Bayesian Analysis and Predictive Modeling

Computational Mathematics and Statistics

Jason Bryer, Ph.D.

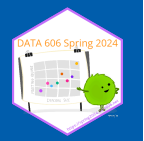
November 19, 2024

One Minute Paper Results

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

What important question remains unanswered for you?

Bayesian Analysis



Bayesian Analysis

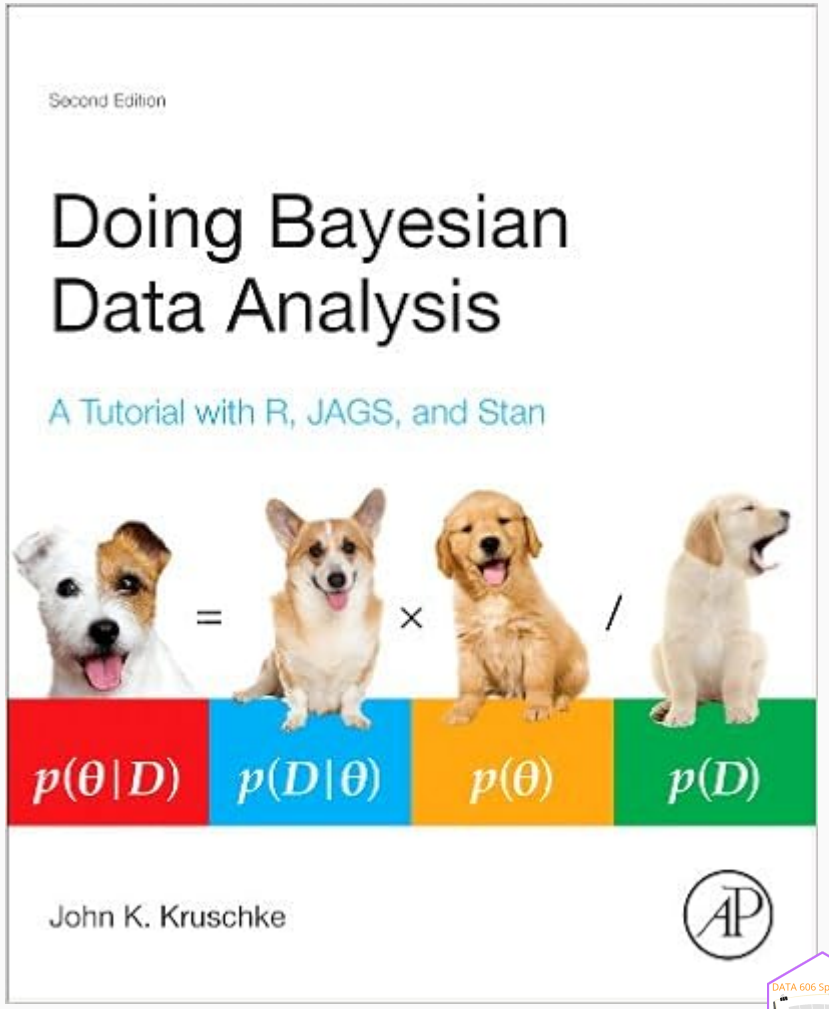
Kruschke's videos are an excelent introduction to Bayesian Analysis https://www.youtube.com/watch?v=YyohWpjl6KU!

Doing Bayesian Data Analysis, Second Edition: A Tutorial with R, JAGS, and Stan

The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, and Emerged Triumphant from Two Centuries of Controversy by Sharon Bertsch McGrayne

Video series by Rasmus Baath Part 1, Part 2, Part 3

Billiards with Fred the Frequentist and Bayer the Bayesian



Bayes Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|A^{'})P(A^{'})}$$

Consider the following data from a cancer test:

- 1% of women have breast cancer (and therefore 99% do not).
- 80% of mammograms detect breast cancer when it is there (and therefore 20% miss it).
- 9.6% of mammograms detect breast cancer when it's not there (and therefore 90.4% correctly return a negative result).

	Cancer (1%)	**No Cancer (99%)**
Test postive	80%	9.6%
Test negative	20%	90.4%

How accurate is the test?

Now suppose you get a positive test result. What are the chances you have cancer?
80%? 99%? 1%?

