UHASSELT
KNOWLEDGE IN ACTION

Maastricht University

Faculty of Sciences
School for Information Technology
Master of Statistics and Data Science

Master's thesis

Feature Discovery in Small-Sized Experiments in Early Drug Development

Mathijs van Westendorp

Thesis presented in fulfillment of the requirements for the degree of Master of Statistics and Data Science, specialization Biostatistics

SUPERVISOR :

Prof. dr. Olivier THAS

2020
2021

Transnational University Limburg is a unique collaboration of two universities in two countries: the University of Hasselt and Maastricht University.

UHASSELT
KNOWLEDGE IN ACTION

www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

Maastricht University

Faculty of Sciences
School for Information Technology
Master of Statistics and Data Science

Master's thesis

Feature Discovery in Small-Sized Experiments in Early Drug Development

Mathijs van Westendorp

Thesis presented in fulfillment of the requirements for the degree of Master of Statistics and Data Science, specialization Biostatistics

SUPERVISOR :

Prof. dr. Olivier THAS

Feature Discovery in Small-Sized Experiments in Early Drug Development

Thesis for the Master in Statistics (UHasselt)

Mathijs van Westendorp (1953773)

Contents

1 Abstract

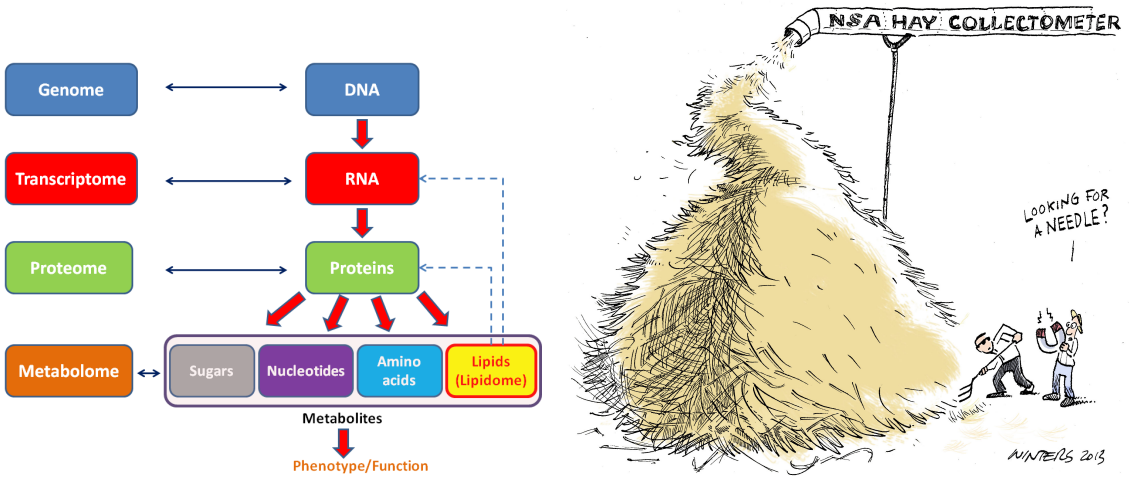
The abstract consists of the thesis' description from the book of abstracts because it does such an excellent job of summarising the research done in this thesis.

> "Nowadays we hear a lot about artificial intelligence (AI) and machine learning (ML) as methods for predicting outcomes. In medical and pharmaceutical sectors, these methods have also become very popular. We will focus on prediction problems in preclinical discovery research in pharmaceutical companies. In this stage of the drug development, no large data sets are available. That is: only a small number of observations on a very large number of features (e.g. data on thousands of gene expression, metabolomics, ... on only 10 to 50 subjects). Scientists use such datasets aiming at finding predictors (e.g. genes) to predict disease status (diagnostics) or to identify responders to a treatment for a disease (personalised medicine). Many of these scientists believe in the power of AI and ML, whereas, however, the success stories of AI and ML come from big data applications (i.e. data from thousands of subjects in the training data).

The goal of this thesis is to study the behavior of AI and ML methods on small sample-sized studies in early drug development. It involves the use of many ML prediction methods in small sample-sized simulation studies. The ultimate goal is to formulate guidelines."

2 Introduction

The advancements in sequencing technology since the Human Genome Project successfully completed in 2003 have led to sharply increased sequencing speeds at dramatic cost reductions. The advancements in technology not only lowers the barrier for utilising sequencing in individual patients but also leads to the possibility of sequencing additional sources of omics data. The omics field has grown in scope to include omics such as transcriptomics (concerning the RNA expression of the genome), proteomics (related to the proteins resulting from the RNA expression) and metabolomics (on the resulting metabolites). Whereas the human genome consists of approximately 20-25 thousand genes, proteomics concerns the study of a number of proteins approximating between 80-400 thousand proteins.[1]



(a) Omics hierarchy (source: Wikipedia)

(b) Haystack (source: Winters 2013

Figure 1: A haystack of Omics data.

Referring to the expression of genes, proteins, metabolites, ... as features, it is evident that sequencing can provide us with a large number of features. As these features are involved in different biological processes, which in turn can be involved in varying diseases when their function is impaired, there is an interest among scientists to identify features related to these processes and diseases. The hope being that pharmaceuticals could be developed that modify these pathways in such a way that these impact disease progression or reverse malign processes (i.e. cure the disease).

Pharmaceuticals are developed in multiple stages, going from discovery in the lab, preclinical research, clinical research, regulatory approval, to post-market safety monitoring and economic feasibility assessments.[2] Focusing on early drug discovery, the aim is to find features that predict treatment effect. The data that is available in this phase of the development of pharmaceuticals consists of few samples: typically between 15-50 samples are available for analysis. The datasets are gathered from previous experiments, where a response was measured for a pharmaceutical, leading to a group of responders and a group of non-responders. Having the possibility of using sequencing techniques, the goal is to gather features that can make a distinction

[1] While estimates, these numbers illustrate the size of potential number of features that the omics fields can supply.

[2] An overview can be found on https://www.fda.gov/patients/learn-about-drug-and-device-approvals/drug-development-process. Evidently, the process cannot be fully discussed within the scope of this thesis given the involvement of a variety of disciplines including: medicine, biology, chemistry, ethics, legal, economics,...

between these two groups, in doing so allowing for further research into that specific feature. However, it is known that few (if any) features may actually be predictive for a specific pharmaceutical. Given the possibility of adding more features, a possible strategy can be to include as many features as one can in the hopes of increasing the chance of having the predictive feature among the features that are analysed. The idea being that adding information increases the prospects of finding a predictive biomarker.[3] However, there are downsides to this approach since it increases the number of non-predictive features in the dataset, which means that when using selection methods like hypothesis tests, lead to a problem of multiple testing. Testing at a certain level of significance for the many features in these datasets entails a correction that is very high, or many hypothesis tests that are false positives, leading to a large Type I or Type II errors.

Making an analogy with the finding of a needle in the haystack, we can imagine a barn full of hay with a few needles that we would like to find. To do this we need to take hay outside for inspection in the farm's courtyard. So using an appropriate agricultural vehicle we pick the desired number of bales which we then inspect.

In this analogy the strategy described above would mean that we would take more bales from the barn to the courtyard, which would evidently increase the probability that these bales include the needles that are in the barn. However, this would also increase the effort required to find the needle in the courtyard, since the amount of hay to look through is larger.

Besides developments in sequencing, another influential development is the resurgence of the machine learning (ML) and artificial intelligence (AI) field through the success of deep learning algorithms in areas such as image recognition, natural language processing and performance in (computer) games such as Go and Starcraft Silver et al. (2016). Different authors, Boniolo et al. (2021), see (great) potential for applying AI in pharmaceutical development, including the applications related to omics data and feature discovery. However, there are several barriers for utilising AI for this purpose, one of which is the requirement of an extensive number of samples to train the models on. However, as mentioned above, the number of samples available in early drug discovery is very small, especially compared to datasets consisting of millions or even billions of samples that these "successful" AI models use. Nonetheless, the aforementioned leads to questions for statisticians involved in early drug discovery about applying AI/ML methods to the problem of feature selections in the small sample datasets that they have available. The reasoning being that AI can deal with adding more features and in the end lead to higher performance in this area. The question is whether the perceived benefit of utilising AI for this purpose is realistic, and if so what the requirements are for obtaining performance benefits in compared to more "traditional" methods.

2.1 Research objective

In this thesis the objective is to assess the above from a statistical perspective. As such, a research objective needs to be formulated in a statistical terminology.

The omics data in early drug development consists of a small number of samples (n), but is high dimensional (p), which means that $p \gg n$. More specifically, the datasets consist of n samples in the range of 10 to 50, with a high dimensions p in the range of 10.000 to 500.000. For this small-sized, high dimensional data we are interested in selecting candidate genes that are predictive for a treatment effect.

As the analogy above illustrates, a higher number of features may not necessarily lead to better results. However, it is not clear how to balance between ensuring that the selected features contain the predictive feature and the ability of identifying such a predictive feature from the set of selected features. Related to this there is the question of the influence of the number of predictive features in the dataset. Is the ratio of the total number of features over the number of predictive features indicative of performance or is the absolute number of predictive features a better indicator for success. While it is often not feasible to substantially increase sample sizes, it is of interest to assess whether there is a difference in performance over the range of sample sizes that are feasible. In addition, public omics data could provide further information

[3] A predictive biomarker is "[a] characteristic that is objectively measured and evaluated as an indicator of normal biological processes, pathogenic processes, or pharmacologic responses to a therapeutic intervention [...] that forecasts the likely response to a specific treatment"(Buyse et al. 2010, 310).

on features in the study. This information would be integrated into the study through a method called 'data integration.' There are different methods to identify predictive features. Since the expectations concerning AI/ML methods are high, a comparison between such methods with other (traditional and contemporary) methods is of interest.

2.1.1 Guidelines for feature selection in small-sample, high-dimensional, datasets

The above leads to the following overall aim, which is to

> **formulate guidelines for the use of different methods in small sample-sized, high dimensional studies in early drug development**.

The guidelines will be on the influence on the performance of selecting predictive features by the following elements:

- the total number of features,
- the number of (expected) predictive features,
- the effect size of these predictive features,
- the sample size of both the responders and non-responders groups, and
- the feasibility of using data integration to pre-select candidate features through the use of existing (public) information on the feature in question.

3 Methods

To formulate guidelines we use simulations to obtain datasets where the predictive features and responders are known. Next, we use a range of methods for their ability to identify the predictive features. This section describes the method of simulating the data, the metrics used to score the results and the different methods that are used to identify predictive features. Further, the data integration method is explained, which attempts to use pre-existing data available (in this case this is also simulated data).

3.1 Simulating the sequencing data

The reason for performing a simulation is that simulation studies allow for controlling the elements of interest, which are relevant in formulating the guidelines.

The simulated data represents sequencing data, which can be the result of next generation sequencing techniques. Because the interest is in higher number of features, the interest lies in not only genomic features but also transcriptomics. For that reason, RNA-sequencing data is simulated using one of the packages available on Bioconductor. First however, the methods are applied to a more theoretical source of simulated data, namely a shifted 2 component Gaussian mixture distribution.

The resulting datasets are stored as Parquet files in separate folders. The metadata is kept in a SQL database, which includes the random seed, number of the generation run, the type of method used for the simulation and the descriptive information of the generated study. The information consists of the following parameters:

- the number of predictive features (n_{pred}),
- the number of features ($n_t = n_{pred} + n_{non\_pred}$),
- the shift in mean between non-predictive and predictive features (μ_{pred}),[4]
- the variance of the predictive features (s^2_{pred}), and

[4]For the `SPsimSeq` package the meaning of this parameter changes given the differences in the underlying method

- the group size for non-responders (n_1) and responders (n_2).

The parameters are iteratively changed to focus on an area of interest.[5] This means that for example, to assess the effect of the total number of genes, only that variable can be varied over a range with a choice of steps, while keeping other parameter values constant.

3.1.1 Simulations using normal distributions

This method generates the data as depicted in Table 1 where the distribution for the specific predictive feature (P) and non-predictive feature (NP) for responders (R) and non-responders (NR) are depicted.

Table 1: A schematic representation of the distributions per feature in the non-responder (NR) and responder (R) groups and for the non-predictive (NP) and predictive (P) features.

	NP_1	$\dots$	$NP_{n_t - n_{pred}}$	P_1	$\dots$	$P_{n_{pred}}$
R_1	$N(0,1)$	$\dots$	$N(0,1)$	$N(\mu_{pred}, s^2_{pred})$	$\dots$	$N(\mu_{pred}, s^2_{pred})$
$\vdots$	$\vdots$	$\ddots$	$N(0,1)$	$\vdots$	$\ddots$	$\vdots$
R_{n_2}	$N(0,1)$	$\dots$	$N(0,1)$	$N(\mu_{pred}, s^2_{pred})$	$\dots$	$N(\mu_{pred}, s^2_{pred})$
NR_1	$N(0,1)$	$\dots$	$N(0,1)$	$N(0,1)$	$\dots$	$N(0,1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
NR_{n_1}	$N(0,1)$	$\dots$	$N(0,1)$	$N(0,1)$	$\dots$	$N(0,1)$

The most basic simulation method involves a location shift of the value of the predictive features for the response group.[6] The data are generated using the following steps:

1. generate a matrix of $n_1 + n_2$ rows and n_t columns sampled from a standard Normal distribution,
2. replace n_1 rows and n_{pred} columns with samples from $N(\mu_{pred}, s^2_{pred})$ corresponding to the predictive features in the response group, and
3. shuffle the resulting columns (i.e. shuffle the features).

Shuffling is required for some methods that consider multiple genes in parallel. Such methods could exploit an artificial structure. For example when all the predictive features are in the bottom-right corner of the matrix, the method may become a predictor for that location in the matrix instead of the predictive features.

The result is a two component Gaussian mixture distribution where one component corresponds to the standard Normal distribution and the other to $N(\mu_{pred}, s^2_{pred})$.

3.1.2 Simulating RNA sequencing data using `SPsimSeq`

The method using the normal distributions described above is unlikely to encompass the complexity and dependencies in a realistic dataset. Therefore, the second simulation method uses the `SPsimSeq` package (Assefa et al. 2021). The package allows for simulating new datasets based on the estimated marginal distributions of an existing RNA sequencing dataset using Gaussian-copulas to retain the dependence between genes using the distribution of gene expression levels from real RNA sequencing data. The latter means that unlike the 'normal' simulation method, it possible to reflect the correlations between features that are involved in a specific biological process.

[5] The generation number is kept in the database of the results.
[6] See the code snippet in section A.1.1.

The package includes a subset of real RNA sequencing data that comes from an article by (Zhang et al. 2015), which is referred to as the "Zhang data" in the package's documentation and this thesis. The ability to use any real (RNA) sequencing data as a source offers the potential to use a range of data sources. However, for convenience this thesis uses the included Zhang dataset.

Similar to the 'normal' method, the predictive features can be created by adding a certain offset to the counts of the relevant features for the responder group. As the data is normalised, it is important to consider whether to add the offset before or after normalising. When done after normalisation, the mean and variance of the predictive features change, which could be used to create a perfect predictor. Evidently this is undesirable but does not preclude the use of such a method when the methods make no use of that specific information. This means that it is preferable to add the offset before normalising.[7]

An alternative is to set a threshold for the log-fold change for predictive features, using the `lfc.thrld` option. However, this entails that features not meeting this threshold are not included in the dataset. Another possibility is to use the package's functionality of specifying a list of features from the 'template' dataset that are differentially expressed.[8] The detractor for these options is that it is harder to control the effect size and total number of features (since the threshold can result in features being dropped from the dataset). In essence, it is convenient for using datasets where the predictive features are known and methods of feature selection need to be assessed for their effectiveness on a predictive feature with those characteristics. This is slightly different from the objective in this thesis of simulating over a range of parameters that may effect the performance of the methods of feature selection.

