Introduction to Regression Trees

Sahir Rai Bhatnagar, PhD Candidate (Biostatistics)

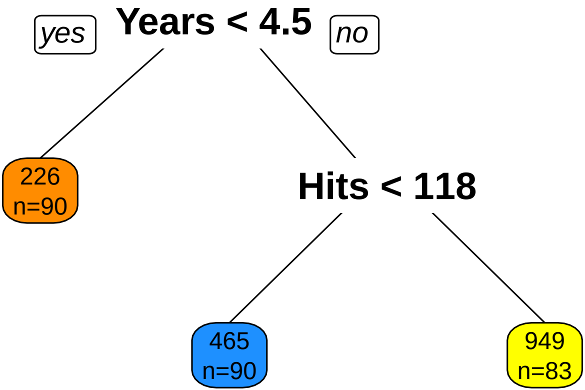
Department of Epidemiology, Biostatistics and Occupational Health

February 19, 2018

Introduction

What?

- A prediction model consisting of a series of **If-Else** statements
- e.g. Vladimir Guerrero: 7 years, 200 hits. Predict his salary for next year?



Background on CART

- Recursive partitioning or segmentation methods were first introduced in the 1960s

- They were formalized by Breiman et al. (1984) [1] under the acronym **CART: Classification and Regression Tree**.

- CART can be applied to both regression and classification problems depending on the response (outcome) variable:
  1. qualitative (classification)
  2. quantitative (regression)

Regression vs. Classification

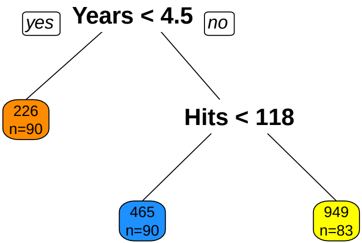


Fig.: Regression

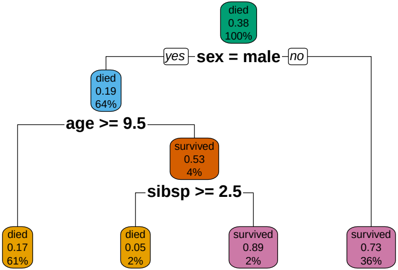


Fig.: Classification

Regression vs. Classification

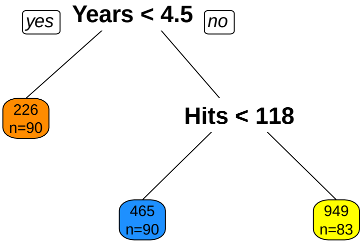


Fig.: Regression

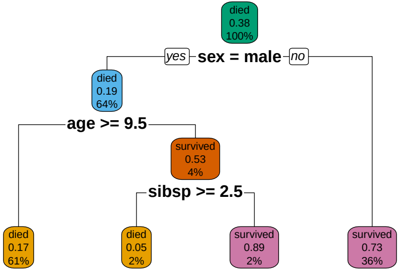
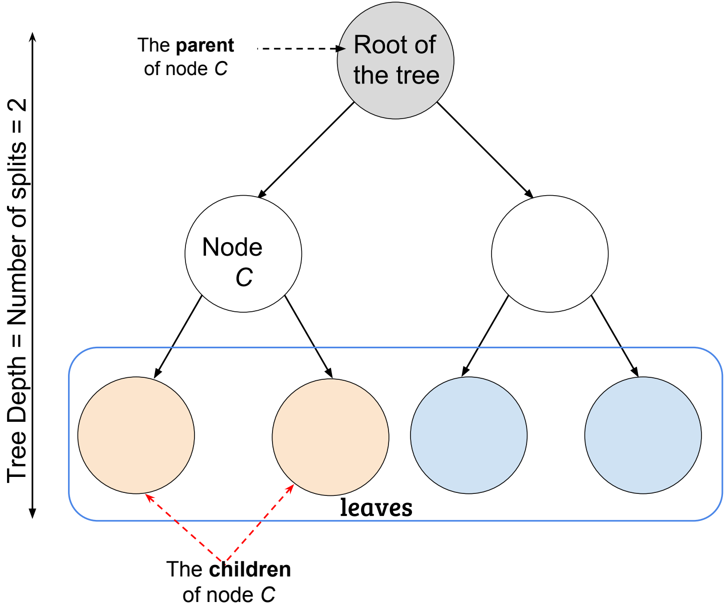


Fig.: Classification

- Today's class → regression

Terminology



A motivating example

Prediction of Major League Baseball Salaries

- Major League Baseball (MLB) data from the 1986 and 1987 seasons. Available in the `ISLR` [4] R package:

```r
library(ISLR)
data(Hitters)
```

Prediction of Major League Baseball Salaries

- Major League Baseball (MLB) data from the 1986 and 1987 seasons. Available in the ISLR [4] R package:

```
library(ISLR)
data(Hitters)
```

- Response variable $y_i, i = 1, \ldots, 263$: 1987 annual salary on opening day in thousands of dollars

Prediction of Major League Baseball Salaries

- Major League Baseball (MLB) data from the 1986 and 1987 seasons. Available in the `ISLR` [4] R package:

```r
library(ISLR)
data(Hitters)
```

- Response variable $y_i, i = 1, \ldots, 263$: 1987 annual salary on opening day in thousands of dollars

- Predictor variables:
  1. X_1: Number of years in the major leagues

Prediction of Major League Baseball Salaries

- Major League Baseball (MLB) data from the 1986 and 1987 seasons. Available in the `ISLR` [4] R package:

```
library(ISLR)
data(Hitters)
```

