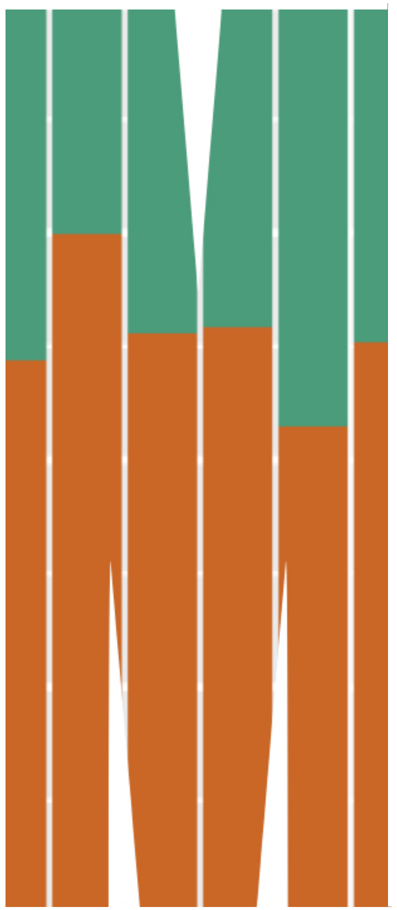


ETC3250/5250: Regression Trees

Semester 1, 2020

Professor Di Cook

Econometrics and Business Statistics
Monash University

Week 6 (b)

Predicting Salary

Using the function `rpart`, we can build a regression tree to predict the `logSalary` of a baseball player, given their `Years` of playing and number of `Hits`.

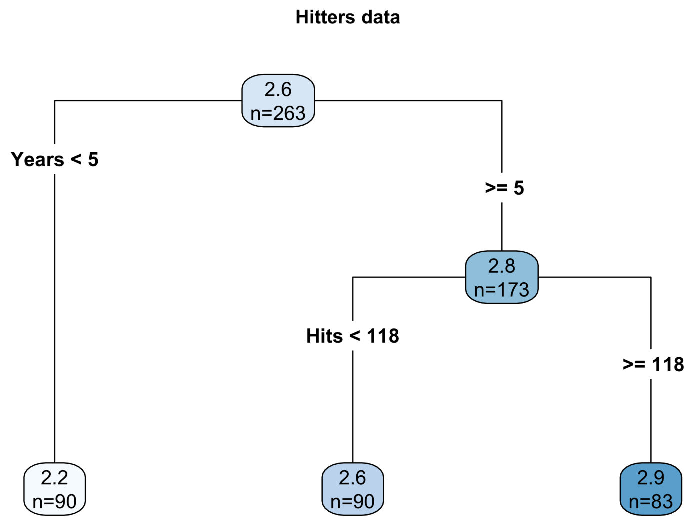
```r
# Fit a regression tree
hitters_rp <- rpart(lSalary~Hits+Years,
             data=Hitters,
             control=rpart.control(cp=0.05))

hitters_rp
```

```
## n= 263
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 263 39.071620 2.574160
##   2) Years< 4.5 90  7.988302 2.217851 *
##   3) Years>=4.5 173 13.713070 2.759523
##     6) Hits< 117.5 90  5.298802 2.605063 *
##     7) Hits>=117.5 83  3.938792 2.927009 *
```

