TREES

Applied Analytics: Frameworks and Methods 1

Outline

- Definition
- Regression Trees
  - *Illustration*
  - *Tree building process*
  - *Tree Pruning*
  - *Variable importance*
- Classification Trees
  - *Illustration*
  - *Tree algorithm*
- Strengths and Weaknesses of Trees

Why Trees? Why not Regression?

- Regression models are well-tested, extensively studied, robust and work even with moderate violations of the assumptions
- However, with large number of variables regression becomes difficult to use when
  - *Relationships are non-linear*
  - *Variables interact*

Decision Trees

- Trees are particularly useful when
  - *using large number of independent variables and*
  - *when there are likely to be non-linear relationships and interactions amongst variables.*
  - *And when interpretability of results is important*
- Trees require minimal preprocessing of data
- At the same time, trees often
  - *perform poorly in predicting relative to the best supervised learning approaches*
  - *tend to overfit the data*

Trees

- A Tree is a predictive model that involves stratifying or segmenting the predictor space into a number of simple regions.
- Predictions are based on a summary statistic of the outcome in a given region.
- It is called a Decision Tree because the set of splitting rules used to segment the predictor space can be summarized in a tree.

Types

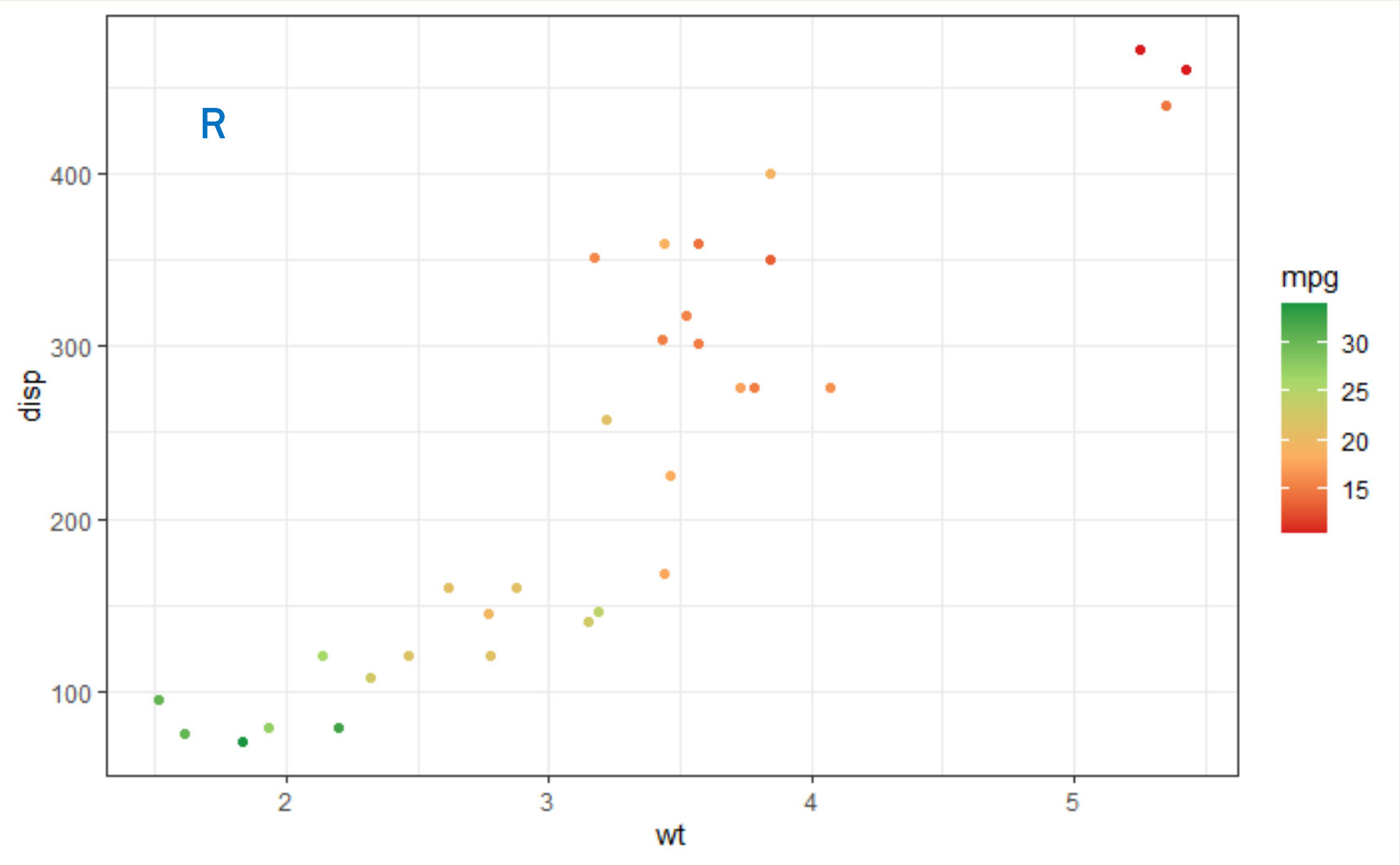
- Based on the nature of the outcome variable, trees can be categorized as
  - *Regression Trees: Outcome is continuous (e.g., price in $), or*
  - *Classification Trees: Outcome is categorical (e.g., whether a product is purchased or not)*

