Predictive Modeling

DATA 606 - Statistics & Probability for Data Analytics

Jason Bryer, Ph.D. and Angela Lui, Ph.D.

May 1, 2024

One Minute Paper Results

What was the most important thing you learned during this class?

What important question remains unanswered for you?

Classification and Regression Trees (CART)

Classification and Regression Trees

The goal of CART methods is to find best predictor in X of some outcome, y. CART methods do this recursively using the following procedures:

- Find the best predictor in X for y.
- Split the data into two based upon that predictor.
- Repeat 1 and 2 with the split data sets until a stopping criteria has been reached.

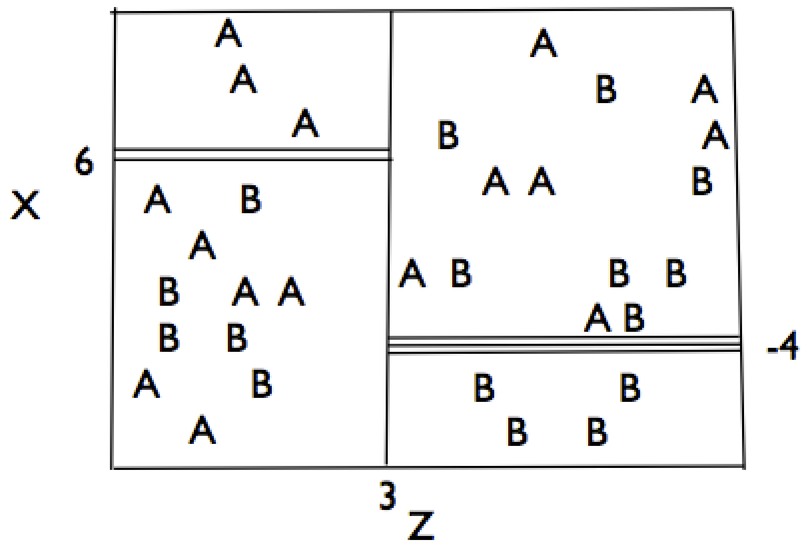
There are a number of possible stopping criteria including: Only one data point remains.

- All data points have the same outcome value.
- No predictor can be found that sufficiently splits the data.

Recursive Partitioning Logic of CART

Consider the scatter plot to the right with the following characteristics:

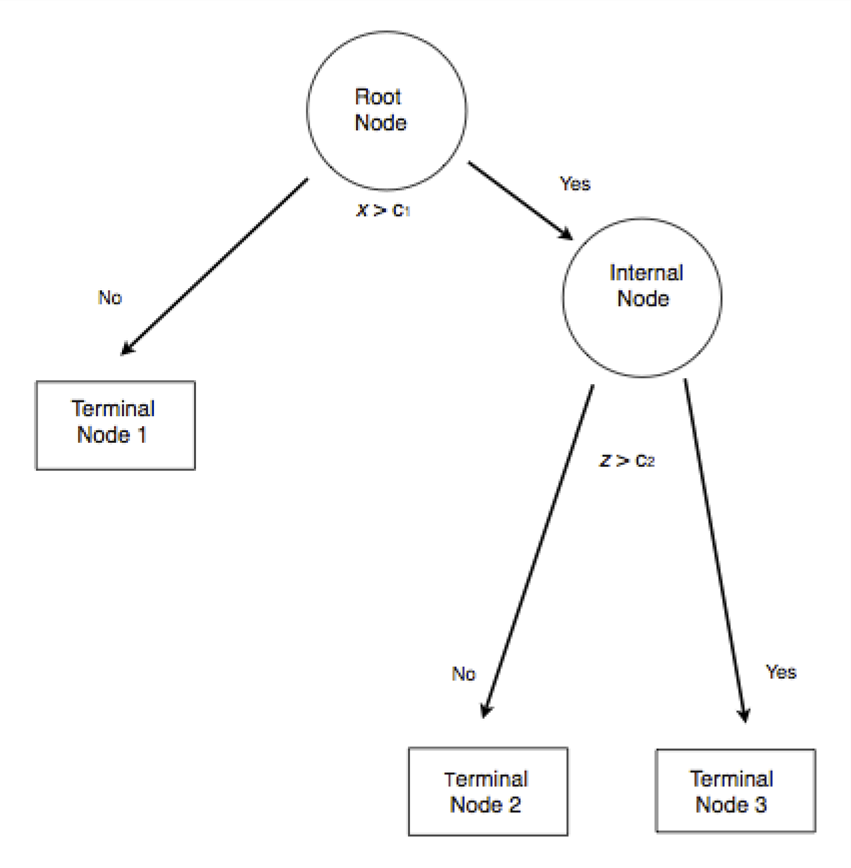
- Binary outcome, G, coded “A” or “B”.
- Two predictors, x and z
- The vertical line at z = 3 creates the first partition.
- The double horizontal line at x = -4 creates the second partition.
- The triple horizontal line at x = 6 creates the third partition.



Recursive Partitioning of a Binary Outcome
(where G = A or B and predictors are Z and X)

Tree Structure

- The root node contains the full data set.
- The data are split into two mutually exclusive pieces. Cases where x > ci go to the right, cases where x <= ci go to the left.
- Those that go to the left reach a terminal node.
- Those on the right are split into two mutually exclusive pieces. Cases where z > c2 go to the right and terminal node 3; cases where z <= c2 go to the left and terminal node 2.



Sum of Squared Errors

The sum of squared errors for a tree *T* is:

$$S = \sum_{c \in leaves(T)} \sum_{i \in c} (y - m_c)^2$$

Where, $m_c = \frac{1}{n} \sum_{i \in c} y_i$, the prediction for leaf \textit{c}.

Or, alternatively written as:

$$S = \sum_{c \in leaves(T)} n_c V_c$$

Where V_c is the within-leave variance of leaf \textit{c}.

Our goal then is to find splits that minimize S.

Advantages of CART Methods

- Making predictions is fast.
- It is easy to understand what variables are important in making predictions.
- Trees can be grown with data containing missingness. For rows where we cannot reach a leaf node, we can still make a prediction by averaging the leaves in the sub-tree we do reach.
- The resulting model will inherently include interaction effects. There are many reliable algorithms available.

Regression Trees

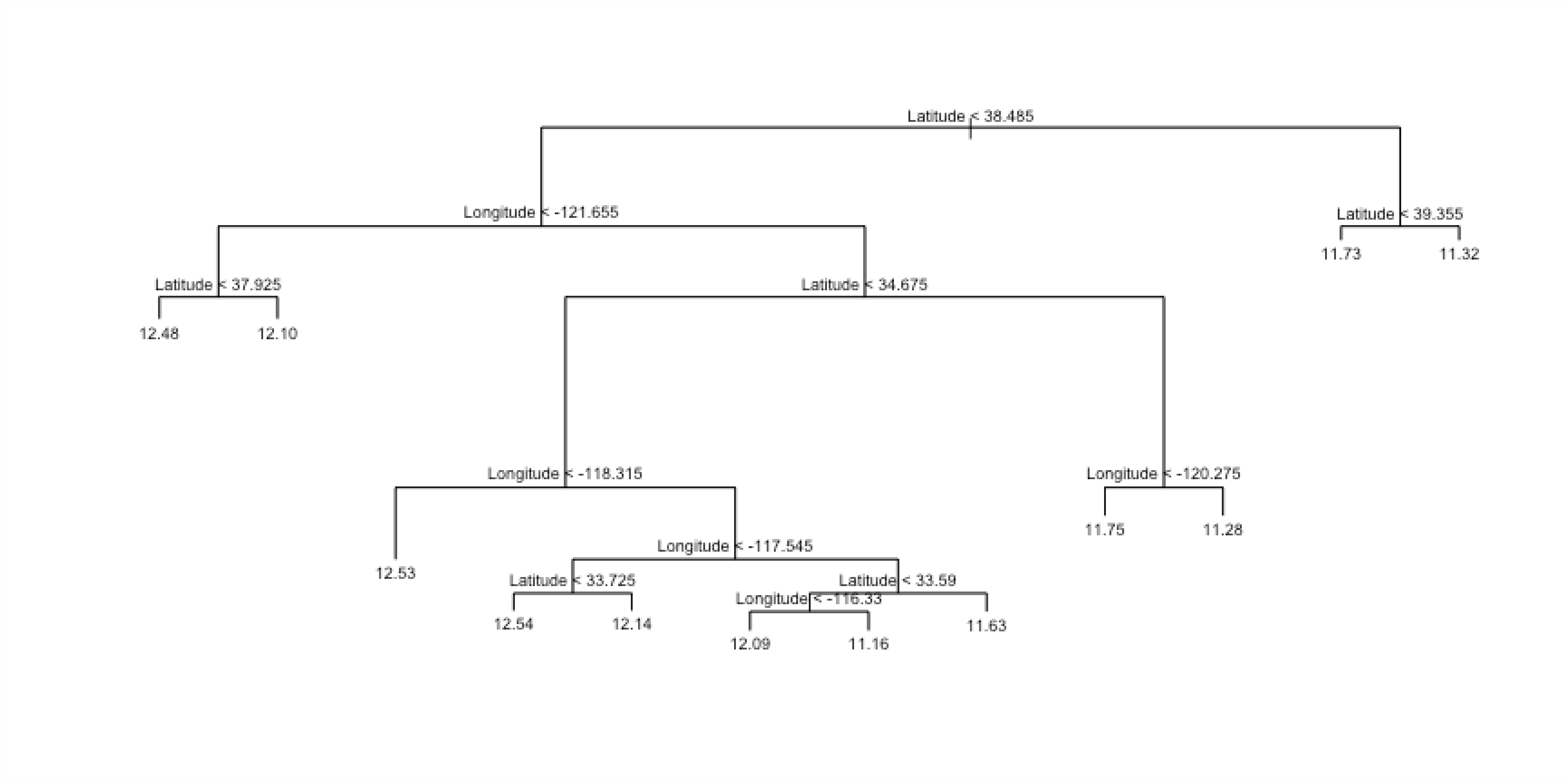
In this example we will predict the median California house price from the house's longitude and latitude.

```
str(calif)
```

