Software Benchmark – Classification Tree Algorithms for Cell Atlases Annotation Using Single-Cell RNA-Sequencing Data

Omar Alaqeeli, Li Xing, & Xuekui Zhang

April 2021

microbiology research

MDPI

Article

Software Benchmark—Classification Tree Algorithms for Cell Atlases Annotation Using Single-Cell RNA-Sequencing Data

Omar Alaqeeli [1], Li Xing [2] and Xuekui Zhang [1,*]

[1] Department of Mathematics and Statistics, University of Victoria, Victoria, BC V8P 5C2, Canada; oalaqeli@uvic.ca
[2] Department of Mathematics and Statistics, University of Saskatchewan, Saskatoon, SK S7N 5A2, Canada; lix491@mail.usask.ca
* Correspondence: xuekui@uvic.ca; Tel.: +1-250-721-7455

Abstract: Classification tree is a widely used machine learning method. It has multiple implementations as R packages; rpart, ctree, evtree, tree and C5.0. The details of these implementations are not the same, and hence their performances differ from one application to another. We are interested in their performance in the classification of cells using the single-cell RNA-Sequencing data. In this paper, we conducted a benchmark study using 22 Single-Cell RNA-sequencing data sets. Using cross-validation, we compare packages' prediction performances based on their Precision, Recall, F_1-score, Area Under the Curve (AUC). We also compared the Complexity and Run-time of these R packages. Our study shows that *rpart* and *evtree* have the best Precision; *evtree* is the best in Recall, F_1-score and AUC; *C5.0* prefers more complex trees; *tree* is consistently much faster than others, although its complexity is often higher than others.

Keywords: classification tree; single-cell RNA-Sequencing; benchmark; precision; recall; F_1-score; complexity; Area Under the Curve; run-time

Citation: Alaqeeli, O.; Xing, L.; Zhang, X. Software Benchmark—Classification Tree Algorithms for Cell Atlases Annotation Using Single-Cell RNA-Sequencing Data. *Microbiol. Res.* **2021**, *12*, 317–334. https://doi.org/10.3390/microbiolres12020022

Academic Editor: Beniamino T. Cenci-Goga

Received: 3 March 2021
Accepted: 1 April 2021
Published: 7 April 2021

1. Introduction

Most human tissues are considered to be a micro-environment, which consists of various types of cells. The Cell Atlas contains information about the proportions of cells in each type and cell type-specific gene expression profiles, which is critical for studying and treating tumors and other diseases [1–5]. Single-cell sequencing is a new technology developed recently, which allows researchers to study gene expression levels at a high resolution of single-cell level [6–13]. In the last few years, using single-cell RNA-sequencing (scRNA-seq) technology, Cell Atlases were studied for many diseases and tumors, such as Fabric Lung [14] and brain tumors [15,16]. To annotate the cell types using scRNA-seq data, researchers first cluster the cells using an unsupervised machine learning approach, and then 'manually' annotate the cell type of cells in each cluster according to their gene expression profiles. This manual approach is time consuming and generates inconsistent results across studies depending on the decisions of researchers.

While many Cell Atlases were built for various cell types and diseases, in newer studies researchers do not have to use an unsupervised learning approach to annotate Cell Atlases. A supervised learning approach can be applied to annotate cell types in a much faster and less subjective way [17–24]. We consider the supervised learning approach for cell-type annotation as a special case of a more general purpose task, that is, classification of cell phenotypes, such as cell development stages, stimulated cells versus wild-type cells and so forth.

A classification tree is a popular supervised machine learning method, applied widely in genomics research [25–30]. The structures of classification trees represent predicted decisions (the higher the level in the hierarchy, the better the prediction) based on analysis

of numerical values or categories. We are interested in applying the classification tree method to annotate cell types using scRNA-seq data.

The classification tree method is implemented in multiple software packages, with different performances with classification and speed. It is unclear which package can perform better than others in our task of cell-type annotation. This concern motivated our work in this paper.

Using 22 scRNA-seq data sets from published studies, we will conduct a benchmark study to investigate the performance of the five most common software R packages for classification. These data sets were pre-processed using a consistent procedure and made available via the Conquer repository [31].

The rest of this paper is organized as follows—in the materials and methods section, we will list methods under testing, describe selected data sets in detail and explain our methodology of conducting our experiment. In the results section, we will share our results and provide analyses. Finally, we will summarize our findings in the discussion section and draw our conclusion.

2. Materials and Method

2.1. Notations

The cell phenotype classification problem is a supervised machine learning task. The predictors are the gene expression profiles of each cell, which is denoted as a matrix X_{ij}, where i represents the index of genes and j represents the index of cells. The outcome of the classification is the cell phenotypes, which is a vector denoted as Y_i, where i represents the genes that have expression.

2.2. Software Packages to Be Evaluated

For multi-class classification, we can construct multiple binary classification models (in-class versus out-of-class), one for each class. Each model can estimate the probability of a class and assign the membership to the class with the largest probability. Furthermore, many cell-type classification problems are binary (e.g., stimulated versus wild-type).

So for simplicity, in this study, we are only considering binary classification performance. When describing classification procedures, we use 1 to denote *True* and 0 to denote *False*. The software packages to be evaluated are listed as below.

rpart [32] (Recursive Partitioning and Regression Trees) is the most commonly used R package that perform both classification and regression. It uses Gini impurity where the probability of any randomly selected elements from a data set incorrectly classified is calculated. The probability score is calculated by summing all probability and subtracting from 1, that is,

$$\Sigma P_i = 1 - P_i$$

ctree [33] (Conditional Inference Trees) is part of *partykit* R package. The tree's algorithm tests a condition or a hypothesis between variables and then splits if the hypothesis was rejected.

evtree [34] (Evolutionary Tree) is based on an evolutionary algorithm. Using the package, a random tree is created at first and then updated periodically after each iteration. The algorithm halts when no change to the most current model is required.

tree package is used for classification or regression. It is based on measuring the impurity and deciding where to create a split. Eventually, splitting halts when the split is not worth following.

C5.0 is a package that extends the C4.5 classification algorithm which itself is an extension of the ID3 algorithm. The algorithm creates a tree by calculating the entropy of samples under testing and splits the branch accordingly.

2.3. The Benchmark Data Sets

The data sets under testing were extracted from the project *Conquer* (consistent quantification of external RNA-Sequencing data) repository developed by C. Soneson and M.

Robinson at the University of Zurich, Switzerland [31]. Three organisms were included in the repository—Homo sapiens, Mus musculus and Danio rerio. Each data set contains a different number of cells. Six protocols were followed to obtain cell sequences—SMARTer C1, Smart-Seq2, SMART-Seq, Tang, Fluidigm C1Auto prep and SMARTer.

We have explored all data sets in the *Conquer* repository. Unfortunately, not all data sets suit our test. For a data set to be suitable for our testing methodology, we have to make sure that its samples can be divided into two groups based on a common phenotype for each group; thus, we can identify that phenotype with 1s or 0s. For example, the data set GSE80032 was excluded because all its samples have the same phenotype thus we have no bases to divide them. Also, each group of samples must not be so small that, when testing, they generate a misclassification or execution errors. We found that both sample groups must have at least 30 samples to avoid errors and misclassification.

There are 22 data sets that fit our test. These data sets are listed in Table 1 as they were presented in the *Conquer* repository along with the protocol type used.

Table 1. List of selected data sets along with their IDs, organisms where the cells were taken from, a brief description, cell counts and sequencing protocols used.