REGRESSION TREES

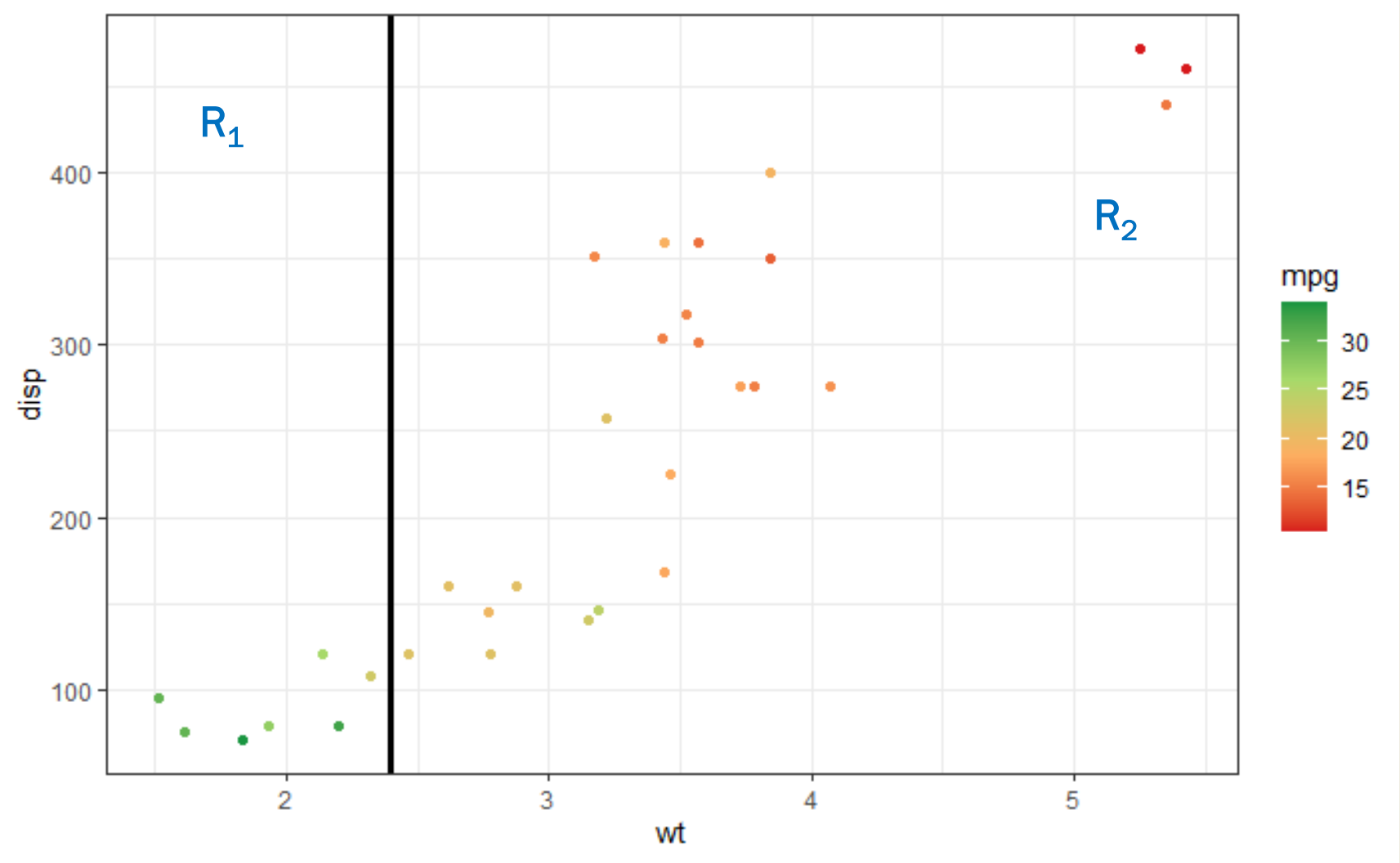
Simple Illustration

- In the next few slides, we will illustrate the process by which a tree models stratify or segment predictor space to make predictions.
- Lets say we are interested in predicting gas mileage for cars (mpg) based on their weight (wt) and displacement in cu. in (disp).

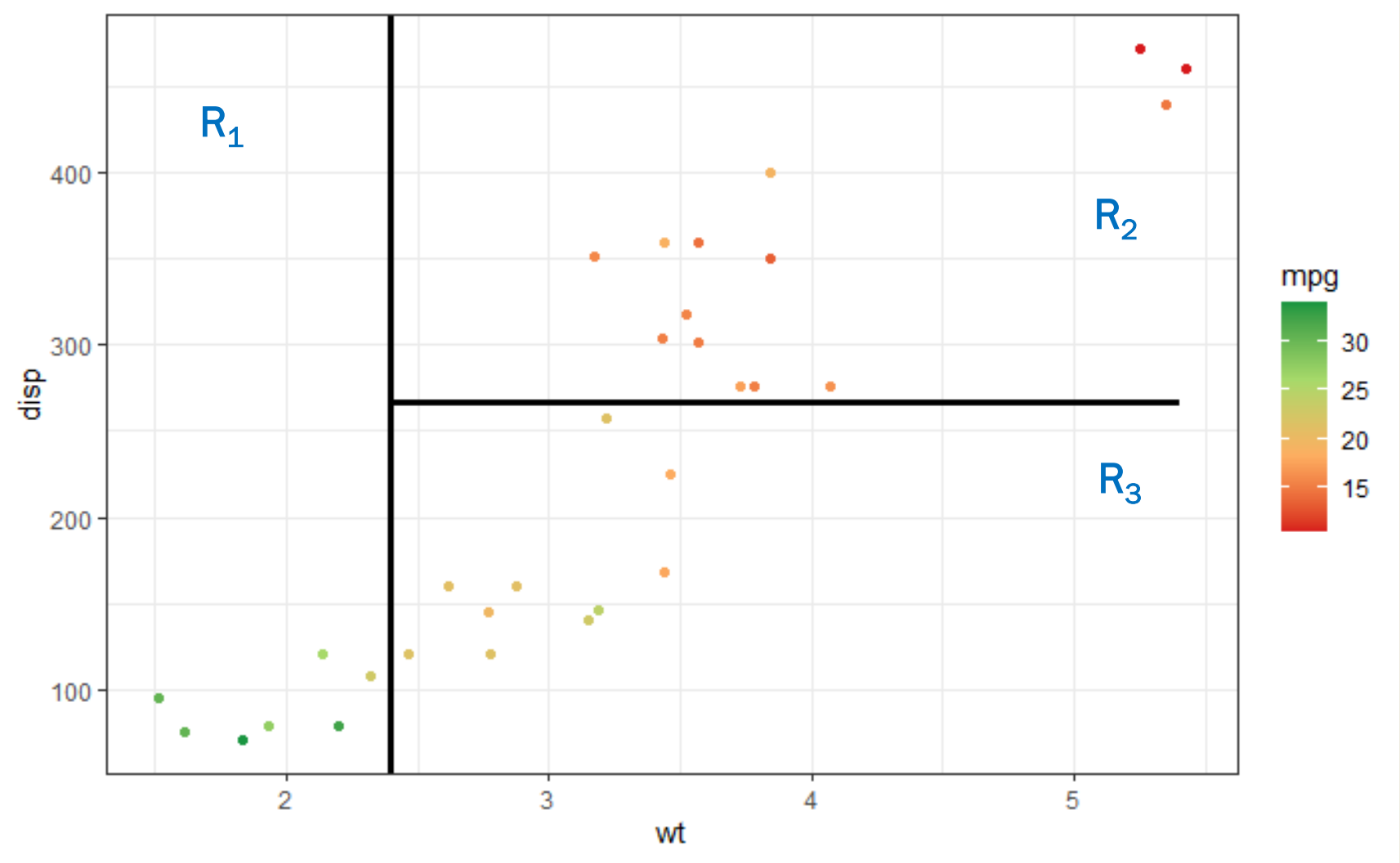
What is the “best” way to partition Predictor Space, R?



