Exploring and Communicating the Pruning Variability of Decision Trees Caused by Cross-Validation

Bresser, Madelief

Universiteit
Leiden
The Netherlands

Exploring and Communicating the Pruning Variability of Decision Trees Caused by Cross-Validation

Madelief Bresser

Thesis advisor: Dr. S.J.W. Willems
Second thesis advisor: Prof.dr. E.M.L. Dusseldorp

Defended on the 9th of July, 2025

MASTER THESIS

STATISTICS AND DATA SCIENCE
UNIVERSITEIT LEIDEN

Contents

Abstract

Decision trees are popular statistical methods because they are both intuitive and easy to implement through R packages, such as `rpart` and `metacart`. They work by iteratively dividing the data using decision rules depending on specific predictors, resulting in a tree-like structure that is easy to interpret. However, one of the main challenges of building a decision tree, is finding the optimal tree size. A tree that is too small can miss important patterns in the data. In contrast, a tree that is too big can overfit the data by including noise and, therefore, might not generalize well to new data. As a solution to this challenge, pruning methods are often used. Pruning simplifies the tree by removing excessive splits that provide little predictive value, or may reflect noise in the data. By removing these splits and nodes, while keeping the main structure and important splits at the top, the risk of overfitting is reduced. However, the pruning process depends on cross-validation, which requires random partitioning of the data. This random nature of data splitting generates variability, which can result in inconsistent pruning outcomes. The focus of this thesis is on pruning variability in random-effects meta-CART models. These models apply decision tree methods to meta-analytic data. In the meta-analytic context, effect sizes are used as the response variable, while moderators are the predictor variables that may explain differences in effect sizes across studies, such as the study characteristics. The aim of this thesis is to investigate how moderator type, effect size strength, moderator correlation, and the pruning strictness, contribute to the pruning variability. Pruning was repeated across 1000 iterations using different seeds on various simulated datasets. Results showed that the pruning outcomes varied significantly and depended on interactions between moderator type, moderator correlation, effect size strength, and pruning strictness. Additionally, three visualization tools were developed, and other suggestions were provided to visualize and communicate the pruning variability.

Chapter 1

Introduction

Decision trees are often used in data analysis for both classification and regression tasks (Mienye and Jere, 2024). This is because they provide an intuitive and practical approach to modeling decision-making. Their interpretability and flexibility make them useful in various fields, such as behavioral science, medicine, finance, and marketing (Finch et al., 2011; Kim et al., 2001; Leach et al., 2016; Li et al., 2020a; Podgorelec et al., 2002; Sarker et al., 2020; Trujillano et al., 2009). Decision trees build a tree-like structure by recursively splitting the data using decision rules that depend on specific predictor variables. The tree ends with terminal nodes that offer predicted values. For categorical response variables, splits aim to maximize the node purity through minimizing the Gini Index or entropy values (Breiman et al., 1984; James et al., 2013). For continuous response variables, they aim to reduce the residual variance within nodes. By building a tree-like structure through this recursive process, the model can detect complex data patterns. However, the main challenge of building a decision tree, is finding the optimal tree size. Underfitting could result from a decision tree that is too small, failing to capture the complex patterns present in the data. Furthermore, overfitting results from decision trees growing too big, capturing noise from the data, and therefore, reducing their generalizing performance (Breiman et al., 1984; Gey and Nedelec, 2005; Isaksson et al., 2008).

As a solution to this challenge, pruning methods are often used in order to remove excessive splits from the fully grown tree. Pruning removes the lower splits and nodes in order to simplify the model, but preserves the most important decision rules and nodes at the top of the tree. By only retaining splits that improve model prediction while removing splits that might be capturing noise, pruning helps to reduce overfitting (James et al., 2013). Cross-validation techniques, such as V-fold cross-validation, are used in the pruning process to test model performance across separate data folds. This helps determine the generalization capabilities of the pruned trees. However, each time cross-validation is repeated with a different partitioning of the data into folds, a different final tree might result (Browne, 2000; Geurts et al., 2006; Isaksson et al., 2008). Therefore, the outcome of the pruning process can vary. This lack of consistency in the pruning

process can make it difficult to maintain a single stable tree, which can lead to unreliable model interpretations and can be confusing to users who do not understand this randomness.

The trees constructed during cross-validation arise from data subsets (cross-validation folds) rather than the entire dataset. Since the process of finding the optimal pruning value depends on cross-validation tree performance, the variability introduced by constructing trees from subsets, affects the pruning process (James et al., 2013). One important cause of this variability, is the nature of the predictor variables. Both continuous and categorical variables influence the tree-building process and, consequently, the pruning process. However, they do so in distinct ways. The variability caused by continuous variables arises from the requirement to find optimal split thresholds, where all potential split points for a variable are evaluated. This approach can introduce bias in variable selection, especially when the number of possible splits differs across predictors (Loh, 2002; Loh and Shih, 1999; Shih, 2004). Continuous variables, which have more potential split points, are more likely to be selected, even when these splits do not significantly improve the model. This bias can cause the tree structure to vary across folds due to small changes in the data, making it harder to identify consistent and important splits. Categorical predictors introduce different forms of variability, namely through biased split determination methods. Variables that contain many categories tend to be selected for splitting more frequently, since they offer more partitioning options (Loh, 2002; Loh and Shih, 1999; Shih, 2004). Splits can then be chosen without adding important information, as they may lower the splitting criterion without improving the predictive or explanatory power of the model.

Another cause of variability is predictor imbalance. If there are predictor variables containing values or categories that are underrepresented, trees can fail to identify consistent splits containing those values. This is because, if predictor imbalance is present, splitting criteria tend to favor majority values, resulting in biased structures and missed patterns associated with the minority values (Chaabane et al., 2020; Chawla, 2010; Cieslak and Chawla, 2008; Liu et al., 2010). Predictor imbalance can also affect cross-validation, as underrepresented categories might not appear in every fold. This can lead to instability in the pruning process. Furthermore, correlations and interactions between predictors also contribute to the variability. When predictors are highly correlated, the algorithm often selects only one of them for splitting, even if multiple are relevant (Doyle, 1973; Loh, 2014). This can lead to instability, as small changes in the data may shift the preference from one correlated variable to another. Furthermore, correlated variables can also introduce bias since decision trees tend to favor the correlated variables during the tree-building process (Strobl et al., 2008). Similarly, interaction effects may only be detected in certain subsets of the data, causing different splits to be chosen across folds. All these characteristics can introduce variability during the tree construction process and, therefore, influence the pruning process. This makes the final pruned trees less stable and more difficult to interpret.

The variability of pruning outcomes presents a challenge for users who depend on decision tree analyses. The developers of the `metacart` R package (Li et al., 2020b) noticed that users frequently encounter unexpected variations in model results from repeated analyses due to varia-

tion in the pruning process. This variability in the pruning results might reduce confidence in the reliability of the model and emphasizes the need for better understanding and communication of the pruning variability. As decision trees are used frequently for important tasks, it is important to ensure that the models are both consistent and reliable.

To address these challenges, this thesis focuses on the following research question: How do characteristics of the generated data, such as moderator type, effect size, and moderator correlation, and the pruning strictness, contribute to the pruning variability? To support this analysis, this thesis first examines how cross-validation introduces variability in pruned decision trees. Next, a simulation study is conducted using true underlying tree models with various characteristics. The goal is to investigate the causes of pruning variability and assess their combined effects on tree structure stability and interpretability. Additionally, this thesis will develop visualizations to communicate the pruning variability to users. These visualizations will show the range of potential pruned trees. Furthermore, they will emphasize areas where the tree might vary due to cross-validation. The goal of these visualizations is to improve model interpretation and, therefore, allow users to better understand their, possibly variable, results of the pruning process. To make them more practical, these methods will serve as an extension to existing decision tree R packages. This will make the results more reproducible and improve the acceptability of decision tree analyses.

Chapter 2

Decision Trees

2.1 Introduction

As one of the most intuitive machine learning methods, decision trees are often used for both regression tasks and classification tasks (Mienye and Jere, 2024). Decision trees are non-parametric models since they avoid making assumptions about the data distribution, unlike linear models that depend on predefined relationships between predictors and response variables. They can discover complex predictor relationships and non-linear patterns by themselves automatically without requiring explicit specification of these interactions (Breiman et al., 1984). Therefore, decision trees are an effective tool for exploratory data analysis.

Decision trees are built using recursive partitioning. Through a series of binary splits that focus on maximizing within-node homogeneity, the data iteratively gets divided into increasingly smaller and uniform regions. The internal nodes of the tree function as decision rules, which are based on predictor variable thresholds, whereas the terminal nodes, or leaves, denote the predicted results. In classification tasks, the predictions take the form of class labels. For regression tasks, they yield numerical predictions. The tree-like structure of the model provides high interpretability since it allows users to visualize splits as decision rules, which makes understanding predictions straightforward and clear. Hence, decision trees are both easy to interpret and very useful in practice. Additionally, decision trees can process both continuous and categorical predictor variables, and are robust to changes in data scale or monotonic transformations (Breiman et al., 1984; James et al., 2013; Loh, 2014). These characteristics make them an effective method for analyzing a wide range of datasets.

This chapter explores the work of Breiman et al. (1984) on decision trees, which presents an essential understanding of Classification and Regression Trees (CART), and how they are used in practice. This chapter examines the main ideas from their book together with more recent studies that have expanded their work. Additionally, the work of James et al. (2013) is used as the foundation for most of the formulas and techniques presented in this chapter, since it offers

a more accessible approach to understanding these concepts.

2.2 Building a Tree

Decision trees start with dividing the data into nodes, or regions, through recursive division that results in more homogeneous regions, with respect to the response variable. The concept of regions is important to how decision trees work, as they represent subdivisions of the predictor space, that are created to maximize the within-node homogeneity of the tree. Binary decision rules form the basis of the algorithm by consistently splitting the data into distinct data regions. The process of determining each split involves evaluating every potential way to divide the data. For numeric predictors, each unique value of the variable serves as a possible threshold to evaluate if the data split at that point results in more homogeneous subsets. For categorical predictors, the algorithm examines every possible combination of category groupings when determining splits. The aim is to split the predictor space into J unique regions that do not overlap: $R_1, R_2, \ldots, R_J$. Each observation in region R_j receives the same prediction, namely the average response value from training observations found within R_j for regression tasks, and the dominant category found within R_j for classification tasks. Although regions may take any form, decision trees usually break down the predictor space into high-dimensional rectangular or box-shaped regions to maintain simplicity and interpretability. Decision trees construct splits through different measures based on whether the tree is designed for regression or classification.

When building regression trees, the goal is to determine regions $(R_1, \ldots, R_J)$ that minimize the Residual Sum of Squares (RSS), defined as

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2, \tag{2.1}$$

where $\hat{y}_{R_j}$ represents the average response for region R_j, J denotes the total number of regions, and y_i indicates the response value for observation number i. The computational effort required to evaluate all possible thresholds to divide the feature space into J regions makes this approach computationally demanding. Therefore, decision trees use a top-down greedy method called recursive binary splitting. The tree-building procedure starts with all observations assigned to one region at the root node of the tree. The method then successively divides the predictor space into two regions. During each step of the recursive binary splitting process, the algorithm selects the optimal split for that specific step without considering future steps that could result in a better overall tree structure. The predictor X_j and cutpoint s that produce the regions with the lowest RSS, determine the split. The recursive binary splitting approach can be summarized as follows. First, we create two regions for every predictor X_j by using every possible cutpoint s:

$$R_1(j, s) = X \mid X_j < s, \quad R_2(j, s) = X \mid X_j \geq s. \tag{2.2}$$

Next, for every candidate split (j, s), we calculate the RSS for the two regions:

$$RSS(j,s) = \sum_{i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2, \tag{2.3}$$

where the mean responses of regions $R_1(j,s)$ and $R_2(j,s)$ are represented by $\hat{y}_{R_1}$ and $\hat{y}_{R_2}$, respectively. Next, we choose the predictor X_j and cutpoint s that produce the smallest $RSS(j,s)$, perform the split, and continue recursively.

Classification trees, like regression trees, use recursive partitioning but are designed to predict categorical outcomes rather than continuous values. In this case, each terminal node assigns the most frequent class label to all observations within it. Classification trees also develop through recursive binary splitting, dividing the predictor space into regions that become progressively more homogeneous, ideally each containing one very dominant class. However, classification trees use the classification error rate, Gini index, or entropy as metrics to determine split quality instead of the RSS. The classification error rate evaluates the proportion of data points in a node that are not from the most frequent class. It can be expressed as

$$\text{Error} = 1 - \max_k(\hat{p}_{mk}), \tag{2.4}$$

where the term $\hat{p}_{mk}$ represents the proportion of observations within node m that are classified as class k, and $\max_k(\hat{p}_{mk})$ calculates the highest proportion across all classes k to determine the proportion of the most frequent class in node m. The classification error represents the proportion of observations that receive incorrect classifications. It serves as a simple and straightforward metric but fails as a sensitive tree construction measure. This is because it focuses only on the most frequent class proportion in each node, while ignoring other class distributions. Therefore, for tree construction, it is common to use the Gini index or entropy instead (James et al., 2013). The Gini index measures how impure a node is by calculating total class variance. Lower values show that the nodes contain more homogeneous classes. It is defined as

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}), \tag{2.5}$$

where K represents the total number of classes, while $\hat{p}_{mk}$ measures the proportion of class k observations within node m. Nodes that contain mostly observations from a single class, show low Gini index values, making this metric valuable for maintaining node purity. Similarly, entropy functions as an impurity evaluation for nodes, where lower entropy values indicate higher node homogeneity. It is defined as

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk}), \tag{2.6}$$

where $\hat{p}_{mk}$ denotes the percentage of data points in node m, which fall under class k, and K stands for the total number of classes. The Gini index and entropy show greater sensitivity to changes in node purity than the classification error rate, making them ideal for assessing splits

when growing trees. However, all three metrics can be used during pruning, a later step where the fully grown tree is simplified to reduce overfitting and improve predictive performance, as we will discuss later in this chapter. In that context, the classification error rate is often preferred if the goal is to maximize the accuracy of the final pruned tree (James et al., 2013).

This process continues until the algorithm produces a fully grown tree. A fully grown tree is one in which all possible splits have been made, subject to the constraints of the data. Usually, the splitting process continues until a stopping criterion is met. The process will, for example, be constrained by setting a minimum number of observations per node, a maximum size (depth) for the tree, or a minimum reduction in impurity required for a split to be considered.

2.3 Example Regression Tree for Predicting Car Prices

Imagine we want to predict the price of a car, a continuous variable, based on various variables from the Automobile Data dataset, published in the April 1990 issue of Consumer Reports (Consumer Reports, 1990). The dataset includes variables such as country of origin, mileage, weight, engine displacement, and horsepower. Using a regression tree, we partition the data into distinct groups based on these variables, allowing us to predict car prices by identifying shared characteristics within each group. An example of such a tree is shown in Figure 2.1. Each terminal node represents the predicted price for cars within that group, while the percentages indicate the proportion of observations in each node relative to the total dataset. Below, we describe the process of constructing the tree.

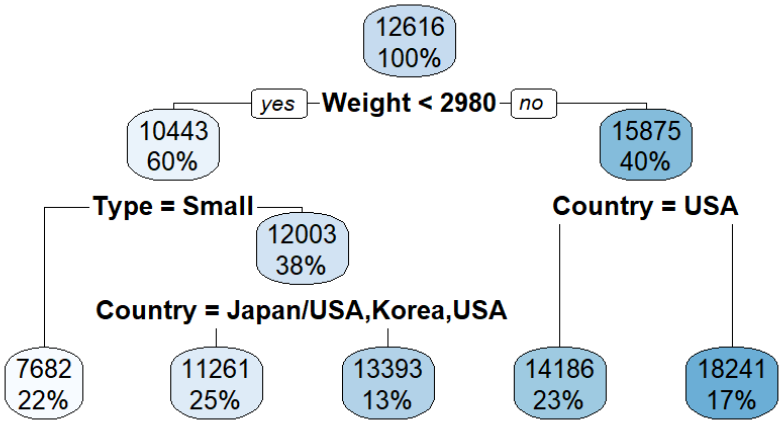
The regression tree algorithm begins with all observations in a single group, represented by the root node. The average variability in car prices is called the Mean Squared Error (MSE), which is the RSS divided by the number of observations within a node. Before making any splits, the mean price is \$12,615.67 and the MSE is 16,392,520. To reduce this variability, the algorithm identifies the optimal variable and threshold for splitting the data. The initial data split uses weight as the criterion, with the splitting threshold set at 2,980 pounds. Cars that are under 2,980 pounds move to one node, while those exceeding 2,980 pounds move to a different node. The data split by weight resulted in two regions with mean prices of \$10,442.58 and \$15,875.29. After splitting by weight, the MSE decreased to 7,880,181 for lighter cars and 11,452,450 for heavier cars. This reduction demonstrates the effectiveness of weight in explaining the variability in car prices.

For lighter cars (weight $<$ 2,980), the next split depends on car type. Small cars form a terminal node with a mean price of \$7682.39 and an MSE of 1,677,286 reflecting the low variability in prices within this group. The node that includes the other car types, now has an average price of \$12,002.70 with an MSE of 4,645,995 and gets divided further by their country of origin. Cars originating from Japan and the USA, Korea, and the USA have an average price of \$11,261.20, with an MSE of 1,418,199, whereas cars from the other countries have an average price of \$13,393.00, with an MSE of 7,734,262. Each split reduced the MSE in the resulting

regions, leading to subsets with increased homogeneity.

Figure 2.1

Example of a Regression Tree for Predicting Car Prices



Note. Terminal nodes show the average predicted price, with percentages indicating the proportion of observations in each node.

For heavier cars (weight $\geq$ 2,980), the next split depends on country of origin. Cars from the USA form a terminal node with a mean price of \$14,185.71 and an MSE of 3,417,360. Cars from the other countries form another terminal node with a mean price of \$18,241 and an MSE of 13,109,880. These splits further reduce the overall variability in car prices by grouping cars with similar characteristics.

No further splits are performed since the stopping criteria have been met. Using the `rpart` R package default values for the stopping criteria, splits are, for instance, only performed if a node contains at least 20 observations (*minsplit*), and terminal nodes must have a minimum of 7 observations (*minbucket*). Additionally, the complexity parameter (*cp*) is set to 0.01 by default, requiring each split to decrease the overall lack of fit by a certain amount before being allowed (Therneau et al., 2023). Concluding, to predict the price of a new car, the variables weight, type, and country of origin, are used to navigate through the tree to a terminal node. The resulting regression tree (Figure 2.1) provides a clear and interpretable approach to understanding how these factors influence car prices. Additionally, an example of a classification tree can be found in Appendix A.