Data Set	ID	Organism	Brief Description	# of Cells	Protocol
EMTAB2805	Buettner2015	Mus musculus	mESC in different cell cycle stages	288	SMARTer C1
GSE100911	Tang2017	Danio rerio	hematopoietic and renal cell heterogeneity	245	Smart-Seq2
GSE102299-smartseq2	Wallrapp2017	Mus musculus	innate lymphoid cells from mouse lungs after various treatments	752	Smart-Seq2
GSE45719	Deng2014	Mus musculus	development from zygote to blastocyst + adult liver	291	SMART-Seq
GSE48968-GPL13112	Shalek2014	Mus musculus	dendritic cells stimulated with pathogenic components	1378	SMARTer C1
GSE48968-GPL17021-125bp	Shalek2014	Mus musculus	dendritic cells stimulated with pathogenic components	935	SMARTer C1
GSE48968-GPL17021-25bp	Shalek2014	Mus musculus	dendritic cells stimulated with pathogenic components	99	SMARTer C1
GSE52529-GPL16791	Trapnell2014	Homo sapiens	primary myoblasts over a time course of serum-induced differentiation	288	SMARTer C1
GSE52583-GPL13112	Treutlein2014	Mus musculus	lung epithelial cells at different developmental stages	171	SMARTer C1
GSE57872	Patel2014	Homo sapiens	glioblastoma cells from tumors + gliomasphere cell lines	864	SMART-Seq
GSE60749-GPL13112	Kumar2014	Mus musculus	mESCs with various genetic perturbations, cultured in different media	268	SMARTer C1
GSE60749-GPL17021	Kumar2014	Mus musculus	mESCs with various genetic perturbations, cultured in different media	147	SMARTer C1
GSE63818-GPL16791	Guo2015	Homo sapiens	primordial germ cells from embryos at different times of gestation	328	Tang
GSE66053-GPL18573	Padovan Merhar2015	Homo sapiens	live and fixed single cells	96	Fluidigm C1 Auto prep
GSE71585-GPL13112	Tasic2016	Mus musculus	cell type identification in primary visual cortex	1035	SMARTer
GSE71585-GPL17021	Tasic2016	Mus musculus	cell type identification in primary visual cortex	749	SMARTer

Table 1. *Cont.*

Data Set	ID	Organism	Brief Description	# of Cells	Protocol
GSE71982	Burns2015	Mus musculus	utricular and cochlear sensory epithelia of newborn mice	313	SMARTer C1
GSE77847	Meyer2016	Mus musculus	Dnmt3a loss of function in Flt3-ITD and Dnmt3a-mutant AML	96	SMARTer C1
GSE79102	Kiselev2017	Homo sapiens	different patients with myeloproloferative disease	181	Smart-Seq2
GSE81903	Shekhar2016	Mus musculus	P17 retinal cells from the Kcng4-cre;stop-YFP X Thy1-stop-YFP Line#1 mice	384	Smart-Seq2
SRP073808	Koh2016	Homo sapiens	in vitro cultured H7 embryonic stem cells (WiCell) and H7-derived downstream early mesoderm progenitors	651	SMARTer C1
GSE94383	Lane2017	Mus musculus	LPS stimulated and unstimulated 264.7 cells	839	Smart-Seq2

Accessing information within these data sets is not a straightforward process. Thus, before proceeding in our test, each data set has to be prepared and normalized before fitting into our methods.

In order to access an abundance of genes, we closely followed procedural steps provided by the *Conquer* repository authors. By using R programming language, we retrieved ExperimentList instances that contained RangedSummarizedExperiment for gene-level object. This allowed us to access abundances which included TPM (Transcripts per million) abundance for each specific gene, gene count, length-scaled TPMs as well as average of transcripts length found in each sample for each particular gene. For our test, we chose genes TPM abundance and used them as input predictor matrix $X_{i,j}$, where i represents samples and j represents genes.

At this point, we had full access to the desired type of information; nevertheless, we had to normalize the matrix $X_{i,j}$ to fit into our test. At first, we rotated the dimensions of X so samples became rows and genes were the columns; thus, $X_{j,i}$.

We then looked into the phenotype associated with the data set, using *table(colData (dataset))* R command, in order to find two distinguished phenotypical characteristics associated with samples group in X. We denoted the first characteristic with 1 and the second with 0. We replaced samples IDs with either 1 or 0 so we could distinguish them within our classified model. For example, in the data set EMTAB2805, there are different phenotypes associated with each group of samples. We picked two phenotypes that were based on stages of *cell cycle stage*, $G1$ and $G2M$. Thus, samples associated with the first stage, $G1$, were replaced by 1, *True*, and samples associated with the second stage, $G2M$ were replaced by 0, *false*. We eliminated any extra samples associated with other cell cycle stages, if any.

At this point, j in $X_{j,i}$ is a binary value where the first group of sample with one distinguished phenotype is represented by 1 and the other group is represented by 0.

Table 2 includes all the aforementioned data sets but after identifying the appropriate phenotypes that were going to be used in our test.

Table 2. List of data sets used in the test along with their associated phenotype that were sat for either 1 or 0.

#	Data Set	Phenotype	# of 1 s	# of 0 s
1	EMTAB2805	Cell Stages (G1 & G2M)	96	96
2	GSE100911	16-cell stage blastomere & Mid blastocyst cell (92–94 h post-fertilization)	43	38
3	GSE102299-smartseq2	treatment: IL25 & treatment: IL33	188	282
4	GSE45719	16-cell stage blastomere & Mid blastocyst cell (92–94 h post-fertilization)	50	60
5	GSE48968-GPL13112	BMDC (Unstimulated Replicate Experiment) & BMDC (1 h LPS Stimulation)	96	95
6	GSE48968-GPL17021-125bp	BMDC (On Chip 4 h LPS Stimulation) & BMDC (2 h IFN-B Stimulation)	90	94
7	GSE48968-GPL17021-25bp	LPS4h, GolgiPlug 1 h & stimulation: LPS4h, GolgiPlug 2 h	46	53
8	GSE52529-GPL16791	hour post serum-switch: 0 & hour post serum-switch: 24	96	96
9	GSE52583-GPL13112	age: Embryonic day 18.5 & age: Embryonic day 14.5	80	45
10	GSE57872	cell type: Glioblastoma & cell type: Gliomasphere Cell Line	672	192
11	GSE60749-GPL13112	culture conditions: serum+LIF & culture conditions: 2i+LIF	174	94
12	GSE60749-GPL17021	serum+LIF & Selection in ITSFn, followed by expansion in N2+bFGF/laminin	93	54
13	GSE63818-GPL16791	male & female	197	131
14	GSE66053-GPL18573	Cells were added to the Fluidigm C1 ... & Fixed cells were added to the Fluidigm C1 ...	82	14
15	GSE71585-GPL13112	input material: single cell & tdtomato labelling: positive (partially)	81	108
16	GSE71585-GPL17021	dissection: All & input material: single cell	691	57
17	GSE71982	Vestibular epithelium & Cochlear epithelium	160	153
18	GSE77847	sample type: cKit+ Flt3ITD/ITD,Dnmt3afl/- MxCre AML-1 & sample type: cKit+ Flt3ITD/ITD,Dnmt3afl/- MxCre AML-2	48	48
19	GSE79102	patient 1 scRNA-seq & patient 2 scRNA-seq	85	96
20	GSE81903	retina id: 1Ra & retina id: 1la	96	96
21	SRP073808	H7hESC & H7_derived_APS	77	64
22	GSE94383	time point: 0 & min time point: 75 min	186	145

Finally, we conducted a *Wilcoxon Test* on the data of X in order to find out p-values associated with each gene.

Due to problems related to memory exhaustion and time for excessive executions we had to trim our matrix; thus, we picked the first 1000 genes with the lowest p-value to include in X. Consequently, the final form of our matrix was $X_{j,i}$ where i ranged from 1 to 1000 and j ranged from 1 to n and separated into two groups of 1 s and 0 s.

At this point, we could proceed with testing and easily split our matrix. At the beginning of each testing iteration, we shuffled the rows of matrix (samples that were denoted by 1 and 0) to prevent any biases that may result from having identical samples IDs.

We proceeded into our 10-fold cross-validation testing. We took 10% of the samples (rows) and set them as our testing set. We took the rest of the samples as our training set. After we measured system's time (method's start time) we fitted the training set into the first method, *rpart*. Immediately after the method stopped its execution, we measured the system's time (the method's end time). We subtracted the starting time from the ending time so the result is considered to be the method's Run-time.