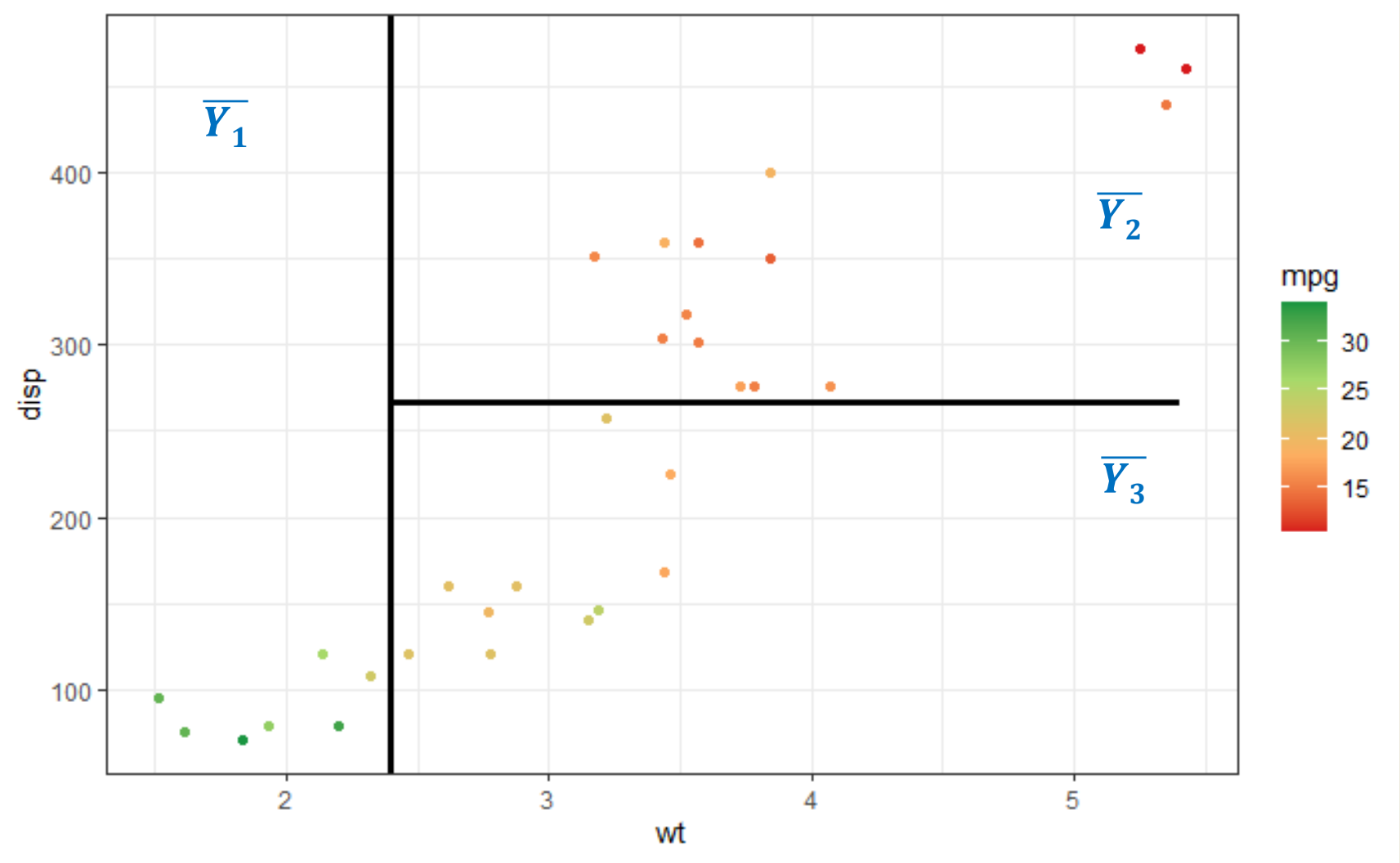
Which of the predictor spaces R_1 or R_2 should be segmented next and where?