2.4 Pruning in Decision Trees

2.4.1 Introduction to Pruning

A primary issue of decision trees, is their tendency to overfit the training data. When grown to their full depth, trees capture both the underlying patterns and the noise in the data, resulting in models that perform poorly on unseen data. This makes tree generalizability an important consideration. To deal with this issue, techniques like pruning and ensemble methods (bagging and boosting) can be applied. The latter will not be explained further since they lie outside the scope of this thesis. Pruning techniques help solve this problem by shrinking the tree structure, enhancing the generalization capability of the model. Pruning works by identifying and removing splits and nodes that add minimal predictive power to the tree, resulting in a simpler model. Two main approaches exist for pruning decision trees: pre-pruning which is known as early stopping and post-pruning referred to as cost-complexity pruning (Breiman et al., 1984; Fürnkranz, 1997; James et al., 2013).

Pre-pruning stops the tree growth before the model becomes too complex. The algorithm may stop splitting a node if it contains too few observations or if further splits do not significantly improve the performance of the model (based on some measure of improvement). Pre-pruning reduces the risk of overfitting but comes with the downside that it might miss important features of the data by stopping too early (Fürnkranz, 1997). In contrast, post-pruning or cost-complexity pruning creates an initial full tree that captures all data splits before removing splits that do not substantially decrease prediction error (Breiman et al., 1984). Achieving the optimal balance between model complexity and accuracy matters because complex trees can show high performance on training data but deliver poor results on test data. Since this thesis specifically aims to investigate the variability arising from post-pruning, only post-pruning will be discussed further.

2.4.2 Post-pruning Process

A common pruning strategy is cost-complexity pruning, also known as weakest-link pruning. This strategy involves first growing an initial tree, T, that fully captures the data. Next, subtrees are generated by iteratively removing splits that contribute the least to reducing the RSS or another error metric. This sequence of subtrees is indexed by a non-negative complexity parameter (cp), α, which controls the trade-off between tree complexity and fit to the data. Additionally, cross-validation is used to select the optimal complexity parameter α. The cost-complexity criterion used to evaluate subtrees is expressed as

$$R_\alpha(T) = R(T) + \alpha|T|, \tag{2.7}$$

where the penalized error $R_\alpha(T)$ results from combining tree error $R(T)$ with a size penalty depending on the number of terminal nodes $|T|$ in tree T, and is regulated by α. A larger α favors simpler trees by penalizing models with many terminal nodes. For example, regression

trees utilize the RSS in order to quantify the error metric, the cost-complexity criterion is then expressed as

$$R_\alpha(T) = \sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|, \tag{2.8}$$

where the predicted response for R_m, denoted as $\hat{y}_{R_m}$, is the average of the training observations within that region.

In order to prevent having to analyze every value of α, we can use geometric means. There are exact values of α that correspond to a point at which the tree structure changes, named critical values (Therneau and Atkinson, 2023). For example, we can have a tree that maintains three splits if we provide an α value equal to or less than cp_1 but greater than cp_2. In order to get a value of α that represents this interval, we can calculate the geometric means. The geometric means, derived from the critical α values, allow us to represent intervals without needing to manage every α individually. Therefore, using these geometric means makes the pruning process more computationally efficient. The geometric means of the intervals can be calculated using the following formula:

$$\text{Geometric Mean} = \sqrt{cp_1 \cdot cp_2}. \tag{2.9}$$

These computed geometric means are the values of α that can now be evaluated using cross-validation (James et al., 2013; Therneau and Atkinson, 2023).

To select the optimal subtree, the performance of the model is evaluated using cross-validation. In V-fold cross-validation, the data is split into V equal parts or folds. The model is first trained on $V - 1$ folds and then tested on the remaining fold, repeating the process V times (Browne, 2000). The average cross-validation error across all folds is used to measure how well the tree generalizes to new data, it is defined as

$$\text{Cross-Validation Error} = \frac{1}{V} \sum_{i=1}^{V} \text{Error}_i, \tag{2.10}$$

where Error_i is the test error for the i-th fold (Browne, 2000). The α that produces the lowest cross-validated error can be chosen as the value for the final model. Alternatively, the one-standard-error or the $c \cdot SE$ rule may be applied, where the value of α that produced the simplest tree whose error is within one or c standard errors of the minimum, is selected (Breiman et al., 1984; Dusseldorp et al., 2010). The entire pruning process can be seen in Algorithm 1.

Now, if we go back to the regression tree example that was visualized in Figure 2.1, we can perform post-pruning on this tree. First, various values of the complexity parameter are evaluated using cross-validation. We can see the results of this cross-validation process visualized in Figure 2.2.

Algorithm 1 Construction of a Decision Tree with Pruning

1. Begin by growing an initial tree on the training dataset through recursive binary splitting. Continue this process until a predefined stopping criterion is met.

2. Perform cost-complexity pruning on the fully grown tree to create a sequence of smaller subtrees, each corresponding to a different value of α.

3. Determine the optimal α using V-fold cross-validation:

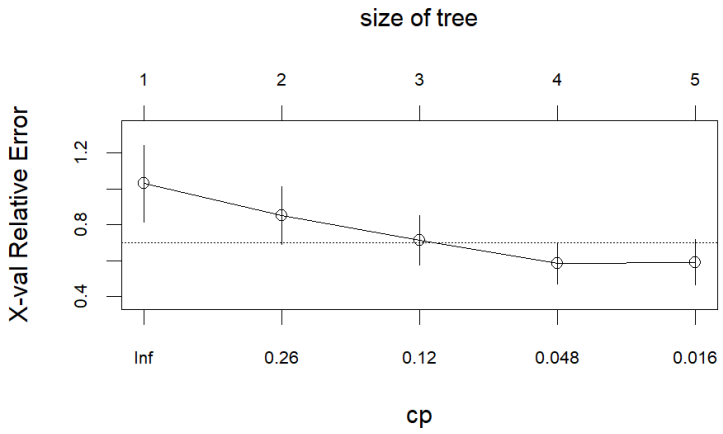
(a) Split the training data into V equal parts (folds). For each fold, apply the steps of growing a full tree and pruning it (Steps 1 and 2) to the remaining $V-1$ folds, leaving the current fold out for testing. Compute the mean squared prediction error for the left-out fold for each subtree and value of α.

(b) Calculate the average error across all V folds for each α. Select the α that results in the lowest average error. Alternatively, the $c \cdot SE$ rule can be used. This rule selects the highest α value where the cross-validated error remains within c standard errors from the minimum cross-validated error (often chosen as 1, referred to as the one-standard-error rule).

4. Choose the subtree corresponding to the selected α.

Figure 2.2

Relationship Between Complexity Parameter α (cp) and Relative Cross-Validation Error

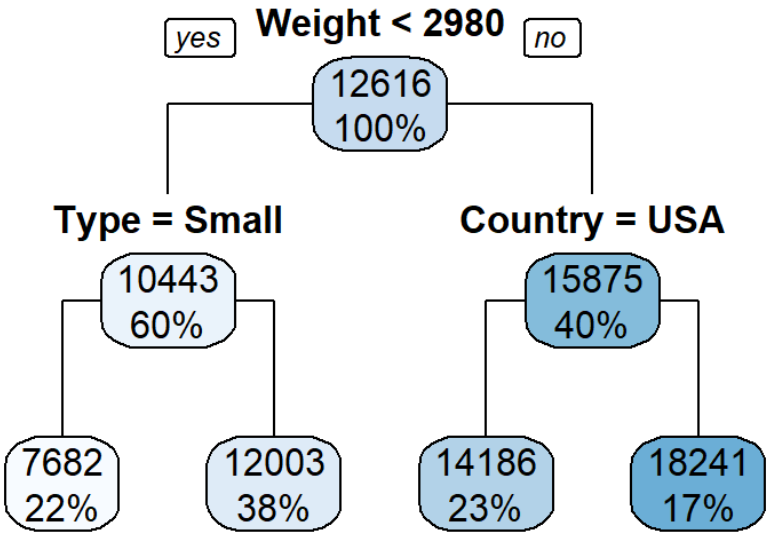


Note. The top x-axis shows tree size (number of terminal nodes), while the bottom x-axis represents α values. The y-axis displays the relative cross-validation error (`xerror`), which measures the ratio of the cross-validated error of a subtree to the cross-validated error of the unpruned tree. The dashed line marks the one-standard-error rule, suggesting the simplest tree within one standard error of the minimum error.

The α value of 0.048 yielded the lowest cross-validation error with the one-standard-error rule. The pruned tree is visualized in Figure 2.3. As we can see, there are fewer terminal nodes now.

Figure 2.3

Example of a Pruned Regression Tree for Predicting Car Prices



Note. Terminal nodes show the average predicted price, with percentages indicating the proportion of observations in each node.

Concluding, cost-complexity pruning involves choosing the optimal tree from the resulting sequence of subtrees that emerges through repeated pruning steps from the fully grown tree. It is important to note that this process maintains the tree structure, ensuring that the sequence keeps both split order and threshold values unchanged. The pruning procedure only selects nodes and splits from the fully grown tree to either keep or remove in order to minimize overfitting and enhance model generalization.

2.5 Decision Trees in R

The `rpart` R package (R Core Team, 2024; Therneau and Atkinson, 2023; Therneau et al., 2023) implements the CART algorithm for constructing decision trees within R. The primary function `rpart()` requires users to provide both a formula and a dataset, while the algorithm internally determines suitable splitting criteria according to the response variable type. The default measure for node purity in classification tasks within `rpart` is the Gini index, but users can select

entropy as an alternative option. The best splits for regression tasks, are determined by evaluating the Residual Sum of Squares (RSS). Furthermore, users can modify the tree-building process with different parameter settings. Different pre-pruning settings can be modified through the `control` argument, which includes parameters such as minimum split observations (`minsplit`), minimum terminal node observations (`minbucket`), the complexity parameter (`cp`), and maximum tree depth (`maxdepth`). Next, users can examine the `rpart` results using the `summary()` function. The `summary()` function provides comprehensive details about the constructed tree. It includes split criteria information, node counts, descriptions of the tree structure, and terminal node predictions.

The `rpart` package also supports post-pruning. The `rpart()` function automatically implements cross-validation to assess multiple pruned trees at various values of α. The package uses the geometric means method to select the α values that it will evaluate using cross-validation. By using the geometric means, all significant pruning options will be assessed while still achieving computational efficiency. Additionally, the parameter `xval` (default value 10) can be adjusted, allowing users to define the number of folds for cross-validation. The cross-validation results can be found in the `cptable`. The `cptable` of the model displays important parameters, including the complexity parameter (`cp`), number of splits (`nsplit`), relative error of the tree (`rel error`), which divides the error of a subtree by the error of the most simple tree (without splits), cross-validated error (`xerror`), and the standard deviation of the cross-validated error (`xstd`). These results can also be visualized using the `plotcp()` function, as shown in Figure 2.2. Next, the `prune()` function allows users to prune the tree by setting a value for the *cp* parameter that specifies how much pruning should occur.

The visualization tools within the `rpart` package are important for interpreting decision trees. The `plot()` function generates a basic tree diagram that can be improved through the `text()` function by labeling splits, adding predictions, and adding sample sizes. These visualizations often require manual modification to improve their interpretability. The `rpart.plot` package delivers advanced visualization features that offer improvements beyond the basic tree visualizations created by `rpart`. This package generates clear visual layouts with distinct nodes that display decision rules and outcome predictions together with sample sizes. The `rpart.plot` package builds on the `ggplot2` R package (Wickham, 2016), and lets users customize their decision tree visualizations with options such as changing the colors, text size, and labels. Examples of figures created with this package are Figure 2.1 and Figure A.1. It is important to note that, even though `rpart()` automatically uses cross-validation to find the best α value for pruning, the `plot()` function will show the fully grown tree if the tree is not manually pruned using the `prune()` function.

In conclusion, the `rpart` package is an important R package for creating decision tree models in R. It can automatically grow trees, apply pruning with cross-validation, and create visualizations, making it a useful choice for those working with CART models. More information and details about this package can be found in the `rpart` package manual (Therneau et al., 2023).

Chapter 3

Trees in Meta-Analysis

3.1 Meta-Analysis

Meta-analysis is an important statistical approach for combining results from different independent studies. By combining the results, they generate more reliable and generalizable answers to research questions (Israel and Richter, 2011). Meta-analysis is often used for healthcare, psychology, and social science research. This is because individual studies in these fields often produce contradictory results or face limitations like small sample sizes and low statistical power (Hedges and Olkin, 1985). Meta-analysis aims to improve statistical precision, while generating stronger overall effect size estimates by combining data. It determines a combined effect size. This effect size represents results from numerous studies, and accounts for the variations found between their effect sizes (Borenstein et al., 2009; Borenstein et al., 2010; Dusseldorp et al., 2014). The observed variability in effect sizes between studies, is referred to as heterogeneity. Heterogeneity can arise from variations in participant characteristics or study design aspects, such as intervention techniques or outcome measurements. Analyzing the heterogeneity prevents oversimplified conclusions that can lead to misinterpretations and incorrect decisions. Hence, for meta-analytic results to be reliable, researchers must understand and address heterogeneity in their analyses.

Fixed-effect and random-effects models are traditional meta-analytic approaches, helping researchers manage heterogeneity in their analyses. Fixed-effect models operate on the assumption that every study included measures one true effect size, with differences arising only from sampling error (Dettori et al., 2022). Fixed-effect models work well when studies share similar designs and aims. Random-effects models accommodate different true effect sizes across studies, while incorporating both within-study variance and between-study variance to account for heterogeneity (Dettori et al., 2022). Random-effects models demonstrate better suitability for datasets with significant differences in study designs, populations, or contexts (Borenstein et al., 2009; Borenstein et al., 2010; Dettori et al., 2022). Both the fixed-effects and random-effects models

remain useful, but they also contain limitations. The models can measure heterogeneity, but face difficulties in explaining it. This happens especially when relationships between study characteristics and effect sizes are complex, non-linear or include moderator interactions (Li et al., 2020a; Tipton et al., 2019). This can become problematic when the number of studies is small, as meta-regression, a common method to explore moderators, would then suffer from low statistical power to test all variables simultaneously (Li et al., 2020a). Additionally, meta-regression requires that all main effects and interaction terms are specified, which becomes impractical when no clear hypotheses exist and the number of possible interactions is large (Li et al., 2020a). These limitations of traditional approaches stimulates researchers to study more flexible and practical methods, including decision tree-based techniques, to enhance their understanding and modeling of heterogeneity.

This chapter will discuss how decision tree methods can enhance traditional meta-analysis techniques, with a focus on the meta-CART approach introduced by Dusseldorp et al. (2014). Decision trees provide improved modeling capabilities for complex data, because they can reveal non-linear relationships and study characteristic interactions, while providing more flexibility than traditional models. The following sections explore the application of decision tree methods to meta-analysis. Additionally, this chapter demonstrates how the `metacart` R package offers a practical approach for applying these methods. The work of Li et al. (2020a) serves as the foundation for the concepts and formulas discussed in this chapter. This is because it offers an accessible and comprehensive understanding of the application of decision tree methods in meta-analysis.

3.2 Meta-CART

Standard CART methods examine individual-level data, whereas meta-CART (Meta-analytic Classification and Regression Trees) applies CART techniques to aggregated data from multiple studies included in a meta-analysis. The effect size functions as the response variable, while study-level characteristics (moderators) operate as predictive variables. The algorithm recursively partitions the dataset to find moderators that explain the variability in effect sizes. It does not work towards individual-level predictions but finds study subgroups displaying similar effect sizes, helping understand the impact of the moderators on study variation (Dusseldorp et al., 2014; Li et al., 2020a). The method uses a recursive partitioning algorithm to iteratively split studies into increasingly homogeneous subgroups. At each step, the algorithm selects the moderator and corresponding threshold that maximizes the difference between subgroups, as measured by a heterogeneity metric such as the Q-statistic.

The process begins like traditional CART with the entire dataset as a single parent node, representing all studies. The algorithm then evaluates potential splits for all available moderators, identifying the split that maximizes between-group heterogeneity while minimizing within-group variability. This split divides the data into two child nodes, which are then subjected to the

same procedure. The algorithm continues this process until stopping criteria are met, such as a minimum number of studies per terminal node. The resulting tree structure provides a visual tree of the relationships between moderators and effect sizes, with terminal nodes representing distinct subgroups of studies and internal nodes showing the moderators and thresholds that define these subgroups. Importantly, meta-CART also includes mechanisms to deal with overfitting, such as tree pruning with cross-validation, which will be explained later.

3.2.1 Random-Effects Meta-CART

Unlike Fixed-Effects (FE) meta-CART, which assumes all variability in effect sizes can be attributed to moderators and within-study sampling error, Random Effects (RE) meta-CART accounts for additional variability between studies that cannot only be explained by observed moderators or sampling error. In this thesis, the focus is only on RE meta-CART since it is more flexible and widely used in research (Li et al., 2025). The random-effects model is represented as

$$d_k = \beta_0 + \beta_1 x_{1k} + \beta_2 x_{2k} + \cdots + \beta_M x_{Mk} + \tau_k + \epsilon_k, \tag{3.1}$$

where τ_k denotes the variation introduced by residual heterogeneity, distributed as $\mathcal{N}(0, \sigma_\tau^2)$. This term captures the difference between the true effect size of the kth study and the population mean from which all study effect sizes are sampled. The term ϵ_k accounts for the sampling error of the observed effect size d_k, distributed as $\mathcal{N}(0, \sigma_{\epsilon_k}^2)$. Thus, total variability in effect sizes arises from two components, namely residual heterogeneity and sampling error. The variance components, $\sigma_{\epsilon_k}^2$ and σ_τ^2, are estimated hierarchically, which means that the sampling variance is estimated first, followed by the between-study variance (Erez et al., 1996).

The within-study variance, $\sigma_{\epsilon_k}^2$, can be estimated using formulas specific to the study design, such as those for Hedges' g:

$$\hat{\sigma}_{\epsilon_k}^2 = \frac{n_k^T + n_k^C}{n_k^T n_k^C} + \frac{d_k^2}{2(n_k^T + n_k^C)}, \tag{3.2}$$

where n_k^T and n_k^C are the sample sizes of the treatment and control groups, respectively. The between-study variance, σ_τ^2, is estimated using the DerSimonian–Laird method, selected for computational efficiency in the `metacart` R package (Li et al., 2020b). Using these variance estimates, the summary effect size under the RE model is computed as a weighted mean, with weights given by $w_k^* = 1/(\sigma_{\epsilon_k}^2 + \sigma_\tau^2)$, and * denoting the random-effects statistics:

$$d_+^* = \frac{\sum_k d_k/(\sigma_{\epsilon_k}^2 + \sigma_\tau^2)}{\sum_k 1/(\sigma_{\epsilon_k}^2 + \sigma_\tau^2)}. \tag{3.3}$$

To quantify heterogeneity, the model employs the Q^* statistic, defined as

$$Q^* = \sum_{k=1}^{K} \frac{(d_k - d_+^*)^2}{\sigma_{\epsilon_k}^2 + \sigma_\tau^2}. \tag{3.4}$$

The tree-growing process in RE meta-CART is similar to traditional decision tree algorithms but requires additional steps to account for residual heterogeneity. At each split, the algorithm estimates σ_τ^2, as the heterogeneity between subgroups is influenced by the residual variance. The between-subgroups heterogeneity, Q_B^*, is defined as

$$Q_B^* = \sum_{t}^{|T|} \sum_{k \in t} \frac{(d_{t+}^* - d_{++}^*)^2}{\sigma_{\epsilon_k}^2 + \sigma_\tau^2}, \tag{3.5}$$

where $|T|$ represents the total number of subgroups, d_{t+}^* is the summary effect size within subgroup t, and d_{++}^* is the overall mean. Unlike standard CART, which uses traditional recursive partitioning where each split is made independently, RE meta-CART affects the overall estimation of σ_τ^2 and the calculation of Q_B^* with every new split. This means the algorithm must evaluate all terminal nodes at once, rather than considering them separately. As a result, the process follows a sequential approach, where each new split depends on the ones that came before it.