After we constructed our model, we used this model for prediction using our testing set. We first extracted the confusion matrix (a 2×2 matrix). In a few cases we had only a one dimension matrix due to the fact that no misclassification occurred, but we programmatically forced the formation of a 2×2 matrix in order to be able to always calculate the *true positive* values and *false positive* values. We then calculated the Precision by dividing the total number of *true positive* values over the total number of *true positive* values and the total number of *false positive* values. The result is the method's Precision. We also calculated the method's recall by dividing the total number of *true positive* values over the total number of *true positive* values and the total number of *false negative* values from the same confusion matrix.

Since we had the method's precision and recall we used them to calculate the value of F_1score that is the method's precision multiplied by the method's recall over the total value of both method's precision and recall, multiplied by 2.

We then proceeded to compute the receiver operating characteristic (ROC) using package **pROC** and, consequently, we obtained the AUC measurement. At the end, we constructed our model tree and computed the number of nodes and leaves (the complexity score).

We proceeded into testing the second method, *ctree*. We repeated the same steps applied on the first method, and we collected scores for *ctree* Precision, Recall, $F + 1$score, AuC, Complexity and Run-time. We then proceeded to test *evtree*, *tree* and C5.0, respectively.

In the second iteration of our cross-validation testing, we took the second fold (the adjacent 10% of the data set) as our testing set and the rest of the samples (including what was in the previously tested fold) as our training set. We repeated the same steps applied previously using these newly created testing and training sets. At the end of this 10-fold cross-validation testing, we had 10 scores for each method's Precision, Recall, F_1score, AUC, Complexity and Run-time. We collected these scores and took their *mean* as the final score.

The previously explained testing procedure (from shuffling samples to collecting mean scores of method's Precision, Recall, F_1score, AUC, Complexity and Runtime) is repeated 100 times. Each time we collect mean scores and we plot our final results thus we proceed to analysis.

2.4. Design of Evaluation Experiments

In this study, we compared the performance of classification using a Cross-validation approach. Each data set was randomly split into different folds. At each iteration, one fold was taken as a testing set (usually represents 10% of the total data set) and the rest of the remaining folds were taken as a training set (90% of the total data set). At the following iteration, the adjacent fold was sat as a testing set and the rest of the remaining folds as a training set and so on. The scores were defined to describe the classification criteria, and the Arithmetic Mean of all scores collected from each iteration is the final score that was used to measure the intended performance. The six scores of evaluation criteria (Precision, Recall, F_1score, AUC statistics, complexity and Runtime) are defined as below.

We define *Precision* as the number of correctly predicted true signals TP over the total number of false signals that are classified as true signals FP and the total number of the correctly predicted true signals. That is denoted as:

$$Precision = \frac{TP}{TP + FP}$$

We define *Recall* as the number of correctly predicted true signals TP over the total number of false signals that are classified as false signals FN and the total number of the correctly predicted true signals. That is denoted as:

$$Recall = \frac{TP}{TP + FN}$$

Consequently, we define the F_1score as:

$$F_1score = 2 \times (\frac{Precision \times Recall}{Precision + Recall})$$

We define *Complexity* as the total number of all splits in the model tree, that is, the total number of all nodes and leaves, including the first node.

The *Area Under the Curve (AUC)* calculates the area under an ROC (Receiver Operating Characteristic) curve, which is basically a plot of scores calculated for both *True Positive Rate* (represented in the y-axis) and *False Positive Rate* (represented in the x-axis) at different variations of thresholds. In other words, an AUC measures the ability for a model to distinguish between positive signals and negative signals. The higher the AUC's score, the better the model in prediction.

The summary of prediction performance of software depends on how the data sets are split into folds (or subsets) but this split effect is not what we intended to show in the evaluation of the methods. So, to remove impact of random split and generate more reliable results, we repeated the test 100 times—each time we randomized the matrix. Then we summarized the performance over 100 repeats and used these on the software.

Algorithm 1 describes the flow of the testing stages after we converted the matrix X into the desirable form.

Algorithm 1 Procedure of Evaluating Software Using Cross-validation

Data set is a matrix X of two dimensions; 1st dimension for Samples and 2nd dimension for Genes, containing TPM genes abundance.
Separate samples into two groups based on selected phenotypes.
In 1st group: replace $Sample\_id \leftrightarrow$ "1"
In 2nd group: replace $Sample\_id \leftrightarrow$ "0"
for $n \in \{1, ..., length(M)\}$ **do** ▷ ***Loop over genes***
    Run Wilcoxon-Test on n
    Extract and Store p-value
end for
Shorten X to include only 1000 genes with the lowest p-value.
for $k \in \{1, ..., 100\}$ **do** ▷ ***General loop***
    Shuffle rows in X
    for $x \in \{1, ..., 10\}$ **do** ▷ ***Loop over samples***
        Set 10% of X as $Testing\_set$
        Set 90% of X as $Training\_set$
        Measure current $time_{start}$
        Fit $Training\_set$ into the first method.
        Measure current $time_{end}$

Algorithm 1 *Cont.*

Predict model on $Testing\_set$
Calculate method's Run-time using $time_{end} - time_{start}$
Calculate method's Precision score using $Confusion\_Matrix$
Calculate method's Recall score using $Confusion\_Matrix$
Calculate method's F_1score score using method's Precision and Recall.
Calculate method's AUC score using ROC function
Calculate method's $Complexity$ score using model tree length
Repeat steps 15-22 on other packages
end for
Calculate *mean* of all scores collected
end for
Plot final scores

2.5. Implementation of Testing Code

The testing codes were implemented in R programming language. All scripts were deposited at https://github.com/Omar-Alaqeeli/ClassificationTreesPackagesTesting/ as of 27 February 2021. There are 22 R scripts that each script retrieves, processes one data set and fit it into all methods. The first script contains installation code lines of necessary packages and all script include code lines that import necessary packages which some of them depend on other packages. Subject data sets can not be deposited in the above link due to their size but they can be found at the *Conquer* repository website: http://imlspenticton.uzh.ch/robinson_lab/conquer_de_comparison/, (accessed on 6 April 2021). All codes were ran on a personal computer using version 3.6.1. of R Studio. Table 3 shows the full specifications of the system and R Studio used:

Table 3. System and R Studio specification details at the time of testing subjected R packages.

platform	×86_64-apple-darwin15.6.0
arch	×86_64
os	darwin15.6.0
system	×86_64, darwin15.6.0
status	
major	3
minor	6.1
year	2019
month	07
day	05
svn rev	76782
language	R
version.string	R version 3.6.1 (05-07-2019)
nickname	Action of the Toes

3. Results

In order to analyse and nominate the best method in each performance category, we collected the mean of all scores obtained from testing each method. The Methods performance was measured and recorded for Precision, Recall, F_1score, AUC, Complexity and Run-time. In each category, 100 scores were recorded at each of the 100 testing iterations. In this section, we analyse the methods' performance in each category (Figure 1) and provide

visualization of their performances with each data set (Figures 2–7), as well as an individual visualization of each method by itself on all data sets (Supplementary Figures S1–S30).