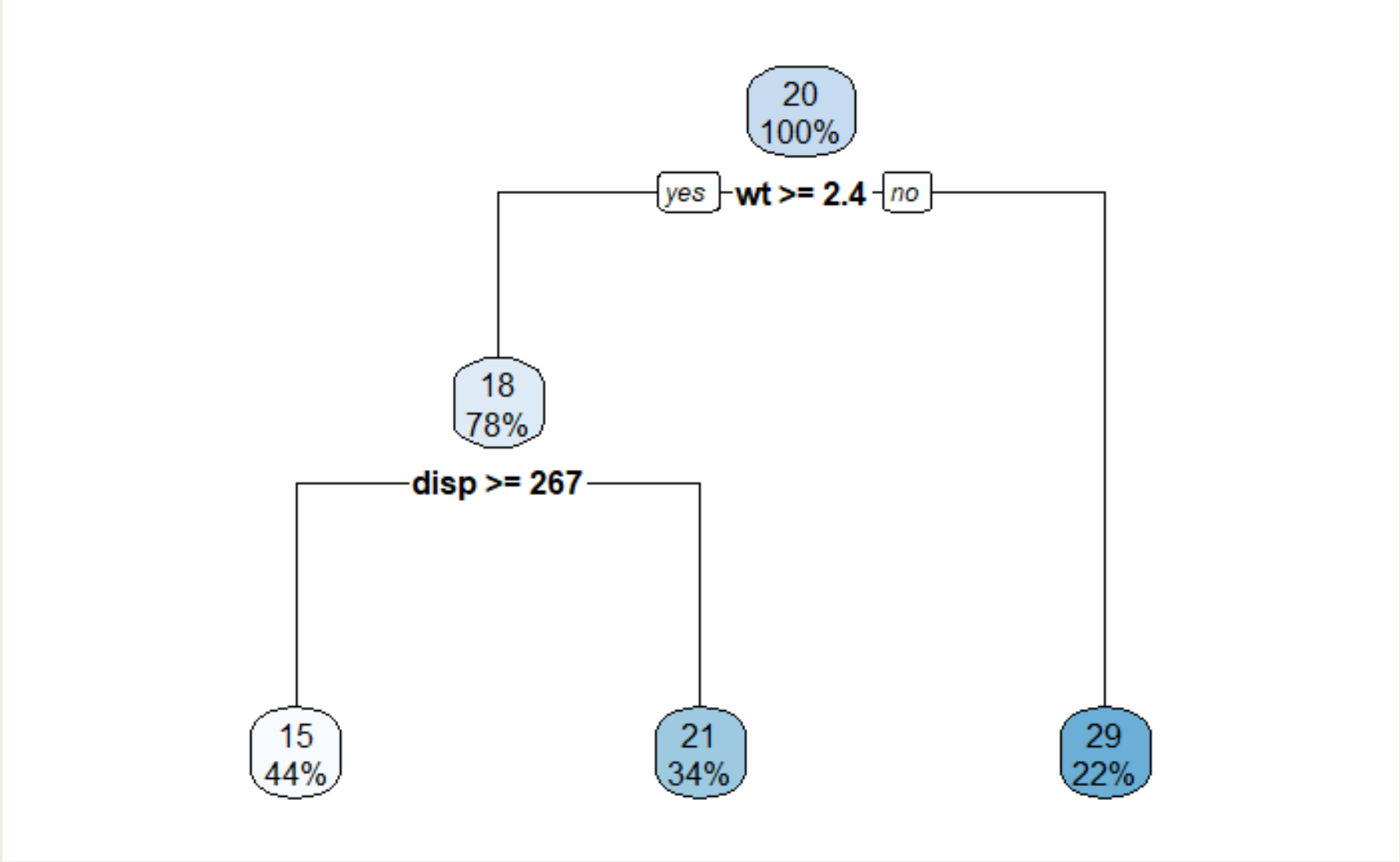
Should we continue segmenting the predictor space of R_1, R_2, and R_3?



What is the predicted outcome for spaces R_1, R_2 and R_3?



Model Representation



Root Node

Interior Node

Leaf or
Terminal Node

Regression Trees

Regression trees partition the data into smaller groups that are more homogeneous with respect to the outcome. There are many approaches to constructing regression trees. One of the oldest and most widely used is the Classification and Regression Tree (CART) methodology.

Tree Building Process

- The predictor space (i.e., set of possible values for $X_1, X_2, \ldots, X_p$) is divided into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
- Every observation that falls into the region R_j , gets the same prediction, which is simply the mean of the response values for the training observations in R_j .

Tree Building Process

- In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or boxes, for simplicity and for ease of interpretation of the resulting predictive model.
- The goal is to find boxes $R_1, \ldots, R_J$ that minimize the RSS, given by

$$\text{SSE} = \sum_{j=1}^{J} \sum_{i \in Rj} (yi - \overline{y_{Rj}})^2$$

  - *where $\overline{y_{Rj}}$ is the mean outcome for training observations in the j^{th} box*

Tree Building Process

- The process begins with the entire data, R, and searches every distinct value of every predictor to find the predictor and split value that partitions the data into two groups (R_1 and R_2) such that the overall sum of squared errors is minimized

$$\text{SSE} = \sum_{i \in R1}(yi - \overline{y_{R1}})^2 + \sum_{i \in R2}(yi - \overline{y_{R2}})^2$$

  - *Where $\overline{y_{R1}}$ and $\overline{y_{R2}}$ are mean training set outcomes within regions R_1 and R_2, respectively*

- Next, the process is repeated, looking for the best predictor and best split value (or cut point) in order to split the data further so as to minimize the SSE within each of the resulting regions.
- However, this time, instead of splitting the entire predictor space, one of the previously identified regions is split. This creates three regions, R_1, R_2 and R_3. Because of the recursive splitting nature of trees, this method is also known as recursive partitioning (and inspiration for the name of the R library, `rpart`).
- This recursive partitioning process continues until a stopping criterion is reached such as number of observations in a region falls below 5.

Computational Shortcuts

- It is computationally infeasible to consider every possible partition of the feature space into J boxes.
- This is the reason for the top-down, greedy approach of recursive binary splitting.
- The approach is top-down because it begins at the top of the tree and then successively splits the predictor space; each split is indicated via two new branches further down on the tree.
- It is greedy because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

Prediction

- In CART models, predicted value is the mean of outcome for a given test observation using the mean of the training observations in the region to which that test observation belongs.

Pruning a Tree

- The process described above may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performance. Why?
- A smaller tree with fewer splits (i.e., fewer regions $R_1, \ldots, R_J$) might lead to lower variance and better interpretation at the cost of a little bias.
- One possible alternative to the process described above is to grow the tree only so long as the decrease in the SSE due to each split exceeds some (high) threshold.
- This strategy will result in smaller trees, but is too short-sighted: a seemingly worthless split early on in the tree might be followed by a very good split — that is, a split that leads to a large reduction in SSE later on.

Pruning a Tree

- A better strategy is to grow a very large tree T_0, and then prune it back in order to obtain a subtree.
- The goal of this process is to find a "right sized tree" that has the smallest error rate.
- Cost complexity pruning — also known as weakest link pruning — is used to do this
- We consider a sequence of trees indexed by a nonnegative tuning parameter, c_p (also known as a complexity parameter). For each value of c_p there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i \in Rm} (yi - \overline{y_{Rm}})^2 + c_p|T|$$

  - *is as small as possible. Here $|T|$ indicates the number of terminal nodes of the tree T, R_m is the rectangle (i.e. the subset of predictor space) corresponding to the m^{th} terminal node, and $\overline{y_{Rm}}$ is the mean of the training observations in R_m.*

Choosing the Best Tree

- The tuning parameter c_p controls a trade-off between the subtree's complexity and its fit to the training data.
- Optimal value of c_p is chosen using using cross-validation.
- We then return to the full data set and obtain the subtree corresponding to c_p.