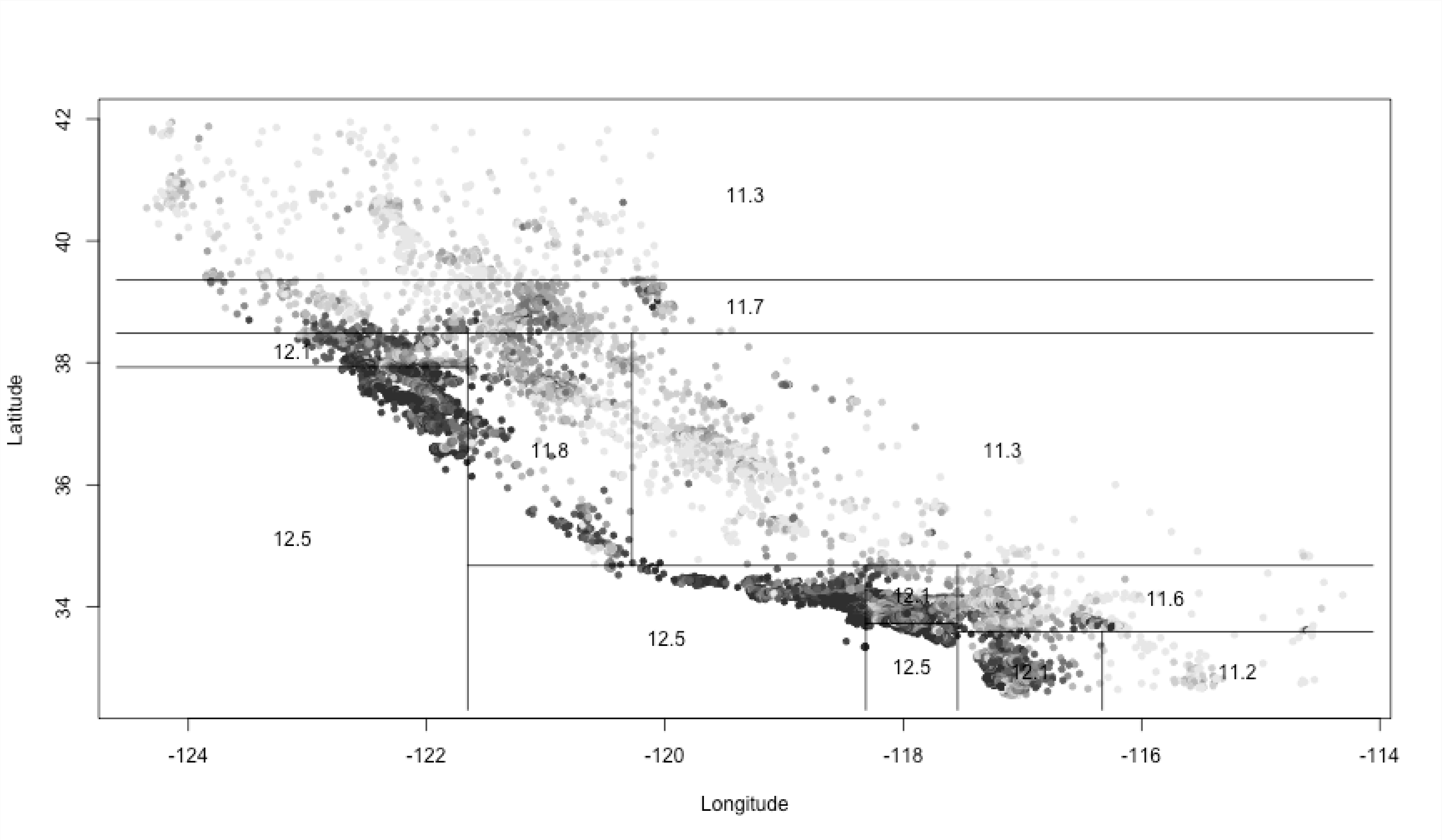
```
## 'data.frame':    20640 obs. of  10 variables:
##  $ MedianHouseValue: num  452600 358500 352100 341300 342200 ...
##  $ MedianIncome    : num  8.33 8.3 7.26 5.64 3.85 ...
##  $ MedianHouseAge  : num  41 21 52 52 52 52 52 52 42 52 ...
##  $ TotalRooms      : num  880 7099 1467 1274 1627 ...
##  $ TotalBedrooms   : num  129 1106 190 235 280 ...
##  $ Population      : num  322 2401 496 558 565 ...
##  $ Households      : num  126 1138 177 219 259 ...
##  $ Latitude        : num  37.9 37.9 37.9 37.9 37.9 ...
##  $ Longitude       : num  -122 -122 -122 -122 -122 ...
##  $ cut.prices      : Factor w/ 4 levels "[1.5e+04,1.2e+05]",..: 4 4 4 4 4 4 4 3 3 3 ...
```

Tree 1

```r
treefit <- tree(log(MedianHouseValue) ~ Longitude + Latitude, data=calif)
plot(treefit); text(treefit, cex=0.75)
```



Tree 1



11.3
11.7
12.1
11.8
11.3
12.5
12.1
11.6
12.5
12.5
12.1
11.2
Latitude
Longitude
42
40
38
36
34
-124
-122
-120
-118
-116
-114

Tree 1

```
summary(treefit)
```

```
##
## Regression tree:
## tree(formula = log(MedianHouseValue) ~ Longitude + Latitude,
##     data = calif)
## Number of terminal nodes:  12
## Residual mean deviance:  0.1662 = 3429 / 20630
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.75900 -0.26080 -0.01359  0.00000  0.26310  1.84100
```

Here “deviance” is the mean squared error, or root-mean-square error of $\sqrt{.166} = 0.41$.