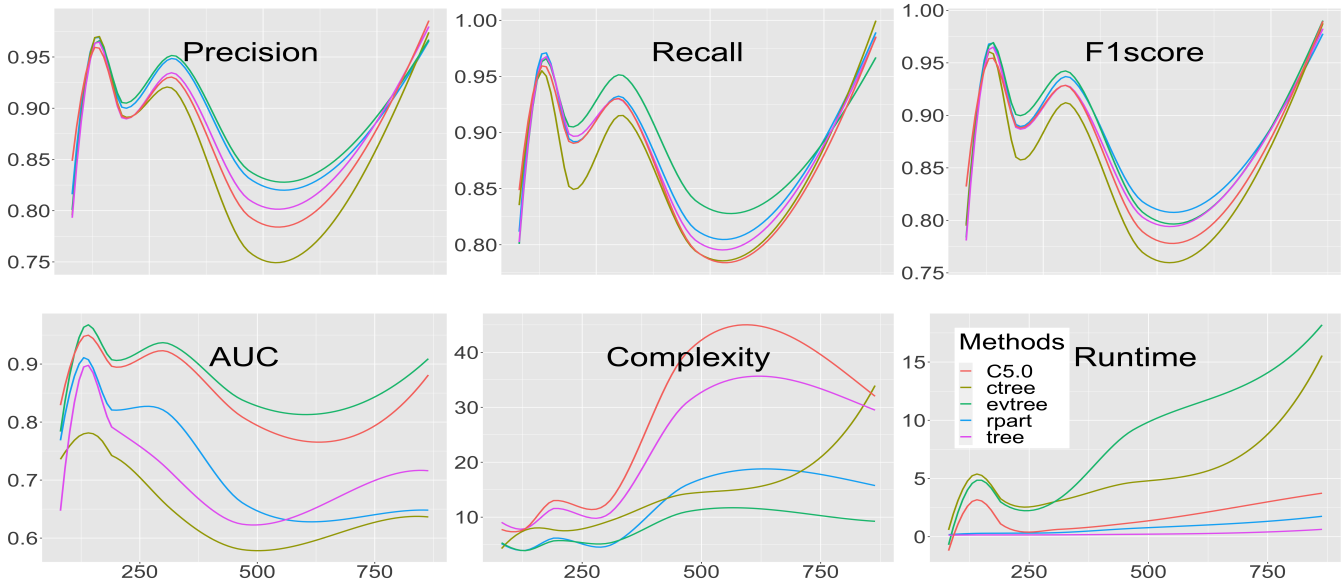

Figure 1. Sample sizes (*x*-axis) versus Precision, Recall, F_1 *Score*, Area Under the Curve (AUC), Complexity and Runtime (*y*-axis).

3.1. Precision

Precision scores for all methods were collected periodically at the end of each testing iteration. In some cases, the values were distributed on a wide Interquartile range, but in other cases they were the complete opposite. We found no correlation between the size of the Interquartile range and the size of the sets or the size of the Interquartile range and the number of 1s and 0s in the training and the testing sets. For example, when we used GSE71585-GPL17021, which has 748 samples, all methods show very narrow Interquartile ranges (approx. 0.95–0.98). Conversely, all methods have wide Interquartile ranges when tested on GSE77847, which has only 96 samples (approx. 0.6–0.9).

To compare the methods' precision, we calculated the arithmetic mean and standard deviation for all scores collected (Supplementary Tables S1 and S2). Although all values were close to each other, it appears that *rpart* and *evtree* scored the highest in 9 data sets, followed by all other methods that scored highest in 8 data sets. However, some methods that scored highest when testing a specific data set have shown a higher standard deviation which indicates that there is a fluctuation in their performances.

For some data sets, all, or nearly all, methods have similar mean values, such as in GSE48968-GPL17021-125bp, GSE60749-GPL13112 and GSE71585-GPL17021.

3.2. Recall

We calculated the methods' recall and their arithmetic mean immediately after the precision (Figure 2). Similarly to the precision, the values are distributed on a wide Interquartile range. Since the size of the sets has no effect on the score of the precision then the situation is similar for the recall.

We calculated the arithmetic mean and standard deviation for all scores collected for methods' recall (Supplementary Tables S3 and S4). *evtree* scored the highest in 10 data sets, followed by *rpart* and *tree* with highest in 8 data sets then *C5.0* in 7 data sets. *ctree* has scored the highest in only 4 data sets. Similar to the precision, all methods scored similar when testing GSE48968-GPL17021-125bp data set.

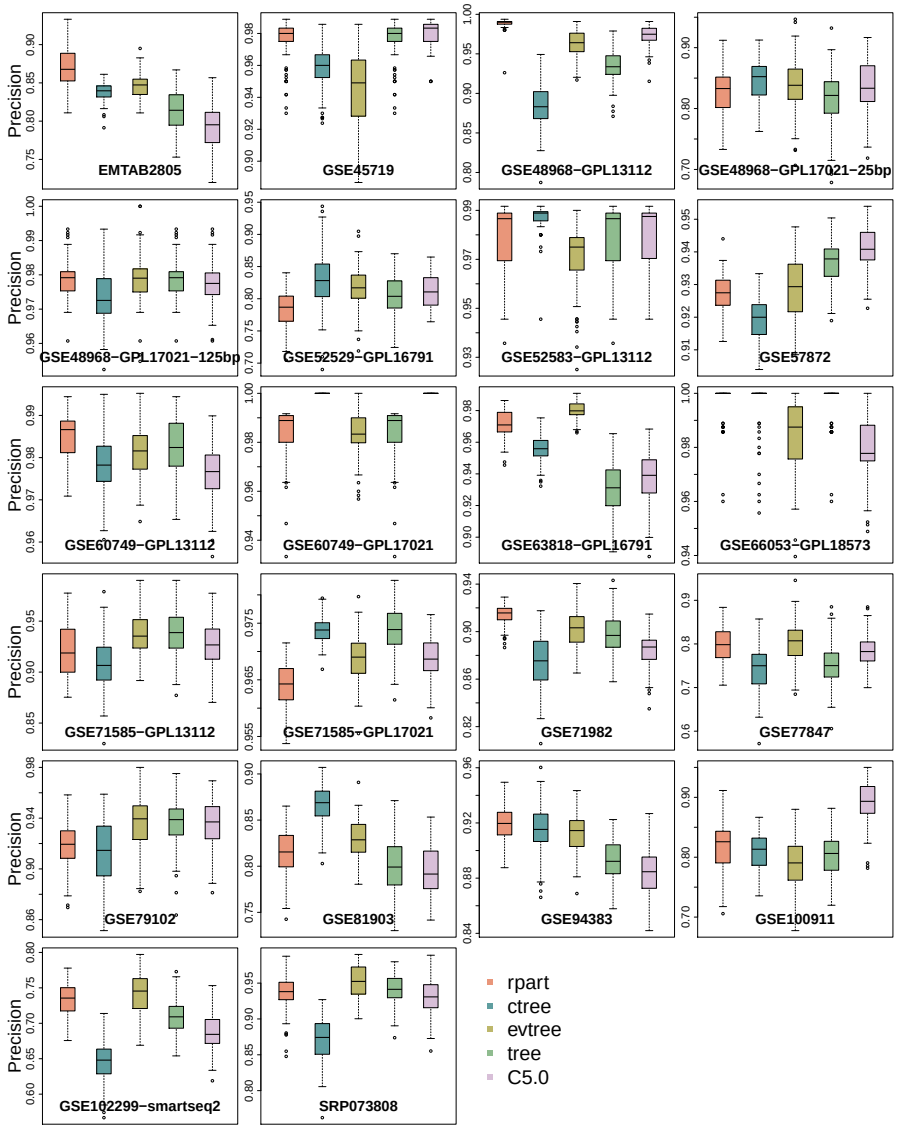

Figure 2. Methods' Precision scores on all data sets.

3.3. F_1-Score

Measuring methods' F_1-scores was a consequence of results obtained from methods' precision and recall (Figure 3).

Similar to calculating the precision and recall, we calculated the arithmetic mean scores and standard deviation of the F_1-scores (Supplementary Tables S5 and S6). *evtree* scored the highest in 13 data sets followed by *rpart* in 11 data sets. *tree* was next with 9 data sets and *C5.0* with highest in 7 data sets. *ctree* came last with highest in 6 data sets.

All methods scored the same in the data sets GSE48968-GPL17021-125bp, GSE52583-GPL13112, GSE60749-GPL17021.