- Response variable $y_i, i = 1, \ldots, 263$: 1987 annual salary on opening day in thousands of dollars

- Predictor variables:
  1. X_1: Number of years in the major leagues
  2. X_2: Number of hits in 1986

Prediction of Major League Baseball Salaries

- Major League Baseball (MLB) data from the 1986 and 1987 seasons. Available in the `ISLR` [4] R package:

```r
library(ISLR)
data(Hitters)
```

- Response variable $y_i, i = 1, \ldots, 263$: 1987 annual salary on opening day in thousands of dollars

- Predictor variables:
  1. X_1: Number of years in the major leagues
  2. X_2: Number of hits in 1986

Objective

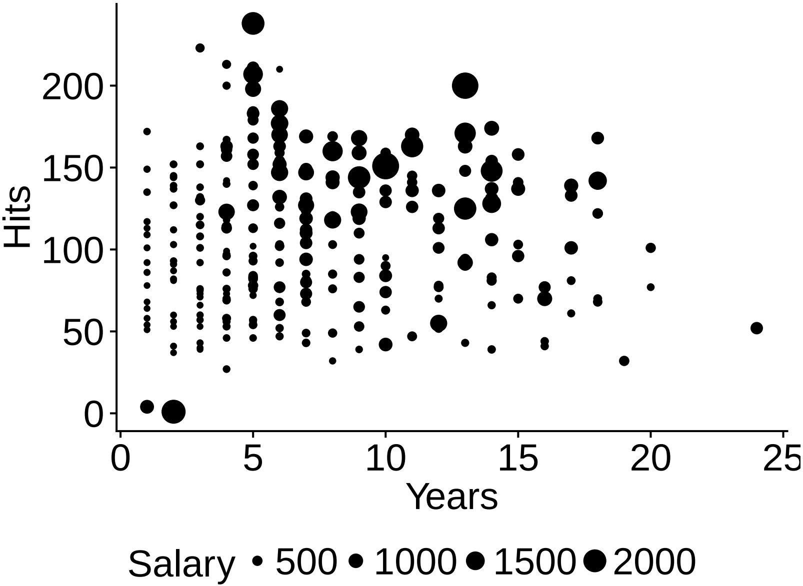
Predict the annual (`salary`) at the start of the 1987 season using the predictor variables (`years` and `hits`).

The Data

A sample of what the data looks like:

	Years	Hits	Salary
-Andre Dawson	11	141	500
-Andres Galarraga	2	87	92
-Barry Bonds	1	92	100
-Cal Ripken	6	177	1350
-Gary Carter	13	125	1926
-Joe Carter	4	200	250
-Ken Griffey	14	150	1000
-Mike Schmidt	2	1	2127
-Tony Gwynn	5	211	740

A Visual Representation of the Data



How does CART work?

Roughly speaking, there are two steps [3]:

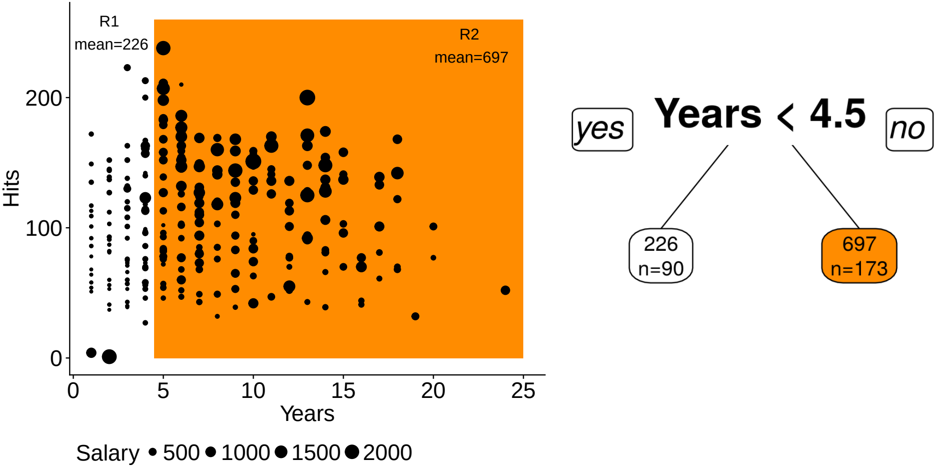
How does CART work?

Roughly speaking, there are two steps [3]:

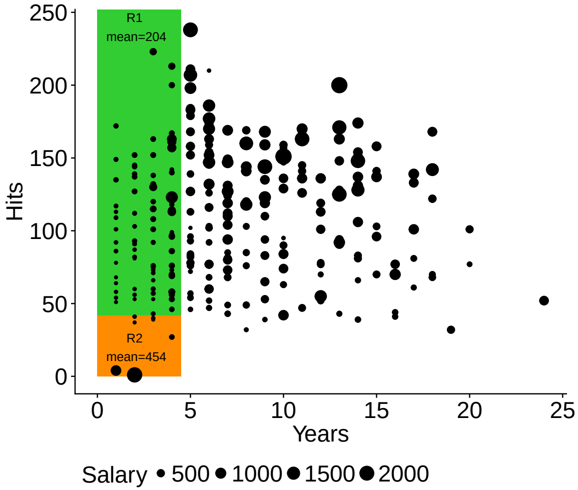
1. We divide the predictor space - that is, the set of possible values for $X_1, X_2, \ldots, X_p$, into J non-overlapping and exhaustive regions, $R_1, R_2, \ldots, R_J$.