Tree 2, Reduce Minimum Deviance

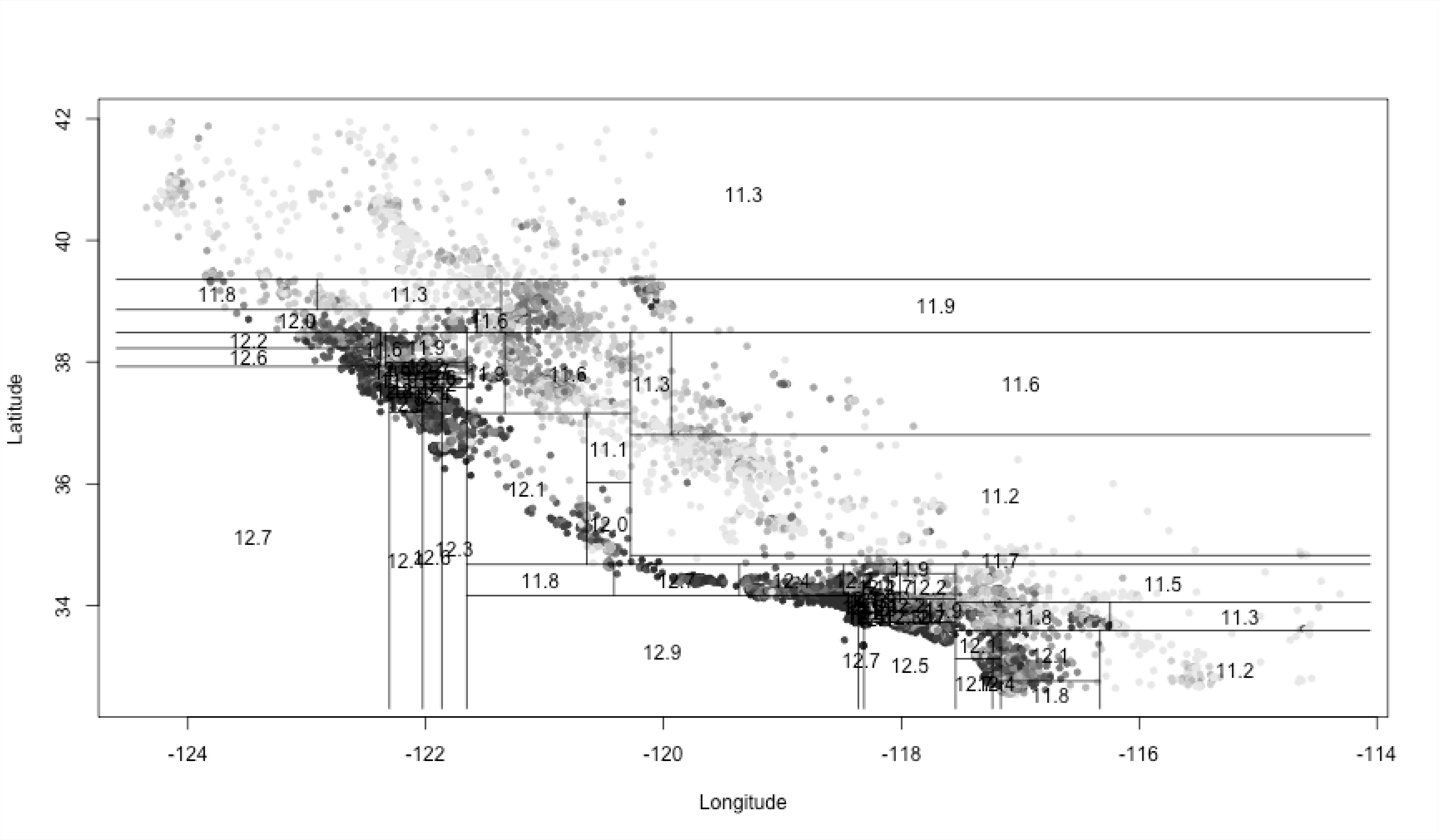
We can increase the fit but changing the stopping criteria with the mindev parameter.

```
treefit2 <- tree(log(MedianHouseValue) ~ Longitude + Latitude, data=calif, mindev=.001)
summary(treefit2)
```

```
##
## Regression tree:
## tree(formula = log(MedianHouseValue) ~ Longitude + Latitude,
##     data = calif, mindev = 0.001)
## Number of terminal nodes:  68
## Residual mean deviance:  0.1052 = 2164 / 20570
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.94700 -0.19790 -0.01872  0.00000  0.19970  1.60600
```

With the larger tree we now have a root-mean-square error of 0.32.

Tree 2, Reduce Minimum Deviance



Tree 3, Include All Variables

However, we can get a better fitting model by including the other variables.

```
treefit3 <- tree(log(MedianHouseValue) ~ ., data=calif)
summary(treefit3)
```

```
##
## Regression tree:
## tree(formula = log(MedianHouseValue) ~ ., data = calif)
## Variables actually used in tree construction:
## [1] "cut.prices"
## Number of terminal nodes:  4
## Residual mean deviance:  0.03608 = 744.5 / 20640
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.718000 -0.127300  0.009245  0.000000  0.130000  0.358600
```

With all the available variables, the root-mean-square error is 0.11.

Classification Trees

Predicting who survived the Titanic.

- `pclass` : Passenger class (1 = 1st; 2 = 2nd; 3 = 3rd)
- `survival` : A Boolean indicating whether the passenger survived or not (0 = No; 1 = Yes); this is our target
- `name` : A field rich in information as it contains title and family names
- `sex` : male/female
- `age` : Age, a significant portion of values are missing
- `sibsp` : Number of siblings/spouses aboard
- `parch` : Number of parents/children aboard
- `ticket` : Ticket number.
- `fare` : Passenger fare (British Pound).
- `cabin` : Does the location of the cabin influence chances of survival?
- `embarked` : Port of embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)
- `boat` : Lifeboat, many missing values
- `body` : Body Identification Number
- `home.dest` : Home/destination

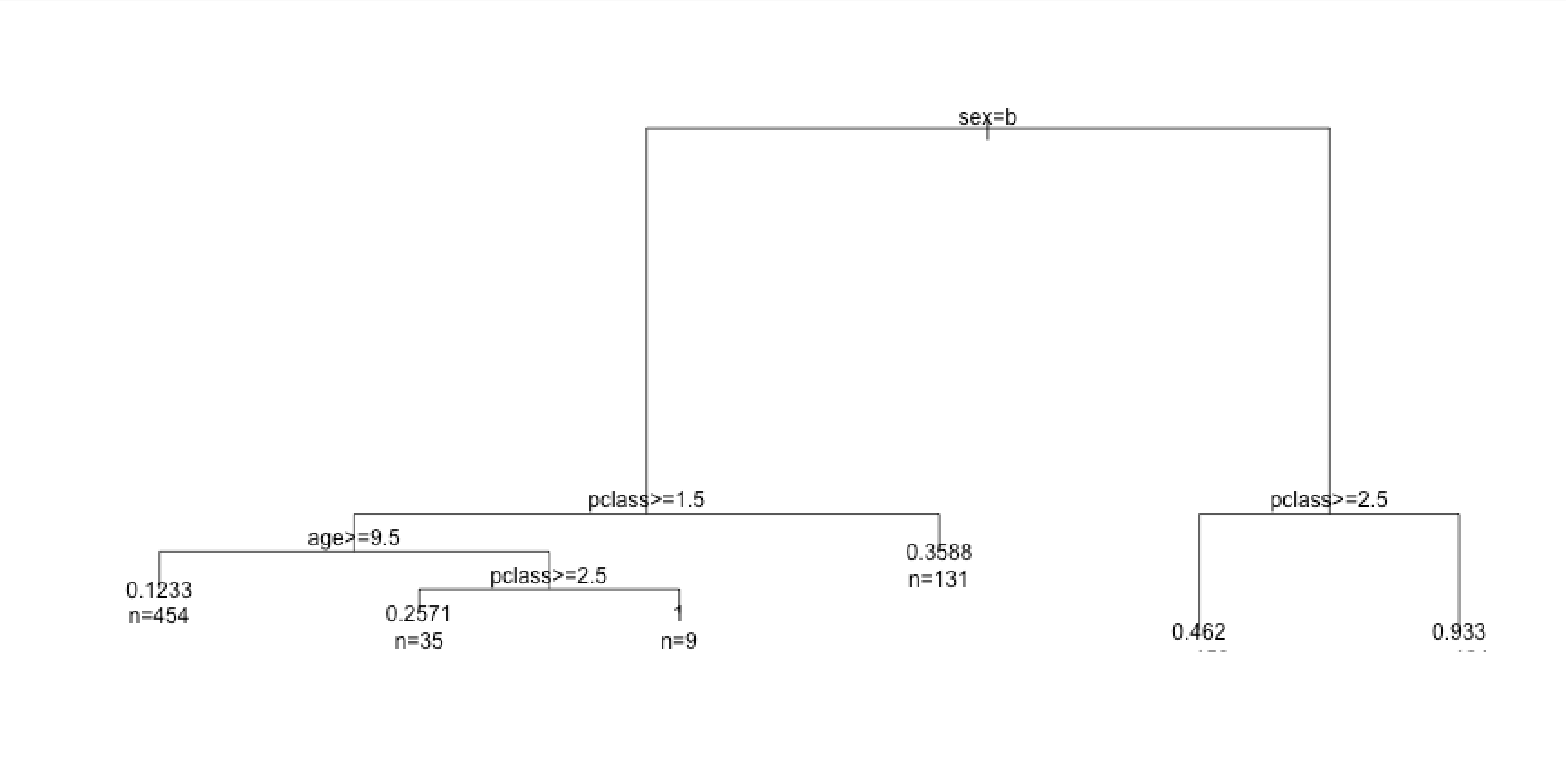
Classification using rpart

```r
(titanic.rpart <- rpart(survived ~ pclass + sex + age + sibsp,
   data=titanic.train))
```

```
## n= 981
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 981 231.651400 0.3822630
##    2) sex=male 629  97.723370 0.1923688
##      4) pclass>=1.5 498  63.004020 0.1485944
##        8) age>=9.5 454  49.092510 0.1233480 *
##        9) age< 9.5 44  10.636360 0.4090909
##         18) pclass>=2.5 35   6.685714 0.2571429 *
##         19) pclass< 2.5 9   0.000000 1.0000000 *
##      5) pclass< 1.5 131  30.137400 0.3587786 *
##    3) sex=female 352  70.715910 0.7215909
##      6) pclass>=2.5 158  39.272150 0.4620253 *
##      7) pclass< 2.5 194  12.128870 0.9329897 *
```