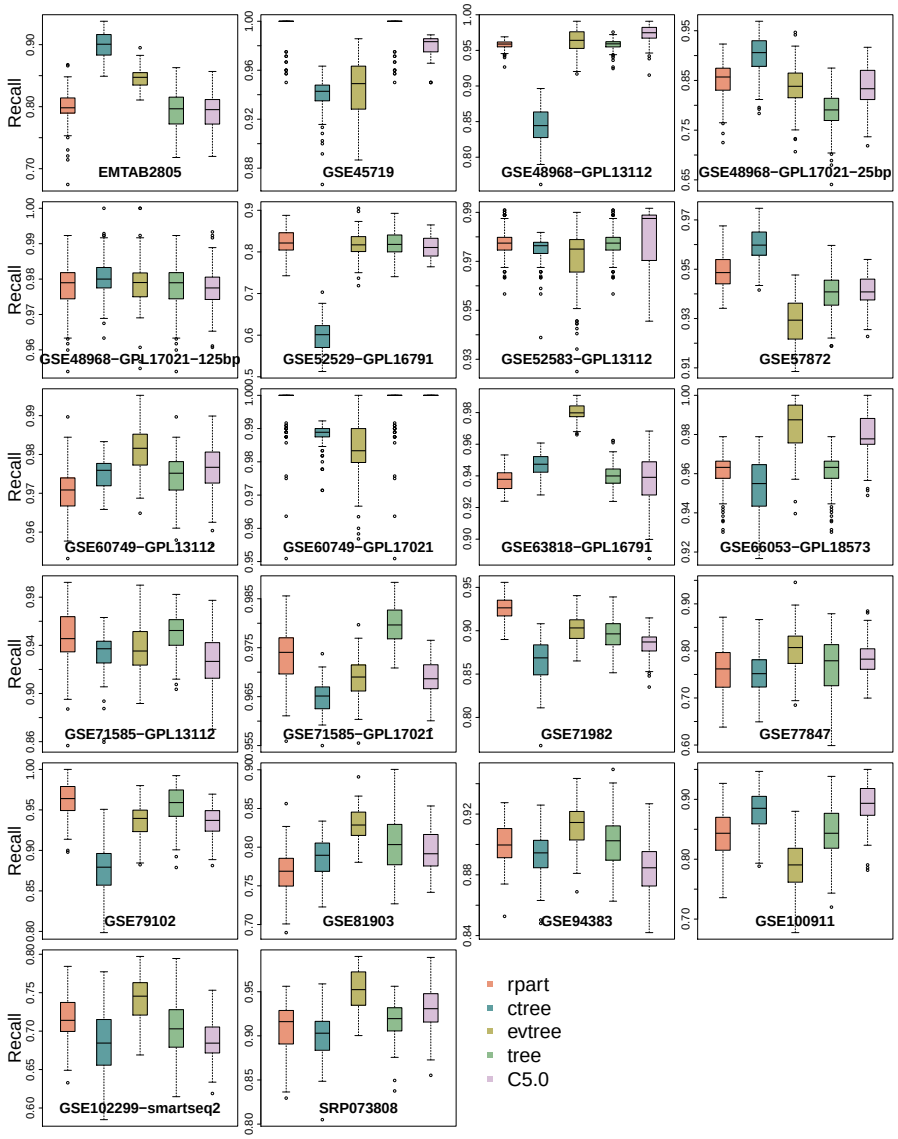

Figure 3. Methods' Recall scores on all data sets.

3.4. Area Under the Curve

Similarly to the Precision, Recall and F_1score, scores for methods' AUC fluctuated. All methods show larger Interquartile ranges when tested on some data sets and narrower Interquartile ranges when tested on others. As with the Precision, there is not any correlation between the size of the training and testing sets and scores recorded for methods' AUC.

We have calculated the mean and standard deviation for scores collected for all methods (Supplementary Tables S7 and S8) to be able to compare them. *evtree* has the highest score in 16 data sets with no method with a score near to it. This score followed by *C5.0* with the highest score in 9 data sets and then *rpart* with the highest score in 7 data sets. *ctree* and *tree* came out the lowest with 2 of the highest scores in only 2 data sets. All methods score the same (0.99) when testing on GSE60749-GPL17021.

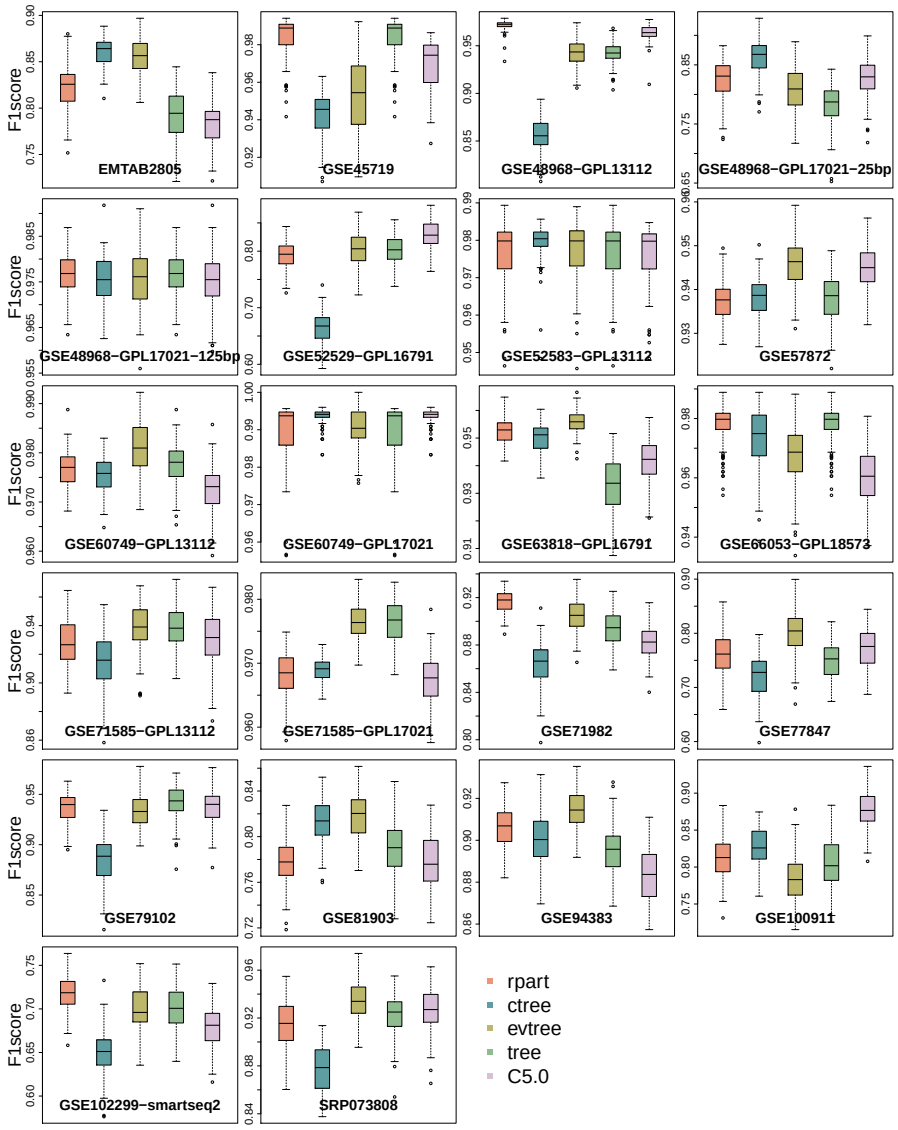

Figure 4. Methods' F1score scores on all data sets.

3.5. Complexity

The complexity scores were collected when creating model trees using the training sets at every iteration. It is normal to assume that a correlation exists between the mean scores of methods' complexities, or the size of the Interquartile range, and the size of the training set; the larger the training set, the higher the complexity mean score, thus the larger the Interquartile range. However, we found that this is not entirely accurate.

We calculated the mean and standard deviation for methods' Complexities (Supplementary Tables S9 and S10). *C5.0* scored the highest in 11 data sets, followed by *tree* with the highest in 8 data sets, then *ctree* in 5 data sets. *rpart* and *evtree* scored no near to them with only highest in 1 data sets.