As such, we use the following steps to simulate the RNA-sequencing datasets using the `SPsimSeq` package.

1. For each study s, call the `SPsimSeq` function with the following study parameters:

- `s.data` with the object's name of the count data of the existing RNA sequencing data,
- `group` a vector containing the group to which a sample belongs,
- `pDE` the proportion of predictive features over the total number of features,
- `group.config` the desired group composition in the simulated dataset,
- `n.genes` the number of features to simulate,
- `tot.samples` the desired number of samples, and
- `lfc.thrld` indicating the minimal log-fold change for predictive features, which was set to a low value to ensure that the predictive features are not left out.

2. Add the location shift to a feature for the response group by using the mean value of that feature, which is multiplied by the $\mu_{pred,s}$ value to obtain a higher value for the predictive features for the responder group and

3. normalise the data (using the R function `scale`).

Because this simulation method is central to this thesis an excerpt from the R code with the implementation of these three steps is given here.

```
# Step 1
sim.data.bulk <- SPsimSeq(n.sim = 1, s.data = zhang.counts,
                          group = MYCN.status, n.genes = study$ngenes,
                          batch.config = 1,
                          pDE = study$pgenes / study$ngenes,
                          group.config = c(study$n0 / (study$n0 + study$n1),
                                           study$n1 / (study$n0 + study$n1)),
                          tot.samples = study$n0 + study$n1,
```

[7] As described in the results section, because a relatively large number of simulations ran where the offset was added after normalising, the results for those method were used despite this caveat.

[8] By providing a lists of null and non-null feature names to the `cand.DE.genes` argument of the `SPsimSeq` function.

```
                            lfc.thrld = 0.001)
# copy the data of interest from the object
df <- as.data.frame(sim.data.bulk$sim.data.list[[1]]$counts)

# Step 2
predictive <- sim.data.bulk$sim.data.list[[1]]$rowData$DE.ind
responder <- sim.data.bulk$sim.data.list[[1]]$colData$Group == '1'
# Add an effect to the predictive features in the responder group
df[predictive, responder] <- df[predictive, responder] +
  rowMeans(df[predictive,]) * study$mu_predictive

# Step 3
df <- as.data.frame(t(apply(df, 1, scale))) %>% replace(is.na(.), 0)
```

3.2 Performance metrics

The performance metrics should ideally provide a single value that represents the performance of a method of feature selection. These metrics should be usable for comparing the methods of feature selection in the context of early drug discovery.

There are a variety of metrics that could be used given that we are dealing with a binary classification problem. As such the basic elements of any performance metric are the true positives (TP), false positives (FP), true negatives (TN) and the false negatives (FN). Because we know the actual number of positives (P) and negative (N) in the dataset and each method of feature selection of predicts whether a feature is positive (PP) or negative (PN), we can combine these measures in different ways to form measures such as the sensitivity, specificity, negative/positive predictive values, false discovery rate (FDR), ... As all these measures can be reconstructed from the four basic elements mentioned above, these are stored as the results for every study and method of feature selection.

In the context of providing an indication of how likely a certain method is, under simulated circumstances, in properly selecting predictive features the precision and sensitivity are the most relevant measures.

The **precision**: $\frac{TP}{TP+FP}$ is the metric that indicates the expected number of true positives when selecting the top x genes ($x_{top} = TP + FP$). The precision is the same as $1 - FDR$, the FDR is often used in this context. However, it seems that in machine learning the combination of precision and recall (i.e. sensitivity) is often used, which makes it a matter of preference.

The **sensitivity** $\frac{TP}{P}$ provides a measurement of the number of predictive features discovered over the number of predictive features actually in the dataset ($P = n_{pred}$).

The precision and sensitivity are complementary measures. While they both look at the true discovery of predictive features, the precision indicates the performance including the selection method (i.e. the choice of x in the top x features). Instead, the sensitivity provides a measure incorporating the number of predictive features that we know are actually in the data.

3.3 Metrics of feature selection

Before discussing the methods used for feature selection, the metrics that are the result outcome of the different methods for feature selection need to be discussed. Herein there are two categories of values: the results of hypothesis tests, and other values that can be used to classify the features.

3.3.1 Hypothesis test metrics

Classical hypothesis testing involves the formulation of a null and alternative hypothesis, the latter which can be one-/two-sided, leading to the calculation of a test statistic and which is based on assumptions on the underlying distribution leads to the well-known and omnipresent p-value (Bijma, Jonker, and van der Vaart 2018, 123).

Statistical significance and p-values are closely related. Taking a p-value of $p = 0.05$ means that there is a 5% chance of falsely rejecting the null hypothesis (Bijma, Jonker, and van der Vaart 2018, 153). However, when multiple tests are conducted simultaneously this can lead to the **multiple testing problem**, where the probability of one incorrect rejections becomes much higher that the 0.05 probability of a false rejection of the null hypothesis for a single test. This can become a problem when this problem is not recognized as such, leading to false rejections of the null hypothesis.

3.3.1.1 Correcting for multiple testing There are different methods of correcting the p-values for multiple simultaneous tests, such as the well-known Bonferroni correction (Bijma, Jonker, and van der Vaart 2018, 153–54). However, especially for larger numbers of tests this correction can be overly conservative.

The issue is that it corrects for probability of *at least one* type 1 error. This may be too ambitious, especially for large scale hypothesis testing with a large number of simultaneous tests being performed (Efron and Hastie 2016, 271). The False Discovery Rate (FDR) instead aims at correcting at a proportion of the falsely rejected hypotheses among the rejected hypotheses. The Benjamini-Hochberg procedure can be used to control for the FDR by (i) ordering the p-values according to size, (ii) reject the null hypotheses corresponding to a jth order statistic and (iii) also reject null hypothesis with a p-value below that of the null hypotheses rejected in step ii. However, at a level where the proportion of correct null hypothesis is close to 1, the Benjamini-Hochberg procedure is also conservative (Bijma, Jonker, and van der Vaart 2018, 155).

The implications of the different procedures of correcting for multiple testing is that it results in a variable number of rejected null hypotheses. There is a balance between rejecting too many null hypotheses when taking the p-values at face value and too few rejections by being overly conservative in correcting for multiple testing. While there is evidently some middle ground to be found, it may be worthwhile to the consider the constraints resulting from the problem's context in which we use hypothesis testing.

3.3.2 SHAP metric for non-hypothesis test based metrics

In a realistic setting it is unknown whether a feature is predictive. However, it is known which samples correspond to responders. As such, we require a method that provides an insight into what features contribute to the classification outcome of a model, i.e. whether a set of features indicates a responder or non-responder and which subset of features is the most relevant for doing that.

The method we use in this thesis to "explain" the models are SHAP (SHapley Additive exPlanations) values (Scott M. Lundberg and Lee 2017). SHAP values are based on Shapley values, which are values used in game-theory that represent the contribution of a 'player' to a 'total payout.' A more formal definition is that

> "[t]he Shapley value is the average marginal contribution of a feature value across all possible coalitions."

In this definition, "coalitions" refers to the different set of features that can have different values. According to (Scott M. Lundberg and Lee 2017) the Shapley values are calculated by

$$\phi_i = \sum_{S \subseteq F} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)]$$

where S is a subset of features and F a set of all features, $f_{S\cup\{i\}}$ is a model trained with the feature i present and f_S is trained without that feature present.

The effect of withholding a feature depends on the other features in the subset. Using a sporting analogy, the contribution of a player in a team depends both on the other players in the team, but also on the position a player plays.[9] In other words, the SHAP values average the player's contribution to the team's score by putting the player in the different positions in a team and averaging over the team's result (e.g. the number of goals scored). The contribution of players can depend whether they are able to work as a 'team.' The idea being that a team can be more than a sum of its parts or that a player can rise above him/herself in the right team.

The formula above implies a large number of computations because of repeated retraining. However, through the use of sampling, Shapley sampling values can be used to explain the model and avoid having to retrain a model for every feature combination (i.e. fewer than $2^{|F|}$ times). There are specific variants that further reduce the computational requirements. As far as relevant for this thesis there is DeepSHAP for neural networks (Scott M. Lundberg and Lee 2017) and for tree-based ML models there is SHAP variant called TreeSHAP (Scott M. Lundberg et al. 2020). For the scope of this thesis it suffices that these methods use different optimized approximations to fit an explanation model, resulting in a outcome similar to that given by the formula above.

The SHAP values have some interesting properties making them a convenient method for interpreting prediction models. These are (i) local accuracy (ii) missingness and (iii) consistency. These properties are only satisfied if ϕ_i are Shapley values.

Local accuracy entails that the explanation model matches the original model for a simplified input that corresponds to the original input. That is "the best explanation of a simple model is the model itself" (Scott M. Lundberg and Lee 2017).

The missingness property ensures that missing data has no attributed impact on the value given to that missing feature by the explanation model, i.e. $\phi_i = 0$ for i missing.

Consistency means that if the value of the model f' for a feature i is equal or higher than that of f for the same feature that value attributed by the explanation model ϕ_i is also equal or higher.

These properties mean that SHAP is more convenient compared to predecessors from which it borrows some of the concepts, e.g. from the local linear explanation models (LIME) that interpret prediction by approximating the model around that give prediction (Ribeiro, Singh, and Guestrin 2016). However, as illustrated in the paper LIME has to violate the consistency property.

The choice for SHAP values as a method to get an insight into the influence of individual features on the prediction was made because it can be applied to all the prediction methods that are delineated below, i.e. the classifications trees and the neural networks.

3.3.3 Selection strategy

As discussed above, the hypothesis test metric allow for a threshold of a false discovery rate of for example 5%. However, the SHAP values don't have such an interpretation. As such, the threshold would have to be determined differently between this two categories of methods. This makes it more difficult to perform an apples to apples comparison. Therefore, the choice for this thesis is to look at taking the top x genes based on the respective scoring methods for further assessment. The exception here is the LIMMA method that would have to be changed significantly to allow for this.

Selecting features for further assessment can be done by selecting the top x genes, varying x between 1 and 15.[10] The reason is that, as we have mentioned above, not all methods return the same type of measure, which would allow selection through a uniform threshold measure such as the FDR. Also the context of feature selection in early drug discovery is that the features that are selected require further lab testing to

[9] That is that the Belgian footballer Lukaku will be more appreciated by his team as a striker than as a goalkeeper.

[10] This higher upper limit is unlikely to be feasible in practice due to the anticipated low cost-effectiveness.

assess there potency as a predictor for response to a pharmaceutical compound. There is a cost involved in the assessment of each predictor, which may be an important constraint in the number of features that can be selected. A top x strategy would keep the cost constant and known upfront. Considering the above, we use the strategy of selecting the top n genes for further assessment.[11]

3.4 Statistical methods for feature selection

The models for feature selection are both classical methods thath already used in the field and some machine learning methods for comparison. An Empirical Bayes (EB) correction is applied to all methods except for the neural networks using Tweedie's formula.

3.4.1 Methods based on hypothesis testing

3.4.1.1 T-test The t-test is the traditional method for testing a hypothesis. In this case we are using a two-sample t-test, assuming independent observations. A t-test assumes that the data are (approximately) normal distribution with an unknown variance. The test uses the mean and variance of both groups for assessing the two-sided hypothesis of equal means among the groups against the alternative of different means. Under the assumption of normality the t-test is known to be the most powerful test (Thas 2010, 231).

For the particular problem at hand, we assume unequal variance between the responder and non-responder groups for predictive features. As such, Welch's t-test, or the unequal variances t-test, is the test that we use. The formula for the test statistic is provided below. There are tables and software available to obtain the corresponding p-value.[12]

$$T = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1}{\sqrt{n_1}}^2 + \frac{s_2}{\sqrt{n_2}}^2}}$$

where $\bar{X}_1$ is the average in the non-responder group, $\bar{X}_2$ is the average in the responder group, s_1^2 the sample variance in the non-responder group, s_2^2 the sample variance in the responder group, n_1 the size of the non-responder group, and n_2 the size of the responder group (Bijma, Jonker, and van der Vaart 2018, 134–36).

Our implementation of this test is done in Python using Jax ("Google/Jax: Composable Transformations of Python+NumPy Programs: Differentiate, Vectorize, JIT to GPU/TPU, and More" n.d.) to allow vectorise the operations for faster execution.[13] As discussed in section 3.3.3, the top x scores are used. Therefore, the test-statistic is sufficient. Making it unnecessary to implement the p-value calculations.

The reason for including this specific hypothesis test is to provide a baseline to compare other method to. The t-test is a well established and widely used method for assessing a difference among groups. As such, it is still a valid testing method for differential expression for sequencing data when controlling for multiple testing.

3.4.1.2 Area Under the Curve (AUC) The idea of using AUC is that instead of using two parameters (the mean and variance) to describe the data, each data point in one group is compared to all data points in the other group. In doing so using more of the information enclosed in the distribution of the data. This potentially allows for a more accurate test when the data deviates from a normal distribution (e.g. due to skewness, variance heterogeneity,...) where Welch's t-test is not necessarily the most optimal test to use (Fagerland and Sandvik 2009). Contrary, the absolute relative efficiency of the WMW for normal data approaches the t-test for normal data and can be higher for other distributions (Thas 2010, 232). It is known that sequencing data are not normally distributed (Thas 2010, 306).

[11] See discussion on effect and alternative methods.
[12] See section 3.3.1 on the problem of large-scale hypothesis testing.
[13] See section A.1.2 for the corresponding Python code.

The AUC value represents the probability that $X_1 \leq X_2$, which can be calculated for very small sample size by checking the equation for all the combinations in the data. However, this quickly becomes computationally intractable. Fortunately, the Mann Whitney (or Wilcoxon) test statistic is an estimator ($\hat{\pi}$) for the AUC (π). Conveniently, this test is available in standard statistical software. The implementation used in this thesis is from the ("Statsmodels/Statsmodels: Statsmodels: Statistical Modeling and Econometrics in Python" n.d.), a python library.

As will become clear in subsection 3.4.1.3 where empirical Bayes correction is discussed, we require (an estimate of) the variance of the AUC to be able to use Tweedie's formula. The variance calculation is given by (Thas 2010, 233–34), which provides the following formulas for estimating the variance of the AUC.

$$Var\{\hat{\pi}\} = \frac{1}{n_1 n_2}\pi(1-\pi)[1 + (n_1 - 1)\rho_1 + (n_2 - 1)\rho_2]$$

with n_1 and n_2 the size of the respective groups, and where the estimators for the ρ are given by:

$$\hat{\rho}_i = \frac{\hat{p}_i - \hat{\pi}^2}{\hat{\pi} - \hat{\pi}^2}$$

with $i = 1, 2$ where

$$p_1 = \frac{1}{n_1 n_2 (n_1 - 1)} \sum_{h \neq i=1}^{n_1} \sum_{j=1}^{n_2} I_{ij} I_{hj}$$

where $I_{ij} = I(X_{1i} \leq X_{2j}$ with $i = 1, \ldots, n_1$ and $j = 1, \ldots, n_2$.

$$p_2 = \frac{1}{n_1 n_2 (n_2 - 1)} \sum_{i=1}^{n_1} \sum_{k \neq j=1}^{n_2} I_{ij} I_{ik}$$

The calculation of the variance of the AUC is partly implemented in the programming language C because the calculations require running a loop that considers the inequality in the $\sum$ of the formulas for calculating $\hat{p}_i$, which is known to be slow in native Python.[14] The EB correction for the AUC are based on the per feature variance. Contrary to the t-test it was thought prudent not to add the constant variance as another method to the comparison.