3.3 Pruning in RE Meta-CART

The RE meta-CART pruning technique differs from standard decision tree pruning, as each split affects the residual heterogeneity variance (σ_τ^2).

Algorithm 2 Construction of a RE Meta-CART Tree with Pruning

1. Use sequential binary splitting on the training data to build a full initial RE meta-CART tree, where after each split the residual heterogeneity (σ_τ^2), is re-estimated. Until the predefined stopping threshold is reached, keep dividing the training data.

2. Start with the complete meta-CART tree size and obtain a sequence of tree sizes by lowering the amount of terminal nodes one by one. Note that at this point, no trees are grown yet.

3. Use V-fold cross-validation in order to find the optimal tree size:

(a) Make V equal folds out of the training data. Using the preset tree sizes from Step 2, fit an RE meta-CART model to $V - 1$ folds for each fold, leaving the current fold out for testing. For every subtree, calculate the mean squared prediction error using the left-out fold.

(b) For every tree size, calculate the average error over all V folds. The tree size with the lowest average error within c standard errors of the overall minimum error should be chosen.

4. Using the training data, fit an RE meta-CART tree using the optimal tree size.

Standard CART pruning removes splits sequentially with the use of cost-complexity pruning, and removing these splits, does not affect previous computations. However, RE meta-CART

pruning calls for a different approach, as eliminating a split will affect the residual heterogeneity variance, σ^2_τ. So, since the cost-complexity pruning method cannot be used in this context, cross-validation is used as the primary approach to find the optimal tree size. The RE meta-CART pruning process evaluates trees of various sizes and identifies the tree that achieves the best combination of model simplicity and prediction accuracy. The selection process prevents overfitting and maintains model stability by choosing the simplest tree within c standard errors of the minimum cross-validation error. The full pruning process for RE meta-CART trees is shown in Algorithm 2.

To demonstrate this process, imagine an initial RE meta-CART tree is developed, producing a tree with seven terminal nodes. The pruning process then creates a sequence of tree sizes with progressively fewer terminal nodes (6, 5, 4, etc.), each indicating a simpler model. These various tree sizes are then used to grow decision trees, which additionally, get evaluated using cross-validation and a mean squared error predictive performance assessment.

In summary, unlike standard CART pruning, which removes tree splits based on complexity penalties, RE meta-CART pruning ensures that the final tree still takes heterogeneity adjustments into account at every split by only using cross-validation.

3.4 Meta-CART in R

The R package `metacart` features implementation of the meta-CART algorithm, created for decision tree meta-analysis methods (Li et al., 2020b; R Core Team, 2024). The `metacart` package includes two main functions called `FEmrt()` and `REmrt()`, which build meta-CART models representing fixed-effects and random-effects models, respectively. The functions require effect sizes, moderator variables, and the within-study variance as inputs to generate a decision tree model. For instance, a call to `REmrt()` might look like:

```
tree <- REmrt(formula = yi ~ m1 + m2 + m3, vi = vark, data = meta_data)
```

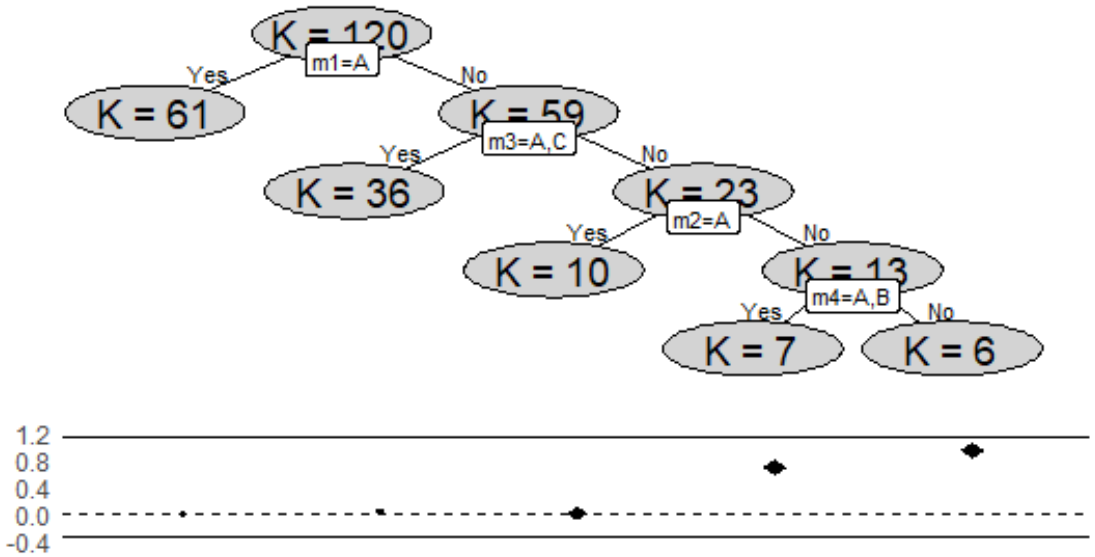
Here, `yi` represents the response variable (effect size), the right-hand side of the formula (`m1 + m2 + m3`) specifies the moderators to be evaluated, and `vark` is the corresponding variance for each effect size. The resulting object, `tree`, contains the fitted meta-CART model, including information about the splits, subgroup characteristics, and effect size estimates for each subgroup. Heterogeneity gets evaluated using the Q-statistic metric. The tree-building process allows for customization through parameters such as the maximum number of splits (`maxL`), the minimum number of studies required in a parent node before a split is attempted (`minsplit`), the minimum number of studies allowed in a terminal node (`minbucket`), and the threshold for reduction in between-subgroup heterogeneity (`cp`) needed for a split to be made. Additional options include the number of folds for cross-validation (`xval`) and the pruning strictness value (`c.pruning`), which determines how aggressively the tree is pruned using the $c \cdot SE$ rule.

Cross-validation results can be accessed through the `cptable` component of the tree object, offering a better understanding of the stability of the model. The `tree$cptable` provides statistics on how the tree performs across different cross-validation folds. Unlike the `rpart` package, where pruning must be carried out manually after fitting the initial tree, the `metacart` package integrates pruning with cross-validation directly into the tree-building process. As a result, the final tree returned by the fitting function is already pruned to the optimal size based on cross-validation results. This can lead to confusion, as pruning results may vary depending on the seed used, causing different pruned trees to be produced across different runs.

Visualization is an important feature of the `metacart` package. The `plot` function provides a clear representation of the decision tree. Nodes display the number of studies they contain (K), with decision rules shown below the internal nodes. This produces a tree structure that emphasizes key moderators influencing heterogeneity and reveals subgroups of studies with similar characteristics. Below the tree, the predicted response values (effect sizes) for each terminal node are shown. Figure 3.1 demonstrates an example of a decision tree generated using `metacart` with a simulated dataset available in the package as well.

Figure 3.1

Example of a Meta-CART Decision Tree Made with Package metacart



Note. Each node displays the number of studies (K) in that subset, with decision rules shown below the internal nodes.

In this example, the root node represents the full dataset of $K = 120$ studies. The first split occurs on the categorical moderator m1, dividing the data into $K = 61$ studies where m1 = A, and $K = 59$ studies where m1 is not A. Additional splits based on moderators m2, m3,

and m4 further refine the subgroups, with the final nodes representing groups of studies with homogeneous effect sizes.

It is also possible to access the full unpruned generated tree by calling the `initial.tree` object. This object contains the decision tree before any pruning has occurred, allowing users to examine the structure and splits that would have been made without the pruning process. The unpruned tree can also be visualized using the `plot()` function, similar to the pruned tree. Additionally, the `summary()` function can again be used to retrieve detailed information about the full tree, including the number of splits, the conditions defining each split, and the characteristics of each subgroup. In summary, the `metacart` package is an important R package for applying decision tree methods to meta-analytic data, helping identifying moderators that explain heterogeneity. A more detailed overview of the functions can be read in the `metacart` package manual (Li et al., 2020b).

Within the scope of this thesis, it is important to understand that the automatic post-pruning feature of the `metacart` package, makes the differences in pruned trees between runs more noticable. Since the `rpart` package requires users to perform post-pruning manually, the variability in pruning results with `rpart` becomes less unexpected. In contrast, the automatic pruning process in the `metacart` package can lead to user confusion due to variability in results. Therefore, examining and communicating the variability present within meta-CART is important.

Chapter 4

Methods

This chapter describes the methods used in this thesis. The main goal of this thesis was to investigate pruning variability in decision trees caused by cross-validation. To achieve this goal, we examined how various characteristics and levels of pruning strictness influenced the stability of the tree structures. Various meta-analytic datasets were simulated, and pruning variability in random-effects meta-CART models, built on these datasets, was evaluated. These datasets were generated by varying the moderator type, effect sizes, correlation structures, and the true underlying tree models. The data simulation approach followed the methodology used by Li et al. (2025). First, an overview of the data simulation process is given, including the process of how the moderators and observed effect sizes were generated. Next, a description of the underlying tree model specifications is provided, where variations in effect sizes across moderators were introduced, creating different levels of true tree model complexity. Finally, the chapter provides a description of how the pruning variability was assessed in the simulated datasets. This includes examining the effects of pruning on the number of terminal nodes and the entropy of these nodes. Additionally, ANOVA tests were used to evaluate how moderator type, effect size strength, moderator correlation, and pruning strictness contributed to the pruning variability. These tests helped to identify which factors most strongly influenced pruning outcomes and how they interacted. All R code used for the simulations and analyses can be found in Appendix B.

4.1 Data Simulation

4.1.1 Study and Sample Size

This study simulated various datasets in order to reflect different meta-analysis scenarios. All the simulated datasets consisted of 120 studies in order to maintain a balance between sufficient tree growth and the restricted availability of studies in meta-analyses. The sample size for each study k, denoted as n_k, was simulated using the methodology described by Viechtbauer (2007). Specifically, the sample sizes followed a normal distribution with a mean of 80 ($\overline{n}$) and a standard

deviation of $\overline{n}/3$ to introduce realistic variability in the study sample sizes.

4.1.2 Moderator Simulation

Each study included ten moderators. The values of these moderators were sampled from a uniform distribution between 0 and 1. This number was chosen to maintain a balance between model complexity and interpretability, allowing for various tree structures while avoiding excessive noise (Li et al., 2025). Only two to three of these moderators were actually used to construct the tree models. The remaining moderators served as noise variables to reflect real-world conditions where multiple potential moderators are available, but only a few are truly relevant. Additionally, a correlation structure was applied to introduce correlation among moderators, with correlations set at 0.3, 0.5, or 0.7. These values were selected to evaluate the impact of varying correlation levels on the pruning variability. A correlation of 0.3 showed a weak correlation, whereas 0.5 denoted a moderate correlation, and 0.7 showed a strong correlation among moderators. Correlated moderator values were generated by simulating multivariate normal vectors using the `mvtnorm` R package (Genz & Bretz, 2009). To convert these values into uniform distributions while preserving their correlation structure, the probability integral transform was applied, as described in Gilli et al. (2011).

In addition to continuous moderators, we investigated the effects of categorical moderators on pruning behavior by applying two discretization methods to the first five moderators (M1 to M5). Under the balanced categorization method, moderator values were split into three categories, where values below 0.3333 were assigned category A, values between 0.3333 and 0.6666 were assigned category B, and values above 0.6666 were assigned category C. In contrast, the unbalanced categorization method assigned values below 0.2222 to category A, values between 0.2222 and 0.8888 to category B, and values above 0.8888 to category C. The second approach resulted in a dominant middle category and two less frequent categories, allowing for an assessment of whether pruning behaves differently when categorical moderators exhibit an imbalanced distribution. The details of these categorizations are presented in Table 4.1.

Table 4.1

Comparison of Balanced and Unbalanced Categorical Moderators

Category	Balanced Categorization	Unbalanced Categorization
A	$x < 0.3333$	$x < 0.2222$
B	$0.3333 \leq x \leq 0.6666$	$0.2222 \leq x \leq 0.8888$
C	$x > 0.6666$	$x > 0.8888$

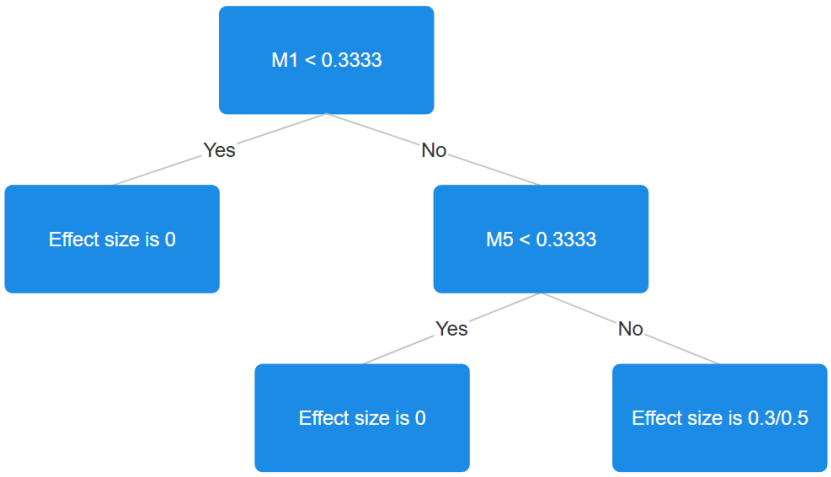
4.1.3 Model Specification

Two different tree models were designed to examine how variations in moderator effects influence the resulting tree structures. These models represented different levels of complexity, allowing for a comparison of pruning behavior under simpler and more complex decision rules. We first discuss the tree models for the datasets that only contained continuous moderators.

The first tree model followed a straightforward structure, where the true effect size (δ) was determined by only two moderators. The effect size was assigned when both the first and fifth moderators (M1 and M5) exceeded 0.3333. This model allowed us to investigate the pruning variability in a model with less interactions and easier splits. The characteristics of this tree model are visualized in Figure 4.1.

Figure 4.1

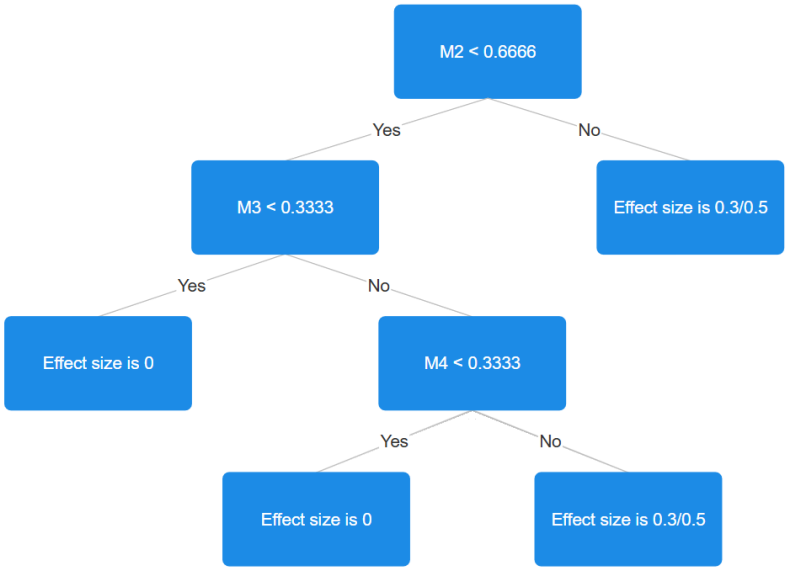
Tree Model 1 for Continuous Moderators



In contrast, the second tree model introduced additional complexity by including more conditions and interactions between moderators. For this model, the effect size was assigned either when the second moderator (M2) was equal to or larger than 0.6666 or when a combination of the second, third, and fourth moderators (M2, M3, and M4) met the following threshold criteria, the second moderator was smaller than 0.6666, the third moderator was larger than 0.3333, and the fourth moderator was larger than 0.3333. This structure increased the depth and complexity of the tree, allowing for an assessment of the stability under more complex conditions that could cause the pruning process to be less stable. The characteristics of this tree model are visualized in Figure 4.2.

Figure 4.2

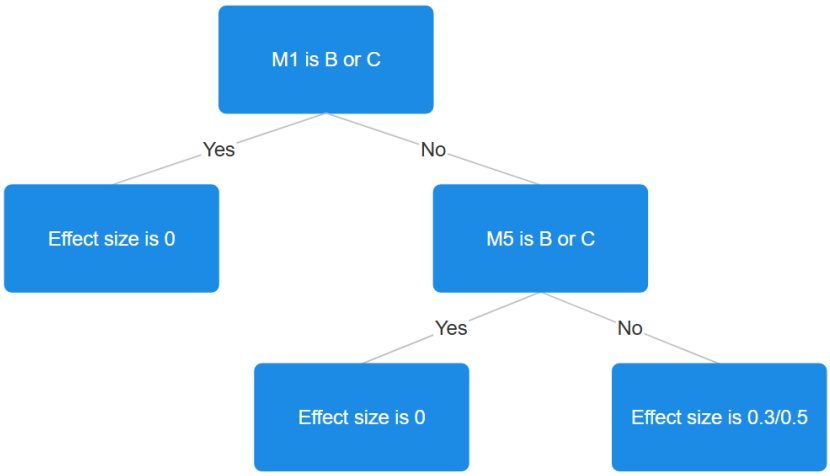
Tree Model 2 for Continuous Moderators



A similar structure was applied when defining effect size structures based on categorical moderators for the categorical datasets. The first tree structure assigned an effect size when the first and fifth moderators (M1 and M5) belonged to category B or C, mirroring the structure of the continuous tree where these moderators had to exceed a given threshold to influence effect size determination. The characteristics of this tree model are visualized in Figure 4.3.