Classification using `rpart`

```
plot(titanic.rpart); text(titanic.rpart, use.n=TRUE, cex=1)
```



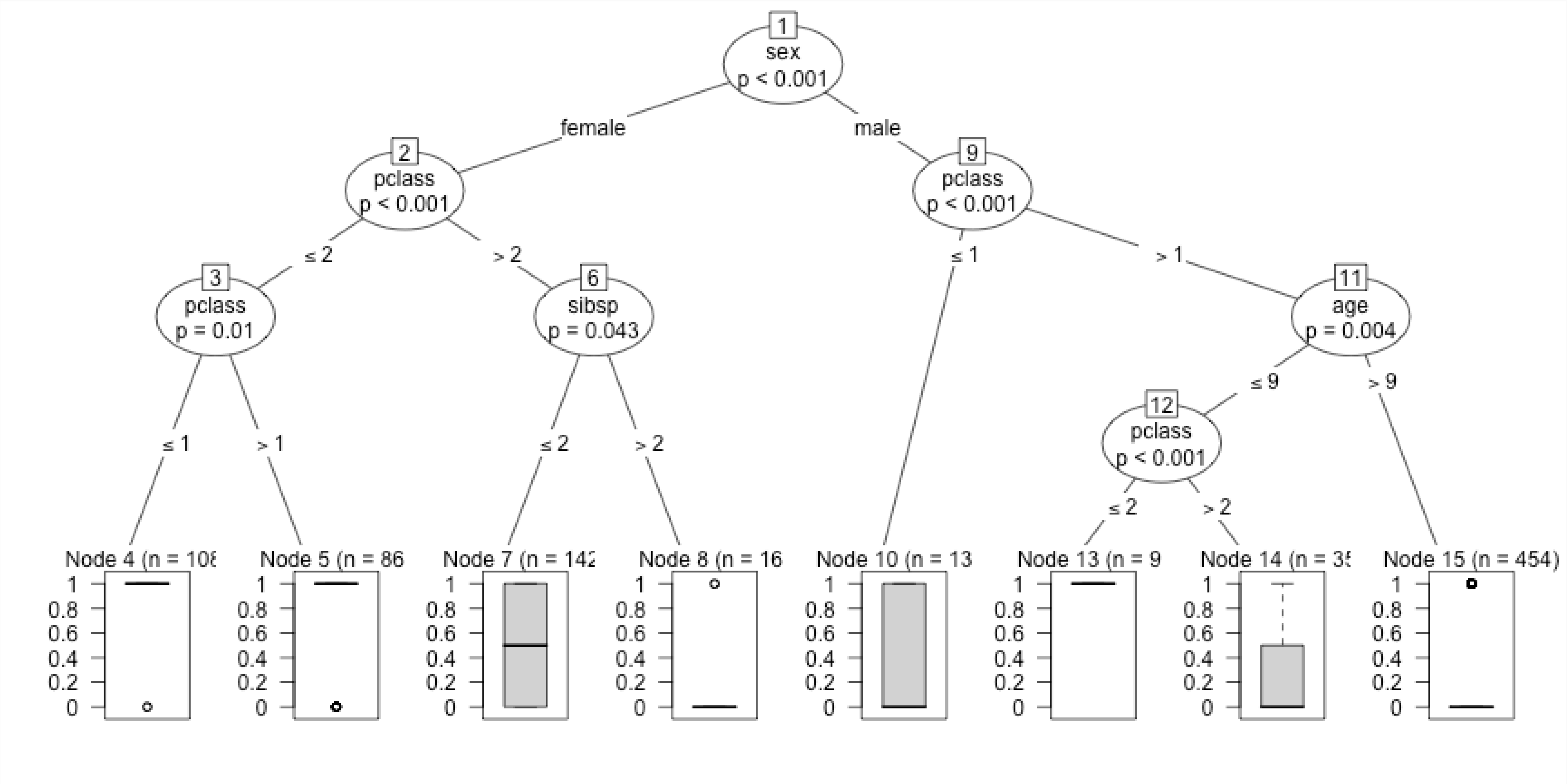
Classification using `ctree`

```r
(titanic.ctree <- ctree(survived ~ pclass + sex + age + sibsp, data=titanic.train))
```

```
##
##      Conditional inference tree with 8 terminal nodes
##
## Response:  survived
## Inputs:  pclass, sex, age, sibsp
## Number of observations:  981
##
## 1) sex == {female}; criterion = 1, statistic = 267.418
##   2) pclass <= 2; criterion = 1, statistic = 91.487
##     3) pclass <= 1; criterion = 0.99, statistic = 9.116
##       4)*  weights = 108
##     3) pclass > 1
##       5)*  weights = 86
##   2) pclass > 2
##     6) sibsp <= 2; criterion = 0.957, statistic = 6.496
##       7)*  weights = 142
##     6) sibsp > 2
##       8)*  weights = 16
## 1) sex == {male}
##   9) pclass <= 1; criterion = 1, statistic = 24.468
##     10)*  weights = 131
```

Classification using `ctree`

```
plot(titanic.ctree)
```



Ensemble Methods

Ensemble methods use multiple models that are combined by weighting, or averaging, each individual model to provide an overall estimate. Each model is a random sample of the sample. Common ensemble methods include:

- *Boosting* - Each successive trees give extra weight to points incorrectly predicted by earlier trees. After all trees have been estimated, the prediction is determined by a weighted "vote" of all predictions (i.e. results of each individual tree model).
- *Bagging* - Each tree is estimated independent of other trees. A simple "majority vote" is take for the prediction.
- *Random Forests* - In addition to randomly sampling the data for each model, each split is selected from a random subset of all predictors.
- *Super Learner* - An ensemble of ensembles. See https://cran.r-project.org/web/packages/SuperLearner/vignettes/Guide-to-SuperLearner.html

Random Forests

The random forest algorithm works as follows:

1. Draw n_{tree} bootstrap samples from the original data.

2. For each bootstrap sample, grow an unpruned tree. At each node, randomly sample m_{try} predictors and choose the best split among those predictors selectedBagging is a special case of random forests where $m_{try} = p$ where *p* is the number of predictors.

3. Predict new data by aggregating the predictions of the ntree trees (majority votes for classification, average for regression).

Error rates are obtained as follows:

1. At each bootstrap iteration predict data not in the bootstrap sample (what Breiman calls "out-of-bag", or OOB, data) using the tree grown with the bootstrap sample.

2. Aggregate the OOB predictions. On average, each data point would be out-of-bag 36% of the times, so aggregate these predictions. The calculated error rate is called the OOB estimate of the error rate.