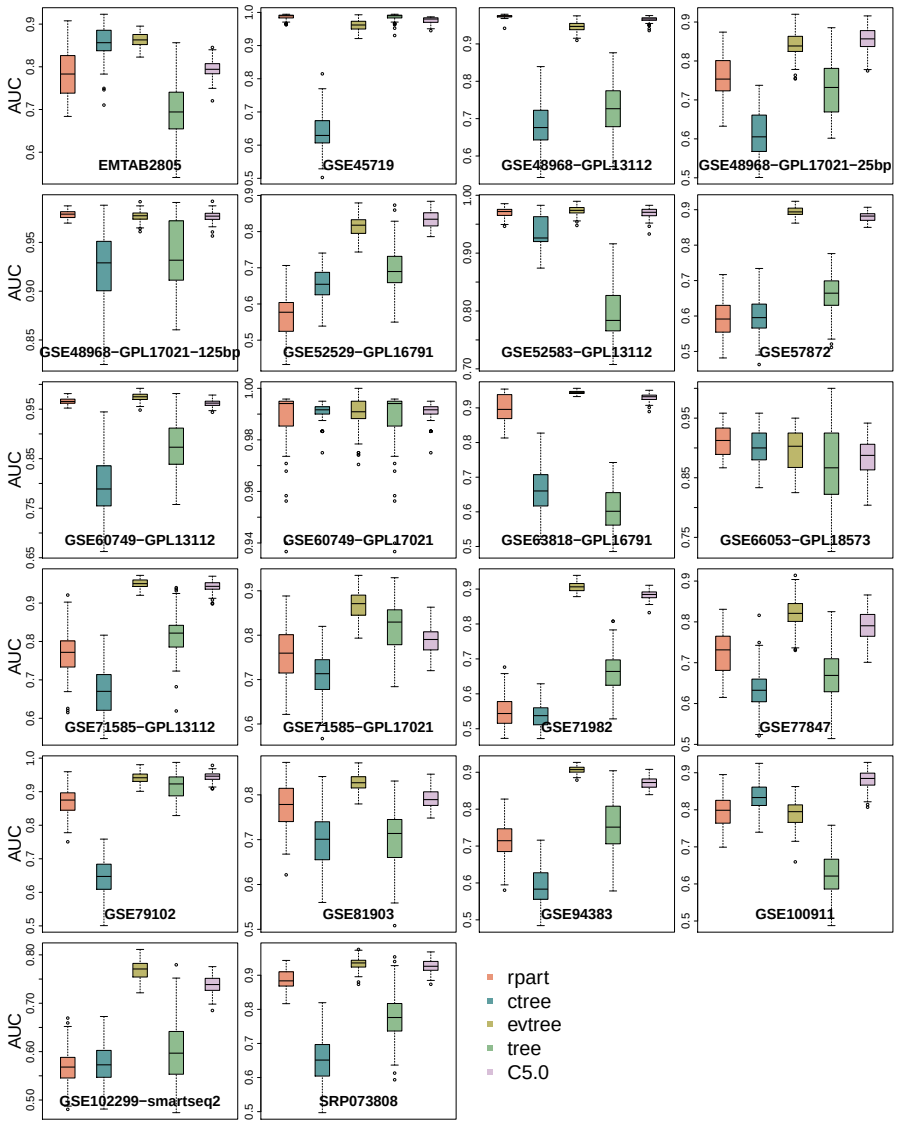

Figure 5. Methods' AUC scores on all data sets.

3.6. Runtime

A method's Run-time was calculated based on how long it takes the method to create a model tree. We calculated the Run-time, in seconds, for all methods at every iteration. The lower the score, the faster the method. Very few scores are outliers thus we eliminated them from the plots (Figure 6) so the y-axis is adjusted correctly thus to be able to view methods' Interquartile range correctly.

We found that there exists a correlation between the size of the run-time score Interquartile ranges and training sets.

Analyzing the mean and standard deviation of Run-times in (Supplementary Tables S11 and S12) we induce that *tree* has scored fastest in all 22 data sets with no method near it. No other method has scored lower in any data sets.

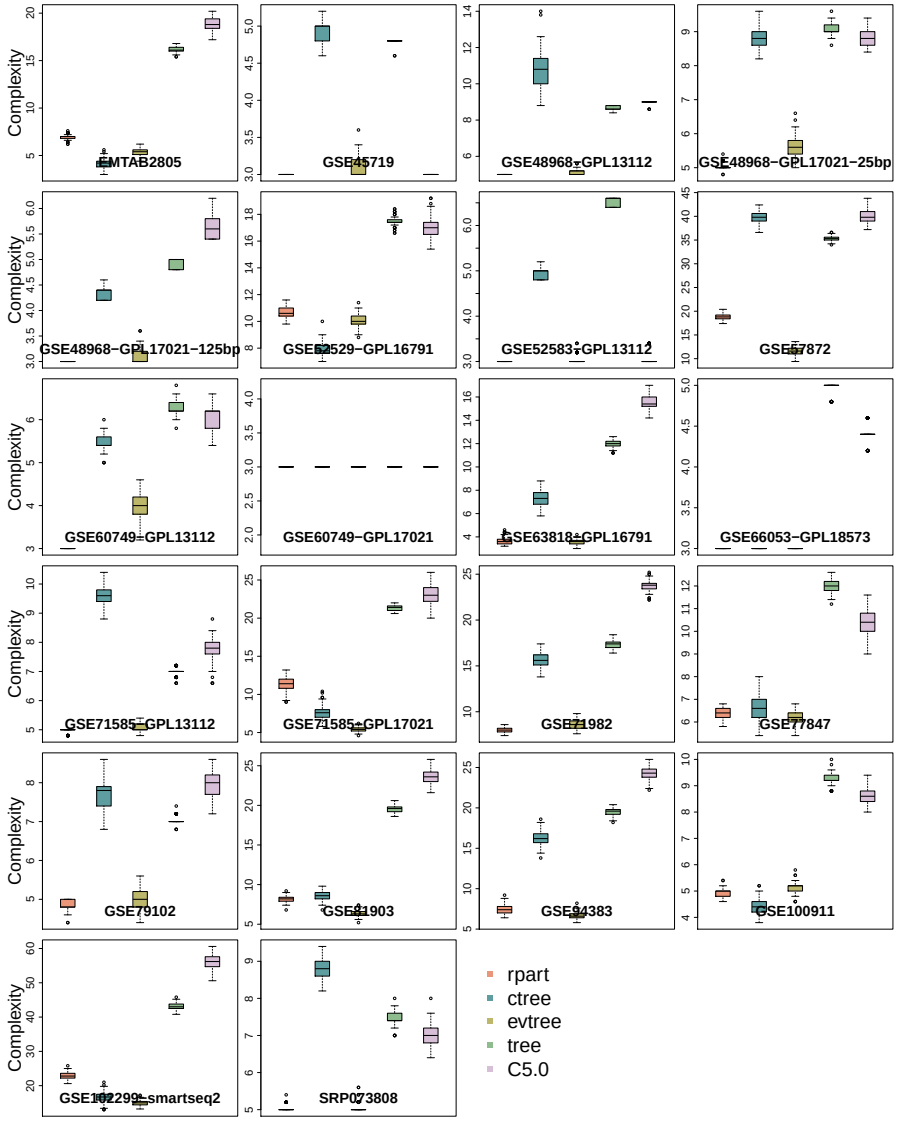

Figure 6. Methods' Complexity scores on all data sets.

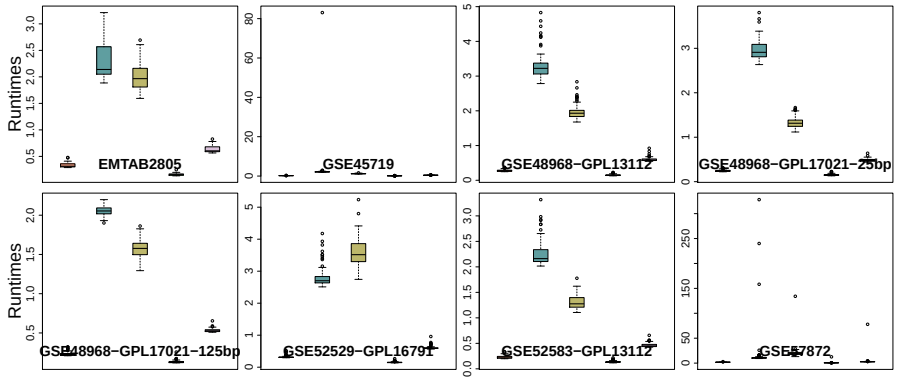

Figure 7. *Cont*.