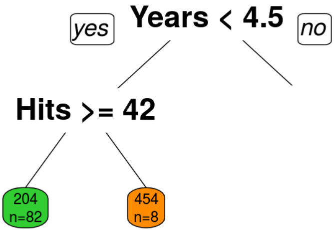
2. For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

First Split


R1
mean=226
R2
mean=697
Hits
200
100
0
0
5
10
15
20
25
Years
Salary
500
1000
1500
2000
yes
Years < 4.5
no
226
n=90
697
n=173

Second Split



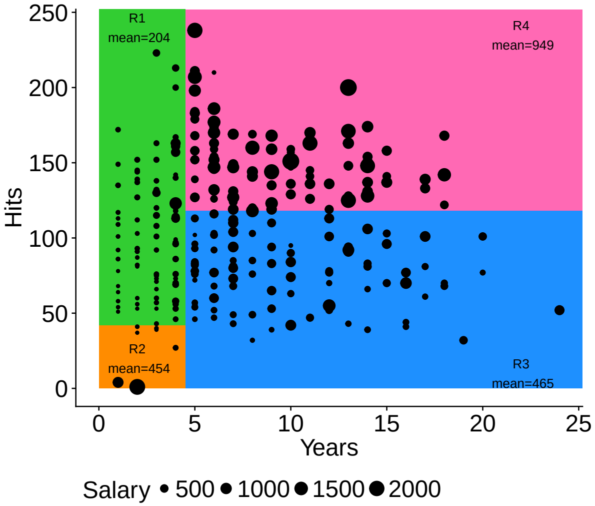


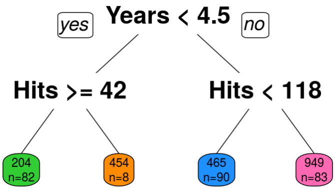
A Mistake in the Data

	Years	Hits	Salary
-Andre Dawson	11	141	500.00
-Andres Galarraga	2	87	91.50
-Barry Bonds	1	92	100.00
-Cal Ripken	6	177	1350.00
-Gary Carter	13	125	1925.57
-Joe Carter	4	200	250.00
-Ken Griffey	14	150	1000.00
-Mike Schmidt	2	1	2127.33
-Tony Gwynn	5	211	740.00

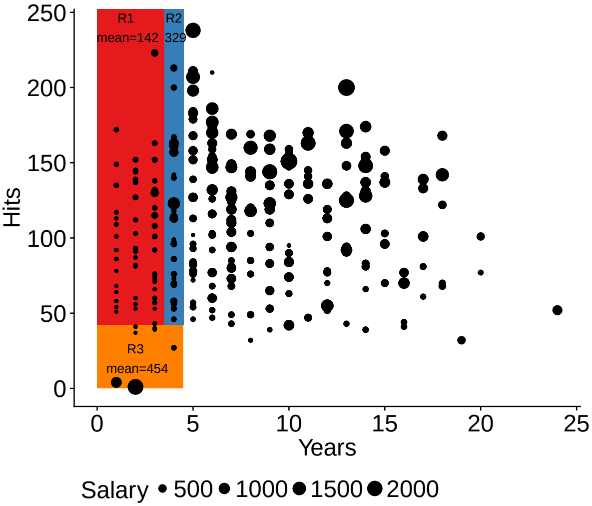
- Mike Schmidt started his career in 1972, and was inducted into the Baseball Hall of Fame in 1995.

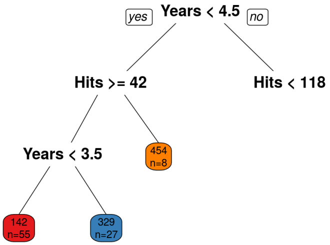
Second Split



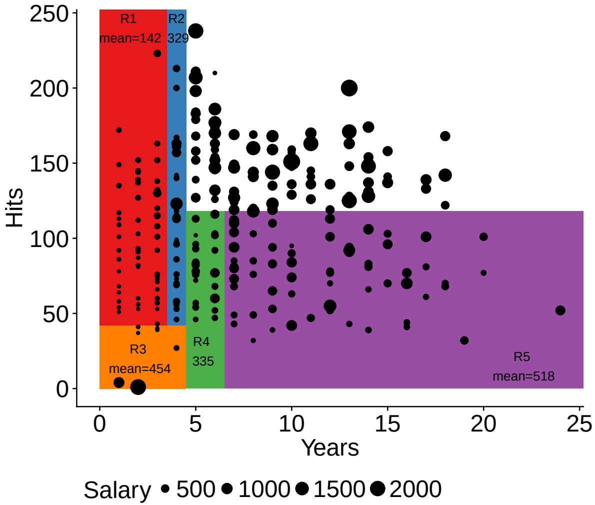


Third Split

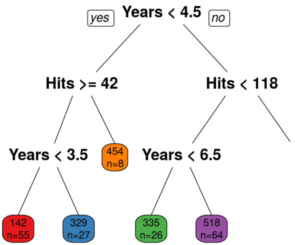

R1
mean=142
R2
329
R3
mean=454
Hits
Years
0
5
10
15
20
25
0
50
100
150
200
250
Salary 500 1000 1500 2000