Random Forests: Titanic

```r
titanic.rf <- randomForest(factor(survived) ~ pclass + sex + age + sibsp,
                           data = titanic.train,
                           ntree = 5000,
                           importance = TRUE)
```

```r
importance(titanic.rf)
```

```
##                0         1 MeanDecreaseAccuracy MeanDecreaseGini
## pclass  98.53696 113.42441             134.5831         45.95406
## sex    231.50133 302.50534             308.4630        129.84180
## age     96.56779  59.30569             121.2686         58.59634
## sibsp   79.02451 -17.28700              62.0706         17.19954
```

Random Forests: Titanic (cont.)

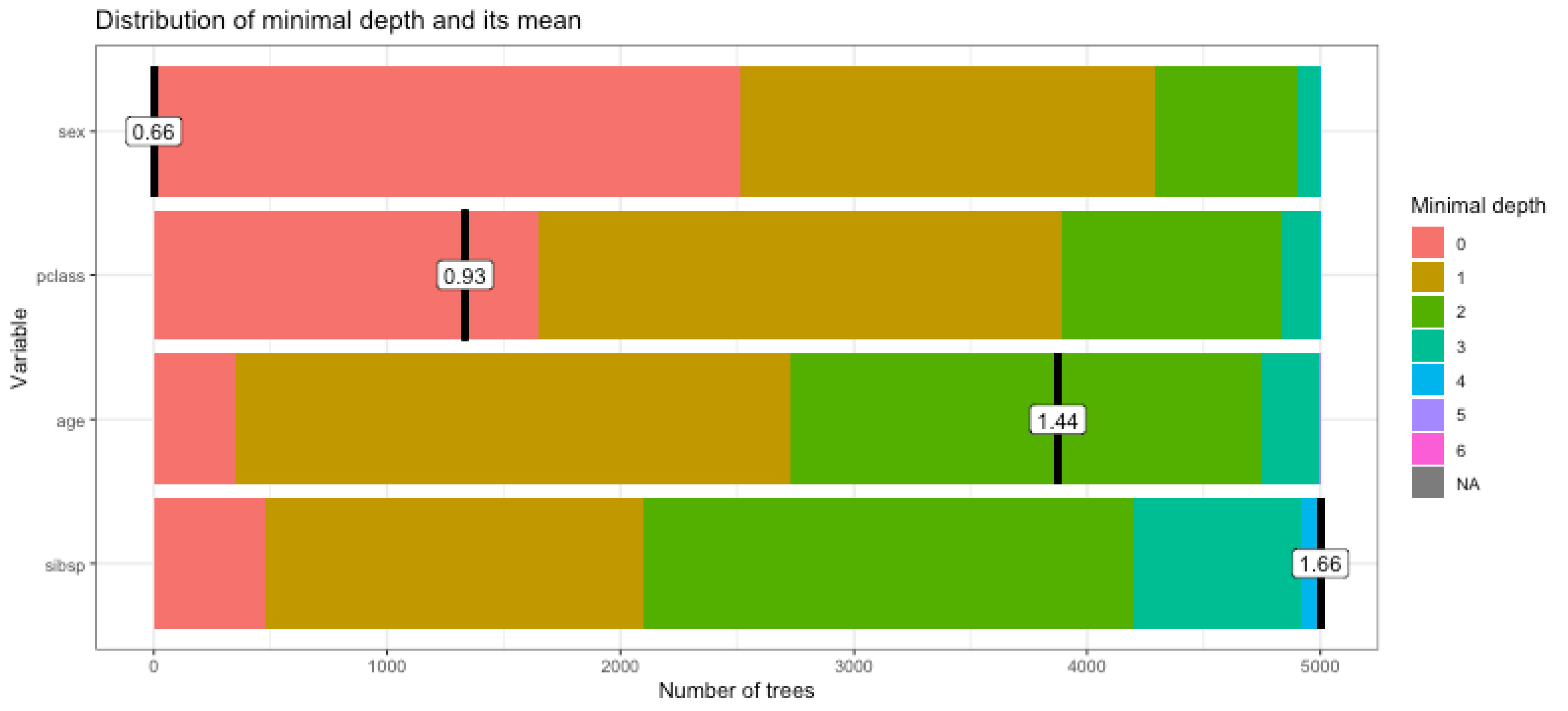
```r
importance(titanic.rf)
```

```
##                0         1 MeanDecreaseAccuracy MeanDecreaseGini
## pclass  98.53696 113.42441             134.5831         45.95406
## sex    231.50133 302.50534             308.4630        129.84180
## age     96.56779  59.30569             121.2686         58.59634
## sibsp   79.02451 -17.28700              62.0706         17.19954
```

Random Forests: Titanic

```
min_depth_frame <- min_depth_distribution(titanic.rf)
```

```
plot_min_depth_distribution(min_depth_frame)
```



Predictive Modeling

Example: Hours Studying Predicting Passing

```r
study <- data.frame(
    Hours=c(0.50,0.75,1.00,1.25,1.50,1.75,1.75,2.00,2.25,2.50,2.75,3.00,
            3.25,3.50,4.00,4.25,4.50,4.75,5.00,5.50),
    Pass=c(0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,1,1,1,1,1)
)
study[sample(nrow(study), 5),]
```

```
##    Hours Pass
## 8   2.00    0
## 16  4.25    1
## 6   1.75    0
## 18  4.75    1
## 20  5.50    1
```

```r
tab <- describeBy(study$Hours, group = study$Pass, mat = TRUE, skew = FALSE)
tab$group1 <- as.integer(as.character(tab$group1))
```

Prediction

Odds (or probability) of passing if studied **zero** hours?

$$log(\frac{p}{1-p}) = -4.078 + 1.505 \times 0$$

$$\frac{p}{1-p} = exp(-4.078) = 0.0169$$

$$p = \frac{0.0169}{1.169} = .016$$

Odds (or probability) of passing if studied **4** hours?

$$log(\frac{p}{1-p}) = -4.078 + 1.505 \times 4$$

$$\frac{p}{1-p} = exp(1.942) = 6.97$$

$$p = \frac{6.97}{7.97} = 0.875$$

Fitted Values

```
study[1,]
```

```
##   Hours Pass
## 1   0.5    0
```

```
logistic <- function(x, b0, b1) {
    return(1 / (1 + exp(-1 * (b0 + b1 * x)) ))
}
logistic(.5, b0=-4.078, b1=1.505)
```

```
## [1] 0.03470667
```

Model Performance

The use of statistical models to predict outcomes, typically on new data, is called predictive modeling. Logistic regression is a common statistical procedure used for prediction. We will utilize a **confusion matrix** to evaluate accuracy of the predictions.

		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR–}}$	F_1 score $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR–) $= \frac{\text{FNR}}{\text{TNR}}$		

Predicting Heart Attacks

Source: https://www.kaggle.com/datasets/imnikhilanand/heart-attack-prediction?select=data.csv

```
heart <- read.csv('../course_data/heart_attack_predictions.csv')
heart <- heart |>
    mutate_if(is.character, as.numeric) |>
    select(!c(slope, ca, thal))
str(heart)
```

```
## 'data.frame':    294 obs. of  11 variables:
##  $ age     : int  28 29 29 30 31 32 32 32 33 34 ...
##  $ sex     : int  1 1 1 0 0 0 1 1 1 0 ...
##  $ cp      : int  2 2 2 1 2 2 2 2 3 2 ...
##  $ trestbps: num  130 120 140 170 100 105 110 125 120 130 ...
##  $ chol    : num  132 243 NA 237 219 198 225 254 298 161 ...
##  $ fbs     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ restecg : num  2 0 0 1 1 0 0 0 0 0 ...
##  $ thalach : num  185 160 170 170 150 165 184 155 185 190 ...
##  $ exang   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ oldpeak : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num     : int  0 0 0 0 0 0 0 0 0 0 ...
```

Note: `num` is the diagnosis of heart disease (angiographic disease status) (i.e. Value 0: < 50% diameter narrowing -- Value 1: > 50% diameter narrowing)

Missing Data

We will save this for another day...

```
complete.cases(heart) |> table()
```

```
##
## FALSE  TRUE
##    33   261
```

```
mice_out <- mice::mice(heart, m = 1)
```

```
##
##  iter imp variable
##   1   1  trestbps  chol  fbs  restecg  thalach  exang
##   2   1  trestbps  chol  fbs  restecg  thalach  exang
##   3   1  trestbps  chol  fbs  restecg  thalach  exang
##   4   1  trestbps  chol  fbs  restecg  thalach  exang
##   5   1  trestbps  chol  fbs  restecg  thalach  exang
```

```
heart <- mice::complete(mice_out)
```