The reason for including this non-parametric test in the comparison is that we expect that for the simulation methods that lead to more realistic datasets that these will not necessarily be normally distributed. Therefore, we expect that the AUC will perform better on the more realistic datasets. Whereas the t-test is expected to exceed the performance of the other methods for the normally distributed data.[15]

Another reason for using the AUC is because the AUC has the convenient and meaningful interpretation as an effect size parameter. It gives $Pr(X_1 \leq X_2)$, which is equal to $\frac{1}{2}$ under the null hypothesis of equal distributions for X_1 and X_2. This means that when the value of the AUC is further away from 0.5 that it is (much) more likely that there is a difference between the groups (Thas 2010, 226). In casu this entails that the closer AUC is to either 0 or 1, the more likely it is that the feature is a good candidate for being a predictive feature.

[14] Loops in Python are known to be slow, a C implementation provides a speed-up in the order of a 100-1000x.

[15] Although not necessarily by much given the high relative efficiency of the WMW test compared to the t-test: $ARE \geq 0.864$ (Thas 2010, 233)

3.4.1.3 Empirical Bayes correction for selection bias using Tweedie's formula The selection bias as referred to by (Efron 2011) as the tendency of the mean μ_i of the features ($i = 1, ..., n_t$) that are selected (e.g. by the top x features as candidate predictive features) to actually be closer to the mean of the other, that is less extreme, than their sampled value. In other words, the actual location shift would be smaller for these selected features meaning that it is more likely that the features have become this outlying due to randomness.

One way of correcting for selection bias, is to use EB methods as a shrinkage estimator (Efron and Hastie 2016, 411). In contrast to full Bayesian methods, where the prior is determined prior to observing the data, EB uses the data to estimate the prior distribution. EB approximates a hierarchical (full) Bayes model where the hyperparameters are set at their most likely values instead of being integrated out (Lesaffre and Lawson 2012, 238–39). Sufficient subjects need to be available to ensure that the approximation is accurate enough. This is because the influence of individual datapoints on the prior becomes negligible for a larger number of datapoints. Fortunately, the large number of features in our data allow for an accurate approximation.

There are multiple EB estimation strategies (Efron and Hastie 2016, 421). In this thesis we use the so-called Tweedie's formula method as described in (Efron 2011).

$$E\{\mu|z\} = z + \sigma^2 \frac{d}{dz} logf(z)$$

which assumes that σ^2 is known (which is why the estimators for the variance were introduced in the previous sections) and where z is a vector of observed values for $i = 1, 2, \ldots, N$. It possible to estimate the μ_i by using the EB version of this formula:

$$\hat{\mu_i} \equiv \hat{E}\{\mu_i|z_i\} = z_i + \sigma^2 \hat{l}'(z_i)$$

The advantage of this method is that it works directly with the marginal density and all observations can be used to obtain a smooth estimate of $\hat{l}(z)$. The method does require a smoothed differentiable estimate of $l(z) = logf(z)$. To obtain such a function, Lindsey's method can be applied, which uses a Poisson regression by assuming that $f(z)$ is of a Jth-degree polynomial.

$$f(z) = exp\{\sum_{j=0}^{J} \beta_j z^j\}$$

having a canonical parameter vector $\beta = (\beta_0, \beta_1, \beta_2, ..., \beta_J)$ with β_0 used to fulfil the requirement that $f(z)$ integrates to 1 over the family's sample space $\mathcal{Z}$, which can be determined by partition over the range of $\mathcal{Z}$ into K bins and computing the counts for each bin: $y_k = \#\{$ z_i in kth bin$\}$ with $k = 1, 2, \cdots, K$. Next Poisson regression can be used taking the model: $y_k \overset{ind}{\sim} Poi(v_k)$.

As mentioned before, and evident from Tweedie's formula, is that an estimate of the variance is necessary. As such, to be able to apply this method, the formula's for the variance are needed for methods to which it is applied. As shown above there are formulas for estimating the variance for the methods based on hypothesis testing, specifically the t-test and AUC. However, for the prediction methods below the formula of the variance is not known, meaning that it is set to 1 because the data are normalised.

Furthermore, assuming that there is a difference between the variance of the predictive features compared to the other features, it may be of interest to adjust the EB correction based on the specific variance of a feature.[16] Therefore, there we use two methods of calculating the variance: (i) determining σ for the whole set at once, and (ii) determining σ per feature with $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ where n is the number of features. This is more computationally intensive, which can be mitigated by parallel computations, and as said will hopefully provide more accurate corrections.

[16] A more extreme form of the mean-variance trend that is used in the LIMMA package that is used to correct for the sequencing technologies reliability at lower intensities (Ritchie et al. 2015)

3.4.1.4 LIMMA The LIMMA package (LIMMA: Linear Models for Microarray Data) offers a relatively popular method for assessing differential expression in sequence data. While the package offers multiple function, in this thesis we use the `limFit` function to fit a linear model to the data and the `eBayes` to perform EB correction on that linear model fit.

Thus, the method consists of fitting a linear model to each row of data, which corresponds to a feature. It also uses the parallel nature of genomic data to borrow strength between these separate models for the genes (Ritchie et al. 2015). The latter step, what the authors call "information borrowing" is done using EB methods(Ritchie et al. 2015). More specifically it concerns a robust EB method, which reduces the probability of selecting variable features based on spurious differences (Phipson et al. 2016).

This way of using the LIMMA package is similar to the t-test with EB correction in the sense that the method consists of a linear model (in which the coefficients undergo a modified t-test) and an EB correction (Phipson et al. 2016, 951). Because it is conceptually somewhat similar to one of the other methods, and given its wider use, we apply this method to assess whether there is a difference between the implementations.

3.4.2 Prediction methods

The following methods don't use hypothesis testing, but instead are prediction models that aim to predict the most likely class. These are the so-called machine learning and AI methods.

3.4.2.1 Xgboost: boosted classification trees A method of predicting a classification are classification trees, see Figure 2,[17] where the parameters are used as splitting variables. In other words, by following the path along the tree the parameters determine which direction to take, which when followed to the root (or leaf depending on your perspective on trees) leads to a certain classification. Evidently there is a balance between the size of the tree in relation to its tendency to under-/overfit(Hastie, Tibshirani, and Friedman 2009, 307).

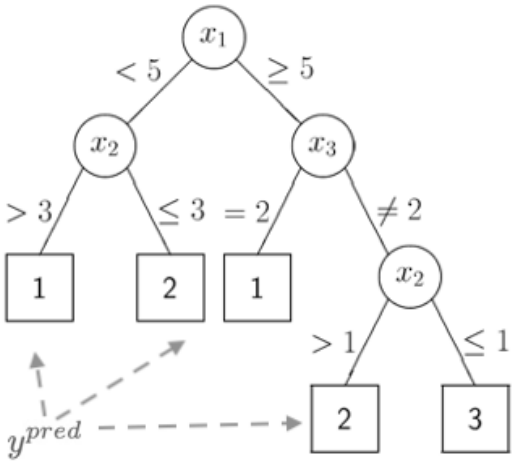


Figure 2: An example of a classification tree.

Classification trees have a high variance because the hierarchical splits and the influence of small changes thereon lead to very different trees (Hastie, Tibshirani, and Friedman 2009, 312). Meaning that classification trees (like regression trees) are so-called "weak learners" or weak classifiers because their error rate is only slightly better than random guessing. Weak learners can be combined into a higher performance model through the use of boosting, which reduces variance by averaging over many (weak) learners (Hastie, Tibshirani, and Friedman 2009, 337).

[17] Image source: https://bookdown.org/tpinto_home/Beyond-Additivity/)

It is infeasible to calculate the best partitioning of a tree in terms of the minimum sum of squares because of the computational complexity this entails (Hastie, Tibshirani, and Friedman 2009, 307). Consequently, different algorithms are used to approximate the "optimal" tree including gradient boosting. Gradient boosting uses the numerical optimization procedure of gradient descent to ascertain the gradient in which to 'travel' the model's parameters to reduce the loss function (Hastie, Tibshirani, and Friedman 2009, 358; Efron and Hastie 2016, 348). It is beyond the scope of the thesis to reiterate the boosting algorith and gradient descent since there are good textbooks on those topics, e.g. (Hastie, Tibshirani, and Friedman 2009, 361).

For this thesis we use the implementation of boosted classification trees in the Xgboost package(Chen and Guestrin 2016). Using `n_estimators=10` for 10 boosting rounds, `eval_metric='mlogloss` suitable for a binary classification problem and 12 simultaneous jobs for exploring the tree through `n_jobs=12`.

To ensure more robust results we use 5-fold cross-validation to balance variance with computational feasibility (Hastie, Tibshirani, and Friedman 2009, 242). This means that the n_t features are split into 5 datasets, leaving one out as a test-set. The SHAP values are calculated on that test-set, meaning that the SHAP values are calculated on unseen data within the context of a fold. This should ensure that a correct cross-validation is done as described in (Hastie, Tibshirani, and Friedman 2009, 245–46).

3.4.2.2 Neural networks Given the popularity of deep learning in big data applications, we will assess the performance of a neural network for feature selection.

Neural networks are two-stage regression or classifications models that can be represented by a network diagram, or a graph (Hastie, Tibshirani, and Friedman 2009, 392). The neural network can be made into a classification model by adding a softmax layer as the final layer (Hastie, Tibshirani, and Friedman 2009, 393). There are several parameters that can be changed or tuned in a neural network. The number, type, size of (hidden) layers can be determined with little constraints. The topology of the network is very flexible where the connection between different layers can be (completely) specified. There are a variety of activation functions for the neurons in the network. There is also the exact method of gradient descent and its hyperparameters such as the learning rate that can be tuned (Yu and Zhu 2020).

We will not bother with tuning all these parameters and use the default parameters in Tensorflow. This is because the competing methods of feature selection require very little effort in terms of tuning. The reasoning behind this choice is that it would be difficult in practice to perform further tuning given the limited availability of data, a lack of understanding of how robust hyperparamaters are across different datasets, and would substantially complicate the comparison between the methods of feature selection.

The only tuning we change in the model is using an increasing number of hidden layers until overfitting becomes problematic and reduce the number of iterations (epochs) to a subjective level of not overfittting. This choice was made because overfitting is an issue in using neural networks (Hastie, Tibshirani, and Friedman 2009, 359), which becomes more likely when adding depth to the network. There are two major interventions possible against overfitting: reducing the size and number of layers, and using drop out layers. The latter is a layer type that randomly drops, or resets, the values of neurons in the network to prevent the model from relying on such nexuses. We use the rectified linear unit (ReLU) as the activation function.

Because the neural network will be trained to classify into responders and non-responders, an additional step is required to obtain a measure that can be used to identify predictive features. The SHAP values, using the DeepSHAP variant that is conceptually similar to the TreeSHAP mentioned above, are calculated and used to identify predictive features.

We did an initial exploration into two implementations for creating the neural network model: Tensorflow and Jax. The latter is a newer, more low-level, library that would allow for parallel training of smaller networks. However, this comes at the cost of support in other packages. In the end in proved to be technically infeasible to create a efficient SHAP value implementation for Jax within the context of this thesis (i.e. this would require low-level coding in both packages to implement). Therefore, we use Tensorflow given it is directly supported by the SHAP package. Unfortunately, there is no straightforward way to do parallel training of neural networks in Tensorflow, which was consequently abandoned.

3.5 Data integration (DI)

Data integration (DI) allows the use of earlier measurements of the same feature(set) to be informative on whether a feature deviates from the wild-type, which is hypothesised to increase the probability of it being a predictive feature. To assess this, the dataset resulting from the simulation method described above are used. However, the predictive features are shuffled in the set as to not coincide with the predictive features in the 'original study' dataset. The reasoning being that in reality the "response" is unknown for datasets available for data integration and that in public sets there is no differential expression that is hypothesised for the predictive features. The underlying assumption for the data integration method that we use is that the wild-type variance of a feature differs from the variance of a predictive feature. Essentially, this assumption revolves around the idea that the influence of a compound on the expression levels and a response is unlikely to be found in the wild, i.e. in publicly available datasets. In essence this is the concept of the filter methods as described in (Kuhn and Johnson 2013, 499).

Revisiting the analogy of the needle in the haystack, the idea of data integration is that we are able to remove part of the haystack of which we are relatively sure that it contains no needle. Thinking of Figure 1b this is similar to using a tool (e.g. the magnet in the figure above) to guess whether the needle is in some well defined part of the hay.

The method used for this thesis is the following for a specific study:

1. Select a random sample of size $n_{pub} = 10$ of other simulated studies with the same number of genes
2. Shuffle the predictive features in the 'integration' dataset
3. Use a hypothesis test (see below) to assess genes that differ between the 'study' dataset and the 'integration' dataset
4. Select those features that have a value below a certain (subjective) threshold.

Two hypothesis tests were used in step 3: a test for equal means and equal variance.

3.5.1 Based on the mean

Initially, the data integration was based on the mean. For this, the hypothesis test that we use is the Mann Whitney U test statistic given it is more efficient when the data is not normally distributed. The resulting p-value was used for filtering of the features below a (pre-determined) threshold of $p \leq 0.65$ feature selection. This is a subjective cut-off point as there is no obvious way to select an objective cut-off point.(Kuhn and Johnson 2013, 499)

3.5.2 Based on variance

Another method do the filtering is based on the (difference in) variance. To assess the difference based on the variance there are a couple of tests that we can use: F-test for equal variance, Bartlett's test, the Levene (Levene 1960) and Brown-Forsythe (Brown and Forsythe 1974) tests (the latter is an extension of the Levene test using the median instead of the mean). Because the F-test and Bartlett's test do not work well on non-normal data and we use the Brown-Forsythe test, which is recommended for skewed distributions.[18]

The null hypothesis is that all variances are equal, $H_0 : \sigma_1^2 = \sigma_2^2 = \cdots = \sigma_k^2$. The alternative hypothesis is that at least one pair of variances (e.g. i and j) that is different, $H_a : \sigma_i^2 \neq \sigma_j^2$. The test statistic is defined as follows, based on (Guthrie 2020).

$$W = \frac{(N-k)}{k-1} \frac{\sum_{i=1}^{k} N_i (\bar{Z_i}. - \bar{Z_{..}})^2}{\sum_{i=1}^{k} \sum_{j=1}^{N_i} (Z_{ij} - \bar{Z_{i.}})^2}$$

[18] See the Scipy documentation, https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.levene.html

where $Z_{ij} = |Y_{ij}| - \tilde{Y}_i$ with $\tilde{Y}_i$ the median of the i-th subgroup, $\bar{Z_{i.}}$ the mean of the i-th subgroup, $Z_{..}$ the overall mean.

The corresponding p-value can be calculated using the F distribution: $W > F_{\alpha,k-1,N-k}$ where $k-1$ and $N-k$ indicate the degrees of freedom and α the significance level.

We use the implementation in (Virtanen et al. 2020) of the Levene test and filter the features that have a p-value resulting in $p \leq 0.65$. The threshold is again subjective.