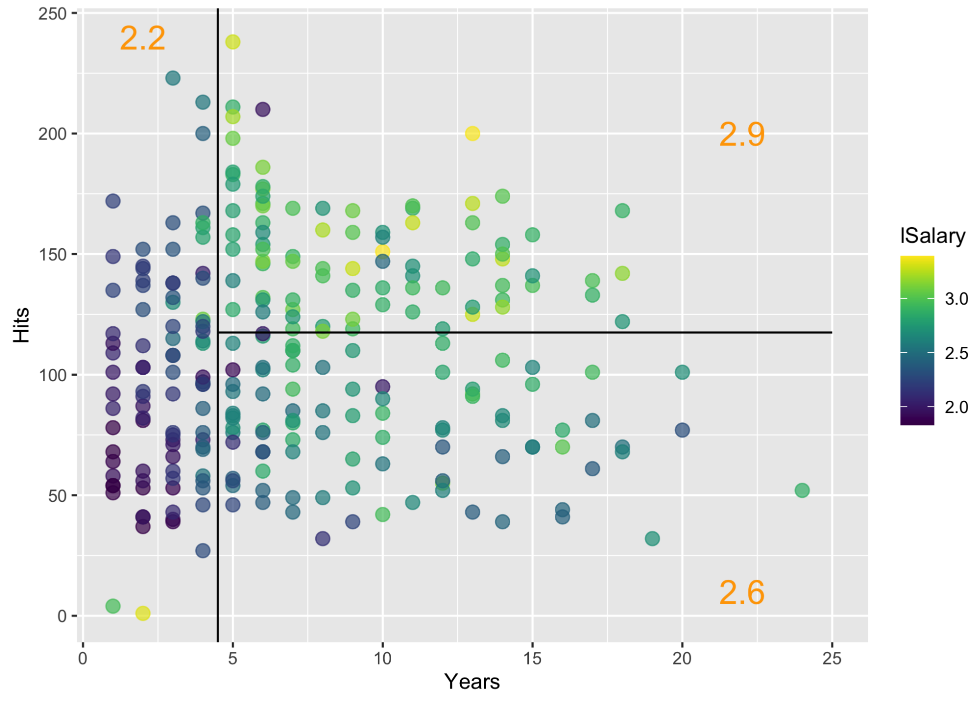
Predicting Salary

Using the function `rpart`, we can build a regression tree to predict the `logSalary` of a baseball player, given their `Years` of playing and number of `Hits`.

`rpart.plot` can be used to visualise the fitted tree.



