Risso, J. Ngai, T. P. Speed, and S. Dudoit. Normalization of RNA-seq data using factor analysis of control genes or samples. *Nature Biotechnology*, 32(9):896–902, sep 2014.

[97] R. Shippy, S. Fulmer-Smentek, R. V. Jensen, W. D. Jones, P. K. Wolber, C. D. Johnson, P. S. Pine, C. Boysen, X. Guo, E. Chudin, Y. A. Sun, J. C. Willey, J. Thierry-Mieg, D. Thierry-Mieg, R. A. Setterquist, M. Wilson, A. B. Lucas, N. Novoradovskaya, A. Papallo, Y. Turpaz, S. C. Baker, J. A. Warrington, L. Shi, and D. Herman. Using RNA sample titrations to assess microarray platform performance and normalization techniques. *Nature biotechnology*, 24(9):1123–31, sep 2006. PMID: 16964226.

[98] J. T. Leek, R. B. Scharpf, H. C. Bravo, D. Simcha, B. Langmead, W. E. Johnson, D. Geman, K. Baggerly, and R. A. Irizarry. Tackling the widespread and critical impact of batch effects in high-throughput data. *Nature Reviews Genetics*, 11(10):733–739, oct 2010.

[99] M.-A. Dillies, A. Rau, J. Aubert, C. Hennequet-Antier, M. Jeanmougin, N. Servant, C. Keime, G. Marot, D. Castel, J. Estelle, G. Guernec, B. Jagla, L. Jouneau, D. Laloe, C. Le Gall, B. Schaeffer, S. Le Crom, M. Guedj, and F. Jaffrezic. A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis. *Briefings in Bioinformatics*, 14(6):671–683, nov 2013.

[100] R. Bacher, L. F. Chu, N. Leng, A. P. Gasch, J. A. Thomson, R. M. Stewart, M. Newton, and C. Kendziorski. SCnorm: Robust normalization of single-cell RNA-seq data. *Nature Methods*, 14(6):584–586, 2017. PMID: 28418000.

[101] C. A. Vallejos, D. Risso, A. Scialdone, S. Dudoit, and J. C. Marioni. Normalizing single-cell RNA sequencing data: Challenges and opportunities. *Nature Methods*, 14(6):565–571, 2017. PMID: 28504683.

[102] B. Wang, J. Zhu, E. Pierson, D. Ramazzotti, and S. Batzoglou. Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning. *Nature Methods*, 14(4):414–416, 2017. PMID: 28263960.

[103] M. K. C. from Jed Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R Core Team, M. Benesty, R. Lescarbeau, A. Ziem, L. Scrucca, Y. Tang, C. Candan, and T. Hunt. *caret: Classification and Regression Training*, 2017. R package version 6.0-77.

[104] Q. Xu and Q. Yang. A Survey of Transfer and Multitask Learning in Bioinformatics. *Journal of Computing Science and Engineering*, 5(3):257–268, 2011.

[105] A. J. Brookes and P. N. Robinson. Human genotype-phenotype databases: aims, challenges and opportunities. *Nature Reviews Genetics*, 16(12):702–715, 2015.

[106] B. Lehner. Genotype to phenotype: lessons from model organisms for human genetics. *Nature Reviews Genetics*, 14(3):168–178, 2013.

[107] N. Collier, A. Oellrich, and T. Groza. Toward knowledge support for analysis and interpretation of complex traits. *Genome Biology*, 14:214–214, 2013.

[108] A. Oellrich, N. Collier, T. Groza, D. Rebholz-Schuhmann, N. Shah, O. Bodenreider, M. R. Boland, I. Georgiev, H. Liu, K. Livingston, A. Luna, A.-M. Mallon, P. Manda, P. N. Robinson, G. Rustici, M. Simon, L. Wang, R. Winnenburg, M. Dumontier, M. ? Dumontier, M. R. Boland, I. Georgiev, H. Liu, K. Livingston, A. Luna, A.-M. Mallon, P. Manda, P. N. Robinson, G. Rustici, M. Simon, L. Wang, R. Winnenburg, and M. Dumontier. The digital revolution in phenotyping. *Briefings in Bioinformatics, bbv083*, pages bbv083–bbv083, 2015.

[109] S. D. M. Brown, W. Wurst, R. Kühn, and J. M. Hancock. The Functional Annotation of Mammalian Genomes: The Challenge of Phenotyping. *Annual Review of Genetics*, 43(1):305–333, 2009.