- Ok, we got a positive result. It means we're somewhere in the top row of our table. Let's not assume anything - it could be a true positive or a false positive.
- The chances of a true positive = chance you have cancer *chance test caught it = 1%* 80% = .008
- The chances of a false positive = chance you don't have cancer *chance test caught it anyway = 99%* 9.6% = 0.09504

	Cancer (1%)	**No Cancer (99%)**	
Test postive	True +: 1% * 80%	False +: 99% * 9.6%	**10.304%**
Test negative	False -: 1% * 20%	True -: 99% * 90.4%	**89.696%**

How accurate is the test?

$$Probability = \frac{desired \quad event}{all \quad possibilities}$$

The chance of getting a real, positive result is .008. The chance of getting any type of positive result is the chance of a true positive plus the chance of a false positive (.008 + 0.09504 = .10304).

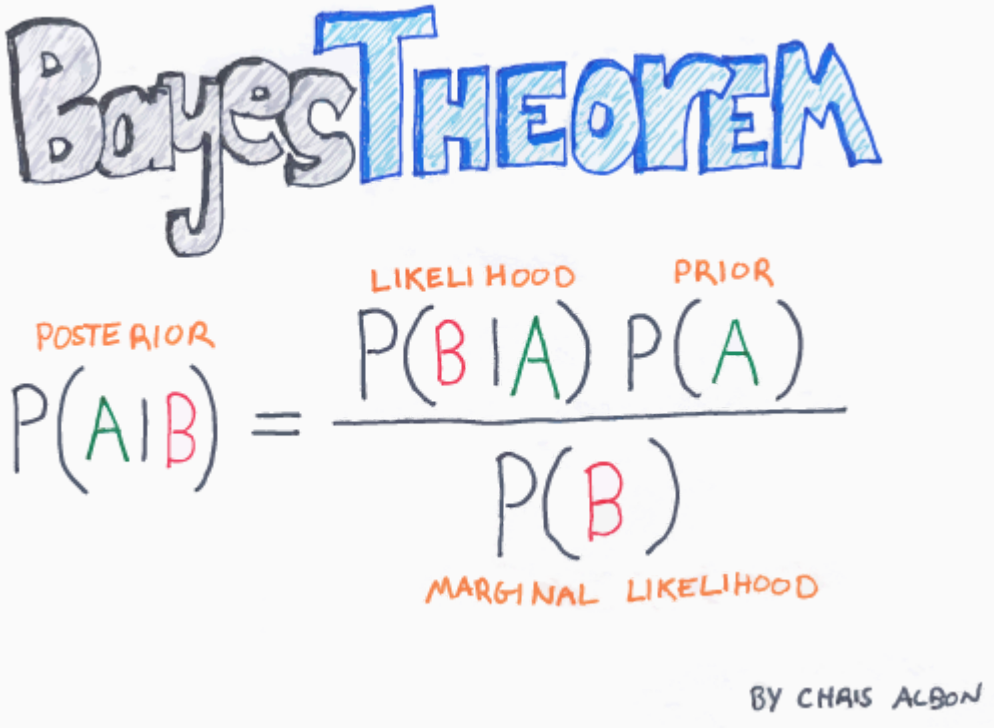
$$P(C|P) = \frac{P(P|C)P(C)}{P(P)} = \frac{.8 * .01}{.008 + 0.095} \approx .078$$

So, our chance of cancer is .008/.10304 = 0.0776, or about 7.8%.

Bayes Formula

It all comes down to the chance of a true positive result divided by the chance of any positive result. We can simplify the equation to:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$


Bayes Theorem
POSTERIOR
LIKELIHOOD
PRIOR
P(A|B) = P(B|A) P(A) / P(B)
MARGINAL LIKELIHOOD
BY CHRIS ALBON

How many fish are in the lake?

- Catch them all, count them. Not practical (or even possible)!
- We can sample some fish.

Our strategy:

1. Catch some fish.
2. Mark them.
3. Return the fish to the pond. Let them get mixed up (i.e. wait a while).
4. Catch some more fish.
5. Count how many are marked.

For example, we initially caught 20 fish, marked them, returned them to the pond. We then caught another 20 fish and 5 of them were marked (i.e they were caught the first time).