Performance Measures

- Quality of predictions can be assessed by indices that compare predicted values to true values
- Root mean squared error (RMSE)
  - $RMSE = \sqrt{\frac{\sum_{i=1}^{n}(yi - \hat{y}_i^{\,2})}{n}}$
- Mean squared error (MSE)
  - $MSE = \frac{\sum_{i=1}^{n}(yi - \hat{y}_i^{\,2})}{n}$
- Mean absolute error (MAE)
  - $MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n}$
- RMSE and MSE punish larger errors more than MAE

Predictor Importance

- Relative importance of predictors in trees can be determined by examining the reduction in SSE attributed to each split.
- Intuitively, predictors that appear higher in the tree, or that appear multiple times in the tree will be more important than predictors that appear lower in the tree or not at all

Summary: Tree Algorithm

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of c_p.
3. Use k-fold cross-validation to choose α. For each k = 1, . . . , K:
   a) *Repeat Steps 1 and 2 on the* $\frac{K-1}{K}$ *th fraction of the training data, excluding the kth fold.*
   b) *Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of* c_p.
   c) *Average the results, and pick* c_p *to minimize the average error.*
4. Return the subtree from Step 2 that corresponds to the chosen value of c_p.

See regressionTree.html for a demonstration of Regression Trees in R

CLASSIFICATION TREES

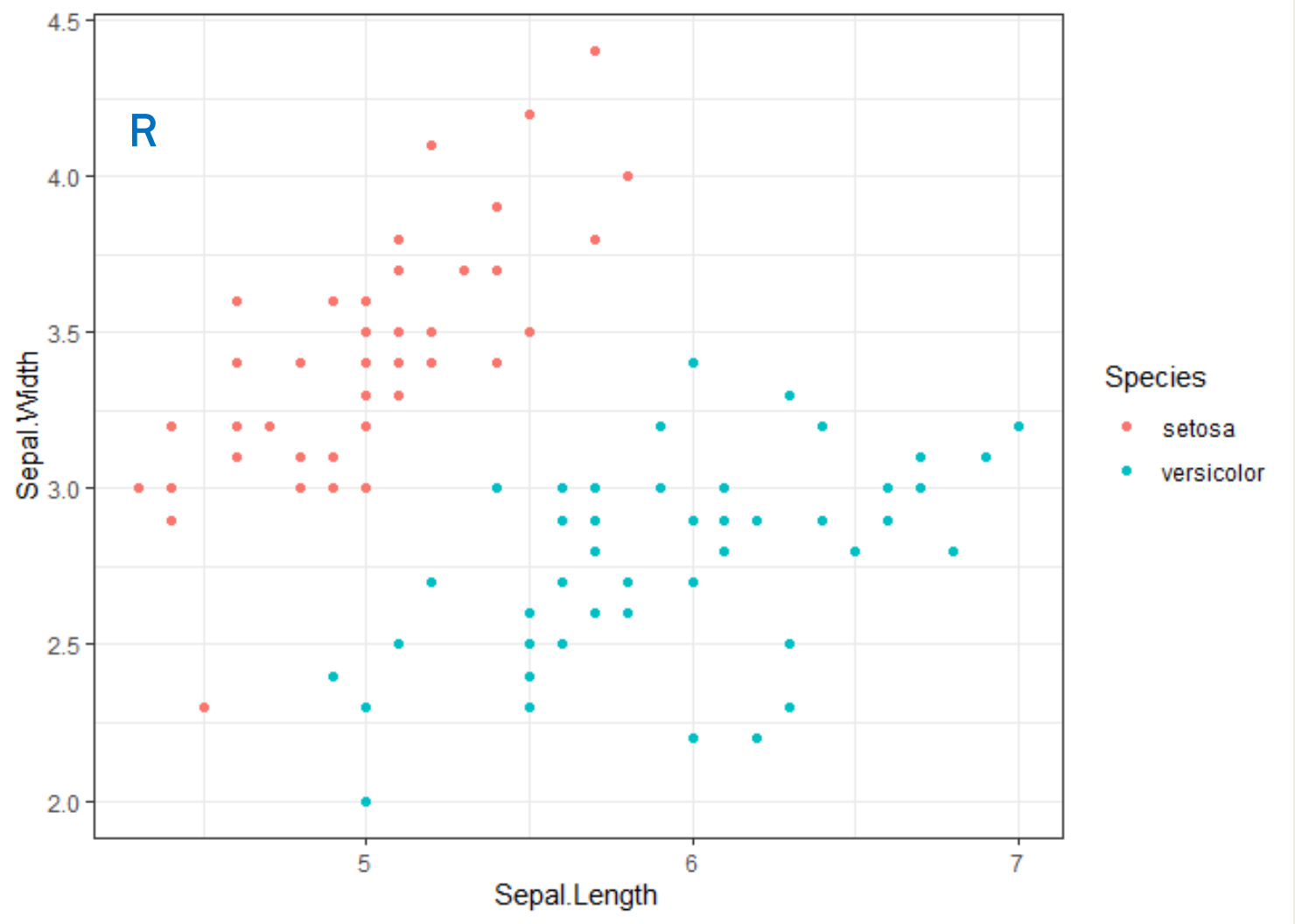
Classification Trees

- Very similar to a regression tree, except that it is used to predict a qualitative response rather than a quantitative one.
- For a classification tree, predicted value is based on the most commonly occurring class of training observations in the region to which it belongs.