[110] H. L. Allen, K. Estrada, G. Lettre, S. I. Berndt, M. N. Weedon, F. Rivadeneira, C. J. Willer, A. U. Jackson, S. Vedantam, S. Raychaudhuri, T. Ferreira, A. R. Wood, R. J. Weyant, A. V. Segrè, E. K. Speliotes, E. Wheeler, N. Soranzo, J.-H. Park, J. Yang, D. Gudbjartsson, N. L. Heard-Costa, J. C. Randall, L. Qi, A. V. Smith, R. Mägi, T. Pastinen, L. Liang, I. M. Heid, J. Luan, G. Thorleifsson, T. W. Winkler, M. E. Goddard, K. S. Lo, C. Palmer, T. Workalemahu, Y. S. Aulchenko, Å. Johansson, M. C. Zillikens, M. F. Feitosa, T. Esko, T. Johnson, S. Ketkar, P. Kraft, M. Mangino, I. Prokopenko, D. Absher, E. Albrecht, F. Ernst, N. L. Glazer, C. Hayward, J.-J. Hottenga, K. B. Jacobs, J. W. Knowles, Z. Kutalik, K. L. Monda, O. Polasek, M. Preuss, N. W. Rayner, N. R. Robertson, V. Steinthorsdottir, J. P. Tyrer, B. F. Voight, F. Wiklund, J. Xu, J. H. Zhao, D. R. Nyholt, N. Pellikka, M. Perola, J. R. B. Perry, I. Surakka, M.-L. Tammesoo, E. L. Altmaier, N. Amin, T. Aspelund, T. Bhangale, G. Boucher, D. I. Chasman, C. Chen, L. Coin, M. N. Cooper, A. L. Dixon, Q. Gibson, E. Grundberg, K. Hao, M. J. Junttila, L. M. Kaplan, J. Kettunen, I. R. König, T. Kwan, R. W. Lawrence, D. F. Levinson, M. Lorentzon, B. McKnight, A. P. Morris, M. Müller, J. S. Ngwa, S. Purcell, S. Rafelt, R. M. Salem, E. Salvi, et al. Hundreds of variants clustered in genomic loci and biological pathways affect human height. *Nature*, 467(7317):832–838, 2010.

[111] W. W. Busse and J. Lemanske, Robert F. Asthma. *New England Journal of Medicine*, 344(5):350–362, 2001.

[112] C. J. Mungall, G. V. Gkoutos, C. L. Smith, M. A. Haendel, S. E. Lewis, and M. Ashburner. Integrating phenotype ontologies across multiple species. *Genome Biology*, 11(1):1–1, 2010.

[113] S. Wenzel. Severe Asthma in Adults. *American Journal of Respiratory and Critical Care Medicine*, 2012.

[114] M. B. Lanktree, R. G. Hassell, P. Lahiry, and R. A. Hegele. Phenomics. *Journal of Investigative Medicine*, 58(5):700–706, 2010.

[115] R. Feil and M. F. Fraga. Epigenetics and the environment: emerging patterns and implications. *Nature Reviews Genetics*, 13(2):97–109, 2012.

[116] J. M. Raser and E. K. O'Shea. Noise in Gene Expression: Origins, Consequences, and Control. *Science*, 309(5743):2010–2013, 2005.

[117] S. L. Rutherford. From genotype to phenotype: buffering mechanisms and the storage of genetic information. *BioEssays*, 22(12):1095–1105, 2000.

[118] S. D. Horn and R. A. Horn. Reliability and Validity of the Severity of Illness Index. *Medical Care*, 24(2):159–178, 1986.

[119] N. Geifman and E. Rubin. Towards an Age-Phenome Knowledge-base. *BMC Bioinformatics*, 12(1):1–1, 2011.

[120] Z. Che, D. Kale, W. Li, M. T. Bahadori, and Y. Liu. Deep Computational Phenotyping. pages 507–516. ACM.

[121] Y. Fukazawa, H. Park, M. J. Cameron, F. Lefebvre, R. Lum, N. Coombes, E. Mahyari, S. I. Hagen, J. Y. Bae, M. D. Reyes, T. Swanson, A. W. Legasse, A. Sylwester, S. G. Hansen, A. T. Smith, P. Stafova, R. Shoemaker, Y. Li, K. Oswald, M. K. Axthelm, A. McDermott, G. Ferrari, D. C. Montefiori, P. T. Edlefsen, M. Piatak, J. D. Lifson, R. P. Sékaly, and L. J. Picker. Lymph node T cell responses predict the efficacy of live attenuated SIV vaccines. *Nature medicine*, 18(11):1673–81, 2012.

[122] R. S. Hotchkiss and I. E. Karl. The Pathophysiology and Treatment of Sepsis. *New England Journal of Medicine*, 348(2):138–150, 2003.