Adopted from Rasmath Bääth useR! 2015 workshop: http://www.sumsar.net/files/academia/user_2015_tutorial_bayesian_data_analysis_short_version.pdf

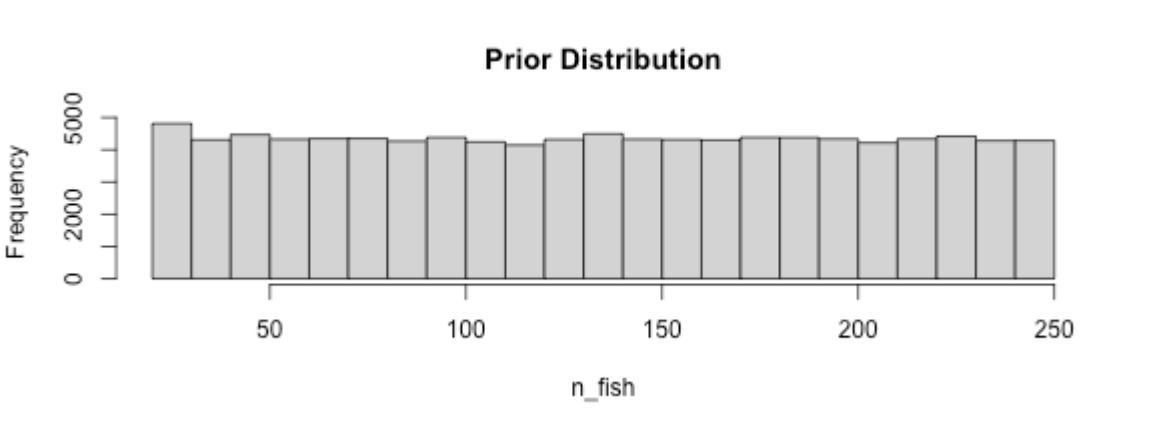
Strategy for fitting a model

Step 1: Define Prior Distribution. Draw a lot of random samples from the "prior" probability distribution on the parameters.

```
n_draw <- 100000
n_fish <- sample(20:250, n_draw, replace = TRUE)
head(n_fish, n=10)
```

```
##  [1] 157  20 106  32 227 100  34 201  93 120
```

```
hist(n_fish, main="Prior Distribution")
```



Strategy for fitting a model

Step 2: Plug in each draw into the generative model which generates "fake" data.

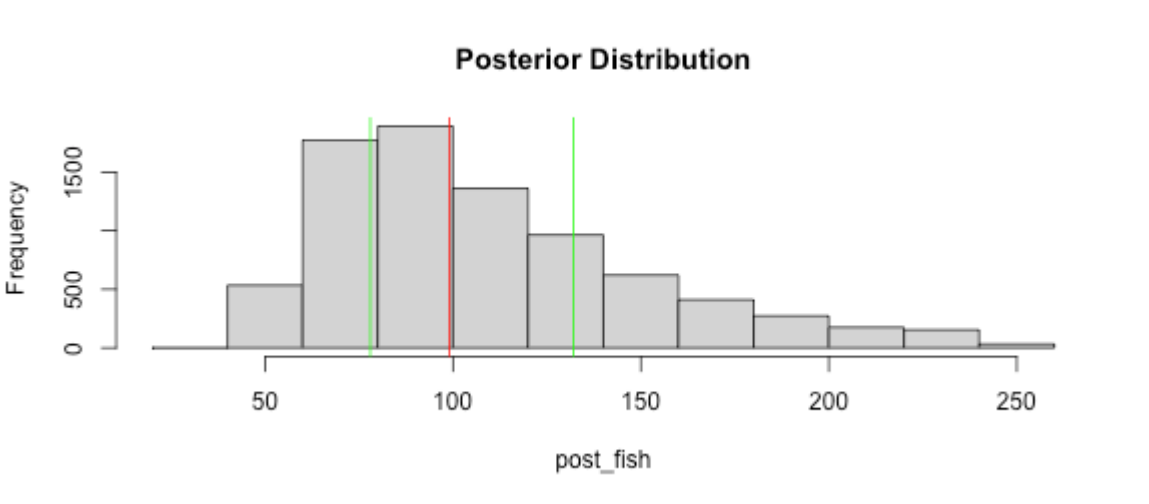
```
pick_fish <- function(n_fish) { # The generative model
    fish <- rep(0:1, c(n_fish - 20, 20))
    sum(sample(fish, 20))
}
n_marked <- rep(NA, n_draw)
for(i in 1:n_draw) {
    n_marked[i] <- pick_fish(n_fish[i])
}
head(n_marked, n=10)
```

```
##  [1]  1 20  1 15  0  5 12  3  2  1
```

Strategy for fitting a model

Step 3: Keep only those parameter values that generated the data that was actually observed (in this case, 5).