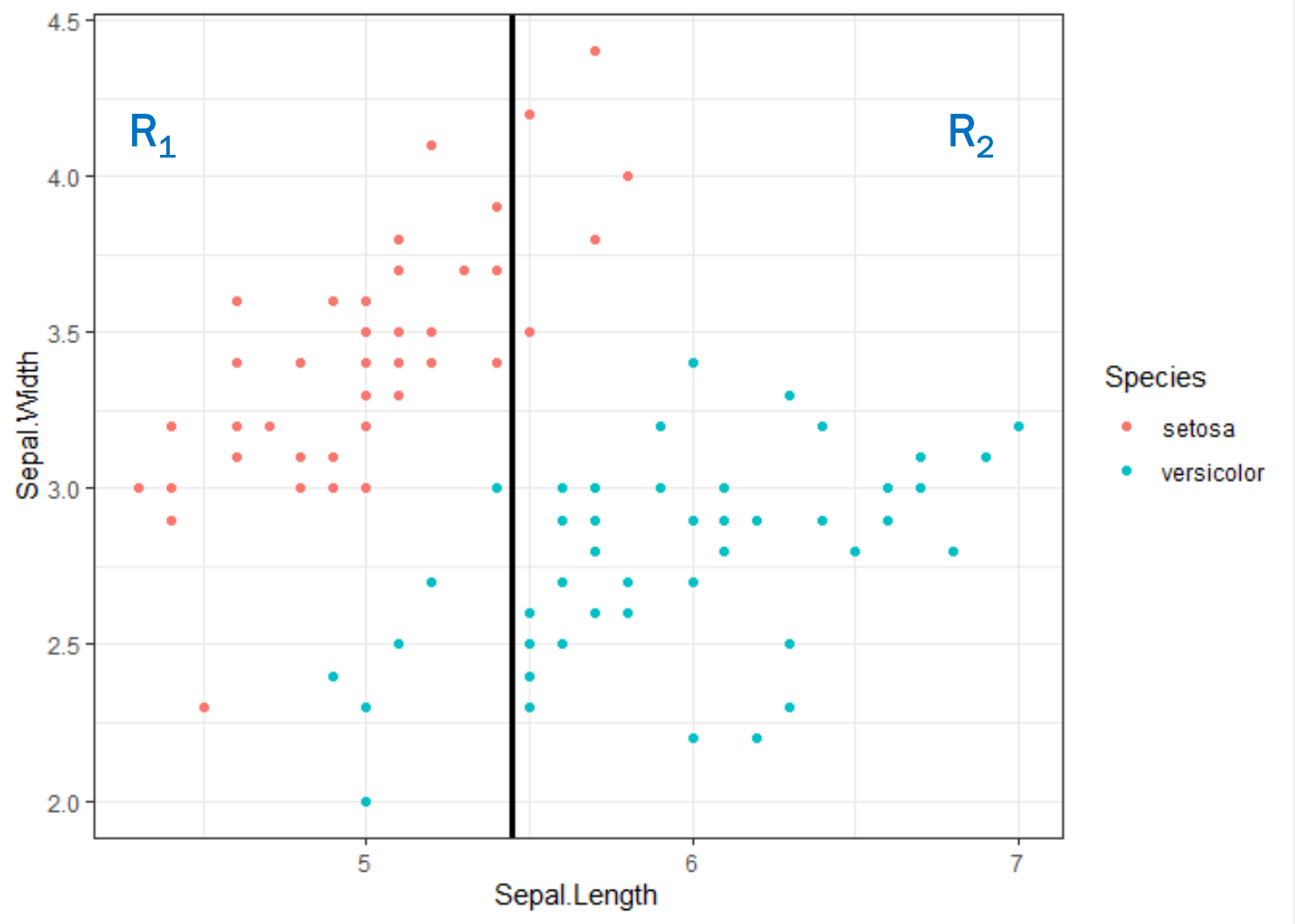
Simple Illustration

- In the next few slides, we illustrate the process by which a classification tree stratifies or segments predictor space to make class predictions.
- Lets say we are interested in predicting flower type (setosa or versicolor) based on sepal length and sepal width.

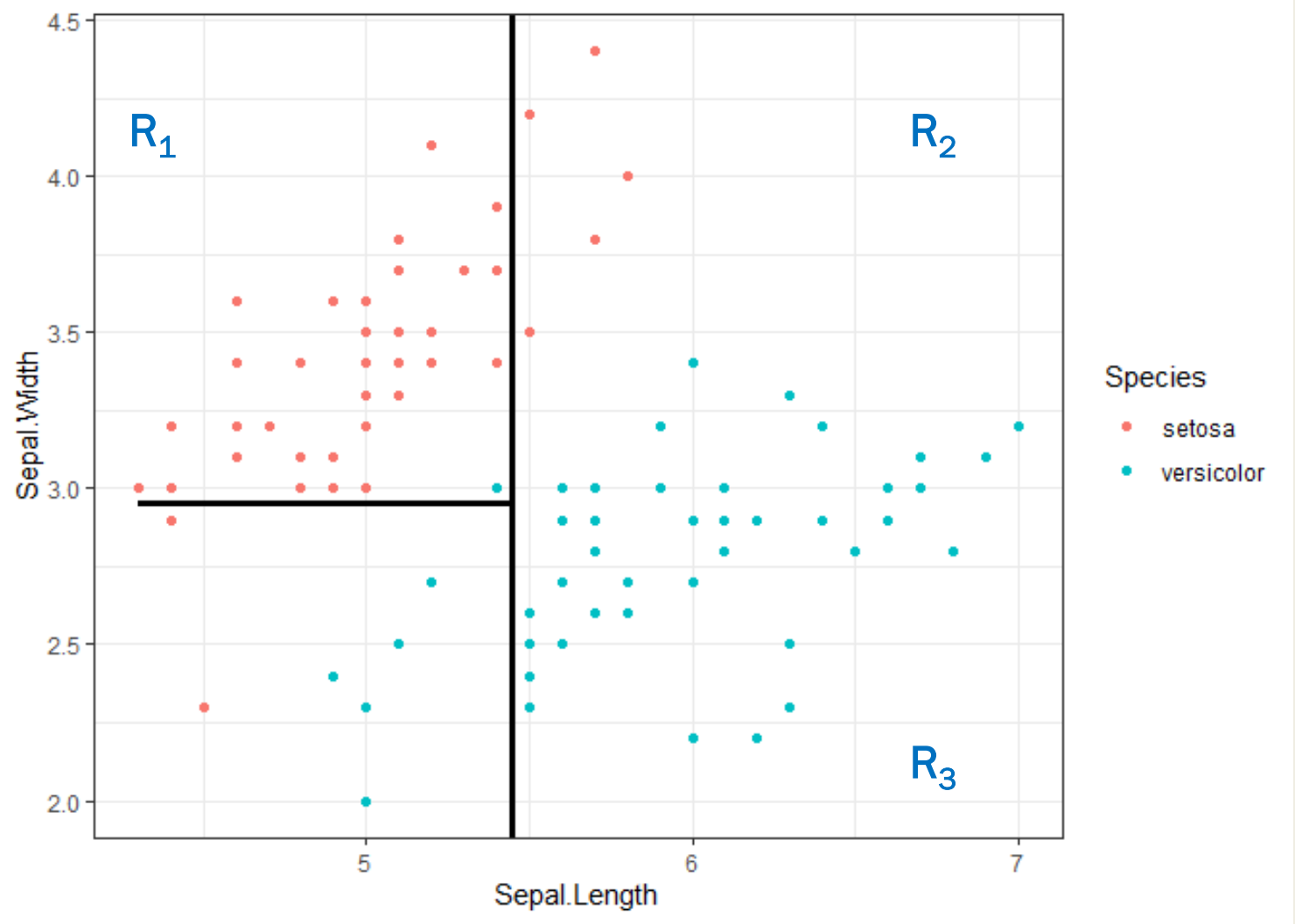
What is the “best” way to partition Predictor Space, R?