Years < 4.5
yes
no
Hits >= 42
Hits < 118
Years < 3.5
454
n=8
142
n=55
329
n=27

Third Split

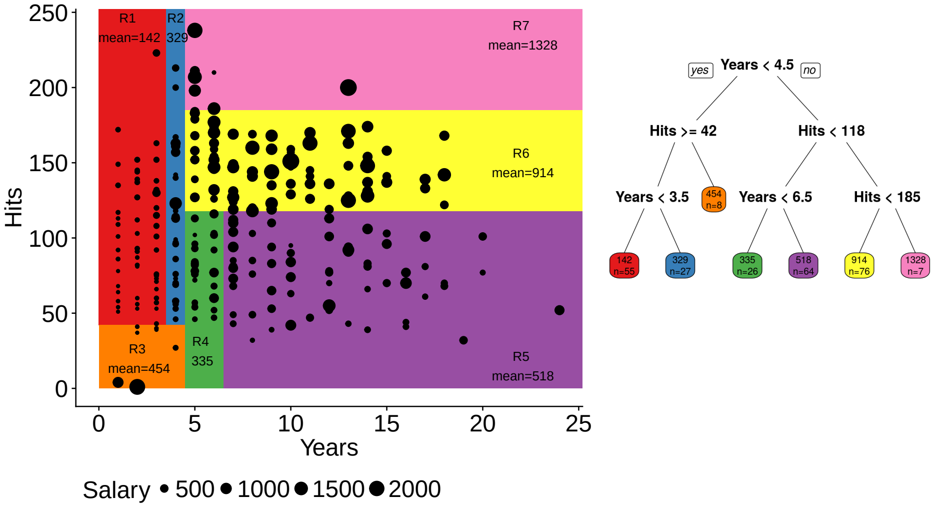


R1
mean=142
R2
329
R3
mean=454
R4
335
R5
mean=518
Hits
Years
0
50
100
150
200
250
0
5
10
15
20
25
Salary 500 1000 1500 2000

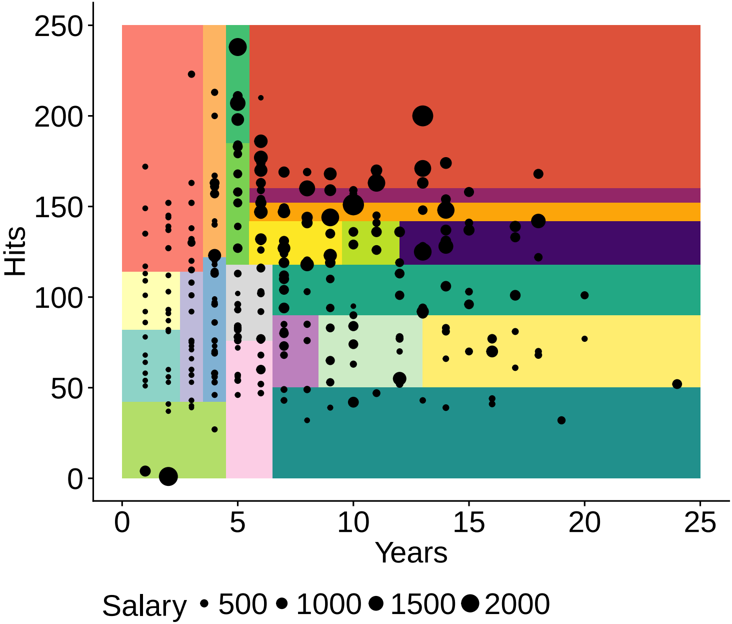


Years < 4.5
yes
no
Hits >= 42
Hits < 118
Years < 3.5
454
n=8
Years < 6.5
142
n=55
329
n=27
335
n=26
518
n=64

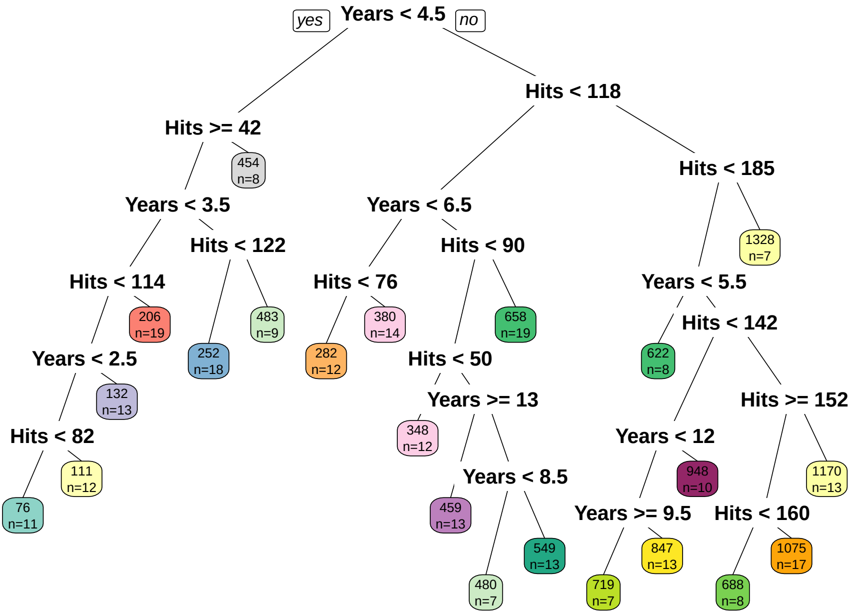
Third Split


R1
mean=142
R2
329
R7
mean=1328
R6
mean=914
R3
mean=454
R4
335
R5
mean=518
Hits
Years
Salary 500 1000 1500 2000
Years < 4.5
yes
no
Hits >= 42
Hits < 118
Years < 3.5
454 n=8
Years < 6.5
Hits < 185
142 n=55
329 n=27
335 n=26
518 n=64
914 n=76
1328 n=7