4 Results

The results start with the outcome of the data simulation, describing what data was generated in the end. Next, the different aspects relevant for the guidelines are discussed, being: number of features, number of predictive features, effect-size of the predictive features, and sample size. There is a specific section focusing on the performance of the EB correction using Tweedie and an assessment of the performance of the data integration.

4.1 Data simulations

While the intention was to use the steps in simulating the data as described in section 3.1.2, small coding errors led to unexpected study data. However, these studies were still usable for specific analyses and therefore kept for further analysis.

The result of the simulation is a dataset that encompasses 1.5 TB, for around ~100.000 studies in total, see Table 2 for an indication of the size of the different simulation generations. Because of time and computational constraints not every simulation run was completed, which is shown by a difference in the generated studies and the number of studies. Consequently, there may be missing combinations of parameters, e.g. there may not be a study for a specific number of features with all the other combinations that are available for another number of features in that same run.[19]

Table 2: Description of the (approximate) number of studies simulated.

Generation	Size	Generated studies	Number of studies
Normal	63.7 GB	52518	52518
Zhang-1	328.8 GB	13167	13167
Zhang-2	762.6 GB	21339	21339
Zhang-2-di-mean	9.2 GB	17770	21339
Zhang-2-di-var	24.9 GB	8800	21339
Zhang-3	323.5 GB	3467	5629

Table 2 shows that there were multiple generations of data simulations. The first generation made use of the simulations method described in section 3.1.1, which consists of the studies generated through the location shifted Gaussian mixture model.

The other methods are using the `SPsimSeq` package, with differences in the way the location shift to the predictive features for the responders group was applied.

The `Zhang-1` set is the initial `SPsimSeq` version the location shift was applied after scaling. Consequently, there is variance of 1 for all but the predictor features (i.e. the variance would be a perfect classifier). The t-test with EB based on the variance is probably affected by this mistake because there the correction term in Tweedie's formula is based on the variance of the individual features. For this method and combined with the `Zhang-1` generation means that there is a disproportionate correction for predictive features because

[19] This is known not to be ideal, see the limitations in the discussion for elaboration on this.

the location shift in that generation influences the variance. For larger μ_{pred} that means there can be an overcorrection to the opposite extreme for those features. For this reason the studies in Zhang-1 generation are not valid for the t-test (EB, variance) method of feature selection. Because the other methods do not seem to be overly affected by this mistake the generation was kept as a whole.

The second generation is `Zhang-2`, which would have been the final version except that due to a coding error the minimal effect size was always a μ_{pred} of 1, leading to effect sizes that were larger than desired. The last generation `Zhang-3` was mainly used for the assessment of the Xgboost method since there was no appreciable decrease in performance at $\mu_{pred} \geq 1$. Because the Zhang-2 and Zhang-2 are otherwise identical, these generations were combined, where they were not needed separately, by correcting the Zhang-2 generation by increasing the metadata by 1 for the μ_{pred}.

`Zhang-2-di-mean` and `Zhang-2-di-var` are the results of the DI methods of using the mean and the variance respectively. As their names imply the DI uses the `Zhang-2` generation as the underlying datasource.

There are three complications that arise from how the simulations were done.

1. Some generations are incomplete for some analyses, meaning that checks are needed to ensure a sufficient number of studies per (combination of) variable(s). To be specific, there may be fewer results for the Xgboost and Neural network models since these are more computationally demanding.[20]
2. The iterative development process and resource constraints entail that not all analyses are applied to every generation. This specifically concerns the LIMMA and the neural network methods of feature selection.[21]
3. Most importantly, within a generation there are multiple 'runs' that are balanced within, but not balanced between. For the normal data this was identified by the time of the calculation of the result, for the Zhang data by a `meta_run` identifier. However, these proved to be unwieldy in practice because repetitions of runs with different seed value also have a different number, so it remains largely unused.

4.2 Influence of number of features

The expectation is that increasing the number of features is the equivalent to adding more hay to a haystack, i.e. it will make it more difficult to find predictive features. Consequently, an increase in the number of features with all other variables constant is expected to reduce performance.

For the 'normal' data the results in Figure 3 show a small, but consistent, decrease in performance when the number of features increases. The exception is the neural network, which is both remarkable in that it performs better on more features as well as the lack of difference between the group sizes. There is variability in the performances that originates from the different effect sizes that are in a generation, which are not balanced over the number of features.[22]

Figure 5 shows the effect of the number of features on the precision in the Zhang-1 and Zhang-2 simulated sets. Depending on the method of feature selection there is a downward trend, although for some combinations of the method of feature selection and the number of predictive features this is not the case. The variability in the results are likely a result of the variance of the precision estimate and the number of samples since the larger number of samples per number of features in the Zhang-1 set is less variable.

One evident result shown in Figure 5 is the very poor performance of the neural network.

First, looking at the overall effect of the number of total features, there is a clear drop in the performance with higher number of features as shown in Figure 4.

Figure 5 shows the effect of the number of features for the different methods of feature selection, applied to all the `SPsimSeq` simulations. The variability in these results led to the choice of binning into 4 groups based on the number of features. The results in these groups are averaged showing that for most studies

[20] These are also implemented separately from the hypothesis test based methods.
[21] LIMMA is not applied to the normal generation, whereas the neural networks were not used in data integration.
[22] This results from doing more runs per simulation generation with slightly differen parameters.

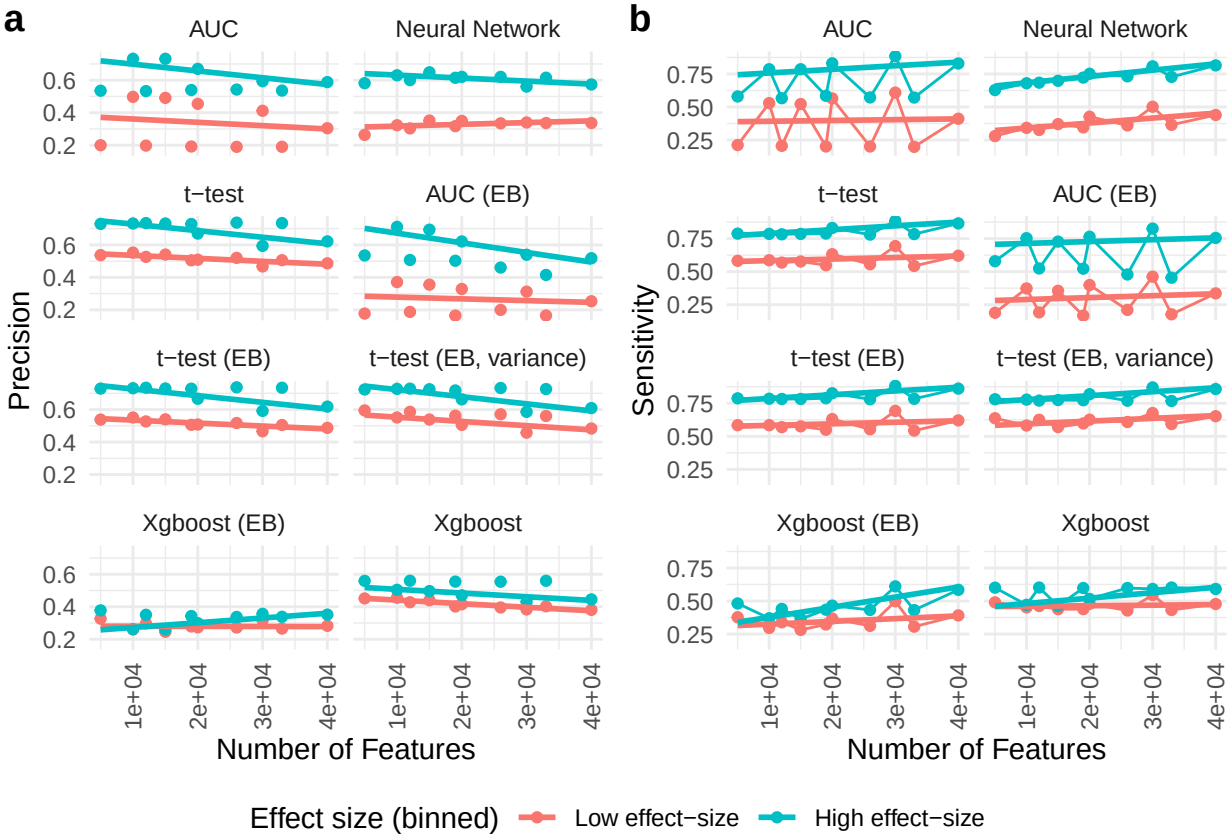


Figure 3: The effect on the precision (a) and sensitivity (b) of number of features in the normal data. The points indicate the mean precision with the solid lines are the linear regression line for a lower effect size and a larger effect size.

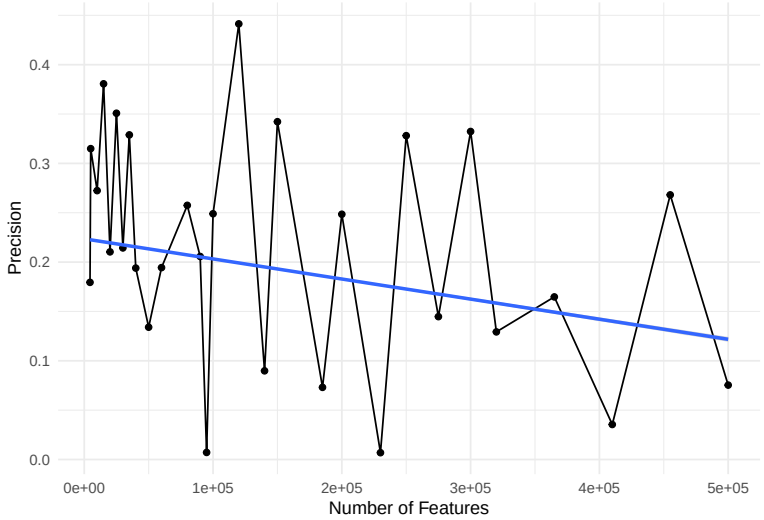
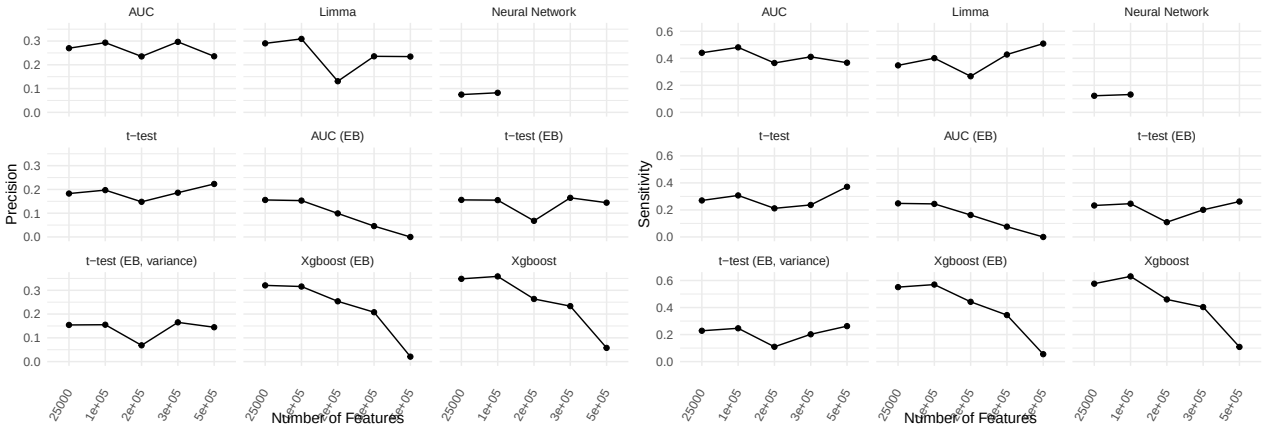


Figure 4: The precision and sensitivity for the Zhang generations, which are averaged over the other parameters. The linear regression line is in blue.

there is a decrease in performance. For the t-test methods there is an increase around the 500.000 features, which may be the result of chance since the number of studies for that group is limited. Of note is that the neural networks with a single hidden layer started overfitting even at very low epochs, meaning that the performance became very poor. Therefore, it was decided to be economical in the use of compute time for that method and to not further consider larger networks.



(a) Effect on the precision

(b) Effect on the sensitivity

Figure 5: The effect of the number of features on larger group sizes for the Zhang generations. The solid line and black dots show the mean precision and sensitivity.

4.3 Number of predictive features

To asses the influence of the number of predictive features on the precision and sensitivity, we look into the effect of the number predictive features for a range of selection strategies using the absolute number of predictive features. After this, the fraction of predictive features in the total number of features is assessed ($\frac{n_{pred}}{n_t}$).

4.3.1 Effect of the absolute number of predictive features

Figure 6a suggests that having a larger number of predictive features increases the precision, which is rather unremarkable. What is notable is that the sensitivity, as show in Figure 6b, drops when there are more predictive features in the data. There are two possible reasons for this. The first is that the top x selection strategy is a limiting factor for selecting the larger number of predictive features. For larger number of predictive features a more liberal selection strategy may be required. In the context of early drug discovery such a situation would be unlikely since the expected number of predictive features is not that high. The second is that it becomes increasingly difficult to select additional predictive features. Given that the features are ranked it makes sense that lower ranked predictive features are more similar to non-predictive features.

4.3.2 Effect of the relative number of predictive features

Figure 7a shows the precision for the fraction of predictive features in the total number of genes ($\frac{n_{pred}}{n_{pred}+n_{non\_pred}}$). The points in the figure indicate the mean precision for that specific fraction The precision increases when there are relatively more predictive features in the dataset, regardless of the selection strategy. The sensitivity is more constant compared to the precision, with the mean sensitivity when selecting only the top 1 feature declines at higher fractions. The reason for this is that a higher fraction either requires more (than 1) predictive features or a lower number of total features. The former affects the sensitivity because it declines with an increased number of predictive features when keeping the selection strategy constant.