Data Setup

We will split the data into a training set (70% of observations) and validation set (30%).

```
train.rows <- sample(nrow(heart), nrow(heart) * .7)
heart_train <- heart[train.rows,]
heart_test <- heart[-train.rows,]
```

This is the proportions of survivors and defines what our "guessing" rate is. That is, if we guessed no one had a heart attack, we would be correct 62% of the time.

```
(heart_attack <- table(heart_train$num) %>% prop.table)
```

```
##
##         0        1
## 0.604878 0.395122
```

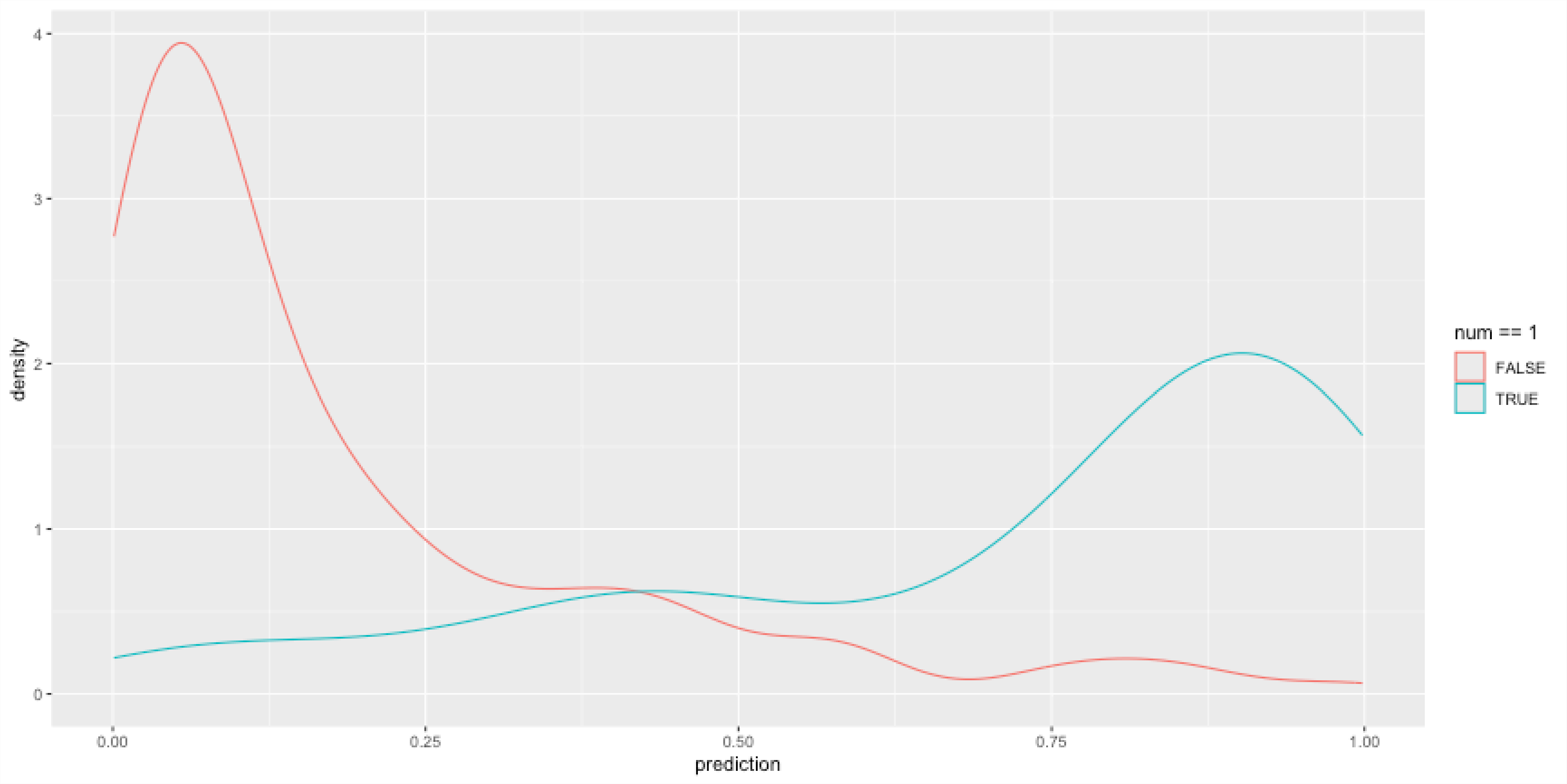
Model Training

```r
lr.out <- glm(num ~ ., data=heart_train, family=binomial(link = 'logit'))
summary(lr.out)
```

```
##
## Call:
## glm(formula = num ~ ., family = binomial(link = "logit"), data = heart_train)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.895542   3.311490  -1.780  0.07502 .
## age          0.033586   0.033259   1.010  0.31257
## sex          1.422810   0.587957   2.420  0.01552 *
## cp           1.059011   0.264157   4.009  6.1e-05 ***
## trestbps    -0.016868   0.013816  -1.221  0.22212
## chol         0.007188   0.003313   2.170  0.03001 *
## fbs          1.474454   0.984983   1.497  0.13441
## restecg     -0.467892   0.551265  -0.849  0.39601
## thalach     -0.008747   0.010904  -0.802  0.42244
## exang        1.182195   0.581334   2.034  0.04199 *
## oldpeak      1.056386   0.324408   3.256  0.00113 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 275.10  on 204  degrees of freedom
## Residual deviance: 148.47  on 194  degrees of freedom
## AIC: 170.47
##
```

Predicted Values

```
heart_train$prediction <- predict(lr.out, type = 'response', newdata = heart_train)
ggplot(heart_train, aes(x = prediction, color = num == 1)) + geom_density()
```



Results

```r
heart_train$prediction_class <- heart_train$prediction > 0.5
tab <- table(heart_train$prediction_class,
             heart_train$num) %>% prop.table() %>% print()
```

```
##
##                  0          1
##   FALSE 0.55121951 0.10243902
##   TRUE  0.05365854 0.29268293
```

For the training set, the overall accuracy is 84.39%. Recall that 60.49% people did not have a heart attach. Therefore, the simplest model would be to predict that no one had a heart attack, which would mean we would be correct 60.49% of the time. Therefore, our prediction model is 23.9% better than guessing.

Checking with the validation dataset

```
(survived_test <- table(heart_test$num) %>% prop.table())
```

```
##
##         0         1
## 0.7191011 0.2808989
```

```
heart_test$prediction <- predict(lr.out, newdata = heart_test, type = 'response')
heart_test$prediciton_class <- heart_test$prediction > 0.5
tab_test <- table(heart_test$prediciton_class, heart_test$num) %>%
    prop.table() %>% print()
```

```
##
##                  0          1
##   FALSE 0.59550562 0.05617978
##   TRUE  0.12359551 0.22471910
```

The overall accuracy is 82.02%, or 10.1% better than guessing.

Receiver Operating Characteristic (ROC) Curve

The ROC curve is created by plotting the true positive rate (TPR; AKA sensitivity) against the false positive rate (FPR; AKA probability of false alarm) at various threshold settings.