And if we continue…



Stop if the number of observations is less than 20



The Details

The Details

The CART algorithm requires 3 components:

1. Defining a criterion to select the best partition among all predictors.

2. A rule to decide when a node is terminal, i.e., it becomes a leaf.

3. Pruning the tree to avoid over-fitting.

1. Selecting the Best Partition

The objective is the find the regions $R_1, \ldots, R_J$ that minimize the squared error loss:

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

- $\hat{y}_{R_j}$: the mean response for the training observations within the jth box

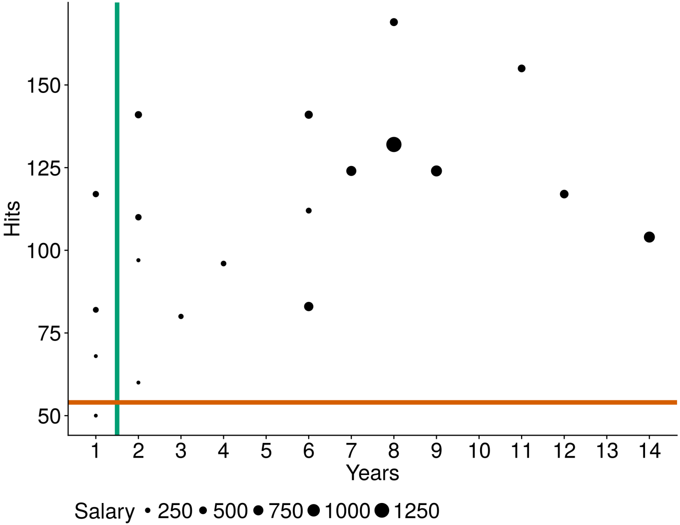
1. Selecting the Best Partition

The objective is the find the regions $R_1, \ldots, R_J$ that minimize the squared error loss:

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

- $\hat{y}_{R_j}$: the mean response for the training observations within the jth box
- Finding the solution to (1) is computationally infeasible (*NP-hard*). Why?

Exhaustive Search for $J = 4$



1. A Top-Down “*Greedy*” Approach

- *Top-Down*: It begins at the top of the tree at which point all observations belong to a single region.

1. A Top-Down "*Greedy*" Approach

- *Top-Down*: It begins at the top of the tree at which point all observations belong to a single region.
- *Binary Splits*: Each split at the value s for the jth predictor creates **exactly** two children; R_1 and R_2 which leads to the greatest possible reduction in the *residual sum of squares*:

$$R_1(j,s) = \{X|X_j < s\} \qquad \text{and} \qquad R_2(j,s) = \{X|X_j \geq s\}$$

1. A Top-Down "*Greedy*" Approach

- *Top-Down*: It begins at the top of the tree at which point all observations belong to a single region.
- *Binary Splits*: Each split at the value s for the jth predictor creates **exactly** two children; R_1 and R_2 which leads to the greatest possible reduction in the *residual sum of squares*:

$$R_1(j,s) = \{X | X_j < s\} \qquad \text{and} \qquad R_2(j,s) = \{X | X_j \geq s\}$$

- The goal is to find the values j and s that minimize the equation:

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

1. A Top-Down "*Greedy*" Approach

- *Top-Down*: It begins at the top of the tree at which point all observations belong to a single region.
- *Binary Splits*: Each split at the value s for the jth predictor creates **exactly** two children; R_1 and R_2 which leads to the greatest possible reduction in the *residual sum of squares*:

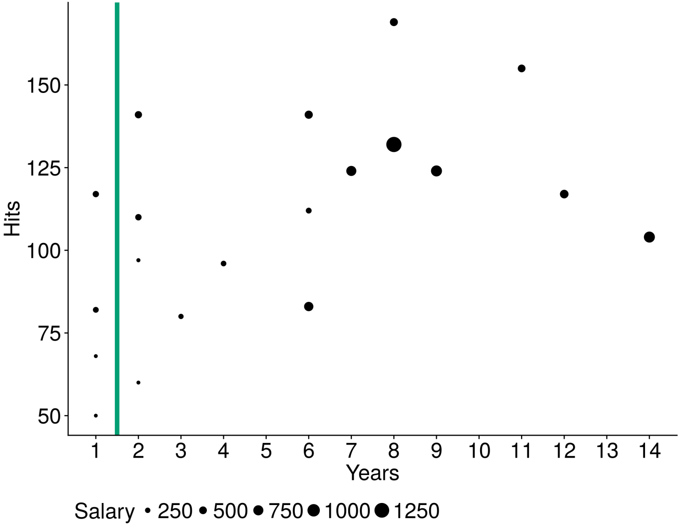
$$R_1(j,s) = \{X | X_j < s\} \qquad \text{and} \qquad R_2(j,s) = \{X | X_j \geq s\}$$

- The goal is to find the values j and s that minimize the equation:

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

- *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

The Best Split Using a “*Greedy*” Approach



2. Stopping Rule

- `minsplit`: To avoid creating splits that will lead to very small leaves, the minimum number of observations that must exist in a node in order for a split to be attempted (`minsplit = 20` is the default in `rpart`).