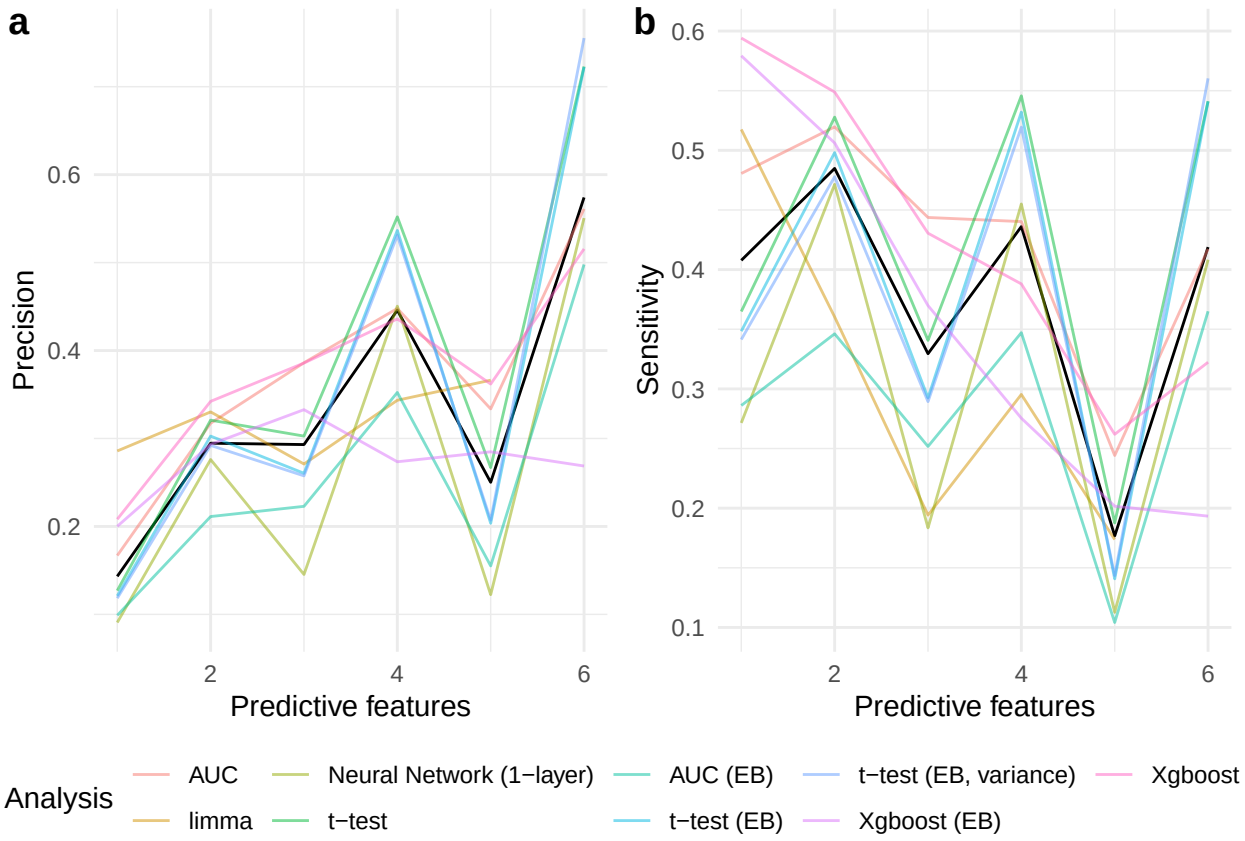


Figure 6: The precision (a) and sensitivity (b) for different numbers of predictive features. The black line is the mean of all methods, which are the coloured lines that show the precision and sensitivity per method of feature selection.

Figure 7b shows the sensitivity for the fraction of predictive features in the total number of features ($\frac{n_{pred}}{n_t}$), similar to the figure of the precision above. Overall the sensitivity is relatively constant over the range of proportions in the simulated studies. For the top $x = 1, 2$ strategies the decrease can be explained because a higher proportion in our simulated set requires more predictive features, since there are few low dimensional studies being simulated. Therefore, these strategies do not perform well because of the inability to select all these predictive features.

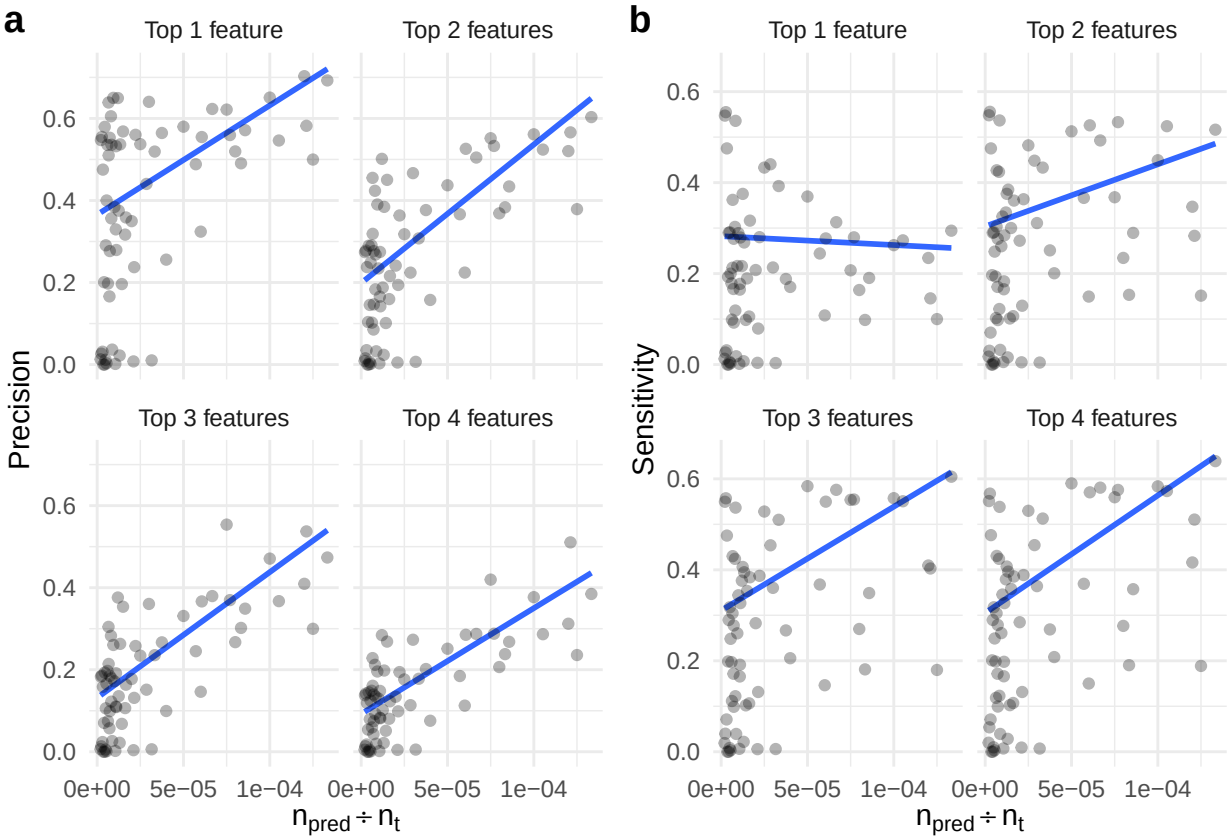


Figure 7: The dots indicate the mean precision (a) and sensitivity (b) for the relative number of predictive features, which means the number of prective features divided by the total number of features.

Figure 8 shows the results for the LIMMA method. These results are presented separately because it doesn't use a top x selection method. Unsurprisingly, for LIMMA there is a benefit of having a higher proportion of predictive features in the data too for the precision. The sensitivity gets lower with a higher fraction of predictive features. In conjunction with the higher performance for the precision, it can be concluded that the reason for this is that the simulations with a higher fraction have a higher absolute number of predictive features, where the sensitivity is reduced by selecting lower number of the top x features. This again confirms that a selection strategy must be adapted to the specific context to ensure high precision and sensitivity.

4.4 Effect size

The previous sections have already hinted at the importance of the effect size, consisting of the location shift and the variance of the predictive features, on the performance of the different methods of feature selection.

The results for the effect size in the Zhang dataset in Figure 9. The results for the normal generation are not shown, but consist of similar results, albeit with more variation due to the higher number of combinations in the normal generation.[23]

[23]The results for the normal dataset are more variable as a result from using more combinations of parameters in the

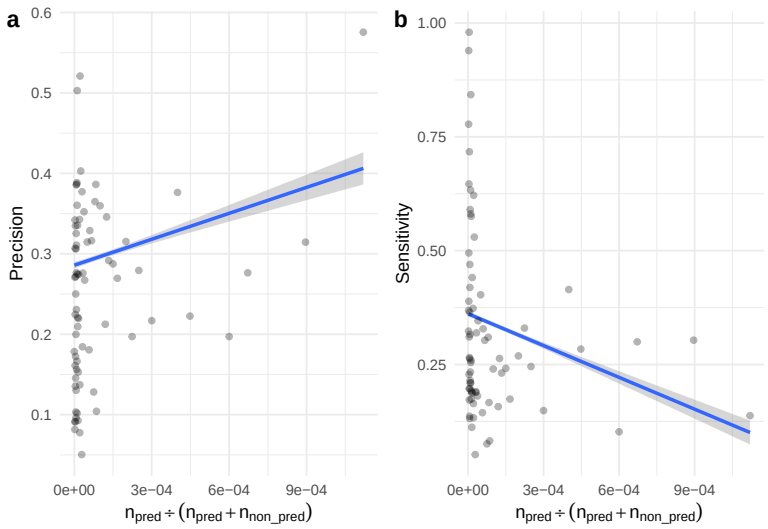


Figure 8: Precision (equal to the sensitivity here) for the LIMMA method for the fraction of predictive features in the total number of features.

Figure 9 shows that the Xgboost method outperforms the other methods of feature selection, especially at lower effect sizes. A notable difference compared to the results on the normal simulation method is that the AUC is now outperforming the t-test. Since the `SPsimSeq` package simulates more realistic, that is non normal, datasets this is in line with the theory discussed above in the method section.

Figure 10 shows that the sensitivity gets higher with a larger location shift. Because the average is taken over the different predictive features, the maximum sensitivity when selecting the top gene is $\frac{1+\frac{1}{2}+\frac{1}{3}}{3} \approx 0.61$. All but the Xgboost method show a slow steady, albeit variable, rise in sensitivity with increased location shift. The AUC method is another good performer, which quickly increases in performance at lower location shifts. Because these results are rather noisy due to the influence of all other parameters, a GAM smoother was used for assessing the trends for the different methods of feature selection.

To assess the limits of the Xgboost method, another simulation run: "Zhang-3" was done, which has lower effect size because of the different ways in which the calculations were done in software. Figure 11 shows that the Xgboost method too has a lower threshold for the effect size that it needs to perform with high precision. A loess smoother is applied to the results to better depict the transition point for the increase in performance of the Xgboost method at a location shift of μ_{pred} in the range of 0.4-0.5.

4.5 Sample size

The size of the responder group is influential on the ability of the different methods to achieve a high precision. The expectation is that balanced groups with larger sample sizes perform best.

There is some variability over the different number of features, resulting from unequal average effect sizes because these come from different runs. Therefore, a linear regression line was used to provided a more convenient picture for comparison. The normal dataset shows clear difference between the group sizes where the larger sample sizes outperform smaller group sizes, see Figure 12. The unbalanced groups are mostly similar over the different methods, with the exception of the neural network, where there is a clear difference. This may be because neural networks do not necessarily work per individual feature, which could partly explain why the larger responder groups performs better using that method of feature selection. When considering the non predictive features as noise, there is more signal available when there are more responders vs. non responders. The results could be better in part because the network could incorporate differences within the responder group as well.

simulations. It can be seen in the Dashboard.

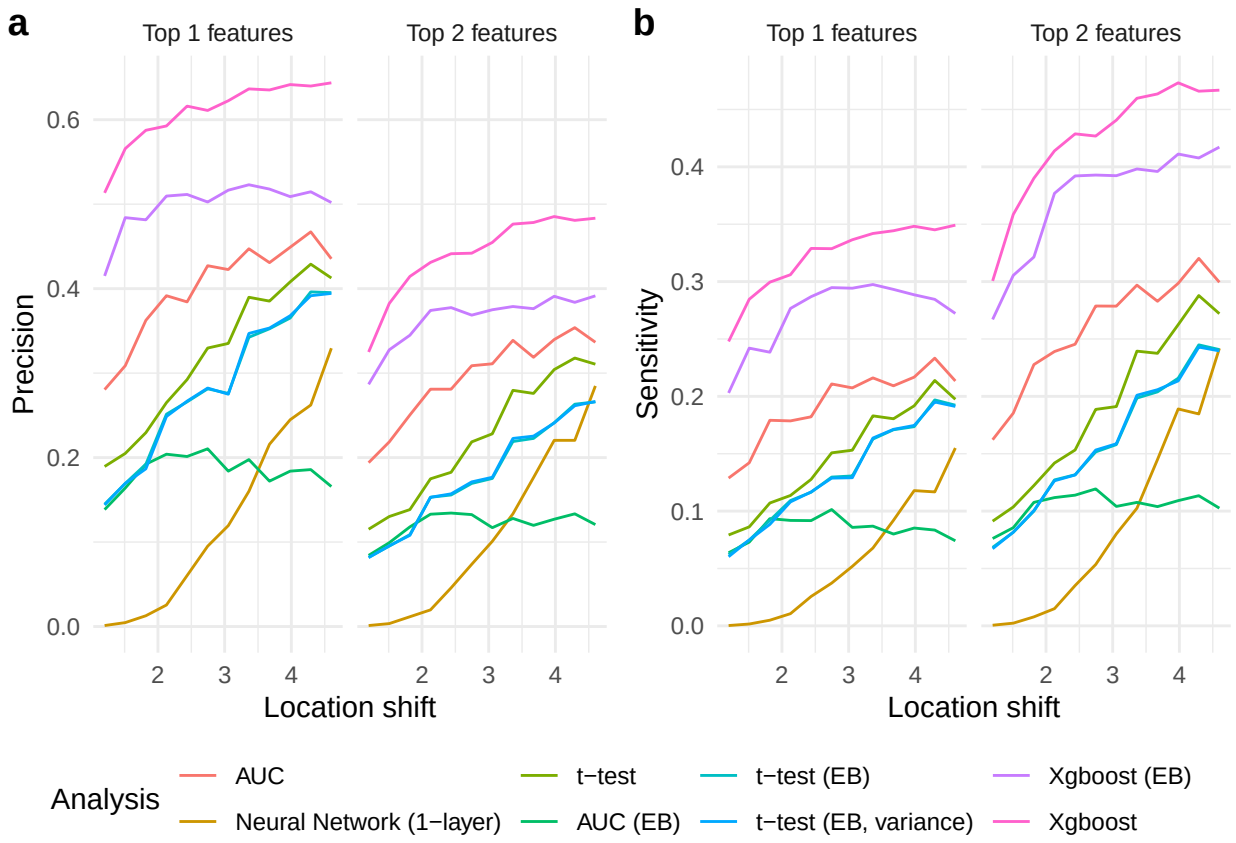


Figure 9: The influence of the location shift on the average sensitivity for the different methods of analyses in the Zhang-1 generation.