In a classification model, outcomes are either as positive (*p*) or negative (*n*). There are then four possible outcomes:

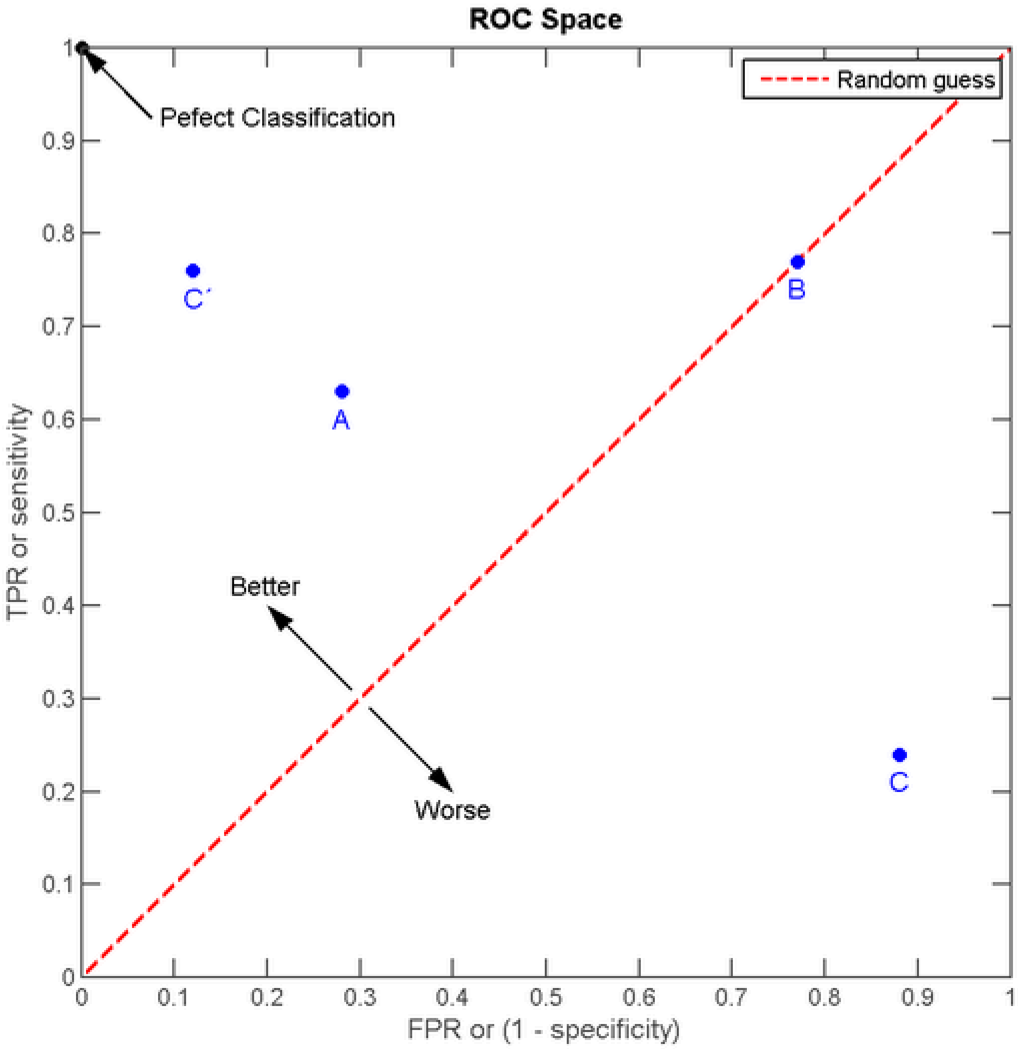
- **true positive** (TP) The outcome from a prediction is *p* and the actual value is also *p*.
- **false positive** (FP) The actual value is *n*.
- **true negative** (TN) Both the prediction outcome and the actual value are *n*.
- **false negative** (FN) The prediction outcome is *n* while the actual value is *p*.

		actual value: *p*	actual value: *n*	total
prediction outcome	*p'*	True Positive	False Positive	P'
	n'	False Negative	True Negative	N'
total		P	N	

```
roc <- calculate_roc(heart_train$prediction,
                     heart_train$num == 1)
summary(roc)
```

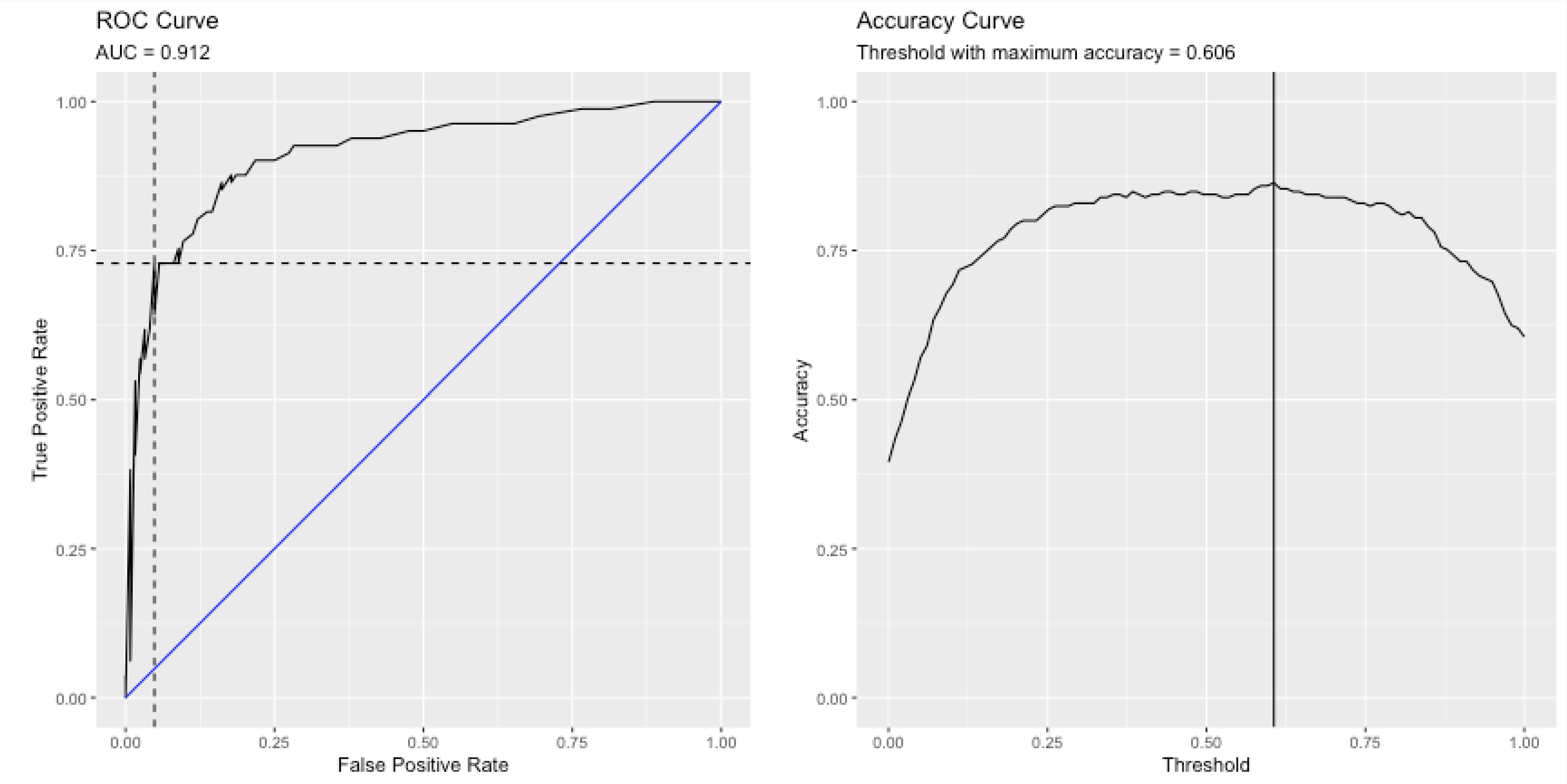
```
## AUC = 0.912
## Cost of false-positive = 1
## Cost of false-negative = 1
## Threshold with minimum cost = 0.606
```

ROC Curve


ROC Space
Random guess
Pefect Classification
C'
A
B
C
Better
Worse
TPR or sensitivity
FPR or (1 - specificity)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