Figure 4.3

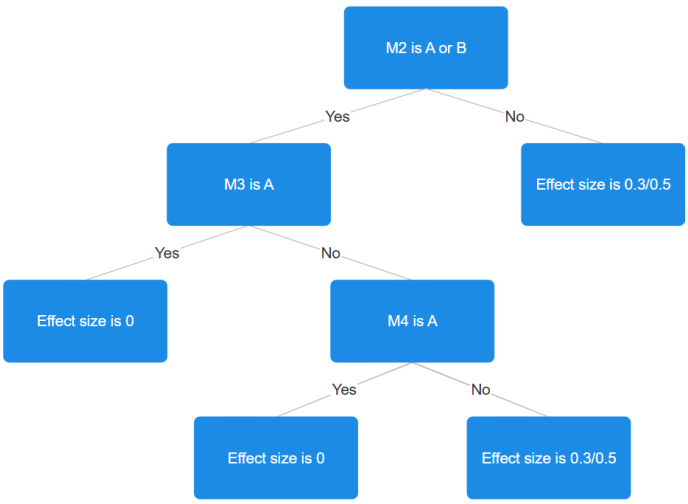
Tree Model 1 for Categorical Moderators



The second tree structure again followed a more complex pattern, where an effect size was assigned when the second moderator (M2) was in category C or when a combination of the second, third, and fourth moderators (M2, M3, and M4) met the required classification criteria, in which the second moderator belonged to category A or B, the third moderator belonged to category B or C, and the fourth moderator belonged to category B or C. This categorization ensured that the decision rules, guiding the effect size assignment, maintained a structure that corresponded to the structure established for the continuous moderators. The specifications of this tree model are visualized in Figure 4.4.

Figure 4.4

Tree Model 2 for Categorical Moderators



4.1.4 Effect Sizes

The true effect sizes (δ) for each of the 120 studies were chosen to be 0.3 or 0.5, depending on the condition. These values indicated small to moderate effect sizes, often reported in meta-analyses, and represented reasonable effect size magnitudes. The choice of these effect size values was inspired by previous meta-analytic studies which found this range to be small to moderate effects generally encountered in empirical studies (Valentine et al., 2015). The goal was to see whether pruning variability changed with the strength of the true underlying effect.

After the assignment of true effect sizes, the datasets were completed by simulating observed effect sizes with the same method that was used by Li et al. (2025) for their simulation. For each study k, an observed effect size g_k was sampled from a normal distribution with mean δ (either 0.3 or 0.5) and variance σ_τ^2 (the between-study heterogeneity, discussed in the next section). The observed effect sizes were then drawn from a non-central t-distribution, and scaled by a small-sample correction factor and sample size:

$$c(m_k)^{-1}(\tilde{n}_k)^{1/2} g_k \sim t_{m_k}(\delta_k \sqrt{\tilde{n}_k}), \tag{4.1}$$

where the value of $\tilde{n}_k$ was computed as

$$\tilde{n}_k = \frac{n_k^2}{n_k + n_k}. \tag{4.2}$$

The correction factor $c(m_k)$ was defined as

$$c(m_k) = 1 - \frac{3}{8n_k - 9}, \tag{4.3}$$

with degrees of freedom $m_k = 2n_k - 2$. This setup reflected the sampling distribution of observed effect sizes while accounting for small-sample bias and the influence of true effects and sample size. The corresponding sampling variance was calculated as

$$\text{Var}(g_k) = c(m_k)^2 \cdot \frac{2n_k - 2}{(2n_k - 4)(n_k/2)} \left(1 + \frac{n_k \delta_k^2}{2}\right) - \delta_k^2. \tag{4.4}$$

4.1.5 Heterogeneity

Set at 0.025, the between-study variance (σ_τ^2) functioned as the variance of the normal distribution from which the real effect sizes were derived and helped to account for the between-study heterogeneity. This value was selected because it is in line with generally reported modest degrees of between-study heterogeneity in meta-analyses (Borenstein et al., 2009; Li et al., 2025; Veroniki et al., 2016). Heterogeneity is important because it added another source of uncertainty that could influence the tree-building process among the various data conditions. However, to not further extend the scope of this study, we did not test multiple values of heterogeneity, but set it to 0.025 for all datasets.

4.1.6 Summary

The simulation process generated 36 ($3 \cdot 2 \cdot 2 \cdot 3$) different datasets. One for each unique combination of the four design factors: types of moderators (3), effect size strengths (2), tree structures (2), and multiple values for the true correlation (3). The details of the simulation conditions are summarized in Table 4.2.

Table 4.2

Simulation Conditions and Options

Design Factor	Levels
Moderator Type	Continuous, Categorical, Unbalanced Categorical
Effect Size Strength	0.3, 0.5
Tree Structure	Model 1 (Figure 4.1/4.3), Model 2 (Figure 4.1/ 4.3)
Correlation Level	0.3, 0.5, 0.7

4.2 Variability Analysis

To analyze the effects of pruning on the random-effects meta-CART models, we generated 1,000 pruned trees for every dataset, each corresponding to a different seed. To ensure reproducibility and allow investigation of the variability caused by random seeds, we first selected a specific initial seed and then sampled 1,000 seeds from a broad range. This method allows others to reproduce the research while still allowing the variation in the pruning results to be investigated.

For each of the 1000 repetitions of the pruning process, we saved the number of terminal nodes for the pruned tree to assess the pruning variability. Based on these results, various analyses were conducted, including examining the number of terminal nodes, computing entropy values, and conducting ANOVA tests to assess whether the variability in pruning could be attributed to the different characteristics, or random effects introduced by the different seeds.

For the analysis, the two different tree models were examined independently. Examining the pruning variability for both models separately helped us to better understand how the influence of the different characteristics varied across different degrees of model complexity.

4.2.1 Pruning

To evaluate how different levels of pruning strictness affect the pruning variability, each dataset was analyzed using a `c.pruning` random-effects meta-CART value of 0 and 0.5. The `c.pruning` parameter represents the c value in the $c \cdot SE$ rule. Higher pruning values ($c = 0.5$) represent a stricter pruning criterion, producing smaller and simpler trees. In contrast, lower pruning values ($c = 0$) apply less strictness, resulting in larger and more complex trees that may be more sensitive to overfitting.

4.2.2 Terminal Nodes

One of the most important aspects of understanding pruning variability, is the number of terminal nodes left in the tree after pruning. In a decision tree, terminal nodes are the last nodes that will not be split further. If the number of terminal nodes is stable across most runs of the pruning process, the pruning process is producing consistent outcomes. In contrast, variations in the number of terminal nodes over several runs imply that slight data fluctuations and the seed used, influence the pruning process, thus resulting in variability in the pruned trees.

The terminal node count results were visualized using stacked bar graphs. These bar graphs displayed the proportion of different terminal node counts across different moderator types, correlations, effect sizes, and pruning values. Additionally, these bar graphs allowed for an assessment of whether the pruning process selected the correct number of terminal nodes (the number that corresponded to the actual underlying tree model).

4.2.3 Entropy

To further assess the variability in pruning, the entropy values of the terminal node distribution were computed. Entropy functions as a measurement of uncertainty that becomes especially valuable for categorical count data since it measures the variability and unpredictability of categories present in a dataset (Alsakran et al., 2014; Archer et al., 2013; Saraiva, 2023). In this context, the entropy quantified the variability in the number of terminal nodes across several runs. The entropy values were calculated using the Shannon entropy (Saraiva, 2023; Shannon, 1948). Shannon entropy was selected because it offers a straightforward measure of variability that treats all outcomes equally without requiring tuning parameters. The following formula defines this entropy:

$$D = -\sum_{k=1}^{K} \hat{p}_k \log_2(\hat{p}_k), \tag{4.5}$$

where $\hat{p}_k$ denotes the proportion of models with terminal nodes classified in category k, and K is the total number of possible categories (the number of terminal nodes). A low entropy value indicated that the pruning process produced trees that were mostly the same size across all runs, whereas a high entropy value meant that the pruning process was more sensitive to the different seeds and resulted in more variability in tree sizes.

4.2.4 ANOVA

ANOVA tests were applied to further investigate how pruning strictness, moderator type, effect size strength, and moderator correlation, influenced the entropy values (representing the pruning variability). Specifically, the ANOVA tests assessed whether these factors had a significant effect on entropy, which had the largest influence, and whether there were interaction effects between them.

To identify which variables contributed to pruning variability and whether these effects depended on interactions between them, we compared models with increasing complexity. Specifically, we wanted to determine whether including interaction terms improved model fit. To do this, we first compared a model with only main effects to a model that included all two-way interactions. Then, we compared the two-way interaction model to a model that also included three-way interactions. These comparisons were made using ANOVA tests to assess whether adding interaction terms led to a significantly better fit. To ensure the validity of the final ANOVA model, the assumptions of normality and homoscedasticity were assessed by inspecting Q-Q plots and residuals versus fitted values plots. In addition, the Shapiro-Wilk test was conducted to further test the normality of the residuals. Line graphs were then used to visualize the significant interactions. These graphs helped us better understand the interactions between the characteristics. Additionally, post-hoc comparisons were done using Tukey's Honest Significant Difference (HSD) test, in order to investigate the significant main effects of characteristics that

did not contain significant interactions.

In summary, this analysis provided a comprehensive approach to investigating the pruning variability in random-effects meta-CART models resulting from cross-validation. We looked at how different characteristics influenced the pruning process using 1,000 distinct seeds, and then evaluated the outcomes in terms of terminal nodes, entropy, and ANOVA testing.

Chapter 5

Results

In this chapter, the results of the pruning variability analysis in random-effects meta-CART models, are presented. Specifically, the influence of pruning with cross-validation on the stability of the tree structures. The main outcomes that are presented, are the number of terminal nodes, entropy values, and the effects of the various characteristics and the pruning strictness, assessed through ANOVA. The results show how pruning affected the stability and interpretability of decision trees under different meta-analytic conditions.

5.1 Terminal Node Variability

The number of terminal nodes remaining after pruning was an important measure of pruning variability. This section examines the distribution of terminal node proportions across 1,000 generated trees for every dataset with each pruning strictness value ($c = 0$ and $c = 0.5$).

5.1.1 Tree Model 1 Results

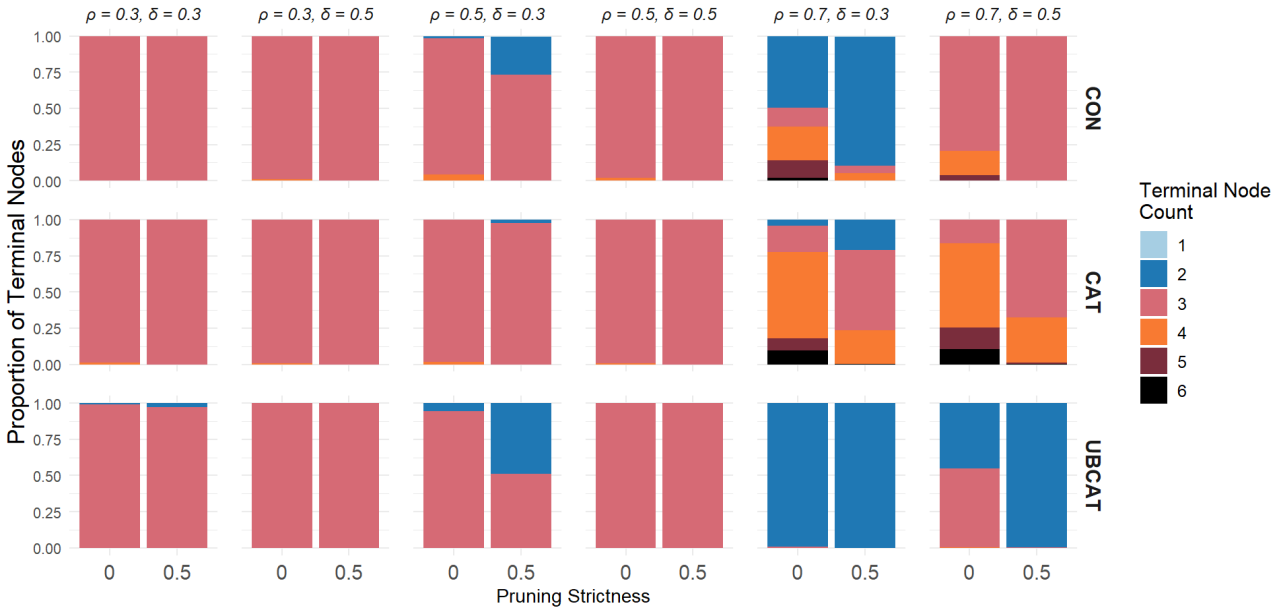
Recovery of the Correct Number of Terminal Nodes

The terminal node proportions for tree model 1 are shown in Figure 5.1. Tree model 1 had a simpler underlying structure, and the correct number of terminal nodes, based on the true data structure, was three, as visualized in Figures 4.1 and 4.3. The correct number of three terminal nodes was often recovered when the correlation between moderators was low or moderate ($\rho = 0.3$ and $\rho = 0.5$). In contrast, for high correlation ($\rho = 0.7$), the correct number of terminal nodes was often not recovered. When the effect size was low ($\delta = 0.3$) and the correlation high ($\rho = 0.7$), underfitting occurred more frequently. This happened particularly for datasets with continuous (CON) and unbalanced categorical (UBCAT) moderators, with the number of terminal nodes being lower than three. For balanced categorical (CAT) moderators, however, overfitting happened more often, with trees often having more than three terminal

nodes. This overfitting effect was less pronounced under stricter pruning ($c = 0.5$). For UBCAT moderators under high correlation, underfitting was often observed regardless of the pruning strictness, suggesting that the combination of collinearity and imbalance made it difficult for the model to detect all important splits. Pruning strictness influenced recovery as expected. For the less strict pruning ($c = 0$), trees tended to have more terminal nodes, leading to more complex models. In contrast, stricter pruning ($c = 0.5$) resulted in trees with fewer terminal nodes, limiting overfitting but potentially increasing the risk of underfitting depending on the data conditions.

Figure 5.1

Proportion of Terminal Nodes for Tree Model 1



Note. The x-axis represents the pruning values ($c = 0$ and $c = 0.5$), and the y-axis shows the terminal nodes proportions. The bars represent the proportion of terminal node count across different moderator types, continuous (CON), categorical (CAT), and unbalanced categorical (UBCAT), and correlation-effect size (ρ and δ) combinations.

Stability of the Tree

In addition to investigating the recovery of the correct number of terminal nodes, the stability of the tree structures across replications could also be explored by looking at Figure 5.1. First, the stability of the tree structures seemed to be influenced by the pruning strictness. With less strict pruning ($c = 0$), the number of terminal nodes seemed to vary more across replications, indicating inconsistent pruning behavior. In contrast, stricter pruning ($c = 0.5$) led to less variation in terminal node counts, suggesting that this pruning level reduced the sensitivity of

the model to small fluctuations in the data. Furthermore, the type of the moderators also affected the stability of terminal nodes. UBCAT in general produced relatively consistent terminal node counts regardless of the correlation value. The low variation suggests that the imbalance in category frequencies might have constrained the ability of the model to include splits that were not important. However, this same restriction may have contributed to underfitting, as we have discussed before. For CON and CAT, there seemed to be more variability, especially combined with lower effect sizes ($\delta = 0.3$).

The correlation between moderators also seemed to influence tree stability. For high correlation ($\rho = 0.7$), there was more variability in the number of terminal nodes compared to the lower correlations, where variability was minimal. Additionally, pruning variability was also affected by the effect size. The variability in terminal node counts across almost all combinations of moderator type, pruning level, and correlation increased when the average effect size was lower ($\delta = 0.3$). This implies that the tree pruning process became more sensitive to random noise in the data when the effect size was less strong, which then influenced how many splits were kept after pruning.

In summary, these findings indicate that an interaction of pruning strictness, moderator type, moderator correlation, and effect size strength, influenced tree complexity and stability in tree model 1. Stricter pruning in general resulted in more stable and simpler trees while sometimes underfitting the data. Less strict pruning resulted in more complicated and variable trees that sometimes overfit the data, especially when combined with strong moderator correlation or low effect sizes. Pruning results were also influenced by moderator structure, where imbalanced categorical moderators produced smaller and more stable models that might have missed important features in the data.

5.1.2 Tree Model 2 Results

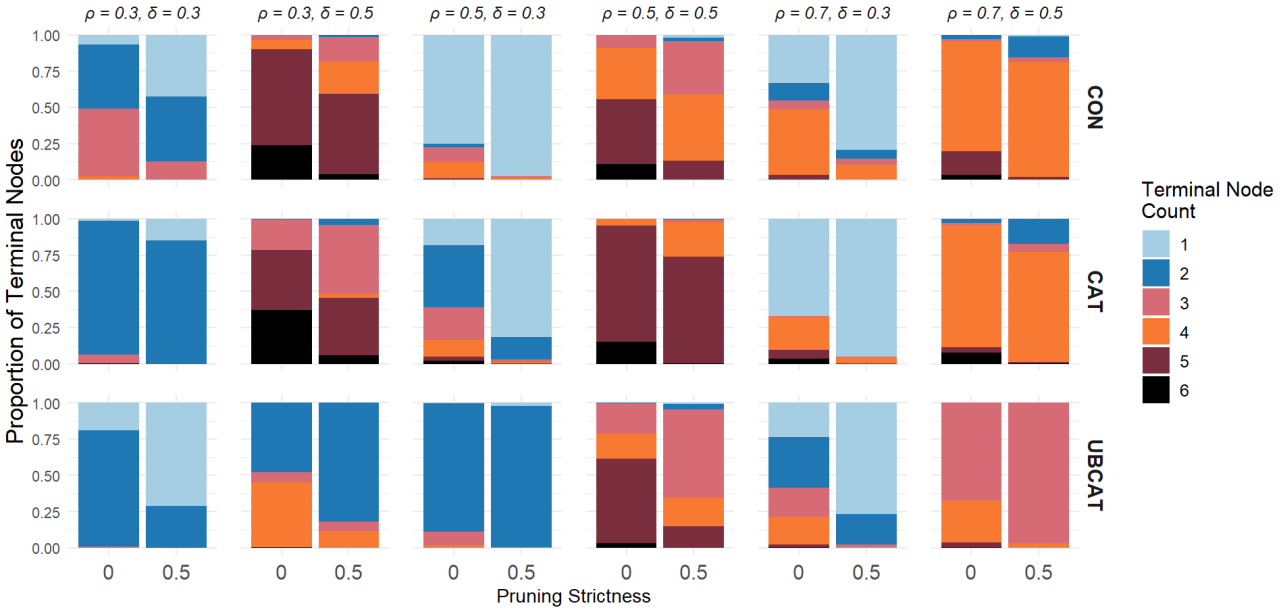
Recovery of the Correct Number of Terminal Nodes

The terminal node proportions for tree model 2 are shown in Figure 5.2. Tree model 2 had a more complex underlying tree structure, compared to tree model 1. The correct number of terminal nodes for this model, based on the true data structure, was four, as visualized in Figures 4.2 and 4.4. For Tree Model 2, the correct number of terminal nodes was often not achieved, especially for small to moderate correlations ($\rho = 0.3$ and $\rho = 0.5$). In most conditions, the trees either underfit or overfit the data. Specifically, for lower effect sizes ($\delta = 0.3$), trees tended to underfit, while for higher effect sizes ($\delta = 0.5$), they overfit. The correct number of terminal nodes was most often achieved under high moderator correlation ($\rho = 0.7$) combined with higher effect sizes ($\delta = 0.5$), specifically for CON and CAT. This suggests that, for tree model 2, high correlation and stronger effect sizes might have helped the model better recover the correct number of terminal nodes. Furthermore, pruning strictness also influenced the results. When pruning was less strict ($c = 0$), trees in general had more terminal nodes, as expected.