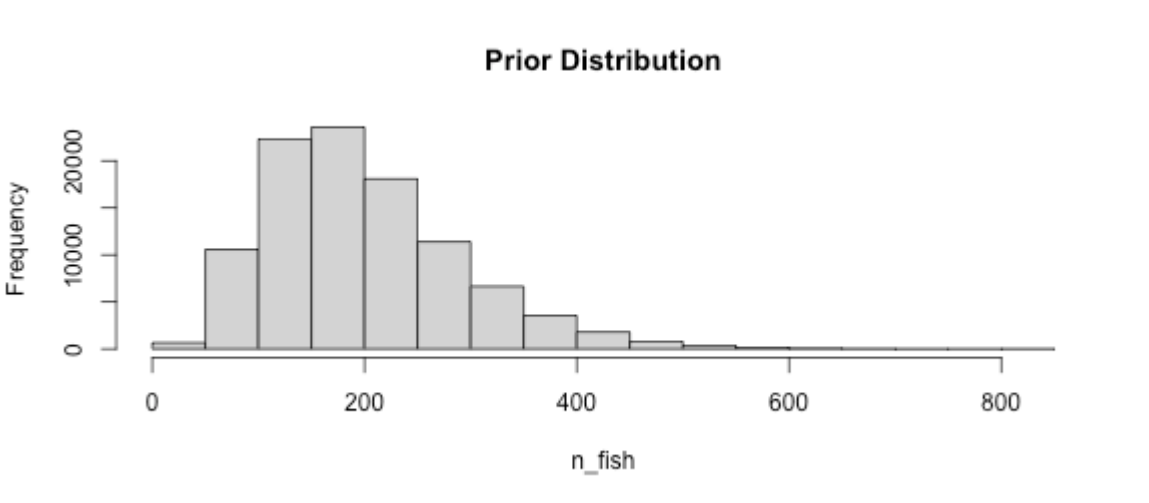
```
post_fish <- n_fish[n_marked == 5]
hist(post_fish, main='Posterior Distribution')
abline(v=median(post_fish), col='red')
abline(v=quantile(post_fish, probs=c(.25, .75)), col='green')
```



What if we have better prior information?

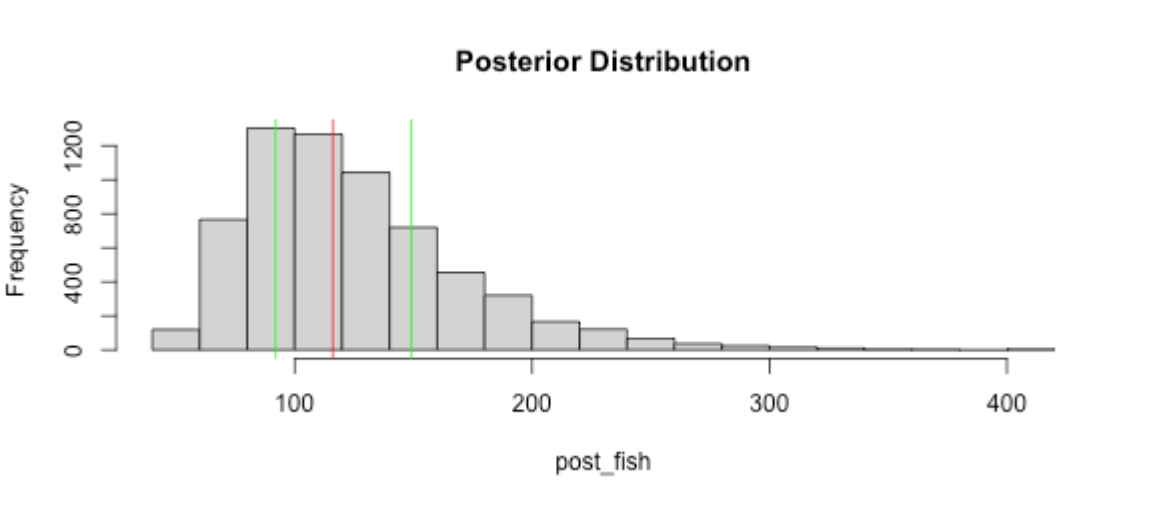
An "expert" believes there are around 200 fish in the pond. Insteand of a uniform distribution, we can use a binomial distribution to define our "prior" distribution.

```
n_fish <- rnbinom(n_draw, mu = 200 - 20, size = 4) + 20
hist(n_fish, main='Prior Distribution')
```