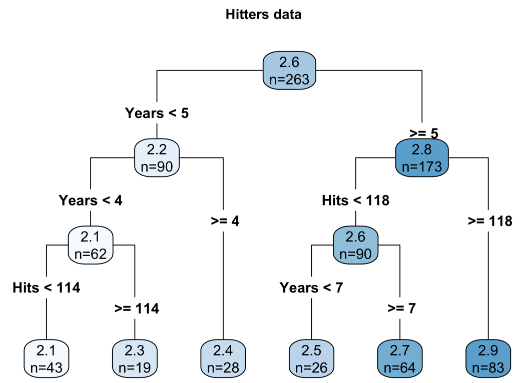
Regions of the decision tree



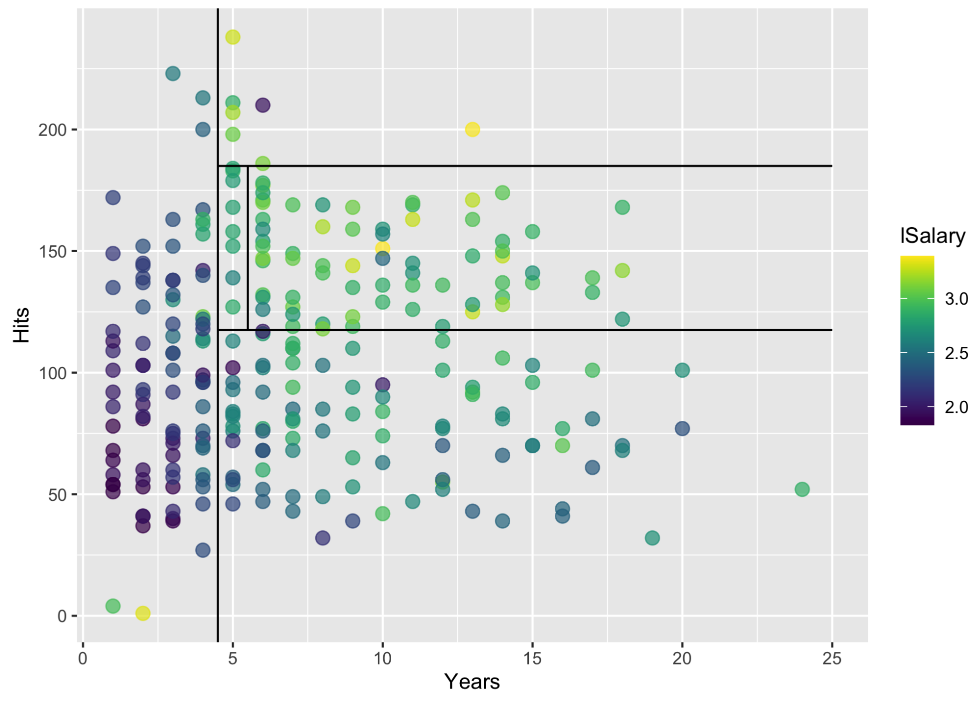
Deeper trees

By decreasing the value of the complexity parameter `cp`, we can build deeper trees.

```
# Fit a regression tree
hitters_rp2 <- rpart(lSalary~Hits+Years,
                     data=Hitters,
                     control=rpart.control(cp=0.012))
```



Regions



Regression trees - construction

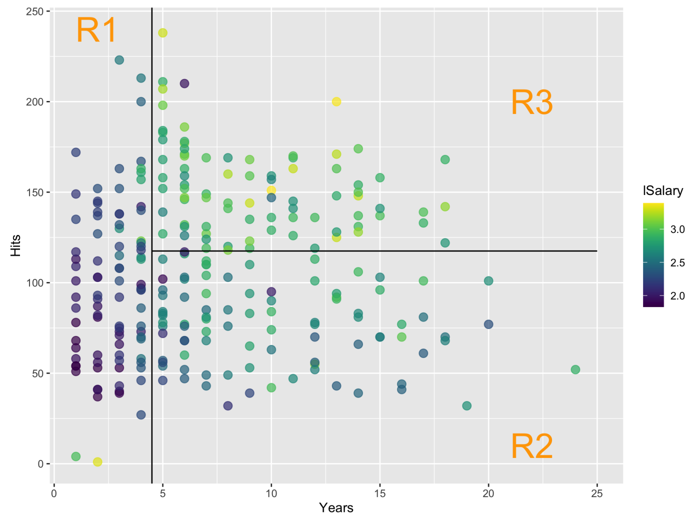
- We divide the predictor space - that is, the set of possible values for $X_1, X_2, \ldots, X_p$ - into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_M$.
- The regions could have any shape. However, for simplicity and for ease of interpretation, we divide the predictor space into high-dimensional rectangles.
- We model the response as a constant c_j in each region $f(x) = \sum_{j=1}^{J} c_j \, I(x \in R_m)$

e.g.

$R_1 = \{X|\text{Years} < 4.5\}$ $R_2 = \{X|\text{Years} \geq 4.5, \text{Hits} < 117.5\}$
$R_3 = \{X|\text{Years} \geq 4.5, \text{Hits} \geq 117.5\}$

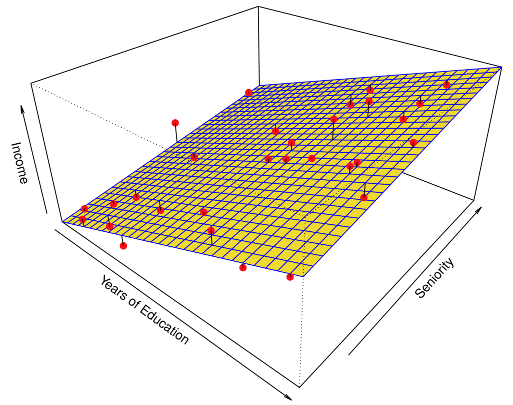
Leaves and Branches

- R_1, R_2, R_3 are terminal nodes or leaves.
- The points where we split are internal nodes.
- The segments that connect the nodes are branches.



Linear regression

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$



Regression trees

$$f(X) = \sum_{m=1}^{M} c_m \, I(X \in R_m)$$

Determining the c_m values and splits

Q1) Given a partition $R_1, R_2, \ldots, R_M$, what are the optimal values of c_m if we want to minimize $\sum_i (y_i - f(x_i))^2$ (the MSE)?