- `minbucket`: the minimum number of observations in any terminal leaf node (`minbucket` = `minsplit/3` is the default in `rpart`)

3. Pruning the 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.

- This is because the resulting tree, T_{max} with $|T_{max}|$ leaves, might be too complex.

- A smaller tree with fewer splits (that is, fewer regions $R_1, \ldots, R_J$) might lead to lower **prediction variance** and better interpretation at the cost of a little **bias**. *What is this phenomenon called?*

3. Pruning the Tree

- We first grow the biggest tree possible T_{max} and then *prune* it back in order to obtain a *subtree*

- We consider adding a **penalty** to our loss function in order to penalize excessively large trees.

- For each value of α, there exists a subtree $T \subset T_{max}$ that minimizes:

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T| \tag{3}$$

- $|T|$ indicates the number of terminal nodes of the tree T, R_m is the rectangle corresponding to the mth leaf, and $\hat{y}_{R_m}$ is the predicted response associated with R_m.

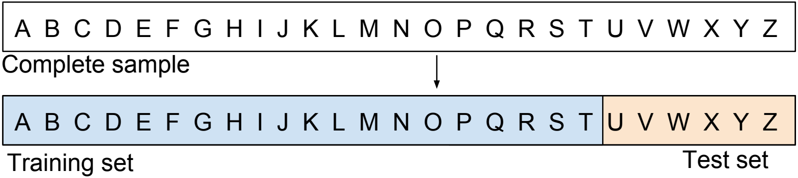
- α is chosen using v-fold cross-validation ($v \rightarrow$ `xval=10` in `rpart` by default).

Background on Cross-Validation

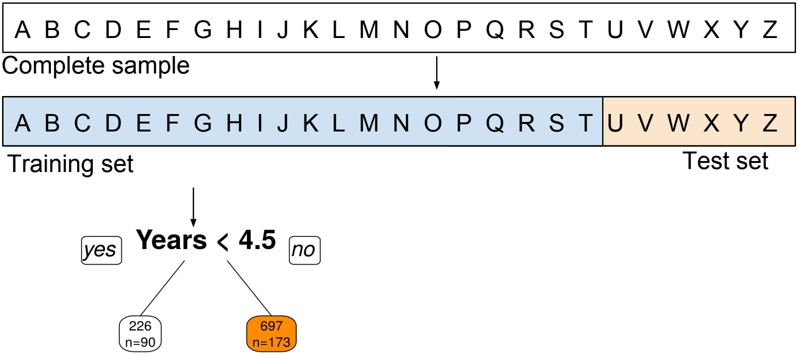
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Complete sample

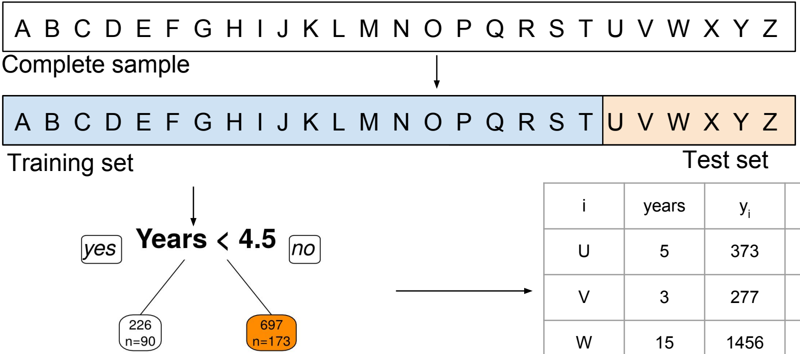
Background on Cross-Validation



Background on Cross-Validation



Background on Cross-Validation



i	years	y_i	$y_i^{(pred)}$
U	5	373	
V	3	277	
W	15	1456	
X	4	455	
Y	1	235	
Z	9	987	

Background on Cross-Validation

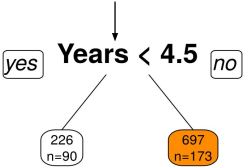
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Complete sample

A B C D E F G H I J K L M N O P Q R S T | U V W X Y Z

Training set

Test set



i	years	y_i	$y_i^{(pred)}$
U	5	373	697
V	3	277	226
W	15	1456	697
X	4	455	226
Y	1	235	226
Z	9	987	697

Background on Cross-Validation

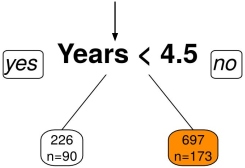
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Complete sample

A B C D E F G H I J K L M N O P Q R S T | U V W X Y Z

Training set

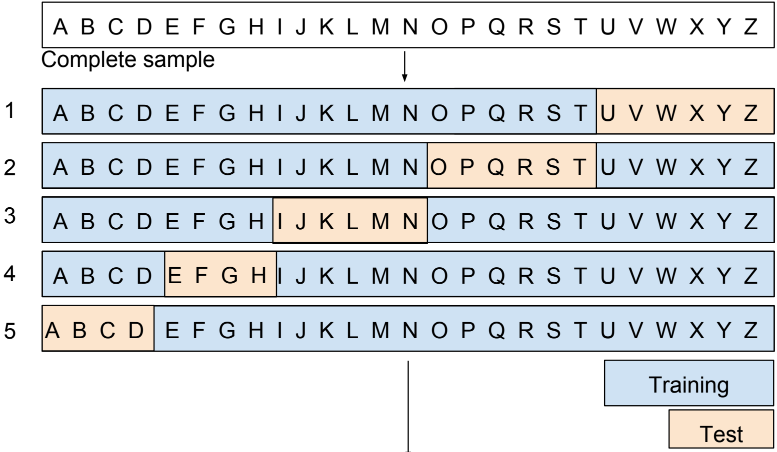
Test set



i	years	y_i	$y_i^{(pred)}$
U	5	373	697
V	3	277	226
W	15	1456	697
X	4	455	226
Y	1	235	226
Z	9	987	697