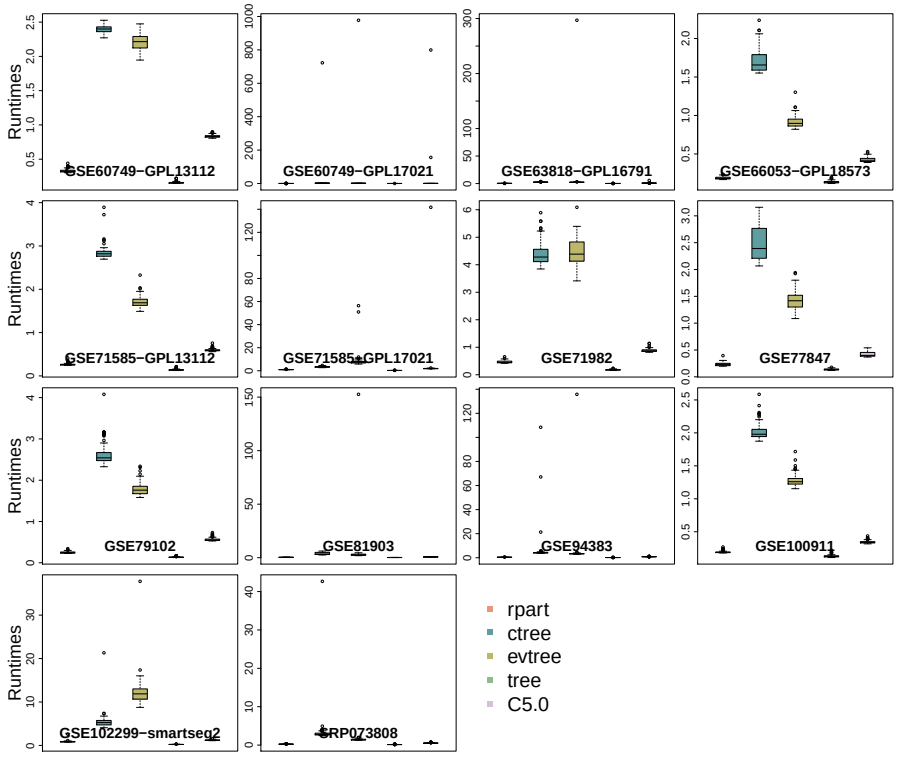

Figure 7. Methods' Runtime scores on all data sets.

4. Discussion and Conclusions

We have evaluated the performance of the five widely used methods using scRNA-seq data sets. We measured Precision, Recall, F_1score AUC, Complexity and Run-time using a cross-validation testing in 100 iterations. At every iteration, and for every performance category, we collected the scores from fitting all methods. We based our comparison on the mean of all those scores.

In terms of Precision, we nominate *rpart* as the best among other methods in this category, although all methods nearly close in performance to each others. For the Recall category, *evtree* has the best performance with a competition from *rpart* and *tree*. Similarly, *evtree* has the best the F_1score among all methods, followed by *rpart*.

In the AUC category, we found out that *evtree* has the top performance and none of the other methods are close to its performance. For Complexity, *C5.0* shows the ability to create more complex trees and only one method was close to its performance.

Although it is hard to measure an R package's Run-time performance, since a lot of factors can affect the accuracy of the measurement such as system hardware, or complexity of the algorithm used by the package, we ensured to measure the methods' Run-times under the same circumstances. Thus, we found out that *tree* is without a doubt the fastest among all methods since its Run-time is the lowest in all data sets.

Table 4 shows that *evtree* has the best performance in four categories; Precision, Recall, F_1score and AUC. *rpart* is competing with *evtree* in precision, *C5.0* always prefers more complex models; *tree* is the fastest consistently although it often fits more complex trees than others.

Table 4. In this summary table, *rpart* and *evtree* scored highest in 9 data sets with regard to Precision. *evtree* scored highest in 10, 13 and 16 data sets with regard to Recall, F_1score and AUC, respectively. *C5.0* scored highest in 11 data sets with regard to creating complex trees. Finally, *tree* shows that it is the fastest. The highest scores in each category are highlighted in bold font.

	rpart	***ctree***	***evtree***	***tree***	***C5.0***
Precision	**9**	8	**9**	8	8
Recall	8	4	**10**	8	7
F_1score	11	6	**13**	9	7
AUC	7	2	**16**	2	9
Complexity	1	5	1	8	**11**
Runtime	0	0	0	**22**	0

Supplementary Materials: The following are available online at https://www.mdpi.com/2036-7481/12/2/22/s1, Figure S1: rpart scores in all datasets., Figure S2: ctree scores in all datasets., Figure S3: evtree scores in all datasets., Figure S4: tree scores in all datasets., Figure S5: C5.0 scores in all datasets., Figure S6: rpart scores in all datasets., Figure S7: ctree scores in all datasets., Figure S8: evtree scores in all datasets., Figure S9: tree scores in all datasets., Figure S10: C5.0 scores in all datasets., Figure S11: rpart scores in all datasets., Figure S12: ctree scores in all datasets., Figure S13: evtree scores in all datasets., Figure S14: tree scores in all datasets., Figure S15: C5.0 scores in all datasets., Figure S16: rpart scores in all datasets., Figure S17: ctree scores in all datasets., Figure S18: evtree scores in all datasets., Figure S19: tree scores in all datasets., Figure S20: C5.0 scores in all datasets., Figure S21: rpart scores in all datasets., Figure S22: ctree scores in all datasets., Figure S23: evtree scores in all datasets., Figure S24: tree scores in all datasets., Figure S25: C5.0 scores in all datasets., Figure S26: rpart scores in all datasets., Figure S27: ctree scores in all datasets., Figure S28: evtree scores in all datasets., Figure S29: tree scores in all datasets., Figure S30: C5.0 scores in all datasets. Table S1: Methods Precision mean scores., Table S2: Methods Precision standard deviation scores., Table S3: Methods Recall mean scores., Table S4: Methods Recall standard deviation scores., Table S5: F1-score mean scores for methods using all data-sets., Table S6: F1-score standard deviation scores for methods using all data-sets., Table S7: AUC mean scores for all methods using all data-sets., Table S8: AUC standard deviation scores for all methods using all data-sets., Table S9: Complexity mean scores for methods using all datasets., Table S10: Complexity standard deviation scores for methods using all datasets., Table S11: Methods Run-time mean scores using all data-sets., Table S12: Methods Runtime standard deviation using all data-sets.

Author Contributions: Conceptualization, X.Z. and L.X.; formal analysis and programming, O.A. under supervision of X.Z.; writing—original draft preparation, O.A.; writing—review and editing, X.Z.; All authors have read and agreed to the published version of the manuscript.

Funding: The research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grants (LX and XZ) and Canada Research Chair Grant (XZ). This research was enabled in part by support provided by WestGrid (www.westgrid.ca, accessed on 6 April 2021) and Compute Canada (www.computecanada.ca, accessed on 6 April 2021).

Data Availability Statement: The testing codes and plots are available and can be downloaded from the GitHub page: github.com/Omar-Alaqeeli/ClassificationTreesTesting, (accessed on 6 April 2021). The scRNA-seq data of 22 studies are available on Conquer repository website: http://imlspenticton.uzh.ch/robinson_lab/conquer_de_comparison/, (accessed on 6 April 2021).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

AUC	Area Under the Curve
scRNA-seq	Single-cell Ribonucleicacid Sequencing
ROC	Receiver Operating Characteristic
TPM	Transcripts per million