Q2) How do we construct the regions $R_1, \ldots, R_M$?

Determining the c_m values and splits

Q1) Given a partition $R_1, R_2, \ldots, R_M$, what are the optimal values of c_m if we want to minimize $\sum_i (y_i - f(x_i))^2$ (the MSE)?

The best c_m is just the average of y_i in region R_m: $\hat{c}_m = \text{average}(y_i | x_i \in R_m)$.

Q2) How do we construct the regions $R_1, \ldots, R_M$?

Finding the best binary partition in terms of minimum sum of squares is generally *computationally infeasible*. For this reason, we take a *top-down*, *greedy* approach that is known as recursive binary splitting.

Strategy for finding good splits

- Top-down: it begins at the top of the tree (all observations belong to a single region) and then successively splits the predictor space; each split is indicated via two new branches further down on the tree.
- Greedy: 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.

Algorithm

1. Start with a single region R_1 (entire input space), and iterate:

   a. Select a region R_m, a predictor X_j , and a splitting point s, such that splitting R_m with the criterion $X_j < s$ produces the largest decrease in RSS

   b. Redefine the regions with this additional split.

2. Continues until stopping criterion reached.

Stopping criterion

- $N_m < a$: Number of observations in R_m is too small to further splitting (`minsplit`). (There is usually another control criteria, even if N_m is large enough, you can't split it small number of observations off, e.g. 1 and $N_m - 1$, `minbucket`.)
- RSS $< tol$: If reduction of error is too small to bother splitting further. (`cp` parameter in `rpart` measures this as a proportional drop - see earlier examples displaying the change in this parameter.)

Diagnostics

Residual Sum of Squared Error

$$\mathrm{RSS}(T) = \sum_{m=1}^{|T|} N_m Q_m(T), \quad N_m = \#\{x_i \in R_m\},$$

where $Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$ and $|T|$ is the number of terminal nodes in T.

Size of tree

- It is possible to produce good predictions on the **training set**, but is likely to overfit the data (trees are very flexible).
- A smaller tree with fewer splits (that is, fewer regions) might lead to lower variance and better interpretation at the cost of a little bias.
- Tree size is a tuning parameter governing the **model's complexity**, and the optimal tree size should be adaptively chosen from the data
- Produce splits only if RSS decrease exceeds some **(high) threshold** can mean that a low gain split early on, might stop the fitting, even though there may be a very good split later.

Pruning

Grow a big tree, T_0, and then **prune** it back. The *pruning* procedure is:

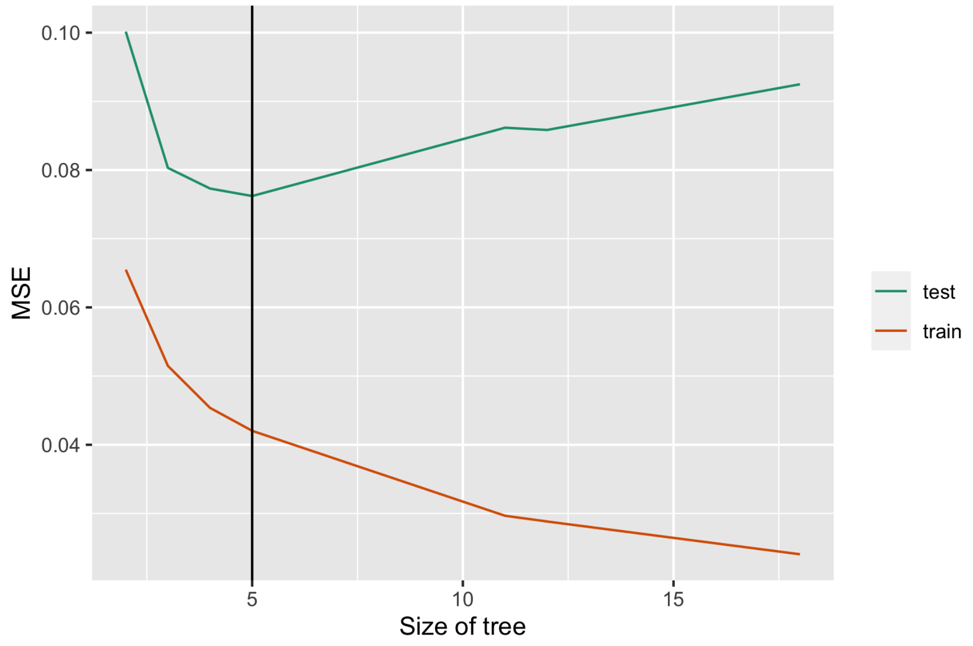
- Starting with with the initial full tree T_0, replace a subtree with a leaf node to obtain a new tree T_1. Select subtree to prune by minimizing