[123] A. M. Guth, W. J. Janssen, C. M. Bosio, E. C. Crouch, P. M. Henson, and S. W. Dow. Lung environment determines unique phenotype of alveolar macrophages. *American Journal of Physiology - Lung Cellular and Molecular Physiology*, 296(6):L936–L946, 2009.

[124] P. W. Kodituwakku. Defining the behavioral phenotype in children with fetal alcohol spectrum disorders: A review. *Neuroscience and Biobehavioral Reviews*, 31(2):192–201, 2007.

[125] A. P. Feinberg. Phenotypic plasticity and the epigenetics of human disease. *Nature*, 447(7143):433–440, 2007.

[126] C. Roadmap Epigenomics, A. Kundaje, W. Meuleman, J. Ernst, M. Bilenky, A. Yen, A. Heravi-Moussavi, P. Kheradpour, Z. Zhang, J. Wang, M. J. Ziller, V. Amin, J. W. Whitaker, M. D. Schultz, L. D. Ward, A. Sarkar, G. Quon, R. S. Sandstrom, M. L. Eaton, Y.-C. Wu, A. R. Pfenning, X. Wang, M. Claussnitzer, Y. Liu, C. Coarfa, R. A. Harris, N. Shoresh, C. B. Epstein, E. Gjoneska, D. Leung, W. Xie, R. D. Hawkins, R. Lister, C. Hong, P. Gascard, A. J. Mungall, R. Moore, E. Chuah, A. Tam, T. K. Canfield, R. S. Hansen, R. Kaul, P. J. Sabo, M. S. Bansal, A. Carles, J. R. Dixon, K.-H. Farh, S. Feizi, R. Karlic, A.-R. Kim, A. Kulkarni, D. Li, R. Lowdon, G. Elliott, T. R. Mercer, S. J. Neph, V. Onuchic, P. Polak, N. Rajagopal, P. Ray, R. C. Sallari, K. T. Siebenthall, N. A. Sinnott-Armstrong, M. Stevens, R. E. Thurman, J. Wu, B. Zhang, X. Zhou, A. E. Beaudet, L. A. Boyer, P. L. De Jager, P. J. Farnham, S. J. Fisher, D. Haussler, S. J. M. Jones, W. Li, M. A. Marra, M. T. McManus, S. Sunyaev, J. A. Thomson, T. D. Tlsty, L.-H. Tsai, W. Wang, R. A. Waterland, M. Q. Zhang, L. H. Chadwick, B. E. Bernstein, J. F. Costello, J. R. Ecker, M. Hirst, A. Meissner, A. Milosavljevic, B. Ren, J. A. Stamatoyannopoulos, T. Wang, and M. Kellis. Integrative analysis of 111 reference human epigenomes. *Nature*, 518(7539):317–330, 2015.

[127] P. Groth and B. Weiss. Phenotype Data: A Neglected Resource in Biomedical Research?, 2006.

[128] P. N. Robinson. Deep phenotyping for precision medicine. *Human Mutation*, 33(5):777–780, 2012.

[129] G. Hripcsak and D. J. Albers. Next-generation phenotyping of electronic health records. *Journal of the American Medical Informatics Association*, 20(1):117–121, 2013.

[130] R. P. Tracy. ?Deep phenotyping?: characterizing populations in the era of genomics and systems biology. *Current Opinion in Lipidology*, 19(2):151–157, 2008.

[131] M. Uschold and M. Gruninger. Ontologies: Principles, methods and applications. *KNOWLEDGE ENGINEERING REVIEW*, 11:93–136, 1996.

[132] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, G. Sherlock, and G. ? Sherlock. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

[133] J. B. L. Bard and S. Y. Rhee. Ontologies in biology: design, applications and future challenges. *Nature Reviews Genetics*, 5(3):213–222, 2004.

[134] G. V. Gkoutos, E. C. J. Green, A.-M. Mallon, J. M. Hancock, and D. Davidson. Using ontologies to describe mouse phenotypes. *Genome Biology*, 6(1):1–1, 2004.

[135] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, S. ? Lewis, L. J. Goldberg, K. Eilbeck, A. Ireland, C. J. Mungall, N. Leontis, P. Rocca-Serra, A. Ruttenberg, S.-A. Sansone, R. H. Scheuermann, N. Shah, P. L. Whetzel, and S. Lewis. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11):1251–1255, 2007.