References

1. Qian, J.; Olbrecht, S.; Boeckx, B.; Vos, H.; Laoui, D.; Etlioglu, E.; Wauters, E.; Pomella, V.; Verbandt, S.; Busschaert, P.; et al. A pan-cancer blueprint of the heterogeneous tumor microenvironment revealed by single-cell profiling. *Cell Res.* **2020**, *30*, 745–762. [CrossRef] [PubMed]
2. Zhou, Y.; Yang, D.; Yang, Q.; Lv, X.; Huang, W.; Zhou, Z.; Wang, Y.; Zhang, Z.; Yuan, T.; Ding, X.; et al. Single-cell RNA landscape of intratumoral heterogeneity and immunosuppressive microenvironment in advanced osteosarcoma. *Nat. Commun.* **2020**, *11*, 6322. [CrossRef]
3. Adams, T.S.; Schupp, J.C.; Poli, S.; Ayaub, E.A.; Neumark, N.; Ahangari, F.; Chu, S.G.; Raby, B.A.; DeIuliis, G.; Januszyk, M.; et al. Single-cell RNA-seq reveals ectopic and aberrant lung-resident cell populations in idiopathic pulmonary fibrosis. *Sci. Adv.* **2020**, *6*, eaba1983. [CrossRef] [PubMed]
4. Seyednasrollah, F.; Rantanen, K.; Jaakkola, P.; Elo, L.L. ROTS: Reproducible RNA-seq biomarker detector—Prognostic markers for clear cell renal cell cancer. *Nucleic Acids Res.* **2015**, *44*, e1. [CrossRef] [PubMed]
5. Arnold, C.D.; Gerlach, D.; Stelzer, C.; Boryń, Ł.M.; Rath, M.; Stark, A. Genome-Wide Quantitative Enhancer Activity Maps Identified by STARR-seq. *Science* **2013**, *339*, 1074–1077. [CrossRef]
6. Nawy, T. Single-cell sequencing. *Nat. Methods* **2014**, *11*, 18. [CrossRef]
7. Gawad, C.; Koh, W.; Quake, S.R. Single-cell genome sequencing: Current state of the science. *Nat. Rev. Genet.* **2016**, *17*, 175–188. [CrossRef]
8. Metzker, M.L. Sequencing technologies—The next generation. *Nat. Rev. Genet.* **2010**, *11*, 31–46. [CrossRef]
9. Elo, L.L.; Filen, S.; Lahesmaa, R.; Aittokallio, T. Reproducibility-Optimized Test Statistic for Ranking Genes in Microarray Studies. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2008**, *5*, 423–431. [CrossRef]
10. Kowalczyk, M.S.; Tirosh, I.; Heckl, D.; Rao, T.N.; Dixit, A.; Haas, B.J.; Schneider, R.K.; Wagers, A.J.; Ebert, B.L.; Regev, A. Single-cell RNA-seq reveals changes in cell cycle and differentiation programs upon aging of hematopoietic stem cells. *Genome Res.* **2015**, *25*, 1860–1872. [CrossRef]
11. Sandberg, R. Entering the era of single-cell transcriptomics in biology and medicine. *Nat. Methods* **2014**, *11*, 22–24. [CrossRef]
12. Islam, S.; Kjällquist, U.; Moliner, A.; Zajac, P.; Fan, J.B.; Lönnerberg, P.; Linnarsson, S. Characterization of the single-cell transcriptional landscape by highly multiplex RNA-seq. *Genome Res.* **2011**, *21*, 1160–1167. [CrossRef]
13. Patel, A.P.; Tirosh, I.; Trombetta, J.J.; Shalek, A.K.; Gillespie, S.M.; Wakimoto, H.; Cahill, D.P.; Nahed, B.V.; Curry, W.T.; Martuza, R.L.; et al. Single-cell RNA-seq highlights intratumoral heterogeneity in primary glioblastoma. *Science* **2014**, *344*, 1396–1401. [CrossRef]
14. Habermann, A.C.; Gutierrez, A.J.; Bui, L.T.; Yahn, S.L.; Winters, N.I.; Calvi, C.L.; Peter, L.; Chung, M.I.; Taylor, C.J.; Jetter, C.; et al. Single-cell RNA sequencing reveals profibrotic roles of distinct epithelial and mesenchymal lineages in pulmonary fibrosis. *Sci. Adv.* **2020**, *6*, eaba1972. [CrossRef] [PubMed]
15. Bauer, S.; Nolte, L.; Reyes, M. Segmentation of brain tumor images based on atlas-registration combined with a Markov-Random-Field lesion growth model. In Proceedings of the 2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Chicago, IL, USA, 30 March–2 April 2011; pp. 2018–2021. [CrossRef]
16. Darmanis, S.; Sloan, S.A.; Zhang, Y.; Enge, M.; Caneda, C.; Shuer, L.M.; Hayden Gephart, M.G.; Barres, B.A.; Quake, S.R. A survey of human brain transcriptome diversity at the single cell level. *Proc. Natl. Acad. Sci. USA* **2015**, *112*, 7285–7290. [CrossRef] [PubMed]
17. Pliner, H.A.; Shendure, J.; Trapnell, C. Supervised classification enables rapid annotation of cell atlases. *Nat. Methods* **2019**, *16*, 983–986. [CrossRef] [PubMed]
18. Jaakkola, M.K.; Seyednasrollah, F.; Mehmood, A.; Elo, L.L. Comparison of methods to detect differentially expressed genes between single-cell populations. *Briefings Bioinform.* **2016**, *18*, 735–743. [CrossRef]
19. Law, C.W.; Chen, Y.; Shi, W.; Smyth, G.K. Voom: Precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome Biol.* **2014**, *15*, R29. [CrossRef]
20. McCarthy, D.J.; Chen, Y.; Smyth, G.K. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic Acids Res.* **2012**, *40*, 4288–4297. [CrossRef]
21. Rapaport, F.; Khanin, R.; Liang, Y.; Pirun, M.; Krek, A.; Zumbo, P.; Mason, C.E.; Socci, N.D.; Betel, D. Comprehensive evaluation of differential gene expression analysis methods for RNA-seq data. *Genome Biol.* **2013**, *14*, 3158. [CrossRef]
22. Miao, Z.; Zhang, X. Differential expression analyses for single-cell RNA-Seq: Old questions on new data. *Quant. Biol.* **2016**, *4*, 243–260. [CrossRef]
23. Stegle, O.; Teichmann, S.A.; Marioni, J.C. Computational and analytical challenges in single-cell transcriptomics. *Nat. Rev. Genet.* **2015**, *16*, 133–145. [CrossRef] [PubMed]

24. Oshlack, A.; Robinson, M.D.; Young, M.D. From RNA-seq reads to differential expression results. *Genome Biol.* **2010**, *11*, 220. [CrossRef] [PubMed]
25. Seyednasrollah, F.; Laiho, A.; Elo, L.L. Comparison of software packages for detecting differential expression in RNA-seq studies. *Briefings Bioinform.* **2013**, *16*, 59–70. [CrossRef]
26. Wang, T.; Li, B.; Nelson, C.E.; Nabavi, S. Comparative analysis of differential gene expression analysis tools for single-cell RNA sequencing data. *BMC Bioinform.* **2019**, *20*, 40. [CrossRef]
27. Krzak, M.; Raykov, Y.; Boukouvalas, A.; Cutillo, L.; Angelini, C. Benchmark and Parameter Sensitivity Analysis of Single-Cell RNA Sequencing Clustering Methods. *Front. Genet.* **2019**, *10*, 1253. [CrossRef]
28. Anders, S.; Pyl, P.T.; Huber, W. HTSeq—A Python framework to work with high-throughput sequencing data. *Bioinformatics* **2014**, *31*, 166–169. [CrossRef]
29. Li, S.; Łabaj, P.; Zumbo, P.; Sykacek, P.; Shi, W.; Shi, L.; Phan, J.; Wu, P.Y.; Wang, M.; Wang, C.; et al. Detecting and correcting systematic variation in large-scale RNA sequencing data. *Nat. Biotechnol.* **2014**, *32*, 888–895. [CrossRef]
30. Delmans, M.; Hemberg, M. Discrete distributional differential expression (D 3 E)—A tool for gene expression analysis of single-cell RNA-seq data. *BMC Bioinform.* **2016**, *17*, 110. [CrossRef]
31. Soneson, C.; Robinson, M.D. Bias, robustness and scalability in single-cell differential expression analysis. *Nat. Methods* **2018**, *15*, 255–261. [CrossRef]
32. Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J. *Classification and Regression Trees*; The Wadsworth & Brooks/Cole statistics/probability Series; Wadsworth & Brooks/Cole Advanced Books & Software: Monterey, CA, USA, 1984.
33. Hothorn, T.; Hornik, K.; Zeileis, A. Unbiased Recursive Partitioning: A Conditional Inference Framework. *J. Comput. Graph. Stat.* **2006**, *15*, 651–674. [CrossRef]
34. Grubinger, T.; Zeileis, A.; Pfeiffer, K.P. evtree: Evolutionary Learning of Globally Optimal Classification and Regression Trees in R. *J. Stat. Softw. Artic.* **2014**, *61*, 1–29. [CrossRef]