$$\frac{\mathrm{RSS}(T_1) - \mathrm{RSS}(T_0)}{|T_1| - |T_0|}$$

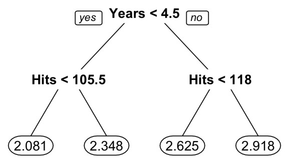
- Iteratively prune to obtain a sequence $T_0, T_1, T_2, \ldots, T_R$ where T_R is the tree with a single leaf node.
- Select the optimal tree T_m by cross validation

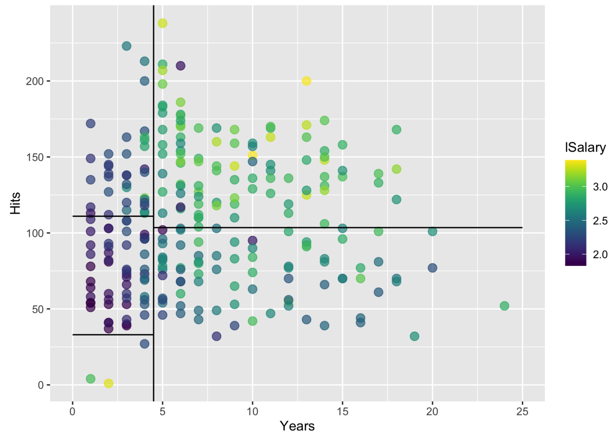
Model selection

Using a 50-50 training test set split.

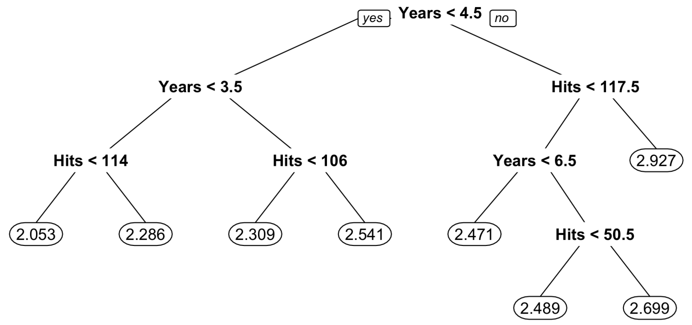


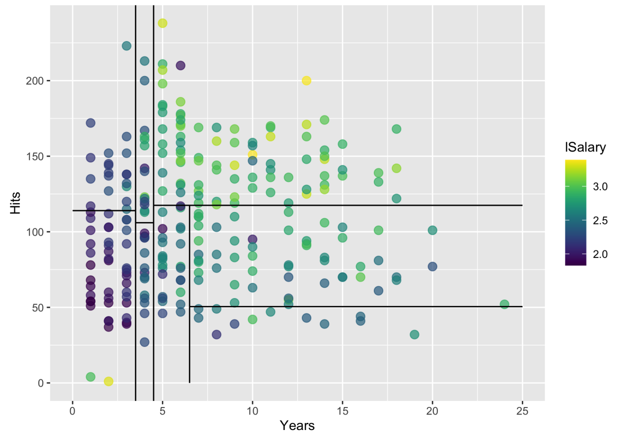
Yielding this model:





Cross-validation recommendation suggests more.





👩‍💻 Made by a human with a computer

Slides at https://iml.numbat.space.

Code and data at https://github.com/numbats/iml.

Created using R Markdown with flair by **xaringan**, and **kunoichi** (female ninja) style.