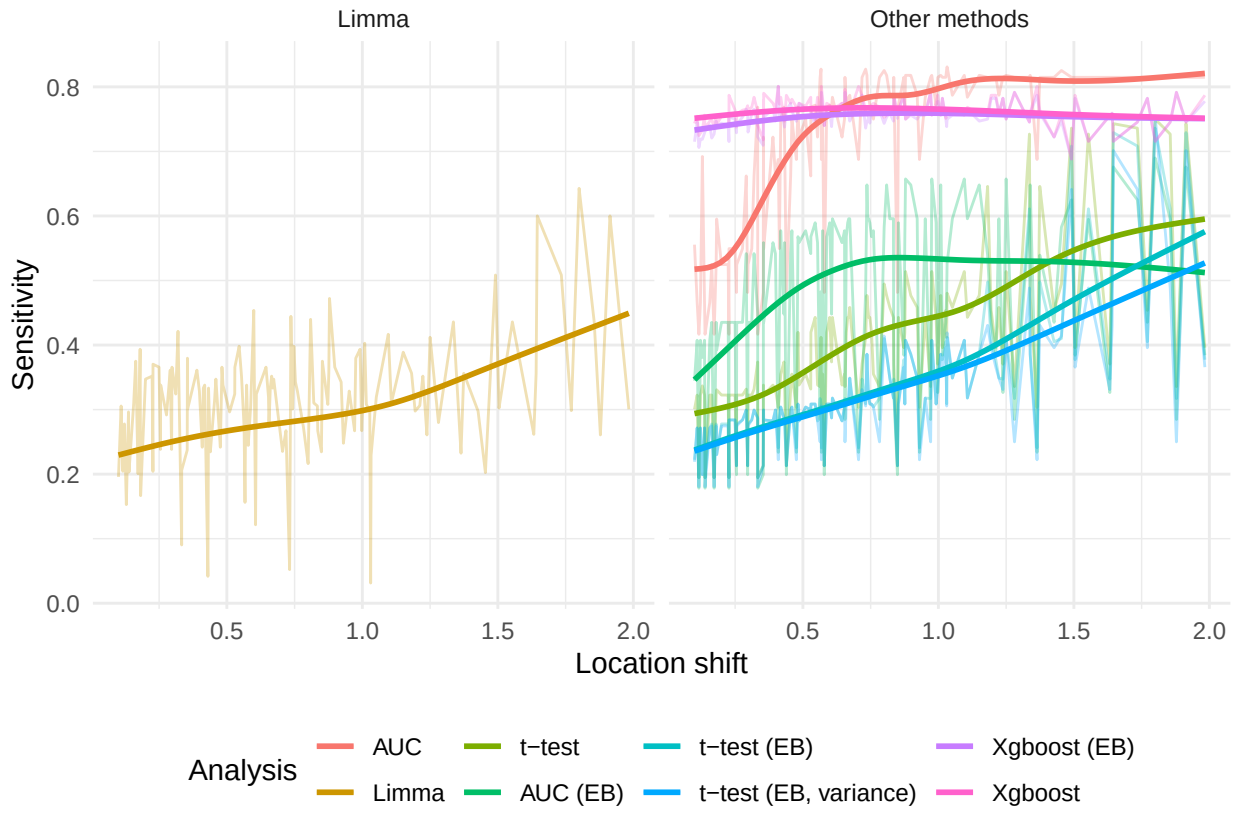


Figure 10: Influence of effect size on the average sensitivity for the different methods of analyis.

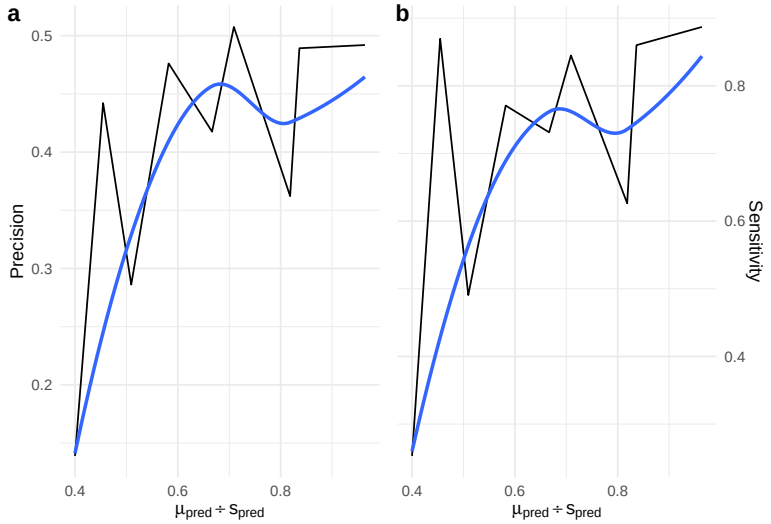


Figure 11: The influence of the location shift on the precision (a) and the sensitivity (b) specifically for the Xgboost method, focusing on smaller location shifts.

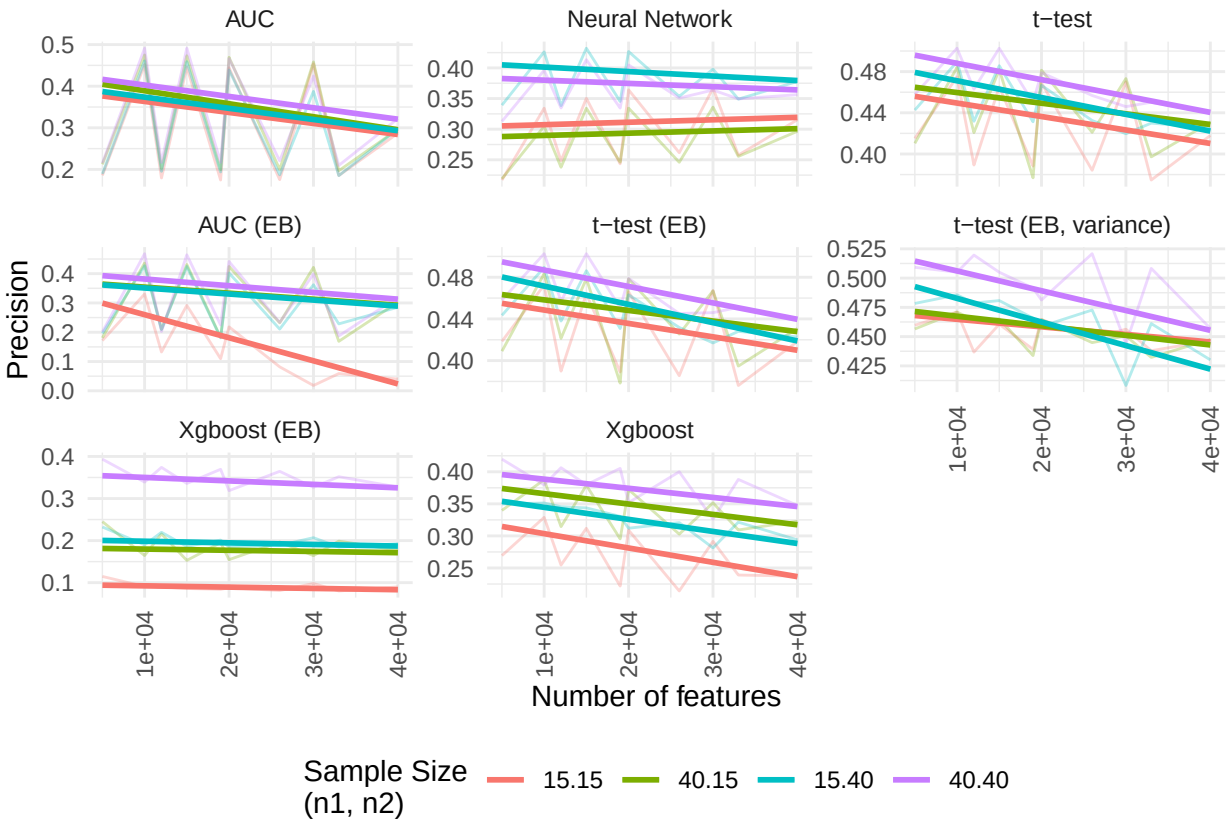


Figure 12: The effect of sample size and balance of groups on the precision of selecting predictive features for the normal dataset.

The same figure, but for the whole Zhang dataset is given in Figure 13 for the range of 20 to 40 thousand features. The effect of the sample size is less clear here. While the largest sample size ($n_1 = n_2 = 40$) generally outperforms the other methods, there are exceptions. A strange phenomenon is the difference in performance for the unbalanced groups in multiple methods. The peculiar very low performance for the $n_1 = 40$, $n_2 = 15$ group suggests an implementation error since there is little theoretical reason to suggest such an effect. A thorough check of the implementation provided no evident reason of an error, making this surprising and unexplained result particularly in light of not observing such an effect in the normal data.

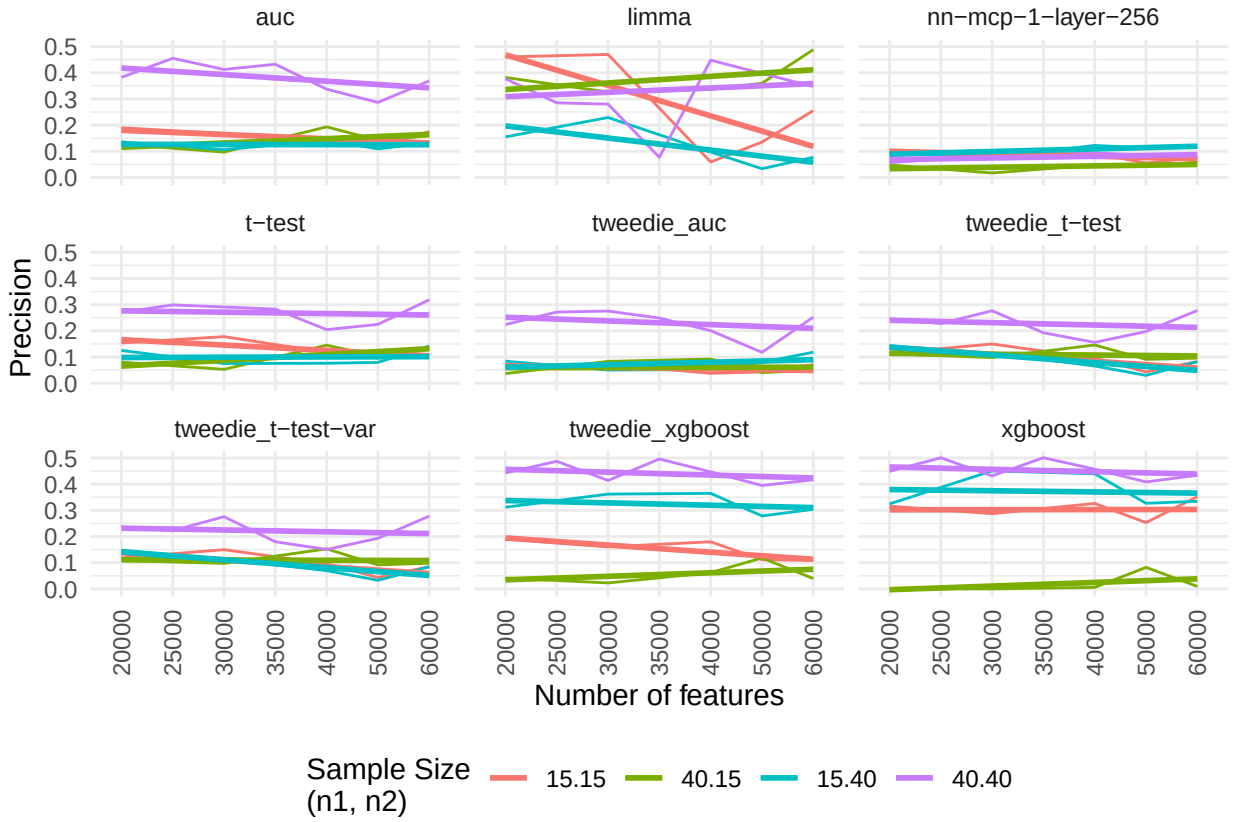


Figure 13: The effect of sample size and balance of groups on the precision of selecting predictive features for the Zhang datasets.

4.6 EB correction using Tweedie

The use of the specific EB correction using Tweedie's formula warrant a further look into how it affects performance in the context of this thesis. While there is no obvious, across the board, performance benefit, there is some suggestion of an effect in the previous results. Although the direction of the effect depends on the specific method of feature selection.

For this section we make the comparison between the performance of the EB correction on the normal dataset and the combination of the Zhang-2 and Zhang-3 datasets. The figures are so called dumbbell plots that show two coloured points connected by a line for easy comparison. The colours of the points indicate whether it corresponds to a EB or non-EB corrected result.[24]

Table 3 gives the average precision and sensitivity for the `normal` generation and the `Zhang-*` generations for the analysis that include an EB correction. The main interest in these two tables is to compare the relative performance of the different methods of feature selection for a specific simulation method (normal vs. Zhang).

[24]Of note is that the LIMMA method was not applied to the normal dataset and as a consequence is missing in this comparison.

Table 3: Averaged precision and sensitivity per analysis.

	Normal		Zhang	
Analysis	Precision	Sensitivity	Precision	Sensitivity
auc	0.35	0.56	0.270	0.44
nn-mcp-1-layer-256	0.34	0.54	0.076	0.12
t-test	0.44	0.69	0.190	0.28
tweedie_auc	0.31	0.48	0.140	0.22
tweedie_t-test	0.44	0.69	0.150	0.23
tweedie_t-test-var	0.45	0.72	0.150	0.23
tweedie_xgboost	0.21	0.40	0.300	0.53
xgboost	0.32	0.49	0.330	0.56

While the t-test based methods perform better for the normal data, the performance for the other methods of feature selection perform relatively better for the more realistic Zhang data.

Figure 14 shows the performance difference for the precision for the t-test versus the EB corrected t-test with the variance calculated for each feature.[25] There is a small, consistent performance benefit for the normal data. For the Zhang data it reduces performance.

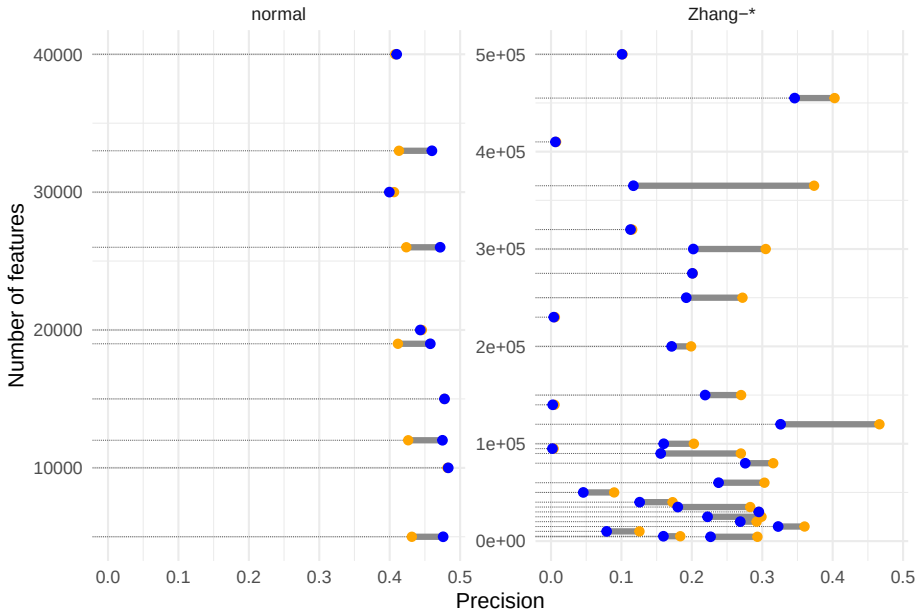
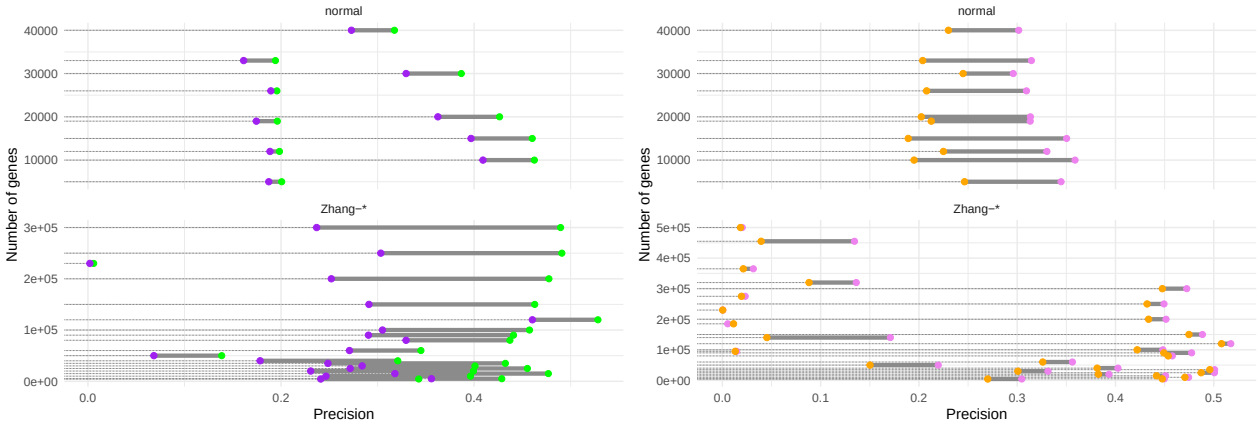


Figure 14: Comparison of the precision of the T-test vs T-test (EB, individual variance), where the EB improves the precision on the normal data.