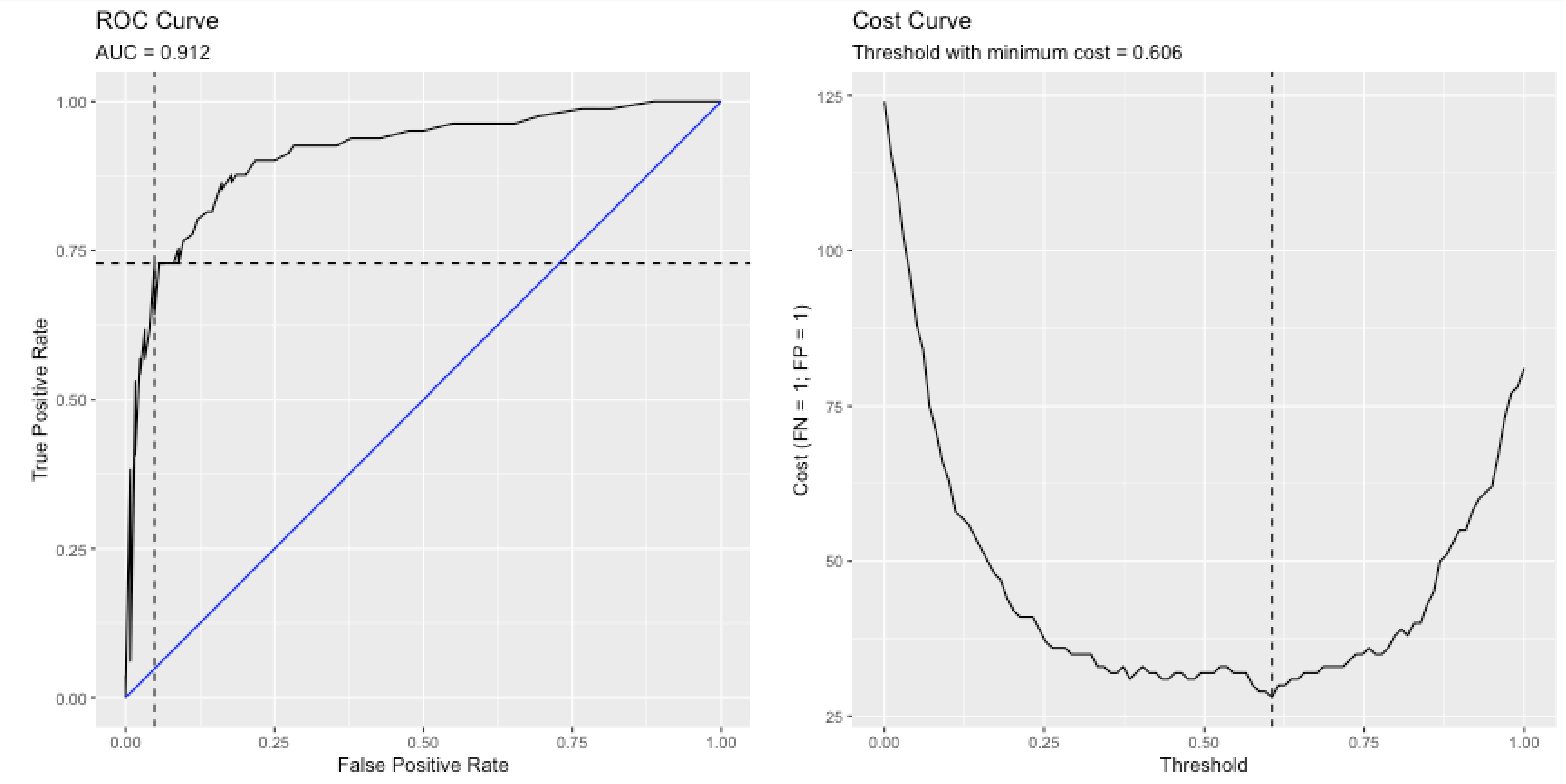
ROC Curve

```
plot(roc, curve = 'accuracy')
```



ROC Curve

```
plot(roc)
```



Caution on Interpreting Accuracy

- Loh, Sooo, and Zing (2016) predicted sexual orientation based on Facebook Status.
- They reported model accuracies of approximately 90% using SVM, logistic regression and/or random forest methods.
- Gallup (2018) poll estimates that 4.5% of the Americal population identifies as LGBT.
- *My proposed model:* I predict all Americans are heterosexual.
- The accuracy of my model is 95.5%, or *5.5% better than Facebook's model!*
- Predicting "rare" events (i.e. when the proportion of one of the two outcomes large) is difficult and requires independent (predictor) variables that strongly associated with the dependent (outcome) variable.

Fitted Values Revisited

What happens when the ratio of true-to-false increases (i.e. want to predict "rare" events)?

Let's simulate a dataset where the ratio of true-to-false is 10-to-1. We can also define the distribution of the dependent variable. Here, there is moderate separation in the distributions.

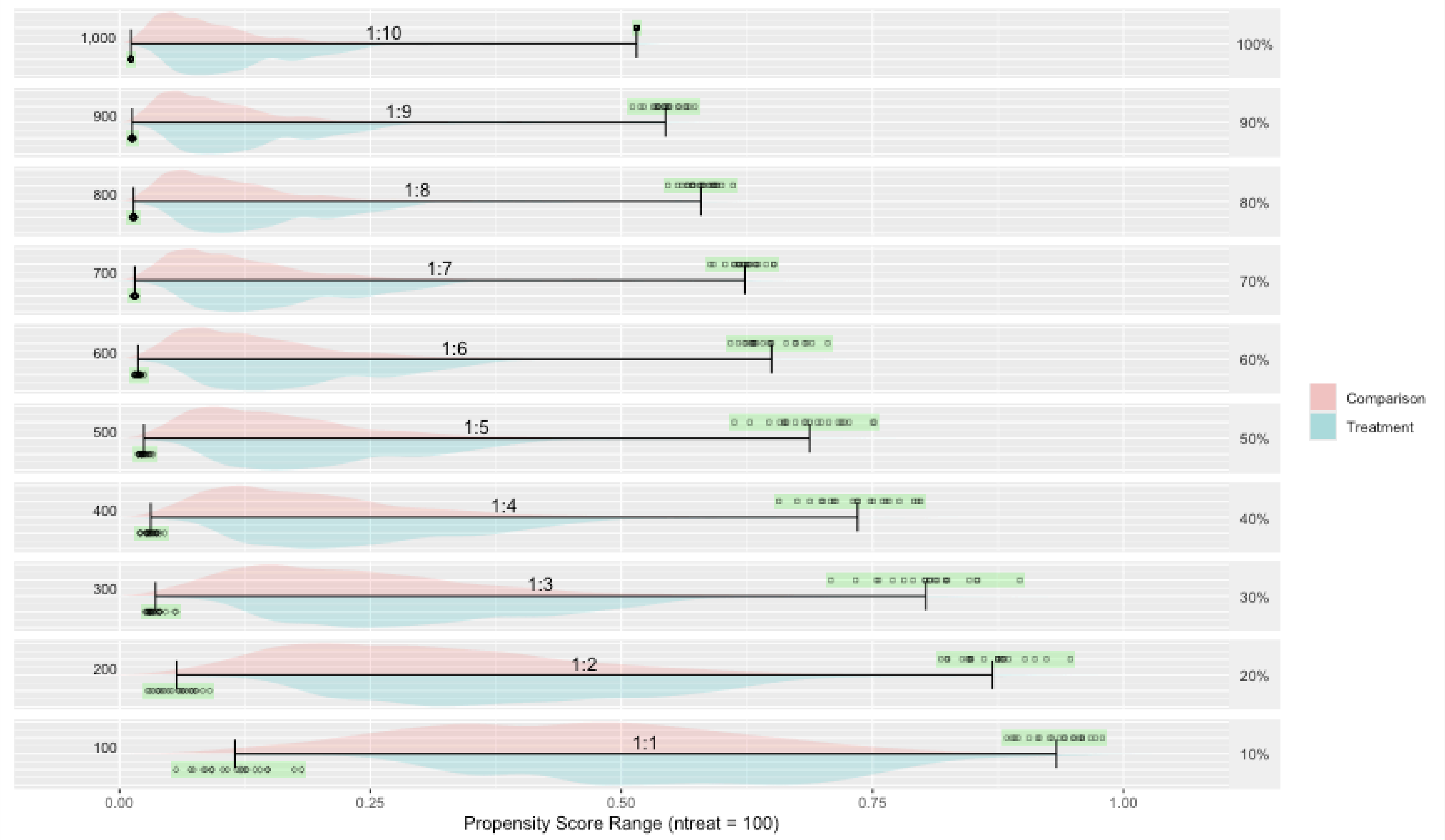
```
test.df2 <- getSimulatedData(
    treat.mean=.6, control.mean=.4)
```

The `multilevelPSA::psrange` function will sample with varying ratios from 1:10 to 1:1. It takes multiple samples and averages the ranges and distributions of the fitted values from logistic regression.

```
psranges2 <- psrange(test.df2, test.df2$treat, treat ~ .,
                     samples=seq(100,1000,by=100), nboot=20)
```

Fitted Values Revisited (cont.)

```
plot(psranges2)
```



One Minute Paper

1. What was the most important thing you learned during this class?
2. What important question remains unanswered for you?

https://forms.gle/Jcw55CYvc6Ym8A5F7