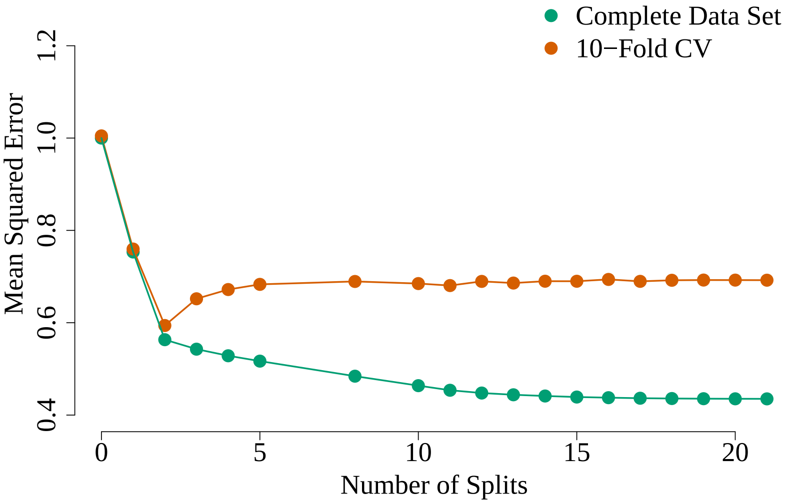
$$MSE^{(test)} = \sum_{i=1}^{6} \left(y_i - y_i^{(pred)}\right)^2 + \alpha|T|$$

Background on Cross-Validation



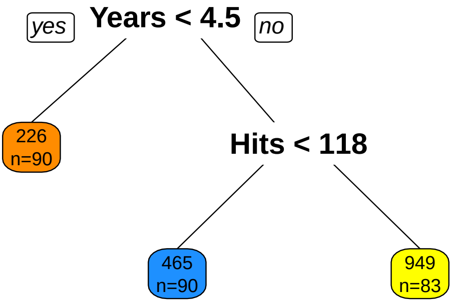
$$CV(\alpha) = \frac{1}{5}\sum_{v=1}^{5} MSE_v^{(test)}$$

Over-fitting



Fitting and Pruning with `rpart`

```
cart_fit <- rpart::rpart(Salary ~ Years + Hits, data = Hitters)
min_ind <- which.min(cart_fit$cptable[, "xerror"])
min_cp <- cart_fit$cptable[min_ind, "CP"]
prune_fit <- rpart::prune(cart_fit, cp = min_cp)
rpart.plot::rpart.plot(prune_fit)
```



Comparison with a Linear Model

Comparison: Linear Model vs. CART

Characteristic[a]	`Linear Model`	CART
Linearity Assumption	✓	✗
Distributional Assumptions	✓	✗
Robust to multicollinearity	✗	✓
Handles complex interactions	✗	✓
Allows for missing data	✗	✓
Confidence Intervals, p-values	✓	✗