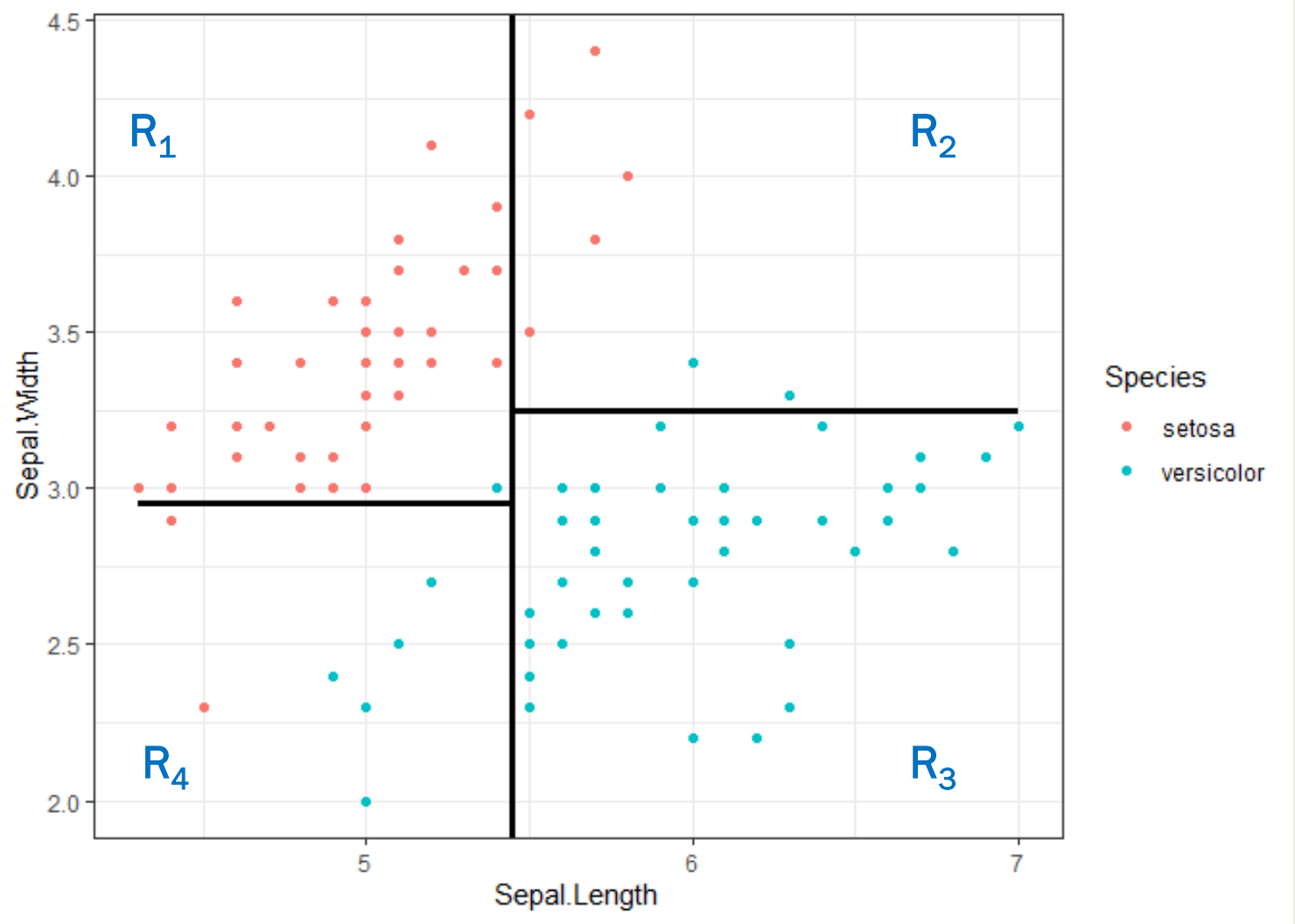
Which of the predictor spaces R_1 or R_2 should be segmented next and where?