What if we have better prior information?

```
n_marked <- rep(NA, n_draw)
for(i in 1:n_draw) {
    n_marked[i] <- pick_fish(n_fish[i])
}
post_fish <- n_fish[n_marked == 5]
hist(post_fish, main='Posterior Distribution')
abline(v=median(post_fish), col='red')
abline(v=quantile(post_fish, probs=c(.25, .75)), col='green')
```



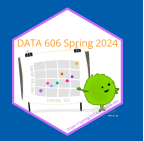
Bayes Billiards Balls

Consider a pool table of length one. An 8-ball is thrown such that the likelihood of its stopping point is uniform across the entire table (i.e. the table is perfectly level). The location of the 8-ball is recorded, but not known to the observer. Subsequent balls are thrown one at a time and all that is reported is whether the ball stopped to the left or right of the 8-ball. Given only this information, what is the position of the 8-ball? How does the estimate change as more balls are thrown and recorded?

```
DATA606::shiny_demo('BayesBilliards', package='DATA606')
```

See also: http://www.bryer.org/post/2016-02-21-bayes_billiards_shiny/

Predictive Modeling



Example: Hours Studying Predicting Passing

```
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
## 10  2.50    0
## 17  4.50    1
## 18  4.75    1
## 5   1.50    0
## 13  3.25    1
```

```
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
```

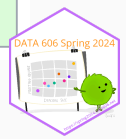
```
logistic <- function(x, b0, b1) {
    return(1 / (1 + exp(-1 * (b0 + b1 * x)) ))
}
logistic(.5, b0=-4.078, b1=1.505)
```

```
## [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−) = $\frac{\text{FNR}}{\text{TNR}}$		



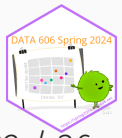
Predicting Heart Attacks

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...

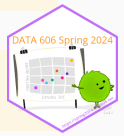
```
complete.cases(heart) |> table()
```

```
##
## 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.

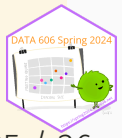
```
(heart_attack <- table(heart_train$num) %>% prop.table)
```

```
##
##         0         1
## 0.6195122 0.3804878
```