Additionally, for stricter pruning ($c = 0.5$), the number of terminal nodes was reduced, as we would expect.

Figure 5.2

Proportion of Terminal Nodes for Tree Model 2



Note. The x-axis represents the pruning values ($c = 0$ and $c = 0.5$), and the y-axis shows the terminal nodes proportions. The bars represent the values of entropy across different moderator types, continuous (CON), categorical (CAT), and unbalanced categorical (UBCAT), and correlation-effect size (ρ and δ) combinations.

Stability of the Tree

In addition to investigating the recovery of the correct number of terminal nodes, the stability of the tree structures across replications, could also be explored by looking at Figure 5.2. The variability in terminal node counts was much higher compared to tree model 1. This higher variability made it more difficult to explore specific patterns. However, pruning strictness seemed to influence the tree stability again. For less strict pruning ($c = 0$), the variability in terminal node counts seemed higher, indicating less stable trees across replications. In contrast, stricter pruning ($c = 0.5$) seemed to have more consistent terminal node counts. This suggests that stricter pruning helped reduce sensitivity to random noise in the data again.

Furthermore, the correlation between moderators also seemed to influence the stability of tree model 2. High correlation ($\rho = 0.7$) often seemed to result in more stable trees. This could specifically be seen when combined with the larger effect size ($\delta = 0.5$). In contrast, lower

correlation values seemed to show more variability in terminal node counts. Additionally, the effect size also seemed to influence tree stability. When the effect size was small ($\delta = 0.3$), the pruning variability seemed to be lower. In contrast, with higher effect sizes, there seemed to be more variability, except for combinations with the highest correlation. This is not consistent with what we had seen for tree model 1, where this effect was reversed and lower effect sizes led to more variable pruning behavior. Finally, for the moderator types, the higher variability in tree model 2 made it difficult to see specific patterns in how different moderator types affected the tree structure stability by just looking at the proportions.

In summary, for tree model 2, the correct number of terminal nodes was often not achieved. Furthermore, tree model 2 had more variability in the number of terminal nodes compared to tree model 1. High moderator correlation, especially when combined with stronger effect sizes, seemed to help with recovering the correct amount of terminal nodes. However, pruning strictness remained important. Stricter pruning provided more stability and simplicity in the tree structures. In contrast, less strict pruning allowed for more complexity, and had more variability.

5.2 ANOVA Results

To investigate the influence of the characteristics of the data and the pruning strictness on pruning variability, a series of ANOVA tests were conducted using the entropy values as the response variable. These analyses were used to confirm earlier observations and to explore how main effects and interaction terms contributed to pruning variability in more depth.

5.2.1 Tree Model 1 Results

Model Comparison

For tree model 1, three ANOVA models were evaluated, one including only main effects, one additionally incorporating two-way interactions, and one extending further to three-way interactions. The main effects model contained the pruning strictness and the characteristics of the data, specifically moderator type, effect size strength, and correlation structure, as the variables. Two-way interactions let these moderators interact with each other, therefore assessing their combined effect on the pruning variability. Three-way interactions allowed for an understanding of their combined effects even further by allowing the interactions between two moderators to be affected by a third variable. The two-way interaction model was initially compared with the main effects model to evaluate model performance. The two-way interaction model was then compared with the three-way interaction model to see if adding more complex interactions had additional explanatory value. Table 5.1 shows the results of these comparisons. The addition of two-way interactions significantly enhanced model fit ($p = .0052$), suggesting that interactions between the characteristics were important for explaining the pruning variability. Adding three-way interactions, however, did not show any significant improvement ($p = .2183$), suggesting

that higher-order interactions did not significantly increase the explanatory power. Thus, the two-way interaction model was selected for further analysis.

Table 5.1

ANOVA Comparison for Tree Model 1

Comparison	Model	Residual *df*	*RSS*	*df*	*SS*	*p*-value
Main vs. 2-way	Main Effects	29	5.519	-	-	-
	Two-Way Interactions	16	1.297	13	4.222	.0052 **
2-way vs. 3-way	Two-Way Interactions	16	1.297	-	-	-
	Three-Way Interactions	4	0.164	12	1.133	.2183

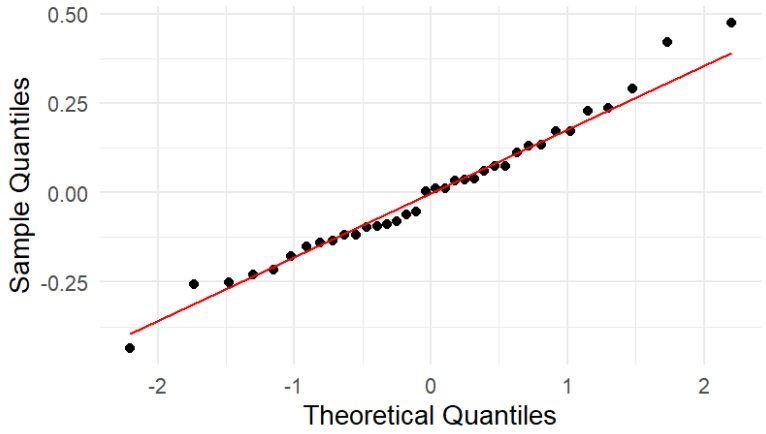
Note. The main effects model was first compared with the two-way interaction model, followed by a comparison of the two-way interaction model with the three-way interaction model. Asterisks indicate statistical significance (*** for $p < .001$, ** for $p < .01$, * for $p < .05$, and . for $p < .1$).

Assumption Checks

Before interpreting the results of the two-way interaction ANOVA model, several assumptions were checked. The Q-Q plot of the residuals (Figure 5.3) indicates that the residuals were approximately normally distributed.

Figure 5.3

Q-Q Plot of Residuals for Tree Model 1



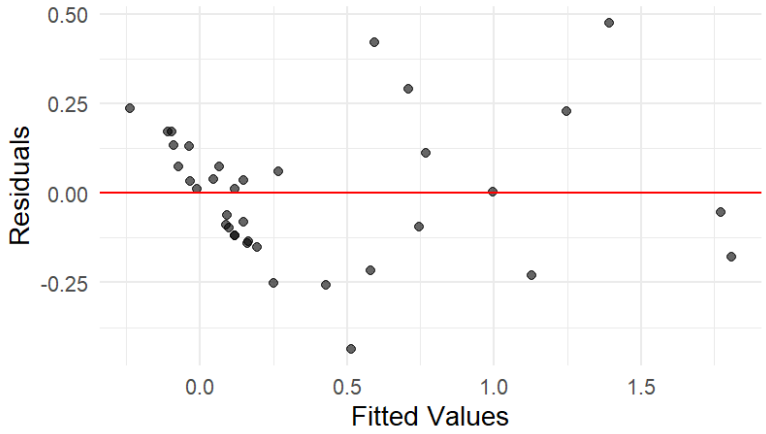
Note. The Q-Q plot checks the normality assumption for the residuals. The x-axis represents the theoretical quantiles, while the y-axis represents the sample quantiles of the residuals. The red reference line shows where the residual points should be, to be normally distributed.

This was supported by the Shapiro-Wilk test ($W = 0.981$, $p = .7743$), which did not provide evidence to reject the assumption of normality. This suggests that the residuals followed a distribution that is close enough to normal for the analysis to be considered valid in this regard.

Homoscedasticity was evaluated by plotting the residuals against the fitted values (Figure 5.4). For the homoscedasticity assumption to be met, the residuals should be randomly scattered across the full range of fitted values. In the figure, we can see that there was random scattering around zero for the higher fitted values. At the lower end, however, a clustering of residuals was observed. This could be due to many entropy values being zero or near zero, resulting in a concentration of residuals in that section of the graph. Even though this trend indicates a potential deviation from the homoscedasticity assumption, it appeared to be caused by particular features of the dataset rather than a universal trend across all fitted values. However, this possible violation should be kept in mind when interpreting the results of the model.

Figure 5.4

Residuals vs Fitted Values for Tree Model 1



Note. The x-axis represents the fitted values from the model, while the y-axis shows the differences between the observed values and the predicted values. This plot is used to assess homoscedasticity. If no clear trend was observed, the assumption of equal variances across fitted values is supported.

Summary of Model Results

The ANOVA summary for the two-way interaction model is shown in Table 5.2. Most of the variation in pruning variability was explained by interaction terms, specifically the interaction between moderator type and correlation ($p = .0017$) and the interaction between correlation and pruning strictness ($p = .0061$). The first significant interaction indicates that the effect of correlation on pruning variability was not consistent across all conditions but varied depending on

the type of the moderator. Furthermore, the significant interaction between correlation and the pruning strictness suggests that the influence of correlation depended on the pruning strictness. Since correlation was involved in multiple significant interactions, the main effect ($p < .0001$) should be interpreted as conditional on other variables. Furthermore, effect size showed a significant main effect ($p = .0294$), indicating that changes in effect size strength influenced pruning variability across conditions, independent of the other variables. This main effect is discussed further in the post hoc analysis section.

Table 5.2

ANOVA Summary for Two-Way Interaction Model (Tree Model 1)

Factor	SS	MS	F-Statistic	p-value
Moderators	0.574	0.287	3.543	.0532 .
Correlation	4.360	2.180	26.897	< .0001 ***
Effect Size	0.463	0.464	5.719	.0294 *
Pruning Strictness	0.260	0.260	3.206	.0923 .
Moderators:Correlation	2.327	0.582	7.177	.0017 **
Moderators:Effect Size	0.209	0.105	1.291	.3021
Moderators:Pruning Strictness	0.097	0.048	0.597	.5622
Correlation:Effect Size	0.245	0.122	1.508	.2511
Correlation:Pruning Strictness	1.158	0.579	7.146	.0061 **
Effect Size:Pruning Strictness	0.186	0.186	2.299	.1489
Residuals	1.297	0.081	-	-

Note. The Sum of Squares (SS) represents the total variation attributed to each factor. The Mean Square (MS) is obtained by dividing SS by the degrees of freedom and reflects the average variance explained by each factor. The F-statistic is the ratio of MS for a given factor to the residual MS, indicating how much the factor contributes relative to unexplained variance. Asterisks indicate statistical significance (*** for $p < .001$, ** for $p < .01$, * for $p < .05$, and . for $p < .1$).

Exploration of Interactions

To better understand the interaction effects that influenced the pruning variability in tree model 1, two interaction plots were generated. Figure 5.5 visualizes the interaction between correlation (ρ) and pruning strictness (c). At $c = 0.5$, entropy increased slightly as the correlation increased from 0.3 to 0.5 and continued to increase slightly more from 0.5 to 0.7. For $c = 0$, entropy slightly increased again between correlation 0.3 and 0.5. However, at correlation 0.7, the entropy value increased substantially. This suggests that when pruning was less strict ($c = 0$), the influence of correlation was minimal until the correlation became very high, where the variability increased substantially. These patterns showed that the pruning strictness value moderated the

influence of correlation on pruning variability, with higher c values reducing the variability for high correlation compared to the lower c value.

Figure 5.5

Interaction Between Correlation and Pruning Strictness (Tree Model 1)

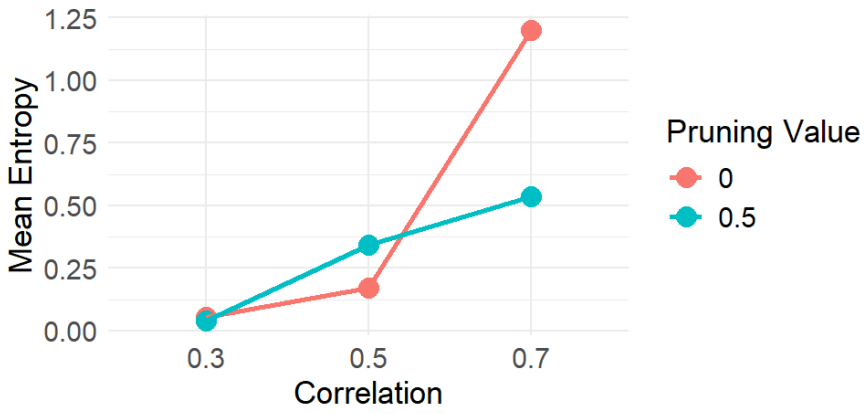


Figure 5.6

Interaction Between Correlation and Moderator Type (Tree Model 1)

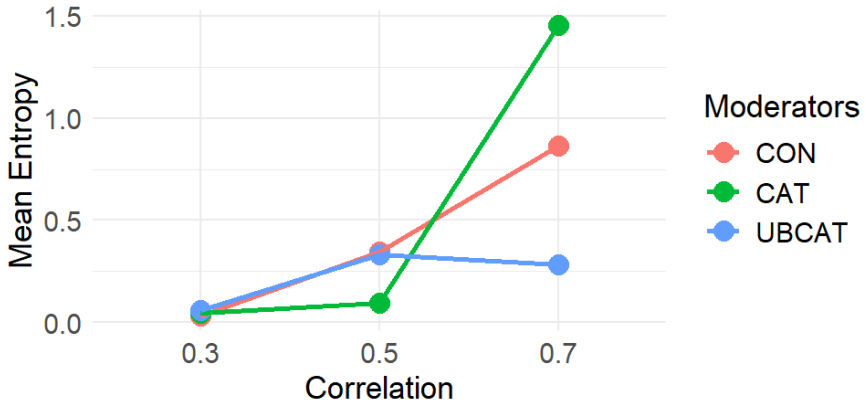


Figure 5.6 visualizes the interaction between correlation (ρ) and moderator type. For continuous moderators (CON), entropy increased as correlation increased, showing a linear trend. For categorical moderators (CAT), entropy only increased very slightly but stayed low between correlations of 0.3 and 0.5. However, it showed a big increase at correlation 0.7, indicating greater sensitivity of categorical moderators to high correlation levels. For unbalanced categorical moderators (UBCAT), entropy was low at correlation 0.3, increased at 0.5, and then slightly decreased

at 0.7. These results demonstrated that the way correlation influenced the pruning variability depended not only on the correlation level but also on the type of moderator. Balanced categorical moderators appeared specifically sensitive to high correlation ($\rho = 0.7$), whereas continuous moderators showed a more consistent pattern of increasing pruning variability.

Post Hoc Analysis

To further examine the effect of the effect size strength, a post-hoc test was conducted since effect size only showed a significant main effect and was not involved in any significant interactions. A Tukey HSD test was used to compare the mean entropy values between the two effect size levels. The test revealed that entropy at $\delta = 0.5$ was significantly lower than at $\delta = 0.3$ (mean difference $= -0.23$, $p = .0294$), suggesting that reduced pruning variability was associated with stronger average effect sizes. This finding implies that the tree models were better at separating important splits from noise when effect sizes were stronger, therefore producing more consistent pruning results across replications.

In summary, the results for tree model 1 showed that pruning variability was mainly influenced by two-way interactions between correlation and moderator type, and between correlation and pruning strictness. These interactions indicate that the effect of correlation depended on both the type of moderator and the level of pruning. The effect size also had a significant effect, with smaller effect sizes increasing the pruning variability compared to higher effect sizes.

5.2.2 Tree Model 2 Results

Model Comparison

For tree model 2, three ANOVA models were evaluated, one including only main effects, one also including two-way interactions, and one additionally including three-way interactions. The main effects model included moderator type, correlation, effect size, and pruning strictness as separate variables. Two-way interactions allowed for combinations of these variables, while three-way interactions examined how the effect of two variables was influenced by a third. The models were compared sequentially to assess the improvement in model fit as more interaction terms were added. The results of these comparisons are presented in Table 5.3.

Adding two-way interactions to the main effects model did not significantly improve the fit ($p = .5074$). However, extending the model to include three-way interactions led to a significant improvement over the two-way interaction model ($p = .0065$). This was also confirmed by the direct comparison between the main effects model and the three-way interaction model, which showed a statistically significant increase in model fit ($p = .0080$). These results suggest that the inclusion of three-way interactions was important for capturing variation in the pruning variability in tree model 2. Thus, the three-way interaction model was selected for further analysis.

Table 5.3

ANOVA Comparison for Tree Model 2

Comparison	Model	Residual *df*	*RSS*	*df*	*SS*	*p*-value
Main vs. 2-way	Main Effects	29	6.915	-	-	-
	Two-Way Interactions	16	3.849	13	3.066	.5074
2-way vs. 3-way	Two-Way Interactions	16	3.849	-	-	-
	Three-Way Interactions	4	0.070	12	3.779	.0065 **
Main vs. 3-way	Main Effects	29	6.915	-	-	-
	Three-Way Interactions	4	0.070	25	6.845	.0080 **

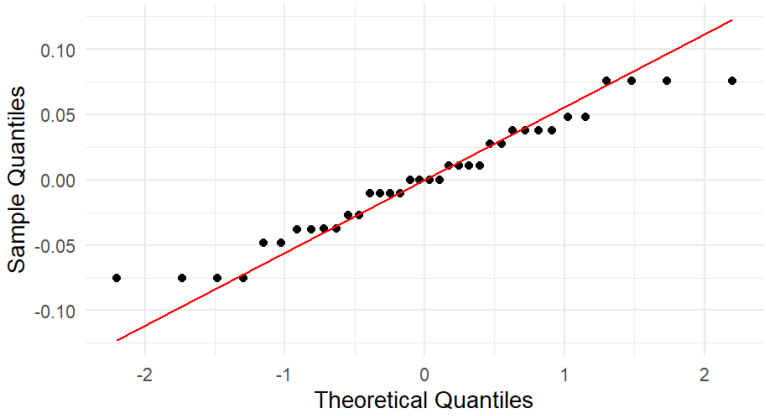
Note. The main effects model was first compared with the two-way interaction model, followed by a comparison of the two-way interaction model with the three-way interaction model. Lastly, the main effects model was compared with the three-way interaction model. Asterisks indicate statistical significance (*** for $p < .001$, ** for $p < .01$, * for $p < .05$, and . for $p < .1$).

Assumption Checks

Before interpreting the results of the three-way interactions ANOVA model for tree model 2, we assessed whether the model assumptions were met. First, we examined a Q-Q plot of the residuals (Figure 5.7), which suggests that the points slightly deviated from the indicated line but were still relatively aligned, indicating a somewhat normal distribution.