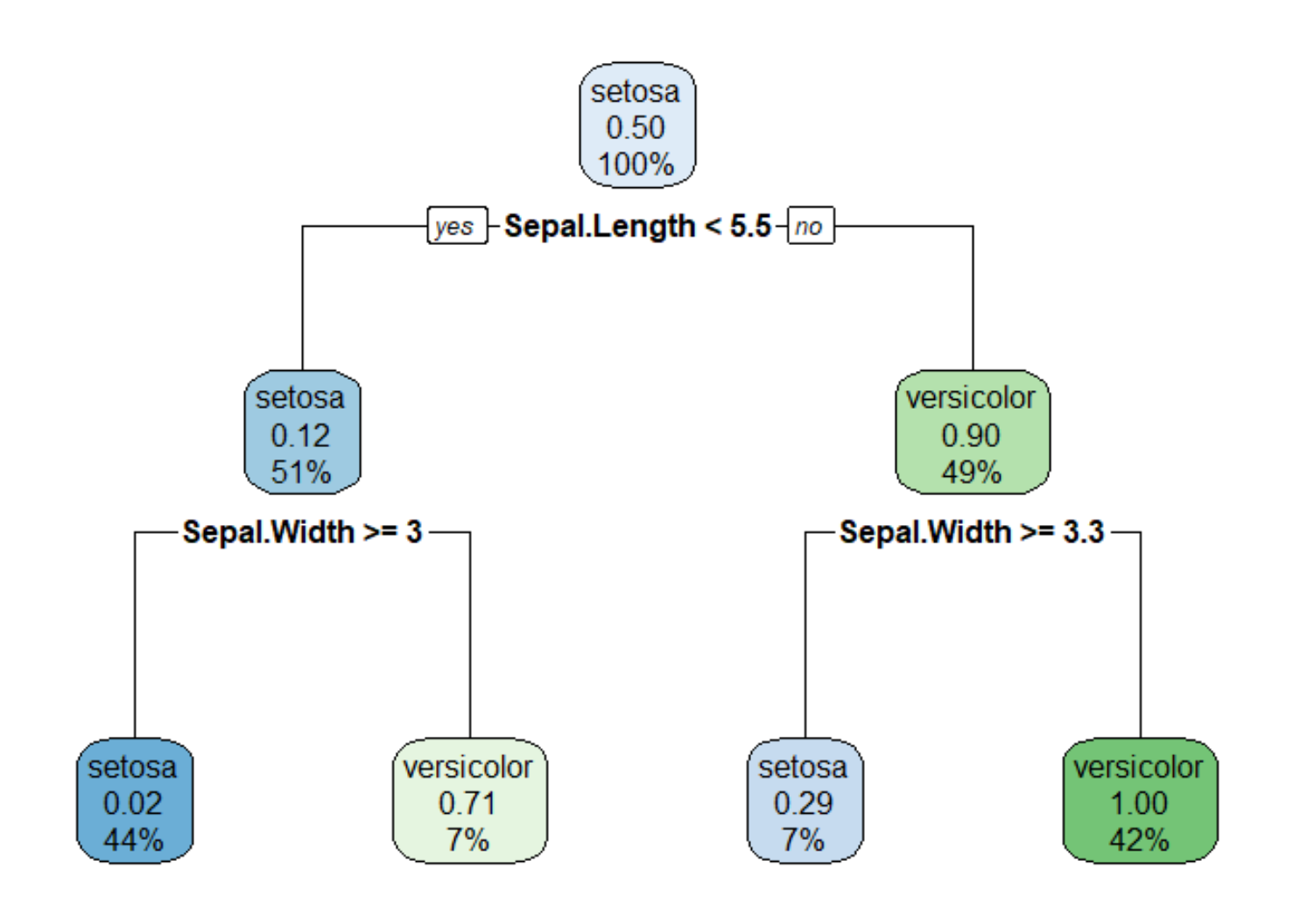
Which of the predictor spaces R_1,R_2, or R_3 should be segmented next and where?



What is the predicted class for each region? How accurate are these predictions?



Model Representation



Root Node

Interior Node

Leaf or
Terminal Node

Classification Tree Algorithm

- Just as in the regression setting, we use recursive binary splitting to grow a classification tree.
- In the classification setting, SSE cannot be used as a criterion for making the binary splits
- A natural alternative to RSS is the misclassification rate, which is simply the fraction of the training observations in that region that do not belong to the most common class:

$$E = 1 - \max(p_{mk})$$

Here p_{mk} represents the proportion of training observations in the mth region that are from the kth class.

- However misclassification rate is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable.

Classification Tree Algorithm

- The Gini index is defined by

  $G = \sum_{k=1}^{K} p_{mk}(1 - pmk)$,

  - *a measure of total variance across the K classes. The Gini index takes on a small value if all of the p_{mk}'s are close to zero or one.*
- For this reason the Gini index is referred to as a measure of node purity — a small value indicates that a node contains predominantly observations from a single class.
- An alternative to the Gini index is cross-entropy, given by

  $G = \sum_{k=1}^{K} p_{mk} log(pmk)$,
- It turns out that the Gini index and the cross-entropy are very similar numerically.

Performance Measures

- Log-Loss
- Classification or Confusion matrix based measures
  - *Accuracy*
  - *Specificity*
  - *Sensitivity*
- AUC

See classificationTree.html for a demonstration of Classification Trees in R

STRENGTHS AND WEAKNESSES

Strengths of Trees

Trees are a popular predictive modeling tool because they

- are easy to interpret and easy to explain to others. The results of a tree model are contained in an inverted tree.
- are easy to implement. Run quite quickly.
- can handle many types of predictors (sparse, skewed, continuous, categorical, etc.) without the need to preprocess them.
- do not require the analyst to specify the form of the predictors' relationship with the outcome (unlike models such as regression). For instance, non-linear relationships can be modeled without data transformation.
- can handle missing data (using surrogate splits)
- implicitly conduct feature selection. If a predictor is never used in a split, then the prediction equation is independent of it. However, this advantage is weakened for highly correlated predictors where the choice of which is used for a split is somewhat random.

Weaknesses of Trees

- Less-than optimal predictive performance.
  - *Tree models partition the predictor space into rectangular regions. If the relationship between predictors and outcome is not adequately described by these rectangles, performance will be poor.*
  - *Number of possible predicted outcomes is finite and is determined by the number of leaves. This limitation is unlikely to capture all of the nuances of the data.*
- Model instability
  - *Slight changes in the data can drastically change the structure of the tree, hence the interpretation. Thus, these models have high variance. Easy to overfit train data.*

Decision trees in R

- Many packages for decision trees including
  - *rpart, tree*
- library(rpart); library(rpart.plot)
- Tuning Parameters to control complexity
  - *control = rpart.control(minsplit, minbucket, maxdepth, cp)*
  - *For more on this try ?rpart.control or see documentation*
- Use method="class" (for classification trees/classification problems). Set method = 'anova' for regression tree.

In closing

- Trees are seldom the best models
- But, they are the simplest to understand
- Trees can be greatly improved by tuning complexity and considering an ensemble of trees

Summary

- In this module, we
  - *defined Trees*
  - *examined regression Trees*
  - *examined classification Trees*
  - *discussed strengths and weaknesses of Trees*