The AUC method and Xgboost result are shown in figure 15. These are shown together because the conclusions on the effect of EB correction are the same for both: it reduces the precision.

Looking into the different methods of feature selection in more detail, reveals a consistent pattern of the Tweedie formula not having a performance benefit, but actually performing worse in all but the EB correction for the t-test. As such there is a very narrow area of applications for this method of correcting for selection bias, namely on a normally distributed dataset, when the method of feature selection is a t-test. However, in none of the more realistic simulations using `SPsimSeq` was there any substantial benefit. To the contrary,

[25] Using the variance per feature has a slight performance benefit, which is why the t-test using the overall variance for EB correction is not shown.



(a) The AUC and AUC with Tweedie corrections

(b) The Xgboost and Xgboost with Tweedie corrections.

Figure 15: The effect of EB correction on the precision for the AUC and Xgboost method shows no benefit.

there was a consistent loss in performance with incidental gains. As such, these results imply that it is not a particularly useful method in practice.

4.7 Data integration

In this section we compare the result on the analysis on the 'complete' Zhang-2 dataset with those that were 'reduced' through the two DI methods. The reason being that the calculations on all datasets would be too time consuming and are unlikely to provide novel insights.

Just like for the section on the results of EB corrections, dumbbell plots are used to compare the results (i.c. the precision) of the data integration compared to the original dataset for both methods.

Figure 16 shows the results of the mean-based method. Overall there does not seem to be a benefit in using this type DI, with the exception of the LIMMMA method where it has a benefit for higher numbers of features. As discussed in the methods section, using a hypothesis test based on the mean is not the recommended option. For one, it requires that there is not a variation of the mean resulting from other sources (e.g. due to variability in the sequencing technology). Besides a lower performance, this specific implementation has no other benefits such as tangible improvements in computation times. Also it requires extra steps meaning an added layer of complexity, which is acceptable only when there are performance benefits.

The results for variance-based method in Figure 17 show a more promising result for using DI. Meaning that it shows consistent benefit for LIMMA. Still, for the other methods the precision is reduced when the precision is already higher. However, when the precision for the original method is low, there does seem to be a benefit of DI. There is an important caveat with this results, namely that the DI was not run to completion, which can affect some methods more than others. Also, there is a problem of the average precision being exactly zero for groups with > 100 studies. This is unlikely and would indicate an error in scoring. However, if that were the case it would show up more consistently because while Xgboost method is mostly implemented separately, the EB-corrected AUC method is implemented together with the other methods.

On the whole, considering the caveats above, the DI show a picture of a sort of shrinkage of the precision. Meaning that except for LIMMA there is a tendency of the direction of the effect of DI to be away from the extremes. This could be the consequence of the results being too variable, however.

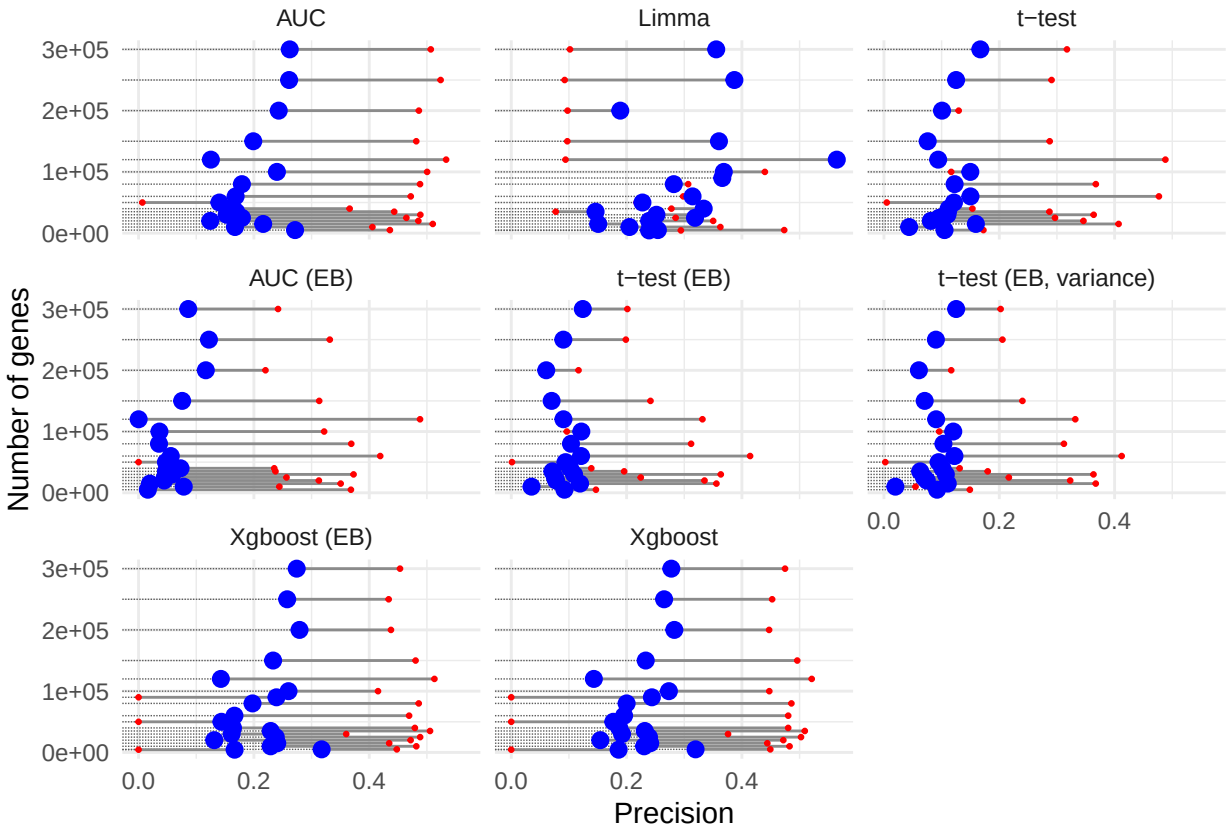


Figure 16: A comparison of the precision for the original datasets and the mean-based DI method. The Zhang-2 is shown in red and the DI in blue.

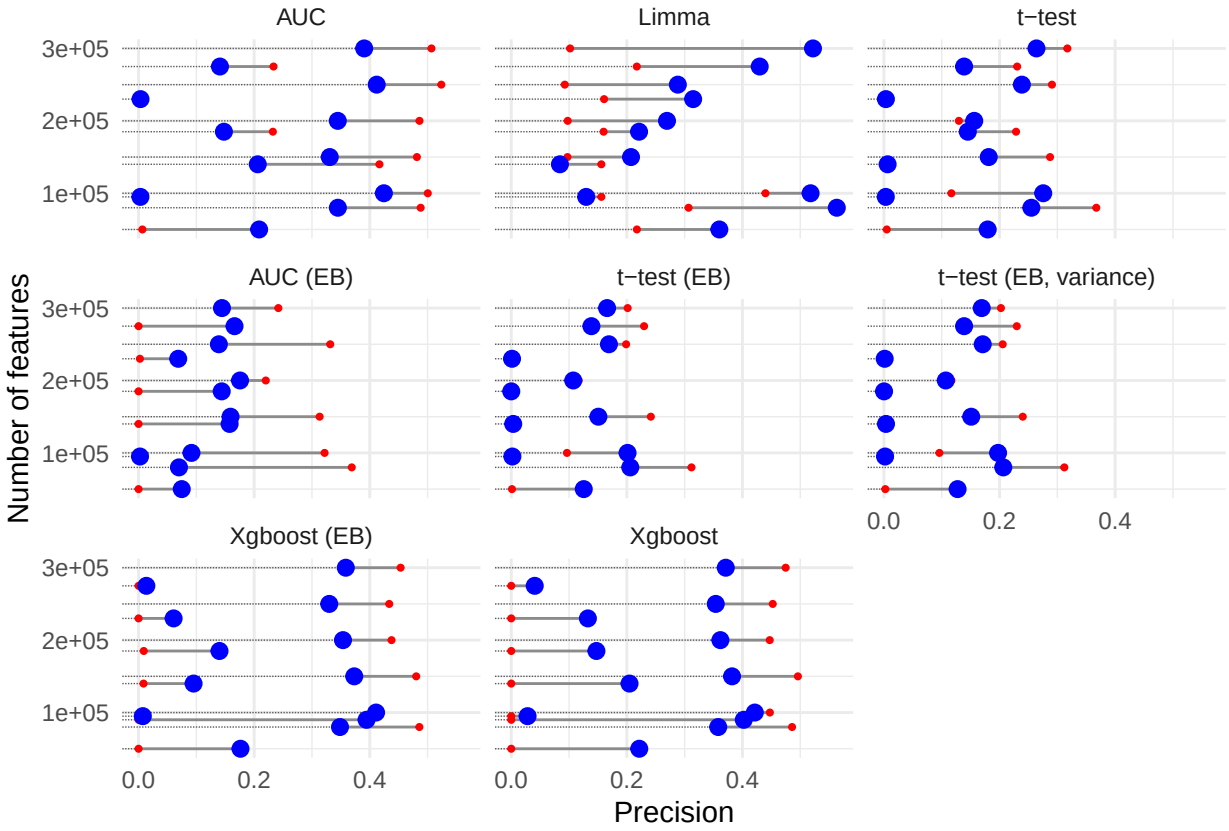


Figure 17: A comparison of the precision for the original datasets and the variance-based DI method. The colours indicate the Original and DI. A caveat in this figure are that there are analyses with a result of exactly zero, which is unlikely.

5 Discussion

The results show that the number of features, the number of predictive features and the sample size affect the precision and sensitivity of the different methods of feature selection. Not all results were completely and unequivocally as expected. In the discussion the limitations of the (results of this) thesis are given, followed by a comparison of the different methods discussing the reasons for the difference in performance, the reasoning behind the results of the data integration methods and possible areas of further research.

5.1 Limitations

Within the scope of a thesis there are limitations on what is feasible. The important limitations were: the computational resources available, the selection strategy may not be optimal, and there are limitations specific to the simulations methods.

5.1.1 Computational resources

For this thesis that involves the computation time available to perform the simulations and subsequent methods. As the results show, it was not always feasible, and also probably not fruitful, to assess every method for each type of simulation. Furthermore, it limited the granularity with which the simulations could run. This has been specifically clear for the number of features, where it was not feasible to perform many iterations at higher numbers, and for sample sizes, where it would take significant computation time to assess more samples per groups and more combinations of responder and non-responder group sizes.

This limitation was the result of having a single node, which became a particularly limiting factor with the generation of more realistic datasets because of RAM limitations. It also meant that it was not feasible to 'redo' the simulations to get a completely balanced result across all analyses and

To solve this multiple nodes in either a cluster (HPC) or cloud environment could be used to have more computational resources available. The reason why this was not used for this thesis was the problem of the additional complexity as a result of having to program for a HPC environment, which would be necessary to make full use of the cluster. A cloud environment would have been prohibitively expensive given the computation and storage requirements. Also, while the cloud environment would be immediately available, it is architecturally different from a HPC, meaning that a rewrite would have been required when converting to the cheaper HPC option (for scientific purposes).

5.1.2 Selection strategy

> The selection strategy of top x features is unlikely to be cost-optimal when many databases have 0 predictive features.

The selection strategy using the top x features can be sub-optimal when the number of predictive features is hard to estimate. The result could be too conservative in selecting features resulting in a lower cost of the further assessment of the features, but also an increase in the probability of missing a predictive feature. In addition, a strategy that always select x features may waste resources in case there are many datasets that have no predictive features.

If it is known that these datasets regularly contain no predictive features, a threshold based method may allow to not select any features for further assessment. Conversely, it may also select too many features to assess further in which case the top x method would still have to be used.

5.1.3 Limitations of the simulated datasets

Lack of balance at the level of the whole simulated dataset

In relation to the computational limitations is the issue of balancing the simulations of a generations to ensure that every stratum is composed equally. Balancing the simulations ensures that comparison are more straightforward.

No full simulation of interactions in the datasets that result from biological pathways

Biological pathways are not really considered. An approach that could amend this is through the use of a package like `SeqNet`.(Grimes and Datta 2021) However, this adds another simulation method and is unlikely to be computationally feasible. Besides, this method focuses more on the connection aspect between genes than on the differential expression in location-shift of a few predictive features.

Only a subset of the features in the Zhang data is used for `SPsimSeq`

Of note is that the package uses a subset of 5000 genes, which is important in the context of the computational load of using the packages (specifically high memory use for larger feature sets). The reason being the memory requirements for larger template datasets exceeding the capacity of the hardware available (32GB RAM).

Limits on the granularity of exploring the parameter space due to computation times

Number of genes especially lead to longer compute times. Cloud/cluster based computing adds complexity in implementing the methods, i.e. distributed systems are harder to develop. In addition, the storage (0.5 TB) and computation requirements (weeks of runtime on single node) are not insignificant. In short, a single node system was used for the analyses, limiting the number of simulations that were possible.

We have not considered batch effects or other steps in a practical sequencing workflow

A sequencing workflow includes steps related to the normalization of data, accounting for batch effect, ... In this thesis these were not considered. However, the `SPsimSeq` package does offer the functionality to incorporate this should this be desirable in further studies.

5.2 Comparison of methods of feature selection

The results have compared the precision and sensitivity of a series of methods of feature selection. We limit the discussion to those findings that are relevant or surprising. While LIMMA is often used, it seems to perform at about the same level as a t-test on our simulated data. It is easily outperformed by the AUC method and Xgboost. Where the AUC provides a convenient balance between high performance and low computational requirements. The Xgboost combined with the feature selection using SHAP works quite well in this context, especially at lower effect sizes. The method could be further improved by looking at alternatives to the K-fold cross validation.

5.2.1 Tweedie

The Tweedie method has a sound theoretical basis and is shown to work for larger number of predictive features.(Efron 2011) However, while there is a small benefit when using t-test hypothesis testing on simulated data based on normal distributions, under the more realistic simulations in this thesis there was no indication of any benefit of this method over the respective uncorrected alternatives. The conclusion is that EB correction using Tweedie did not produce sufficient evidence to consider this method in the context of small sampled sequencing data.

5.2.2 Using AI, the potential of neural networks?

The relatively simple neural networks already suffered from overfitting. This comes as no surprise given the limited sample size of the datasets that were used. Even considering that the underlying correlations of biological pathways expressed in the sequencing data may not have been fully kept because of the subset of the Zhang dataset that was used for simulation, the neural networks severely underperformed all other methods despite being more difficult to set up and taking longer to compute.