Figure 5.7

Q-Q Plot of Residuals for Tree Model 2

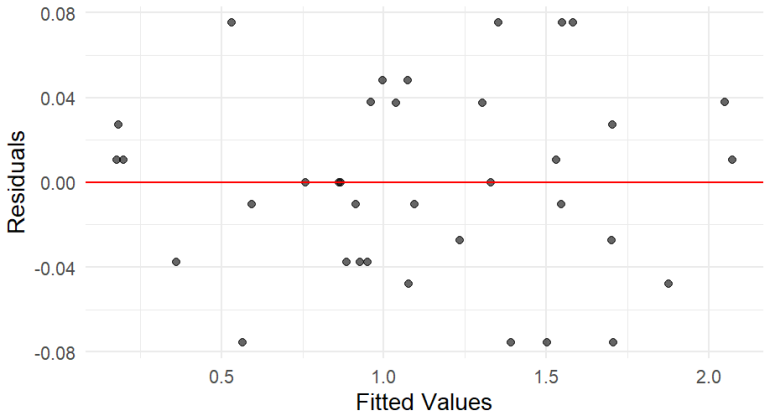


Note. The Q-Q plot checks the normality assumption for the residuals. The x-axis represents the theoretical quantiles, while the y-axis represents the sample quantiles of the residuals. The red reference line shows where the residual points should be, to be normally distributed.

To test this further, a Shapiro-Wilk test was conducted, yielding a test statistic of $W = 0.956$ and a p-value of .1603. The relatively high p-value indicates that the residuals did not significantly deviated from normality, and thus the normality assumption appeared to be met. Next, we inspected the residuals versus fitted values plot (Figure 5.8) to evaluate homoscedasticity. In this plot, residuals should be evenly scattered across the range of fitted values without showing any trend or pattern. The plot for tree model 2 displayed an even spread of residuals around zero, without any noticeable trends or patterns. Unlike in tree model 1, no clustering at the lower fitted values was observed. This suggests that the variance of the residuals was relatively constant across different levels of the response, supporting the assumption of homoscedasticity. So, the Q-Q plot, Shapiro-Wilk test, and residuals versus fitted values plot all supported the conclusion that the assumptions underlying the ANOVA model were reasonably well satisfied for tree model 2.

Figure 5.8

Residuals vs Fitted Values for Tree Model 2



Note. The x-axis represents the fitted values from the model, while the y-axis shows the differences between the observed values and the predicted values. This plot is used to assess homoscedasticity. If no clear trend was observed, the assumption of equal variances across fitted values is supported.

Summary of Model Results

The summary of the ANOVA results for the three-way interactions model is presented in Table 5.4. The results showed that several three-way interactions significantly contributed to pruning variability in tree model 2. Specifically, the interactions between moderators, correlation and effect size, the interaction between moderators, effect size and pruning strictness, and the interaction between correlation, effect size and pruning strictness, all reached statistical

significance ($p = .0018$, $p = .0489$, and $p = .0191$, respectively). These findings suggest that pruning variability was influenced by the combined effects of the characteristics rather than by any single variable or two-way interaction. When higher-order interactions are present and significant, the individual and lower-level interaction effects involving those variables are usually less informative. Therefore, the significant results suggest that the relationship between pruning variability and the characteristics depended on specific combinations of moderator type, effect size strengths, and correlation levels.

Table 5.4

ANOVA Summary for Three-Way Interactions Model (Tree Model 2)

Factor	SS	MS	F-Statistic	p-value
Moderators	0.6320	0.3160	18.075	.0099 **
Correlation	0.0452	0.0226	1.294	.3687
Effect Size	0.4432	0.4432	25.348	.0073 **
Pruning Strictness	1.0878	1.0878	62.219	.0014 **
Moderators:Correlation	0.3021	0.0755	4.321	.0927 .
Moderators:Effect Size	0.0031	0.0015	0.088	.9179
Moderators:Pruning Strictness	0.0687	0.0343	1.965	.2545
Correlation:Effect Size	1.4693	0.7346	42.021	.0021 **
Correlation:Pruning Strictness	0.6589	0.3294	18.844	.0092 **
Effect Size:Pruning Strictness	0.5640	0.5640	32.261	.0047 **
Moderators:Correlation:Effect Size	2.7856	0.6964	39.834	.0018 **
Moderators:Correlation:Pruning Strictness	0.3112	0.0778	4.450	.0886 .
Moderators:Effect Size:Pruning Strictness	0.2462	0.1231	7.042	.0489 *
Correlation:Effect Size:Pruning Strictness	0.4361	0.2180	12.472	.0191 *
Residuals	0.0699	0.0175	-	-

Note. The Sum of Squares (SS) represents the total variation attributed to each variable. The Mean Square (MS) is obtained by dividing SS by the degrees of freedom and reflects the average variance explained by each variable. The F-statistic is the ratio of MS for a given variable to the residual MS, indicating how much the variable contributes relative to unexplained variance. Interaction effects are indicated with colons. Asterisks indicate statistical significance (*** for $p < .001$, ** for $p < .01$, * for $p < .05$, and . for $p < .1$).

Exploration of Interactions

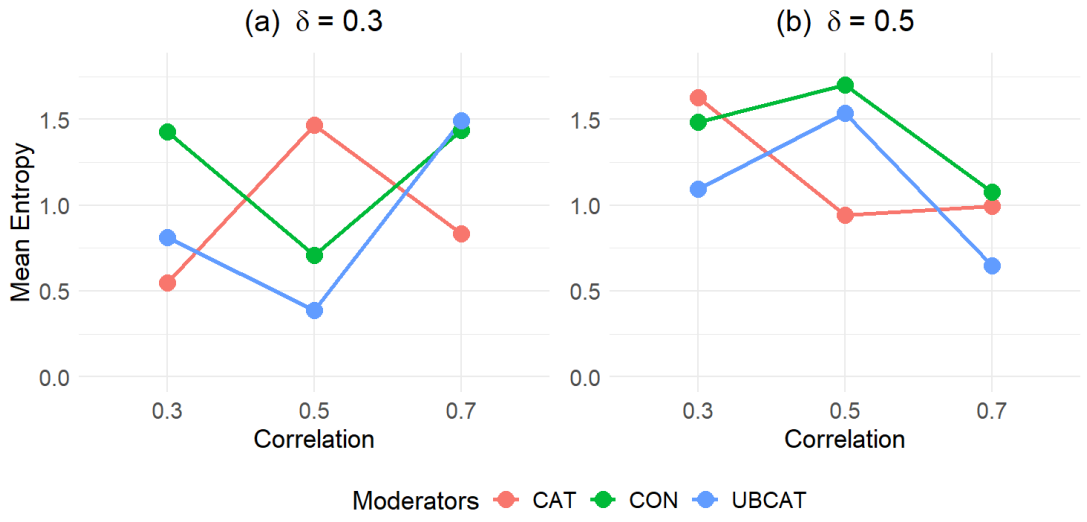
To better understand the complex interactions that influenced the pruning variability in tree model 2, various interaction plots were generated. These figures examined the combined effects of moderator type, correlation, and pruning strictness, at two different levels of effect size. The two panels in the figures (Figure 5.9, 5.10, and 5.11) show the results for $\delta = 0.3$ (panel a) and

$\delta = 0.5$ (panel b). This allowed for a comparison across effect size levels and an assessment of whether the strength or the form of the interactions changed across these levels.

Figure 5.9 visualizes the interaction between moderator type and correlation. At $\delta = 0.3$, the shape of the interaction was different between moderator types. For continuous moderators (CON), entropy followed a U-shape with entropy being the highest at correlation values of 0.3 and 0.7, and lowest at 0.5. Unbalanced categorical moderators (UBCAT) showed a similar U-shaped pattern, but the entropy was overall lower, suggesting that pruning was more stable for UBCAT which might be due to the imbalance making certain splits more likely to be kept. For balanced categorical moderators (CAT), the pattern was reversed. Here, entropy had a peak at correlation 0.5 and was lower at the extremes. This suggests that pruning was most variable when the correlation was moderate for CAT. At $\delta = 0.5$, the patterns were reversed. For the higher effect size, CON and UBCAT showed reversed U-shapes, with entropy highest at correlation 0.5, and lower at 0.3 and 0.7. For CAT, the shape both reversed and flattened. Entropy decreased from 0.3 to 0.5 and slightly increased from 0.5 to 0.7, forming a weaker U-shape and a flatter trend.

Figure 5.9

Interaction Between Correlation and Moderator Type at Different Effect Size Levels (Tree Model 2)



Comparing the two effect size levels, entropy was in general higher at $\delta = 0.5$ than at $\delta = 0.3$, but more importantly, the shape of the interaction changed. At $\delta = 0.3$, the relationship between correlation and entropy was clearly different across moderator types, showing an U-shape for CON and UBCAT, and a reversed U-shape for CAT. At $\delta = 0.5$, these patterns reversed or

flattened. This suggests that the way correlation affected pruning variability depended not only on moderator type but also on the effect size strength. As the effect size increased, the correlation point where pruning was most unstable shifted, and the shape of the interaction changed.

Figure 5.10 shows the interaction between correlation and pruning strictness. At $\delta = 0.3$, the strength of the interaction was clear, with entropy increasing with correlation when pruning was less strict ($c = 0$), but showing a non-linear pattern under stricter pruning ($c = 0.5$), with entropy lowest at a correlation of 0.5. This suggests that stricter pruning reduced variability when correlation was moderate, while less strict pruning resulted in steadily increasing variability as correlation increased. At $\delta = 0.5$, the interaction mostly disappeared. For both pruning levels, entropy was relatively stable between correlations of 0.3 and 0.5, and then dropped at 0.7. The drop was more pronounced for stricter pruning, indicating that at larger effect sizes, pruning helped reduce pruning variability under strong correlation. However, in general the difference between pruning levels was much smaller than at $\delta = 0.3$.

So, comparing the two effect size levels, the overall entropy was again, in general, higher at $\delta = 0.5$. However, the most important difference was the difference in strength of the interaction. At $\delta = 0.3$, there was a clear difference between the pruning conditions across correlation levels, while at $\delta = 0.5$, the patterns were often parallel. This suggests that pruning interacted with correlation more strongly when the effect size was smaller.

Figure 5.10

Interaction Between Correlation and Pruning Strictness at Different Effect Size Levels (Tree Model 2)

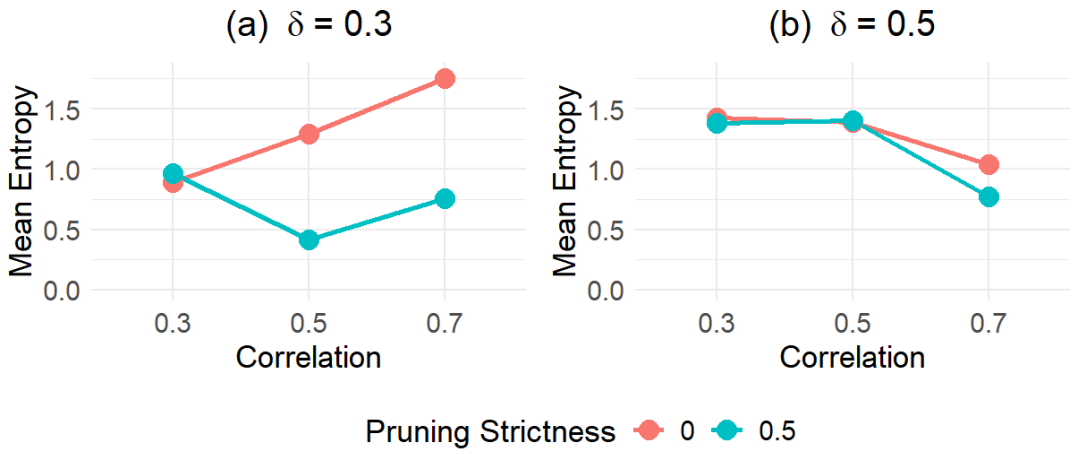


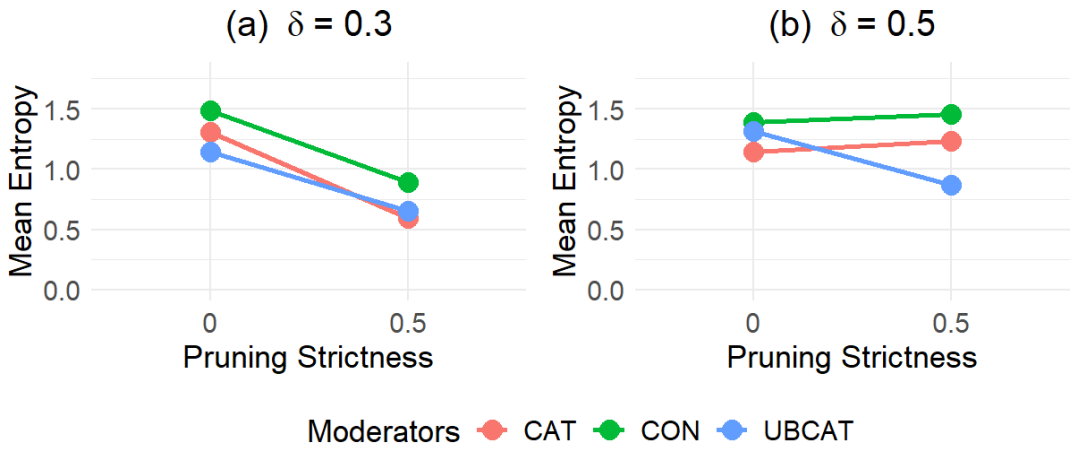
Figure 5.11 shows the interaction between moderator type and pruning strictness. At $\delta = 0.3$, entropy was higher for all moderator types when pruning was less strict ($c = 0$), with continuous moderators (CON) showing the most variability, followed by balanced categorical moderators

(CAT), and then unbalanced categorical moderators (UBCAT). When pruning became stricter ($c = 0.5$), entropy decreased for all three moderator types. At $\delta = 0.5$, the strength and shape of the interaction were quite different. For CON and CAT, entropy slightly increased under stricter pruning, indicating that pruning no longer stabilized the tree as it did at lower effect sizes. In contrast, UBCAT showed a decrease in entropy as pruning became stricter, maintaining the same shape of effect as seen at $\delta = 0.3$. This implies that the effectiveness of pruning at reducing variability in this case depended more on the moderator type. While it still decreased entropy for UBCAT, it increased entropy slightly for CON and CAT.

When comparing both effect size levels, aside from the entropy again being higher in general at $\delta = 0.5$, it can be seen that the strength of the interaction between pruning strictness and moderator type changed for the different effect sizes. At $\delta = 0.3$, the patterns across moderator types were similar, with stricter pruning leading to lower entropy in all cases, suggesting little interaction. However, at $\delta = 0.5$, this changed. UBCAT still showed a decrease in entropy under stricter pruning, but CON and CAT showed slight increases. This difference indicates that the interaction between pruning and moderator type became stronger at higher effect sizes.

Figure 5.11

Interaction Between Pruning Strictness and Moderator Type at Different Effect Size Levels (Tree Model 2)



In summary, the results for tree model 2 showed that including three-way interactions significantly improved model fit, and these interactions were important for explaining pruning variability. The influence of moderator correlation, effect size, pruning strictness, and moderator type on pruning variability depended on their combinations rather than on individual effects alone.

Chapter 6

Discussion

6.1 Findings

This thesis provided a comprehensive analysis of the pruning variability in random-effects meta-CART models caused by cross-validation, by investigating how pruning strictness and characteristics of the data influence this variability. Studying the effect of different moderator types, effect size strengths, correlation levels, and pruning strictness on pruning variability, revealed that the variability often depended on the interaction between these factors, rather than a single factor.

For the more simple model (tree model 1), stricter pruning led to fewer terminal nodes, as expected, and produced more stable trees. However, this came at the cost of potentially discarding useful splits, especially when moderators were imbalanced or strongly correlated. This issue could be due to the splits involving minority categories that are often pruned during cross-validation. This issue has been mentioned before by researchers, where decision trees struggle with imbalanced variables and do not prioritize the minority categories (Chaabane et al., 2020; Liu et al., 2010). In contrast, less strict pruning retained more complexity, as expected. However, it also introduced greater variability, especially when effect sizes were weak and moderators were highly correlated. Under those conditions, the model might include noise, making the pruning outcomes less consistent. These results align with previous findings that decision trees have to deal with a trade-off between simplicity and sensitivity, where simpler models are more stable but risk missing important patterns, while more complex models capture more detail but can be more sensitive to random variation (Breiman et al., 1984; James et al., 2013). This study extends that idea to meta-analysis, where the data can often be more variable to begin with.

The more complex model (tree model 2) had a different pruning variability pattern than tree model 1. In general, the pruning behavior of tree model 2 was more variable, compared to tree model 1. As expected, less strict pruning resulted in more terminal nodes on average. However, it also led to higher variability, implying higher sensitivity to random variation. Consistent

with expectations, stricter pruning on average produced simpler trees with fewer terminal nodes. Furthermore, they also had lower variability on average. However, despite this lower variability, pruning rarely resulted in the correct number of terminal nodes. In most conditions, trees either underfit or overfit the data, depending on the effect size. Specifically, when effect sizes were low, the models tended to underfit the data, regardless of the moderator type or correlation. When the effect sizes were high, overfitting happened more often. Interestingly, higher correlation between moderators did not increase variability, as it did in tree model 1. For tree model 2, high correlation often helped the model achieve the correct number of terminal nodes when combined with the stronger effect sizes. This result suggests that in more complex trees, the combination of high correlation between moderators and strong effect sizes, can improve the stability of the model by detecting the correct number of terminal nodes. However, despite this stability, pruning in tree model 2 was still less accurate compared to tree model 1.

These results were further investigated using ANOVA models. The ANOVA results for tree model 1 indicated that pruning variability was mostly influenced by two-way interactions. Specifically, the interaction between moderator type and correlation suggests that the effect of correlation on pruning variability was different across moderator types. For continuous moderators, higher correlation increased entropy, this effect of collinearity may cause the model to emphasize less important moderators while missing important moderators. This result is consistent with prior research, which suggests that collinearity can lead trees to select only one variable from a group of correlated predictors (Loh, 2014; Strobl et al., 2008). In contrast, categorical moderators were very sensitive to high correlation, while unbalanced categorical moderators did not show as big of an increase in entropy. This result suggests that imbalance in categorical moderators might stabilize pruning by making certain splits more likely to be retained. Furthermore, the pruning strictness had a significant interaction with correlation. For less strict pruning, entropy increased when the correlation became higher. In contrast, for stricter pruning, the increase in entropy was much less strong as correlation became higher. This result suggests that stricter pruning can help stabilize the model even when high correlation is present. Effect size also played a significant role in pruning variability. Lower effect sizes had higher entropy values than higher effect sizes in general. This result suggests that weaker effect sizes made it harder for the model to identify stable splits.

The ANOVA results for tree model 2 showed that pruning variability was influenced by higher-order interactions rather than by single factors. The interaction between moderator type, correlation, and effect size suggests that the effect of correlation on pruning variability was not consistent across all moderator types or effect size levels. For example, correlation had a U-shaped effect on entropy for continuous moderators at the lower effect size, but this pattern reversed at the higher effect sizes. Furthermore, the interaction between correlation, effect size, and pruning strictness showed that the way correlation affects pruning variability also depended on how strict the pruning was, and how strong the effect sizes were. At lower effect sizes, correlation and pruning strictness interacted more clearly. For example, entropy values were in

general lower for stricter pruning at the lower effect sizes, while at higher effect sizes, entropy was generally less affected by the pruning strictness level. The last significant interaction was between moderator type, effect size, and pruning strictness. This interaction showed that stricter pruning had different effects depending on the type of moderator, and how strong the true effect sizes were. At lower effect sizes, stricter pruning reduced variability for all moderator types, but at higher effect sizes, the effect changed and became less strong for some moderator types.