Model Training

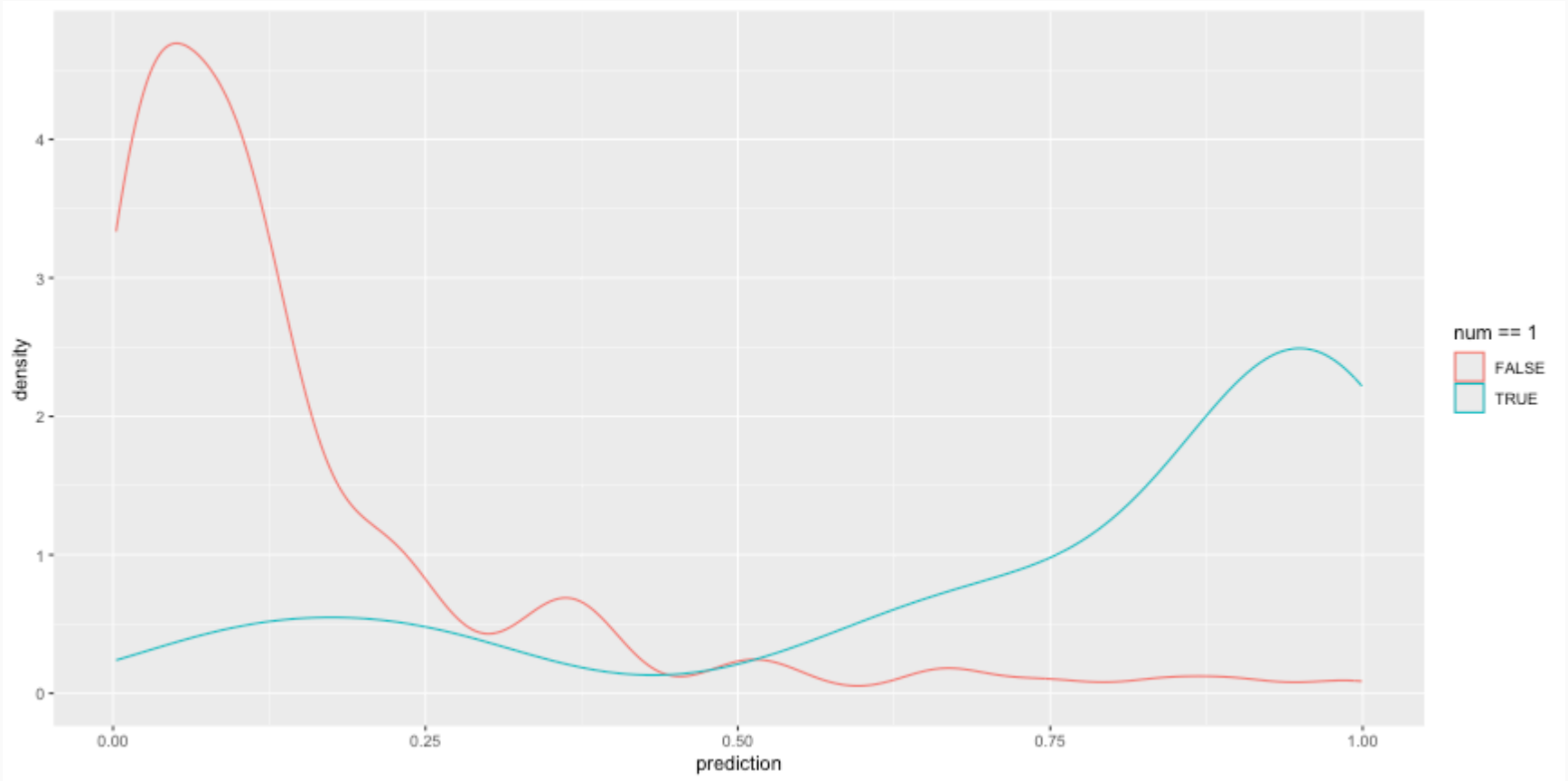
```
lr.out <- glm(num ~ ., data=heart_train, family=binomial(link = 'logit'))
summary(lr.out)
```

```
##
## Call:
## glm(formula = num ~ ., family = binomial(link = "logit"), data = heart_train)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.4376133  3.7007438  -1.199  0.23048
## age         -0.0142032  0.0351409  -0.404  0.68608
## sex          1.5565850  0.6263848   2.485  0.01295 *
## cp           0.5689681  0.2744767   2.073  0.03818 *
## trestbps    -0.0006805  0.0132357  -0.051  0.95899
## chol         0.0056677  0.0038395   1.476  0.13990
## fbs          1.7388039  0.9186366   1.893  0.05838 .
## restecg     -1.0103190  0.5983549  -1.688  0.09132 .
## thalach     -0.0079635  0.0123780  -0.643  0.51999
## exang        1.9721121  0.6105352   3.230  0.00124 **
## oldpeak      1.4684462  0.3439190   4.270 1.96e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 272.36  on 204  degrees of freedom
## Residual deviance: 128.00  on 194  degrees of freedom
## AIC: 150
##
```



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

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

```
##
##                    0          1
##    FALSE 0.57560976 0.06829268
##    TRUE  0.04390244 0.31219512
```

For the training set, the overall accuracy is 88.78%. Recall that 61.95% 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 61.95% of the time. Therefore, our prediction model is 26.83% better than guessing.

Checking with the validation dataset

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

```
##
##         0         1
## 0.6853933 0.3146067
```

```
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 0.5730337 0.1235955
##    TRUE  0.1123596 0.1910112
```

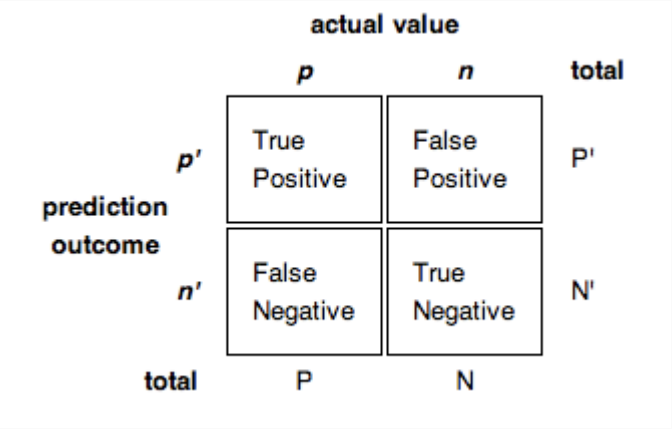
The overall accuracy is 76.4%, or 7.9% better than guessing.