[136] S. Köhler, S. C. Doelken, C. J. Mungall, S. Bauer, H. V. Firth, I. Bailleul-Forestier, G. C. M. Black, D. L. Brown, M. Brudno, J. Campbell, D. R. FitzPatrick, J. T. Eppig, A. P. Jackson, K. Freson, M. Girdea, I. Helbig, J. A. Hurst, J. Jähn, L. G. Jackson, A. M. Kelly, D. H. Ledbetter, S. Mansour, C. L. Martin, C. Moss, A. Mumford, W. H. Ouwehand, S.-M. Park, E. R. Riggs, R. H. Scott, S. Sisodiya, S. V. Vooren, R. J. Wapner, A. O. M. Wilkie, C. F. Wright,

A. T. V.-v. Silfhout, N. d. Leeuw, B. B. A. d. Vries, N. L. Washingthon, C. L. Smith, M. Westerfield, P. Schofield, B. J. Ruef, G. V. Gkoutos, M. Haendel, D. Smedley, S. E. Lewis, P. N. Robinson, and P. N. ? Robinson. The Human Phenotype Ontology project: linking molecular biology and disease through phenotype data. *Nucleic Acids Research*, 42(D1):D966–D974, 2014.

[137] P. N. Robinson and C. Webber. Phenotype Ontologies and Cross-Species Analysis for Translational Research. *PLoS Genetics*, 10(4):e1004268, apr 2014. PMID: 24699242.

[138] J. T. Eppig, J. E. Richardson, J. A. Kadin, C. L. Smith, J. A. Blake, C. J. Bult, A. Anagnostopoulos, R. M. Baldarelli, J. S. Beal, S. M. Bello, J. Berghout, O. Blodgett, N. Butler, J. W. Campbell, L. E. Corbani, H. Dene, H. J. Drabkin, K. L. Forthofer, P. Frost, S. C. Giannatto, P. Hale, L. Hutchins, M. Knowlton, M. Y. Law, J. R. Lewis, M. McAndrews, D. B. Miers, H. Motenko, L. Ni, H. Onda, J. E. Ormsby, W. Pitman, J. M. Recla, B. Richards-Smith, D. R. Shaw, D. Sitnikov, K. R. Stone, M. Tomcuk, L. L. Washburn, and Y. Zhu. Mouse Genome Database: From sequence to phenotypes and disease models. *Genesis*, 53(8):458–473, aug 2015. PMID: 26150326.

[139] X. Chen and J. Zhang. The Ortholog Conjecture Is Untestable by the Current Gene Ontology but Is Supported by RNA Sequencing Data. *PLOS Computational Biology*, 8(11):e1002784–e1002784, 2012.

[140] A. M. Altenhoff, R. A. Studer, M. Robinson-Rechavi, and C. Dessimoz. Resolving the Ortholog Conjecture: Orthologs Tend to Be Weakly, but Significantly, More Similar in Function than Paralogs. *PLOS Computational Biology*, 8(5):e1002514–e1002514, 2012.

[141] M. A. Ferguson-Smith. Karyotype-phenotype Correlations in Gonadal Dysgenesis and Their Bearing on the Pathogenesis of Malformations. *Journal of Medical Genetics*, 2(2):142–142, 1965.

[142] D. Botstein and N. Risch. Discovering genotypes underlying human phenotypes: past successes for mendelian disease, future approaches for complex disease. *Nature Genetics*, 33:228–237, 2003.

[143] M. A. Hamburg and F. S. Collins. The Path to Personalized Medicine. *New England Journal of Medicine*, 363(4):301–304, 2010.

[144] R. Mirnezami, J. Nicholson, and A. Darzi. Preparing for Precision Medicine. *New England Journal of Medicine*, 366(6):489–491, 2012.

[145] A. M. Valdes, S. K. McWeeney, and G. Thomson. Evidence for linkage and association to alcohol dependence on chromosome 19. *Genetic Epidemiology*, 17(S1):S367–S372, 1999.

[146] R. Hitzemann, S. Edmunds, W. Wu, B. Malmanger, N. Walter, J. Belknap, P. Darakjian, and S. McWeeney. Detection of reciprocal quantitative trait loci for acute ethanol withdrawal and ethanol consumption in heterogeneous stock mice. *Psychopharmacology*, 203(4):713–722, 2009.

[147] L. Kruglyak. The road to genome-wide association studies. *Nature Reviews Genetics*, 9(4):314–318, 2008.

[148] J. C. Denny, M. D. Ritchie, M. A. Basford, J. M. Pulley, L. Bastarache, K. Brown-Gentry, D. Wang, D. R. Masys, D. M. Roden, and D. C. Crawford. PheWAS: demonstrating the feasibility of a phenome-wide scan to discover gene?disease associations. *Bioinformatics*, 26(9):1205–1210, 2010.

[149] R. Hitzemann, D. Bottomly, P. Darakjian, N. Walter, O. Iancu, R. Searles, B. Wilmot, and S. McWeeney. Genes, behavior and next-generation RNA sequencing. *Genes, Brain and Behavior*, 12(1):1–12, 2013.