[a] ✓: yes, ✗: no

Linear Model

```r
lm(Salary ~ Years * Hits, data = Hitters)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	159.55	95.65	1.67	0.10
Years	-16.08	11.38	-1.41	0.16
Hits	0.60	0.87	0.69	0.49
Years:Hits	0.54	0.11	5.08	0.00

Table: R2 = 0.41

Regression Surface

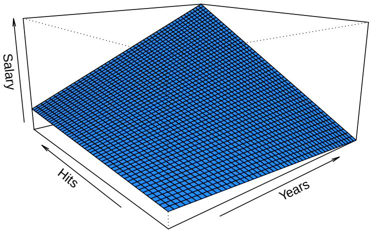


Fig.: Linear Model

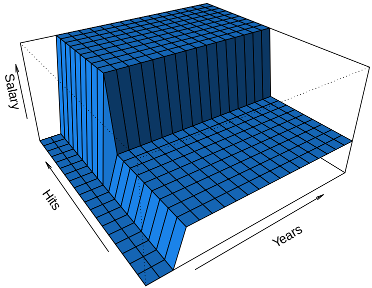
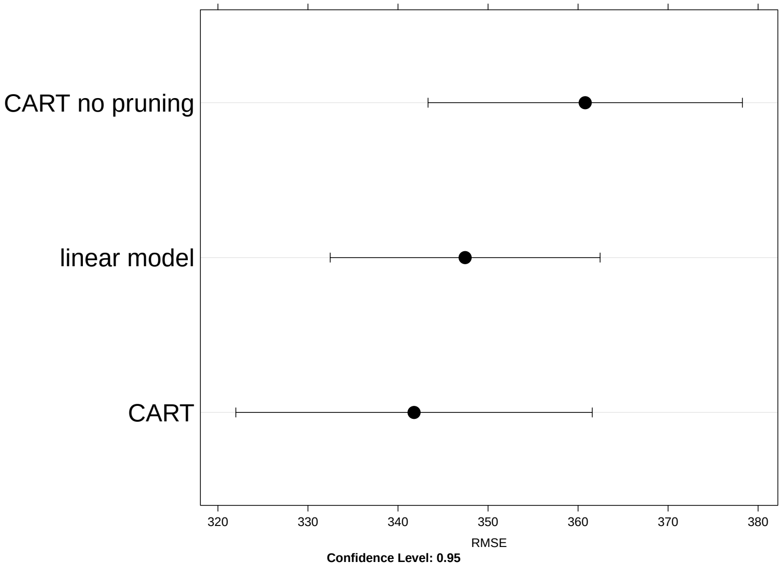
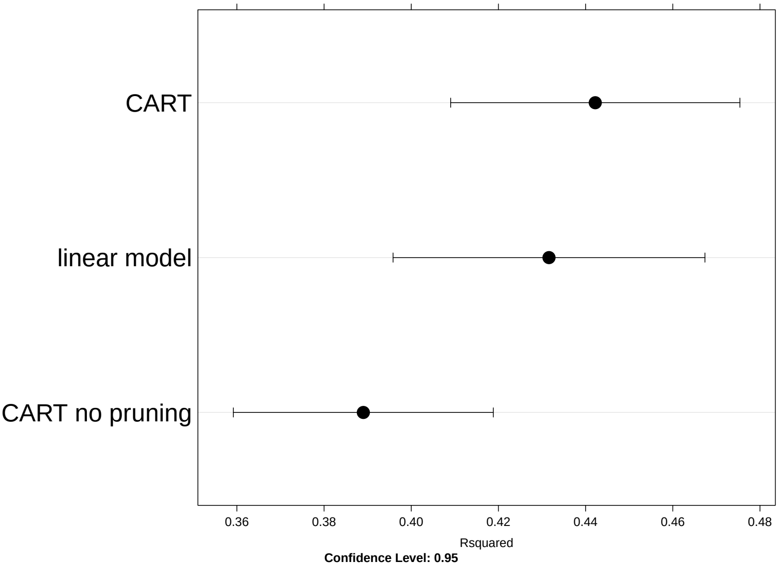


Fig.: CART

RMSE Performance: 10 times 10-fold CV



R^2 Performance: 10 times 10-fold CV



Advantages

- CART models are easy to interpret
- You don’t need to pre-define relationships between variables
- Automatically handles higher-order interactions

Limitations

- CART models generally produce unstable predictions (next class $\rightarrow$ random forests)

Exercise

Build a tree by hand = no software!

- Build a tree using the dataset provided below

- Use the parameters `minsplit = 6` and `minbucket = 2`

	Years	Hits	Salary
-Rey Quinones	1	68	70
-Barry Bonds	1	92	100
-Pete Incaviglia	1	135	172
-Dan Gladden	4	97	210
-Juan Samuel	4	157	640
-Joe Carter	4	200	250
-Tim Wallach	7	112	750
-Rafael Ramirez	7	119	875
-Harold Baines	7	169	950

References I

[1] Leo Breiman et al. *Classification and regression trees.* CRC press, 1984.

[2] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning.* Vol. 1. Springer series in statistics New York, 2001.

[3] Gareth James et al. *An introduction to statistical learning.* Vol. 112. Springer, 2013.

[4] Gareth James et al. "Package 'ISLR'". In: (2017).

[5] Olivier Lopez, Xavier Milhaud, and Pierre-Emmanuel Thérond. "Arbres de régression et de classification (CART)". In: *l'actuariel* 15 (2015), pp. 42–44.

Session Info

```
R version 3.4.1 (2017-06-30)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.3 LTS

Matrix products: default
BLAS: /usr/lib/openblas-base/libblas.so.3
LAPACK: /usr/lib/libopenblasp-r0.2.18.so

attached base packages:
[1] methods   stats     graphics  grDevices utils     datasets  base

other attached packages:
 [1] dplyr_0.7.2          purrr_0.2.3          readr_1.1.1
 [4] tidyr_0.7.1          tibble_1.4.2         tidyverse_1.1.1
 [7] caret_6.0-77         lattice_0.20-35      plotmo_3.3.4
[10] TeachingDemos_2.10   plotrix_3.6-6        visreg_2.4-1
[13] sjmisc_2.6.1         sjPlot_2.3.3         cowplot_0.8.0.9000
[16] ggplot2_2.2.1.9000   xtable_1.8-2         rpart.plot_2.1.2
[19] rpart_4.1-11         data.table_1.10.4-3 ISLR_1.2
[22] knitr_1.19

loaded via a namespace (and not attached):
 [1] TH.data_1.0-8      minqa_1.2.4        colorspace_1.3-2
 [4] class_7.3-14       modeltools_0.2-21  sjlabelled_1.0.1
 [7] glmmTMB_0.1.1      DRR_0.0.2          DT_0.2
[10] prodlim_1.6.1      mvtnorm_1.0-6      lubridate_1.6.0
[13] xml2_1.1.1         coin_1.2-1         RSkittleBrewer_1.1
[16] codetools_0.2-15   splines_3.4.1      mnormt_1.5-5
[19] robustbase_0.92-7  effects_3.1-2      RcppRoll_0.2.2
[22] jsonlite_1.5       nloptr_1.0.4       broom_0.4.2
[25] ddalpha_1.2.1      kernlab_0.9-25     shiny_1.0.5
[28] compiler_3.4.1     httr_1.3.1         sjstats_0.11.0
[31] assertthat_0.2.0   Matrix_1.2-11      lazyeval_0.2.1
[34] htmltools_0.3.6    tools_3.4.1        bindrcpp_0.2
[37] coda_0.19-1        gtable_0.2.0       glue_1.1.1
```