The idea of using neural networks in this context is not strange however. Given that sequencing is quite comparable in its output to regular images. That is a two dimensional grid of values. Similar to regular images where there is a spatial between pixels based on their location in the grid (e.g. along the borders of an object), there are correlations between the measurements in sequencing data. This is based in the biological processes that result from the expression profiles of different genes and their expression. This is by no means intended as an exhaustive comparison nor a full motivation of the transferability of methods used in image recognition to the analysis of sequencing data. However, it does give rise to the idea that if the underlying biological process could identified by the neural network, this could benefit the performance of the neural network as a prediction model for predictive features. It is clear, however, that similar to image recognition that this requires much more data to train such models than is available. Some author (e.g. (Azuaje 2019)) have suggested transfer learning in this context, which is routinely used in image recognition. It remains unclear how that should be done for small sampled datasets.

5.2.3 Data integration

There was a general performance reduction when using data integration. This is contrary to the idea that increasing the fraction of predictive features increases performance, the latter which was something that the data integration did accomplish. However, there were features removed so it may be that those features that were removed to eagerly, or that it removed to few of the non-predictive features for an appreciable effect. It is also important that some features could have gone undetected by the method of feature selection while being kept in the dataset and vice versa. The latter would be problematic for the performance.

Another caveat is that only a single, subjective, threshold of $p \leq 0.65$ was used. This is by no means the optimal threshold. As such, there is room for improvement in data integration.

6 Guidelines

This thesis set out to provide guidelines on five aspects of the statistical analysis of sequencing data. Based on the results of this thesis, considering the limitations as discussed above, the following guidelines are formulated.

On the number of (predictive) features

Adding more total features is like adding more hay when trying to find a needle in a haystack. **A higher number of total features does not improve performance**.

Having **more predictive features** does improve performance. This is only true however when the relative number of predictive features in the total number of features increases, see Figure 7. Therefore, every reasonable effort should be put into reducing the number of features. Features should only be added when it is likely that these increase the fraction of predictive features in the datasets. However, should that be possible, there should be reflection on why that specific data was not used from the outset. Using the haystack analogy, the hay that is added should contain more needles than the hay already under scrutiny.

The effect size of the predictive features

As there is no influencing the effect size, the main recommendation in this regard is to select the method with the highest precision. While our simulations show that specific methods work better for higher effect sizes, it seems that these are unrealistically large in practices. Therefore, it is preferable to use the method most sensitive at lower effect-sizes. Methods such as the **AUC** and especially **Xgboost** perform better for lower effect sizes, which we would therefore recommend among the methods used in this thesis.

The sample size of both the response and non-response group

Unfortunately, it remains unclear why there is a difference between unbalanced groups of the same total size and why this differs per method of feature selection, it is not possible to provide guidance on whether a certain balance is preferable.

What is clear is that sample size and whether the groups are balanced affects the performance of the methods of feature selection. Overall, **a higher number of samples in balanced groups has the highest performance.**

Data integration

The first point of guidance stresses the importance of a higher fraction of predictive features. As such, it seems logical that look at data integration to improve that fraction by removing non-predictive features. Based on the results in this thesis there was no clear gain in using data integration. However, there were exceptions and improvements could be made to the methods used. Therefore, there is **no clear guidance** on whether data integration is beneficial in this context.

7 Further research

Use larger, but more complex, data for training the neural networks

Of note is that the popular applications for neural networks are in imaging and text analysis. Where there is a clear spatial correlation between pixels and words respectively. The spatial and/or biological correlations were not considered in the simulation study. As a result the performance of neural networks may be lacking. Using the package `SPsimSeq` more extensively or `SeqNet` to simulate networks behind gene expression may improve performance for neural networks.

However, to fit such complex data more samples are needed. Would these be available, it could potentially become more feasible to look at transfer learning to apply these mostly pre-trained networks to smaller sample datasets.

Bootstrap the models for reducing uncertainty by considering the distribution of the results. Specifically, bootstrap many small neural networks in parallel.

A technical barrier stood in the way of using bootstrapped neural networks because of the lack of specific support for Jax in the Shap package. This would have meant long computation times because of the use of the more generally applicable, but slower, methods in the SHAP package. It would be of interest to have a user friendly way of working with Jax for neural network modelling and obtaining the SHAP values to be able to combine bootstrapping neural networks and SHAP values for the interpretation of the resulting model.

Data integration

A single threshold was used in this thesis. Further research could explore the effect of different thresholds on the precision and sensitivity using a constant dataset (e.g. the Zhang-2 used in this thesis).

Sample size simulations are not very fine-grained nor balanced

Simulating in a more fine-grained manner over the group sizes can further clarify the influence of sample size. Also simulating larger sample sizes than would be reasonable for the context could allow for making the guidelines more precise.

8 References

Assefa, Alemu Takele, Olivier Thas, Joris Meys, and Stijn Hawinkel. 2021. *SPsimSeq: Semi-Parametric Simulation Tool for Bulk and Single-Cell RNA Sequencing Data* (version 1.2.0). Bioconductor version: Release (3.13). https://doi.org/10.18129/B9.bioc.SPsimSeq.

Azuaje, Francisco. 2019. "Artificial Intelligence for Precision Oncology: Beyond Patient Stratification." *Npj Precision Oncology* 3 (1, 1): 1–5. https://doi.org/10.1038/s41698-019-0078-1.

Bijma, Fetsje, Marianne Jonker, and A. W. van der Vaart. 2018. *An Introduction to Mathematical Statistics*. Amsterdam University Press.

Boniolo, Fabio, Emilio Dorigatti, Alexander J. Ohnmacht, Dieter Saur, Benjamin Schubert, and Michael P. Menden. 2021. "Artificial Intelligence in Early Drug Discovery Enabling Precision Medicine." *Expert Opinion on Drug Discovery* 0 (0): 1–17. https://doi.org/10.1080/17460441.2021.1918096.

Brown, Morton B., and Alan B. Forsythe. 1974. "Robust Tests for the Equality of Variances." *Journal of the American Statistical Association* 69 (346): 364–67. https://doi.org/10.2307/2285659.

Buyse, Marc, Daniel J. Sargent, Axel Grothey, Alastair Matheson, and Aimery de Gramont. 2010. "Biomarkers and Surrogate End Points—the Challenge of Statistical Validation." *Nature Reviews Clinical Oncology* 7 (6): 309–17. https://doi.org/10.1038/nrclinonc.2010.43.

Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost: A Scalable Tree Boosting System." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August, 785–94. https://doi.org/10.1145/2939672.2939785.

Efron, Bradley. 2011. "Tweedie's Formula and Selection Bias." *Journal of the American Statistical Association* 106 (496): 1602–14. https://www.jstor.org/stable/23239562.

Efron, Bradley, and Trevor Hastie. 2016. *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. Institute of Mathematical Statistics Monographs. New York, NY: Cambridge University Press.

Fagerland, Morten W., and Leiv Sandvik. 2009. "Performance of Five Two-Sample Location Tests for Skewed Distributions with Unequal Variances." *Contemporary Clinical Trials* 30 (5): 490–96. https://doi.org/10.1016/j.cct.2009.06.007.

"Google/Jax: Composable Transformations of Python+NumPy Programs: Differentiate, Vectorize, JIT to GPU/TPU, and More." n.d. GitHub. Accessed August 24, 2021. https://github.com/google/jax.

Grimes, Tyler, and Somnath Datta. 2021. "SeqNet: An R Package for Generating Gene-Gene Networks and Simulating RNA-Seq Data." *Journal of Statistical Software* 98 (1, 1): 1–49. https://doi.org/10.18637/jss.v098.i12.

Guthrie, William F. 2020. "NIST/SEMATECH e-Handbook of Statistical Methods (NIST Handbook 151)." National Institute of Standards and Technology. https://doi.org/10.18434/M32189.

Hastie, Trevor, Robert Tibshirani, and J. H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer Series in Statistics. New York, NY: Springer.

Kuhn, Max, and Kjell Johnson. 2013. *Applied Predictive Modeling*. New York, NY: Springer New York. https://doi.org/10.1007/978-1-4614-6849-3.

Lesaffre, Emmanuel, and Andrew Lawson. 2012. *Bayesian Biostatistics*. Statistics in Practice. Chichester: Wiley.

Levene, H. 1960. In *Contributions to Probability and Statistics; Essays in Honor of Harold Hotelling.*, by Ingram Olkin, 278–92. Stanford, Calif.: Stanford University Press.

Lundberg, Scott M., Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. 2020. "From Local Explanations to Global Understanding with Explainable AI for Trees." *Nature Machine Intelligence* 2 (1, 1): 56–67. https://doi.org/10.1038/s42256-019-0138-9.

Lundberg, Scott M, and Su-In Lee. 2017. "A Unified Approach to Interpreting Model Predictions." In *Advances in Neural Information Processing Systems*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

Phipson, Belinda, Stanley Lee, Ian J. Majewski, Warren S. Alexander, and Gordon K. Smyth. 2016. "Robust Hyperparameter Estimation Protects Against Hypervariable Genes and Improves Power to Detect Differential Expression." *The Annals of Applied Statistics* 10 (2): 946–63. https://doi.org/10.1214/16-

AOAS920.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier." August 9, 2016. http://arxiv.org/abs/1602.04938.

Ritchie, Matthew E., Belinda Phipson, Di Wu, Yifang Hu, Charity W. Law, Wei Shi, and Gordon K. Smyth. 2015. "Limma Powers Differential Expression Analyses for RNA-Sequencing and Microarray Studies." *Nucleic Acids Research* 43 (7): e47–47. https://doi.org/10.1093/nar/gkv007.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, et al. 2016. "Mastering the Game of Go with Deep Neural Networks and Tree Search." *Nature* 529 (7587, 7587): 484–89. https://doi.org/10.1038/nature16961.

"Statsmodels/Statsmodels: Statsmodels: Statistical Modeling and Econometrics in Python." n.d. GitHub. Accessed August 10, 2021. https://github.com/statsmodels/statsmodels.

Thas, Olivier. 2010. *Comparing Distributions*. Springer Series in Statistics. New York, NY: Springer New York. https://doi.org/10.1007/978-0-387-92710-7.

Vamathevan, Jessica, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, et al. 2019. "Applications of Machine Learning in Drug Discovery and Development." *Nature Reviews. Drug Discovery* 18 (6): 463–77. https://doi.org/10.1038/s41573-019-0024-5.

Vinyals, Oriol, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, et al. 2019. "Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning." *Nature* 575 (7782, 7782): 350–54. https://doi.org/10.1038/s41586-019-1724-z.

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, et al. 2020. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." *Nature Methods* 17: 261–72. https://doi.org/10.1038/s41592-019-0686-2.

Yu, Tong, and Hong Zhu. 2020. "Hyper-Parameter Optimization: A Review of Algorithms and Applications." March 12, 2020. http://arxiv.org/abs/2003.05689.

Zhang, Wenqian, Ying Yu, Falk Hertwig, Jean Thierry-Mieg, Wenwei Zhang, Danielle Thierry-Mieg, Jian Wang, et al. 2015. "Comparison of RNA-Seq and Microarray-Based Models for Clinical Endpoint Prediction." *Genome Biology* 16 (1). https://doi.org/10.1186/s13059-015-0694-1.

A Appendix

A.1 Selected code

A.1.1 Code for Normal simulation method

```
arr_non_pred = jax_normal(key=key, mu=0, sigma=1,
                    shape=(int((n0 + n1) * int(n_t)-int(n_pred)),))
arr_pred = jnp.array([])

for i in range(int(n_pred)):
    arr_pred = jnp.append(arr_p,
                  np.append(
                          jax_normal(key=key + i,
                                     mu=meta.mu_non_predictive,
                                     sigma=0,
                                     shape=(int(n0),)),
                          jax_normal(key=key + i,
                                     mu=mu_pred,
                                     sigma=jnp.sqrt(s_2_pred),
                                     shape=(int(n1),))
                          )
                      )
```

A.1.2 Welch's t-test

```
from jax import vmap, jit
import jax.numpy as jnp

@jit
def welch_ttest(x1, x2):
    x_1 = x1.mean()
    x_2 = x2.mean()
    s1 = jnp.sqrt(x1.var())
    s2 = jnp.sqrt(x2.var())
    n1 = len(x1)
    n2 = len(x2)
    return (x_2 - x_1) / (jnp.sqrt(s1 ** 2 / n1 + s2 ** 2 / n2))
t = vmap(welch_ttest)(g0, g1)
```

A.1.3 Variance calculation of the AUC

Variance calculation is implemented in the `calc_auc_var` function in `tweedie.py`. The calculation of p is in the C file `calc_rho.c`.

```
def calc_auc_var(pi, x_1, x_2):
    n_1 = x_1.shape[1]
    n_2 = x_2.shape[1]
    n_genes = x_2.shape[0]
```

```
    # Load C functions
    lib = cdll.LoadLibrary('calc_rho.so')
    calc_p_1 = lib.calc_p_1
    calc_p_1.restype = ndpointer(dtype=c_double, shape=(n_genes,))
    calc_p_2 = lib.calc_p_2
    calc_p_2.restype = ndpointer(dtype=c_double, shape=(n_genes,))
    # Prepare data
    x_1_np = np.asarray(x_1)
    x_1_np = np.require(x_1_np, float, ['CONTIGUOUS', 'ALIGNED'])
    x_2_np = np.asarray(x_2)
    x_2_np = np.require(x_2_np, float, ['CONTIGUOUS', 'ALIGNED'])
    # Call functions
    p_1 = calc_p_1(c_void_p(x_1_np.ctypes.data), c_void_p(x_2_np.ctypes.data),
        c_int(n_1), c_int(n_2),
        c_int(n_genes))
    p_2 = calc_p_2(c_void_p(x_1_np.ctypes.data), c_void_p(x_2_np.ctypes.data),
        c_int(n_1), c_int(n_2),
        c_int(n_genes))
    # Complete the calculation
    pi = np.where(pi == 0, 1e-3, pi) # to prevent numerical issues due to estimated probability of 0
    rho_1 = (p_1 - pi**2) / (pi - pi**2)
    rho_2 = (p_2 - pi**2) / (pi - pi**2)
    # Replace 0 with atom of probability to prevent dividing by 0 leading to NaN in calculation
    v = n_1 * n_2 / ((1+(n_1-1)*rho_1)/(1-1/n_2) + (1+(n_2-1)*rho_2)/(1-1/n_1))
    auc_var = (pi*(1-pi)) / v
    C = 1.96/jnp.sqrt(v)
    low = (1/(1+C)) * (pi + .5*C - jnp.sqrt(C*(pi*(1-pi) + C/4)))
    high = (1/(1+C)) * (pi + .5*C + jnp.sqrt(C*(pi*(1-pi) + C/4)))
    return {'variance': auc_var, 'ci_low': low, 'ci_high': high}
```

C-code for calculating p_1

```
double * calc_p_1(const double * x_1, const double * x_2, int n_1, int n_2, int n_genes){
    double * pointer_p_1 = (double *)malloc(sizeof(double) * n_genes);
    for (int index=0; index <n_genes; index++){
        int index_1 = index * n_1;
        int index_2 = index * n_2;
        pointer_p_1[index] = 0;
        for (int h=0; h<n_1; h++){
            for (int i=0; i<n_1; i++){
                for (int j=0; j<n_2; j++){
                    if (h != i) {
                        if ((x_1[index_1 + i] <= x_2[index_2 + j]) &&
                        (x_1[index_1 + h] <= x_2[index_2 + j])){
                            pointer_p_1[index] += 1;
         } } } } }
        pointer_p_1[index] = pointer_p_1[index] / (n_1 * n_2 * (n_1 - 1));
    }
    return pointer_p_1;
}
```