So, the research question of whether characteristics of the data and pruning strictness contribute to pruning variability is confirmed. The results indicated that pruning variability depended on interactions between moderator type, effect size, correlation, and pruning strictness.

6.2 Limitations

Several methodological limitations should be considered when interpreting these results. While the simulation design allowed for strict control over moderator characteristics, it does not capture the full complexity of real-world meta-analytic datasets. Real empirical data can often involve issues such as missing moderator values, publication bias, and reporting errors. These factors introduce additional randomness and noise that were not simulated. Especially missing data can be common in practice. Since the simulations did not include missing data in either moderators or responses, the interaction between missingness and pruning decisions remains untested. This could affect pruning variability in practice. Furthermore, the simulations used fixed design choices, such as a constant number of studies ($n_k = 120$), a constant heterogeneity variance ($\sigma^2_\tau = 0.025$), a set of correlation levels ($\rho = 0.3, 0.5, 0.7$), and two pruning strictness levels ($c = 0$, $c = 0.5$). These choices made comparisons clearer but reduced generalizability. In practice, sample sizes, effect sizes, and moderator structures vary noticeably. It is not known how pruning variability would behave under, for example, smaller samples, greater heterogeneity, or even more complex structures.

Next, another limitation is about the assumptions in the ANOVA models. Normality of residuals was generally supported in the simulations, and the residual plots did not show major deviations. However, some minor heteroscedasticity was present. In particular, the entropy values showed zero inflation, meaning many values were clustered at or near zero. This led to residuals being more concentrated at the lower end of the scale. This pattern should be taken into account as it violates the assumption of constant variance and may affect the reliability of model comparisons.

Furthermore, the discretization of continuous moderators into categorical variables changed the correlation structure. Unbalanced categorization in particular tended to weaken correlations. For example, variables with $\rho = 0.7$ in the continuous condition often dropped to around $\rho = 0.6$ after discretization. Although discretization was necessary for modeling balanced and unbalanced categorical moderators in this study, it introduced systematic bias that should be taken into account.

In addition, a sample check of the fully grown trees revealed that the true underlying model with the correct set of splits was often not recovered. This means that all subsequent pruned trees were already based on suboptimal initial trees, which limits how much we can interpret the results in terms of recovering the correct tree structure. As a consequence, observed patterns in the number of terminal nodes or the entropy of pruned trees should be interpreted with caution. They reflect variability within a set of already imperfect trees, not variability around a known correct solution. However, even when starting from an incorrect full tree, repeated cross-validation still led to different pruning outcomes. So users would still see variation in the final pruned trees each time they rerun the process. It remains unclear whether greater pruning variability is linked to incorrect initial trees, but it is possible that an incorrect starting point makes the pruning process more sensitive to random variation in the cross-validation splits.

Finally, this study focused only on random-effects meta-CART models with specific levels of pruning strictness. Other decision tree methods, such as standard CART, fixed effects meta-CART, or ensemble methods like Random Forests, follow different principles for split selection and pruning. It is not clear if the results observed here would be the same for other decision tree methods. Therefore, comparisons across methods would be needed to generalize these findings.

Overall, these limitations mean that although the findings provide information about the pruning variability under controlled conditions, further research is needed to understand how these effects behave with missing data, more varied data structures, and other modeling methods.

6.3 Future Research

Building on the findings of this thesis, future research could address several important areas. First, future research could apply the variability analysis to real-world meta-analytic datasets. This could involve selecting datasets that differ in various characteristics, such as heterogeneity, sample size, and moderator type, to test whether the effects observed in the simulations hold in practice. Another option is to run new simulations based on the structure of existing datasets, rather than generating data randomly. Both approaches would help evaluate how pruning variability behaves outside controlled settings and whether the patterns generalize. Doing this, also addresses the external validity concerns mentioned before, and could reveal additional factors influencing variability that were not captured in the current design. Next, the ANOVA analysis showed zero inflation in the entropy values, which caused heteroscedasticity. Future work could consider alternative metrics that are less affected by the zeros, or future work could modify the analysis to account for this feature. For example, it could use zero-inflated models, or apply transformations. Additionally, the discretization of continuous moderators reduced the correlation strength, which may have affected model behavior. Although this discretization was necessary in the current design, it introduced distortions that could be avoided by using methods that model categorical moderators directly. Comparing different approaches to discretization, or avoiding it, could help clarify how data processing decisions affect pruning results. Next, another

issue observed in this study was that the fully grown decision trees often failed to recover the true underlying data structure. This raises concerns about whether pruning variability is being measured relative to an accurate reference. Future research could explore strategies to improve the reliability of the fully grown trees, such as adjusting the splitting criteria or increasing the sample size, to ensure the initial model captures the true underlying structure in the data. In addition, it could be useful to investigate whether the correctness of the initial tree influences the pruning variability. For example, if the starting tree is already incorrect, the pruning behavior might show more variation.

All in all, these suggestions can extend the current findings and test how pruning variability is affected by factors that are common in practice but not included in this simulation design.

6.4 Conclusion

In this thesis we investigated whether pruning variability caused by cross-validation is influenced by characteristics of the data and pruning strictness in random-effect meta-CART models. The results showed that pruning variability depended on interactions between multiple factors, including effect size strength, moderator type, correlation, and pruning strictness. While decision trees offer a useful balance between flexibility and interpretability, their stability remains sensitive to these characteristics. The pruning variability creates challenges for interpretation and reproducibility, specifically in meta-analytic settings. The findings of this thesis emphasize the importance of evaluating this variability. Addressing the limitations of this study and expanding the analysis to real datasets and alternative models, can help clarify when decision tree results can be considered reliable in practice. In the next section, we present and discuss suggestions for how to communicate the pruning variability effectively to users.

Chapter 7

Visualization

The results from this thesis show that pruning variability in random-effects meta-CART, caused by cross-validation, was influenced by several characteristics, including moderator type, effect size strength, correlation, and pruning strictness. In some cases, this variability led to large differences in the resulting tree structures. To make this variability easier to interpret, and help users evaluate their results more clearly, we propose several tools to communicate and visualize the pruning variability. These tools are implemented in R and work as an extension of the `metacart` package. They take a previously fitted RE meta-CART model (`REmrt`) and repeatedly apply automatic cross-validation pruning while recording which nodes are kept across runs. The aim is to show which parts of the tree are stable and which are not, so that users can assess the reliability of individual nodes and splits.

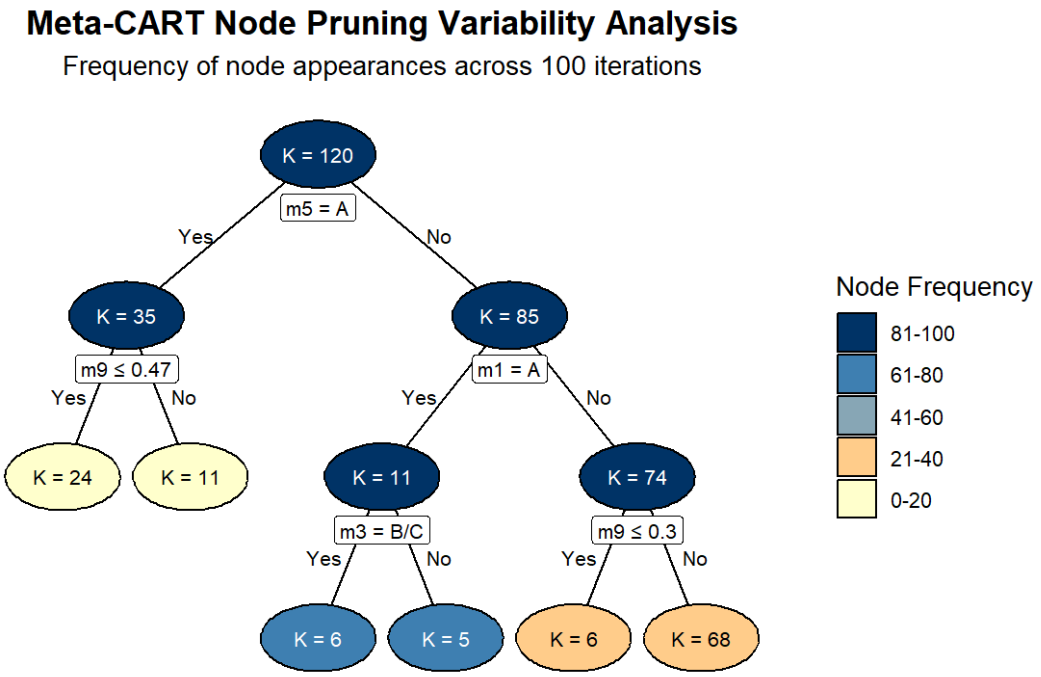
Three types of visualizations are introduced. First, a static plot with node coloring based on how often each node is retained. Second, an interactive tree plot that allows users to explore pruning stability by hovering over the nodes. Third, a plot that shows how often each node appears as a terminal node. The color schemes used for all the visualizations, are designed to be inclusive for most types of color vision deficiencies. In addition, we suggest other strategies for improving transparency, including adding a seed argument to the tree-building functions and showing a summary of the variability using numerical diagnostics. Together, these additions aim to communicate the pruning variability and make the interpretation easier. All code used for the visualizations can be found in Appendix B.

7.1 Node Frequency Visualization

The function `plot.PV()` creates a static plot that shows how frequently each node of the full initial tree is kept in the final pruned trees across repeated cross-validation runs. An example of this plot can be seen in Figure 7.1.

Figure 7.1

Pruning Variability Visualization Showing Node Frequency across Iterations



The function takes three main arguments, namely the fitted model object (`y`), the number of iterations (`iter`), and the pruning strictness value (`c.pruning`).

```
library(metacart)

y <- REmrt(efk ~ m1 + m2 + m3 + m4 + m5 + m6 + m7 + m8 + m9 + m10,
           vi = vark, data = dat, c.pruning = 0)

plot.PV(y, iter = 100, c.pruning = 0)
```

During each iteration, the tree is pruned using a new cross-validation split. The function then tracks which nodes are retained and colors them based on how often they are kept across the iterations. A divergent color scale is used, where darker blue indicates nodes that are often retained and lighter yellow shows nodes that are frequently pruned. Although the scale is continuous, the plot divides the values into five bins to make the legend easier to read. This helps users interpret the plot more quickly, especially when the tree contains many nodes. The number of bins was set to five to keep the output clear and easy to interpret. More bins result in small color differences, which could be harder to distinguish, especially when there are many nodes. Fewer bins would

reduce the amount of detail and could make it harder to distinguish different levels of variation. Five bins provide a balance by simplifying the color scale without losing too much information. Furthermore, the color scheme was chosen to ensure good contrast between different levels and to be accessible for users with different types of color vision deficiency. Next, all nodes are shown as ovals, matching the default plotting style in `metacart`, so the output remains familiar to users of the package. This visualization helps identify which parts of the initial tree tend to remain stable across repeated pruning.

7.2 Interactive Node Frequency Visualization

The `plot.PV.I()` function builds on `plot.PV()` by providing an interactive version that allows users to explore pruning variability in a more interactive way. This function uses the `visNetwork` R package for constructing and visualizing the tree (Thieurmel et al., 2022). An example of this plot is shown in Figure 7.2.

Figure 7.2

Interactive Pruning Variability Visualization Showing Node Frequency across Iterations

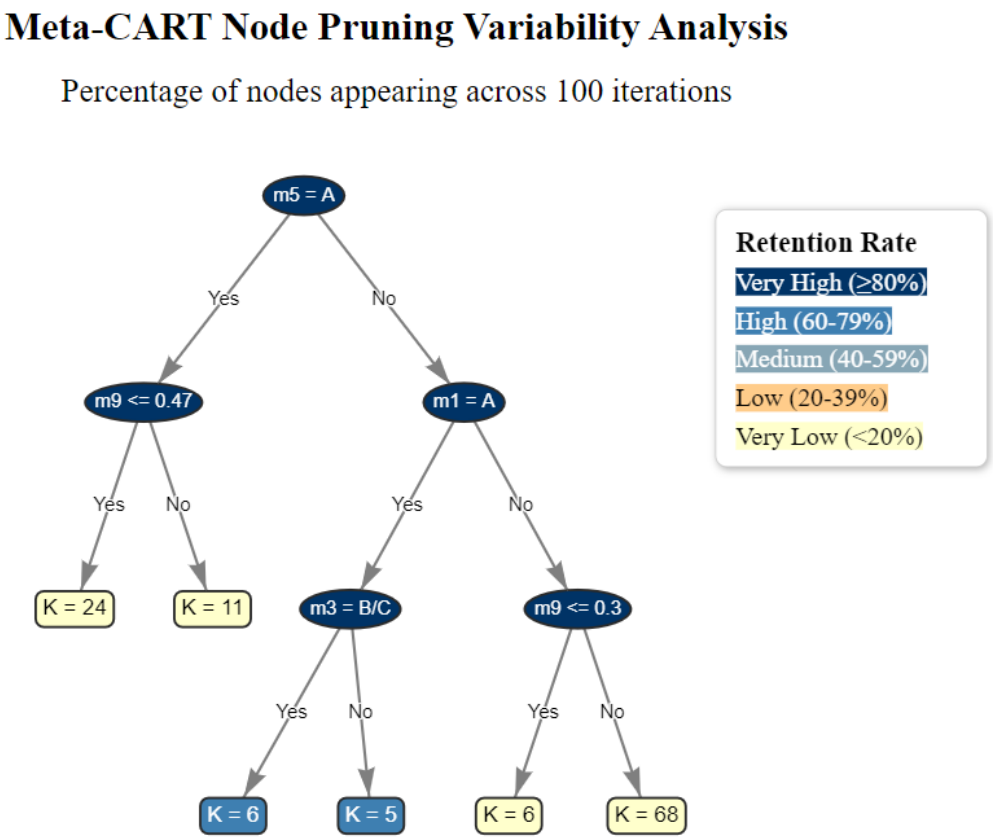
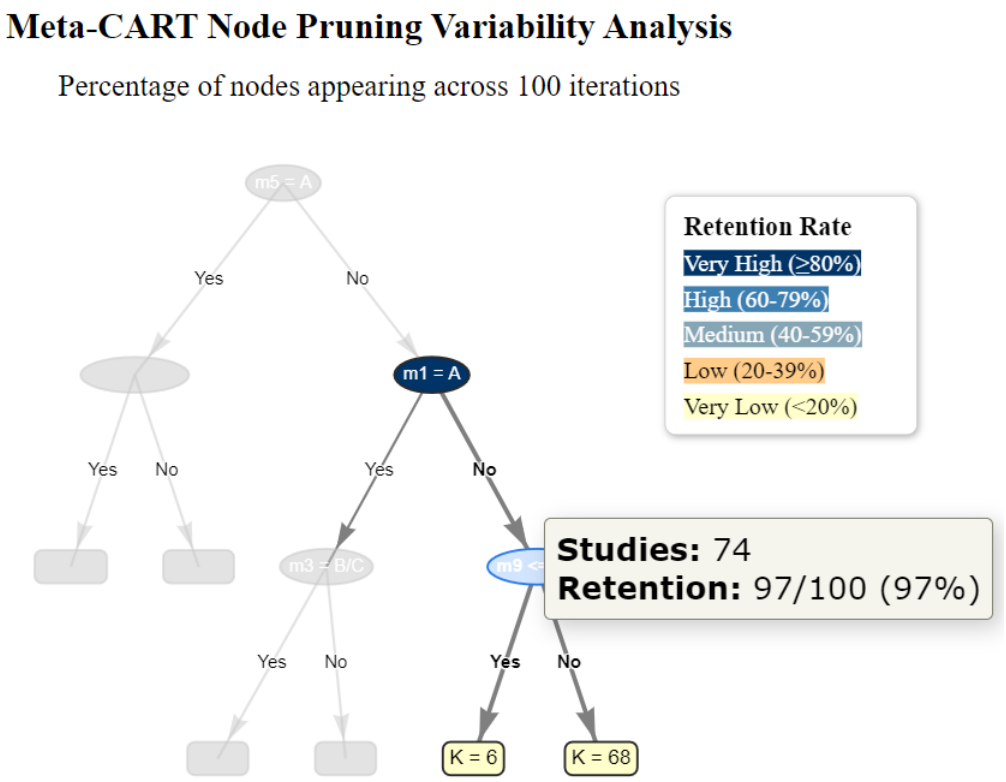


Figure 7.3

Interactive Pruning Variability Visualization Showing Node Frequency across Iterations



Like the static plot, the function requires a fitted model (`y`), number of iterations (`iter`), and pruning strictness value (`c.pruning`).

```
plot.PV.I(y, iter = 100, c.pruning = 0)
```

The function again performs repeated cross-validation and tracks how often each node in the initial tree is retained. The resulting plot lets users hover over nodes to see extra information, including how often a node was kept and the sample size in that node. Figure 7.3 shows how this hover functionality works. This is especially helpful in larger trees, where pruning behavior can be harder to summarize by just looking at the plot. Node color is based on the percentage of times a node was retained, using a divergent color scale that goes from light yellow (low retention) to dark blue (high retention). The percentages are again grouped into five bins, for the same reasons explained earlier. The colors were also chosen with accessibility in mind, as mentioned before. Unlike the static plot function, this visualization is built with the `visNetwork` package, not with the `metacart` code. Because of that, the node shapes follow common decision tree conventions where the splitting nodes are shown as ovals, and terminal nodes are shown

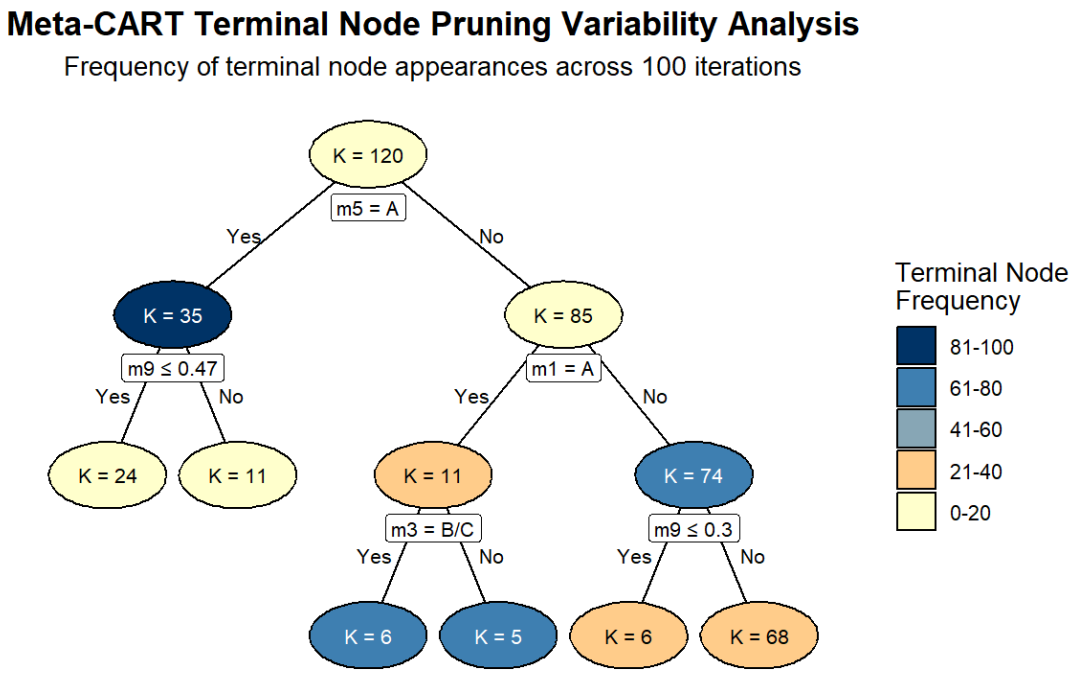
as rectangles. This makes it easier to distinguish between them when exploring the tree. This visualization is useful for seeing which parts of the tree are kept most often and for finding unstable nodes.

7.3 Terminal Node Frequency Visualization

The `plot.PV.TN()` function shows how often each node appears as a terminal node across pruning iterations. An example of this visualization is shown in Figure 7.4. This provides a different perspective compared to the other plots by focusing specifically on the terminal nodes of the tree structures. As with the other visualizations, the function takes a fitted model (`y`), number of iterations (`iter`), and pruning strictness value (`c.pruning`).

```
plot.PV.TN(y, iter = 100, c.pruning = 0)
```

Figure 7.4
Pruning Variability Visualization Showing Terminal Node Frequency across Iterations



In each iteration, the function records which nodes end up as terminal nodes. The resulting plot uses the same divergent color scale as the first plot function (`plot.PV`) in order to show how often each node is a terminal node in the pruned trees. The frequencies are again, proportional

to the number of iterations, divided into five bins. Nodes that often appear as terminal nodes are shown in darker blue, while less frequent nodes are shown in lighter yellow. All nodes are displayed as ovals, using the same format as the first plot function. This ensures consistency with the default `metacart` style. The only difference from the first plot function is that this plot emphasizes how often nodes are terminal nodes, rather than how often they are included in the tree in general. This helps users see which nodes of the tree are often terminal nodes.

7.4 Pruning Variability Diagnostics

The visualizations above are useful for assessing pruning variability, but they require looking at plots. For a quicker summary, we suggest communicating numerical diagnostics that summarize how stable the pruned trees are across repeated runs. One useful metric is the Shannon entropy of the terminal node distribution, which we used before. The entropy captures how consistently the same nodes end up as terminal nodes. Low entropy (below 0.5) suggests high consistency, moderate entropy (0.5 to 1.5) indicates moderate variability, and high entropy (above 1.5) means that the pruned tree structure varies considerably between iterations. In addition, communicating the mean number of terminal nodes provides a sense of the average tree size, while the standard deviation gives an idea of how much this size fluctuates. These values can be obtained using the `summary.PV()` function.

```
summary.PV(y, iter = 100, c.pruning = 0)

Pruning Variability Diagnostics
-------------------------------
Entropy of terminal node distribution: 1.76 (high)
Mean number of terminal nodes: 4.0
SD of terminal nodes across runs: 0.9
```

This output can help users decide whether the pruning strictness value is reasonable, whether a fixed seed might be useful for reproducibility, or whether the results should be interpreted with caution due to an unstable tree structure. Since these diagnostics involve concepts like entropy, they are especially useful for advanced users who are familiar with these diagnostics.

7.5 Reproducibility Through Seed

Finally, to make pruning reproducible, we suggest adding a `seed` argument to the `REmrt` function.

```
y <- REmrt(efk ~ m1 + m2 + m3 + m4 + m5 + m6 + m7 + m8 + m9 + m10,
vi = vark, data = dat, c.pruning = 0, seed = 123)
```

This controls randomness in the pruning step, so repeated runs return the same pruned tree. This does not affect the full initial tree but ensures that pruning results are consistent across runs. This is helpful for reproducibility or when tables or plots have to be generated for reports.

Bibliography

Alsakran, J., Huang, X., Zhao, Y., Yang, J., & Fast, K. (2014). Using entropy-related measures in categorical data visualization. *2014 IEEE Pacific Visualization Symposium*, 135–142. https://doi.org/10.1109/pacificvis.2014.43

Archer, E., Park, I., & Pillow, J. (2013). Bayesian entropy estimation for countable discrete distributions. *Journal of Machine Learning Research*, *15*.

Borenstein, M., Hedges, L. V., Higgins, J. P., & Rothstein, H. R. (2009). *Introduction to meta-analysis*. John Wiley & Sons.

Borenstein, M., Hedges, L. V., Higgins, J. P., & Rothstein, H. R. (2010). A basic introduction to fixed-effect and random-effects models for meta-analysis. *Research Synthesis Methods*, *1*(2), 97–111. https://doi.org/10.1002/jrsm.12

Breiman, L., Friedman, J., Olshen, R., & Stone, C. J. (1984). *Classification and regression trees* (1st). Chapman; Hall/CRC. https://doi.org/10.1201/9781315139470

Browne, M. W. (2000). Cross-validation methods. *Journal of Mathematical Psychology*, *44*(1), 108–132. https://doi.org/10.1006/jmps.1999.1279

Chaabane, I., Guermazi, R., & Hammami, M. (2020). Enhancing techniques for learning decision trees from imbalanced data. *Advances in Data Analysis and Classification*, *14*, 677–745. https://doi.org/10.1007/s11634-019-00354-x

Chawla, N. V. (2010). Data mining for imbalanced datasets: An overview. In O. Maimon & L. Rokach (Eds.), *Data mining and knowledge discovery handbook* (pp. 875–886). Springer. https://doi.org/10.1007/978-0-387-09823-4_45

Cieslak, D. A., & Chawla, N. V. (2008). Learning decision trees for unbalanced data. *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2008)*, *5211*, 241–256. https://doi.org/10.1007/978-3-540-87479-9_34

Consumer Reports. (1990). The 1990 cars. *Consumer Reports*, *55*(4), 235–288.

Dettori, J. R., Norvell, D. C., & Chapman, J. R. (2022). Fixed-effect vs random-effects models for meta-analysis: 3 points to consider. *Global Spine Journal*, *12*(7), 1624–1626. https://doi.org/10.1177/21925682221110527

Doyle, P. (1973). The use of automatic interaction detector and similar search procedures. *Operational Research Quarterly*, *24*, 465–467.

Dusseldorp, E., Conversano, C., & Van Os, B. J. (2010). Combining an additive and tree-based regression model simultaneously: Stima. *Journal of Computational and Graphical Statistics*, *19*(3), 514–530. https://doi.org/10.1198/jcgs.2010.09060

Dusseldorp, E., van Genugten, L., van Buuren, S., Verheijden, M. W., & van Empelen, P. (2014). Combinations of techniques that effectively change health behavior: Evidence from meta-cart analysis. *Health Psychology*, *33*(12), 1530–1540. https://doi.org/10.1037/hea0000018

Erez, A., Bloom, M. C., & Wells, M. T. (1996). Using random rather than fixed effects models in meta-analysis: Implications for situational specificity and validity generalization. *Personnel Psychology*, *49*(2), 275–306.

Finch, W. H., Chang, M., Davis, A. S., Holden, J. E., Rothlisberg, B. A., & McIntosh, D. E. (2011). The prediction of intelligence in preschool children using alternative models to regression. *Behavior Research Methods*, *43*(4), 942–952. https://doi.org/10.3758/s13428-011-0102-z

Fürnkranz, J. (1997). Pruning algorithms for rule learning. *Machine Learning*, *27*(2), 139–172. https://doi.org/10.1023/A:1007329424533

Genz, A., & Bretz, F. (2009). *Computation of multivariate normal and t probabilities*. Springer-Verlag.

Geurts, P., Ernst, L., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42.

Gey, S., & Nedelec, E. (2005). Model selection for cart regression trees. *IEEE Transactions on Information Theory*, *51*(2), 658–670. https://doi.org/10.1109/TIT.2004.840903

Gilli, M., Maringer, D., & Schumann, E. (2011). *Numerical methods and optimization in finance*. Academic Press.

Hedges, L. V., & Olkin, I. (1985). *Statistical methods for meta-analysis*. Academic Press.

Isaksson, A., Wallman, M., Goransson Kultima, H., & Gustafsson, M. (2008). Cross-validation and bootstrapping are unreliable in small sample classification. *Pattern Recognition Letters*, *29*, 1960–1965. https://doi.org/10.1016/j.patrec.2008.06.018

Israel, H., & Richter, R. R. (2011). A guide to understanding meta-analysis. *Journal of Orthopaedic & Sports Physical Therapy*, *41*(7), 496–504. https://doi.org/10.2519/jospt.2011.3333

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: With applications in r*. Springer. https://www.statlearning.com

Kim, J. W., Lee, B. H., Shaw, M. J., Chang, H.-L., & Nelson, M. (2001). Application of decision-tree induction techniques to personalized advertisements on internet storefronts. *International Journal of Electronic Commerce*, *5*(3), 45–62. https://doi.org/10.1080/10864415.2001.11044215

Leach, H. J., O'Connor, D. P., Simpson, R. J., Rifai, H. S., Mama, S. K., & Lee, R. E. (2016). An exploratory decision tree analysis to predict cardiovascular disease risk in african american women. *Health Psychology*, *35*(4), 397–402. https://doi.org/10.1037/hea0000267

Li, X., Dusseldorp, E., Su, X., & Meulman, J. (2020a). Multiple moderator meta-analysis using the r-package meta-cart. *Behavior Research Methods*, *52*. https://doi.org/10.3758/s13428-020-01360-0

Li, X., Dusseldorp, E., Claramunt González, J., Su, X., van Megen, J., & Meulman, J. J. (2025). *Enhanced tree-based subgroup identification in meta-analysis* [Manuscript submitted for publication].

Li, X., Dusseldorp, E., Liu, K., Claramunt, J., & Meulman, J. (2020b). *Metacart: Meta-cart: A flexible approach to identify moderators in meta-analysis* [R package version 2.0-3]. https://CRAN.R-project.org/package=metacart

Liu, W., Chawla, S., Cieslak, D. A., & Chawla, N. V. (2010). A robust decision tree algorithm for imbalanced data sets. *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM)*, 766–777. https://doi.org/10.1137/1.9781611972801.67

Loh, W.-Y. (2002). Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica*, *12*(2), 361–386. https://www.jstor.org/stable/24306967

Loh, W.-Y. (2014). Fifty years of classification and regression trees. *International statistical review*, *82*(3), 329–348. https://doi.org/10.1111/insr.12016

Loh, W.-Y., & Shih, Y.-S. (1999). Split selection methods for classification trees. *Statistica Sinica*, *7*.

Mienye, D., & Jere, N. (2024). A survey of decision trees: Concepts, algorithms, and applications. *IEEE Access*, *PP*, 1–1. https://doi.org/10.1109/ACCESS.2024.3416838

Podgorelec, V., Kokol, P., Stiglic, B., & Rozman, I. (2002). Decision trees: An overview and their use in medicine. *Journal of Medical Systems*, *26*(5), 445–463. https://doi.org/10.1023/A:1016409317640

R Core Team. (2024). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/

Saraiva, P. (2023). On shannon entropy and its applications. *Kuwait Journal of Science*, *50*(3), 194–199. https://doi.org/10.1016/j.kjs.2023.05.004

Sarker, I. H., Colman, A., Han, J., & Van, H. (2020). Behavdt: A behavioral decision tree learning to build user-centric context-aware predictive model. *Mobile Networks and Applications*, *25*, 1151–1161. https://doi.org/10.1007/s11036-019-01443-z

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, *27*(3), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Shih, Y.-S. (2004). A note on split selection bias in classification trees. *Computational Statistics & Data Analysis*, *45*(3), 457–466. https://doi.org/10.1016/S0167-9473(03)00064-1

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., & Zeileis, A. (2008). Conditional variable importance for random forests. *BMC bioinformatics*, *9*, 307. https://doi.org/10.1186/1471-2105-9-307

Therneau, T. M., Atkinson, B. J., & Ripley, B. (2023, December 4). *Recursive partitioning and regression trees* [Producer of the initial R port: Brian Ripley (maintainer 1999-2017)]. Version 4.1.23. https://cran.r-project.org/package=rpart

Therneau, T. M., & Atkinson, E. J. (2023, December). *An introduction to recursive partitioning using the rpart routines*. Mayo Foundation. https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf

Thieurmel, B., B.V., A., & Contributors. (2022). *Visnetwork: Network visualization using 'vis.js' library* [R package version 2.1.2]. https://CRAN.R-project.org/package=visNetwork

Tipton, E., Pustejovsky, J. E., & Ahmadi, H. (2019). A history of meta-regression: Technical, conceptual, and practical developments between 1974 and 2018. *Research Synthesis Methods*, *10*(2), 161–179. https://doi.org/10.1002/jrsm.1338

Trujillano, J., Badia, M., Serviá, L., March, J., & Rodriguez-Pozo, A. (2009). Stratification of the severity of critically ill patients with classification trees. *BMC Medical Research Methodology*, *9*, 83. https://doi.org/10.1186/1471-2288-9-83

Valentine, J. C., Aloe, A. M., & Lau, T. S. (2015). Life after nhst: How to describe your data without "p-ing" everywhere. *Basic and Applied Social Psychology*, *37*(5), 260–273. https://doi.org/10.1080/01973533.2015.1060240

Veroniki, A. A., Jackson, D., Viechtbauer, W., Bender, R., Bowden, J., Knapp, G., Kuss, O., Higgins, J. P., Langan, D., & Salanti, G. (2016). Methods to estimate the between-study variance and its uncertainty in meta-analysis. *Research Synthesis Methods*, *7*(1), 55–79. https://doi.org/10.1002/jrsm.1164

Viechtbauer, W. (2007). Accounting for heterogeneity via random-effects models and moderator analyses in meta-analysis. *Zeitschrift für Psychologie/Journal of Psychology*, *215*(2), 104–121. https://doi.org/10.1027/0044-3409.215.2.104

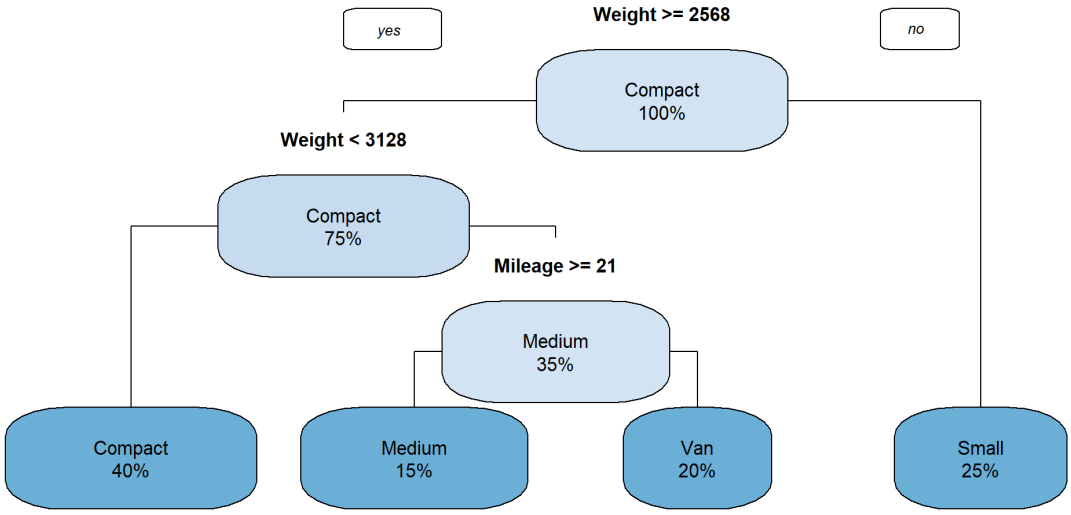
Wickham, H. (2016). *Ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org

A Classification Tree Example

Imagine we want to predict the type of a car, a categorical variable with categories small, sporty, compact, medium, large, and van, based on various variables from the Automobile Data dataset we used previously (Consumer Reports, 1990). Using a classification tree, we partition the data into distinct groups using the variables of the dataset, allowing us to predict car types by identifying shared characteristics within each group. An example of such a tree is shown in Figure A.1. Each terminal node represents the predicted car type for the corresponding group of cars, while the percentages indicate the proportion of observations in each node relative to the total dataset. Below, we describe the process of constructing the tree.

Figure A.1

Example of a Classification Tree for Predicting Car Types



Note. Terminal nodes show the predicted car type, with percentages indicating the proportion of observations in each node.

The classification tree algorithm begins with all observations in a single group, represented by

the root node. At this stage, the total impurity, measured by the Gini index, is calculated. The Gini index for the root node is 0.75, representing the overall impurity of the car types within the group. The tree then proceeds to find the optimal feature and threshold for splitting the data. The first split is based on weight, with a threshold of 2,567.5 pounds. Cars under 2,567.5 pounds are directed to one node, while cars equal to or above that weight go to another node. The split generates new nodes with Gini indexes of 0.67 for the left node containing 45 observations and 0.13 for the right node containing 15 observations, indicating a substantial reduction in impurity.

The left node shows a Gini index value of 0.67, suggesting moderate impurity and predicts compact as the class. This node is further split based on weight at a threshold of 3,127.5 pounds. The weight subset under 3,127.5 pounds (24 observations), shows a Gini index reduction to 0.375, while keeping compact as the predicted class. The Gini index is 0.57 for vehicles that weigh 3,127.5 pounds or more, which demonstrates moderate impurity, while the model predicts medium as the class. The next split examines mileage, using 20.5 miles per gallon as the dividing point. The dataset shows that vehicles with mileage greater than or equal to 20.5 miles per gallon (9 observations), fall under the medium category according to the model, with a Gini index measuring impurity at 0.11. The model assigns cars with mileage smaller than 20.5 (12 observations) to the van class, while showing a moderate Gini index of 0.42.

Hence, the Gini index at each tree node demonstrates the effectiveness of data splits by producing more homogeneous groups. Each data split shows a decline in the Gini index, proving that decision rules effectively reduce the impurity. The tree used default stopping criteria from the `rpart` R package (Therneau et al., 2023). Concluding, to predict the type of a new car, the variables weight, mileage, and country of origin, are used to navigate through the tree to a terminal node. The resulting classification tree (Figure A.1) provides a clear and interpretable approach to understanding how these factors influence car types.

B Code Availability

All the R code used in this thesis is available at https://github.com/madeliefeliss/ThesisCode. This includes the data simulation, pruning variability analysis, and visualization code.