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:

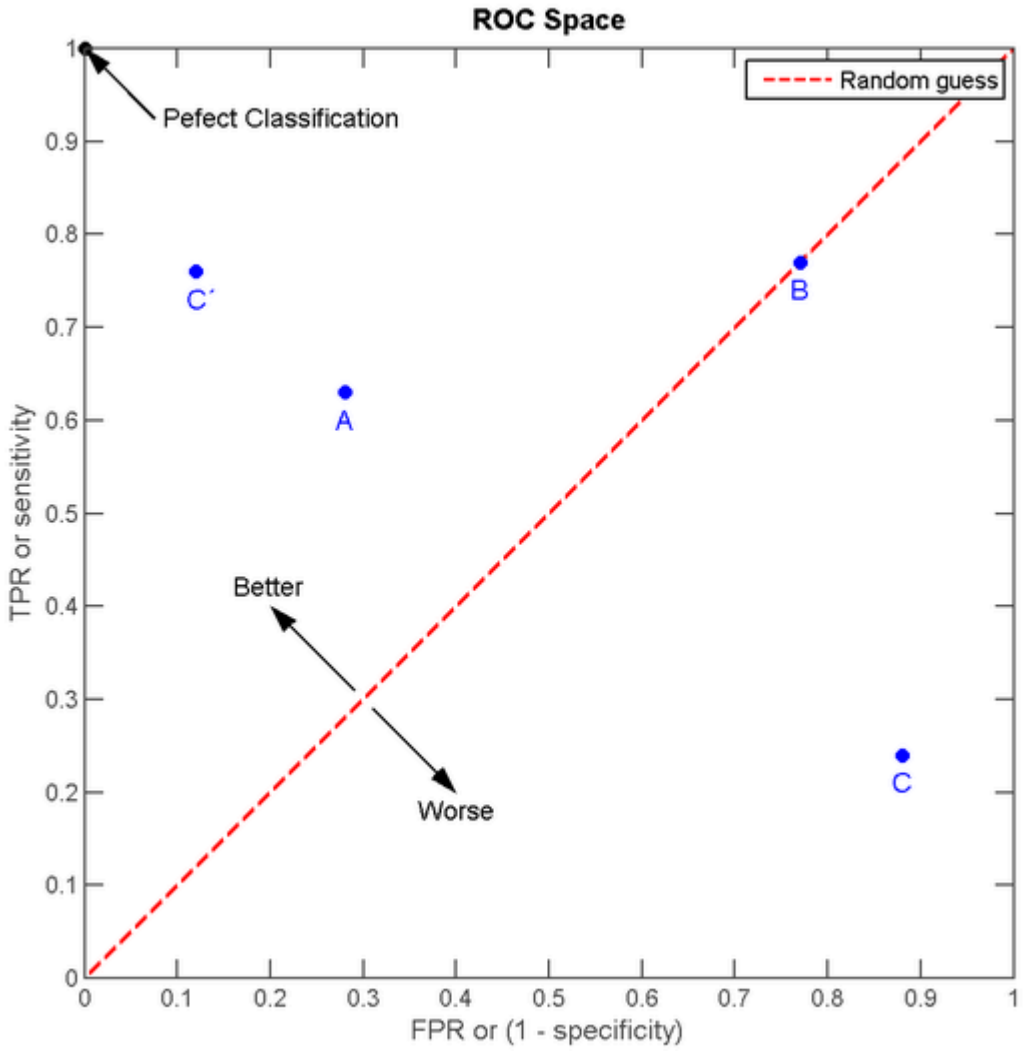
- **true positive** (TP) The outcome from a prediction is *p* and the actual value is also *p*.
- **false positive** (FP) The actual value is *n*.
- **true negative** (TN) Both the prediction outcome and the actual value are *n*.
- **false negative** (FN) The prediction outcome is *n* while the actual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




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

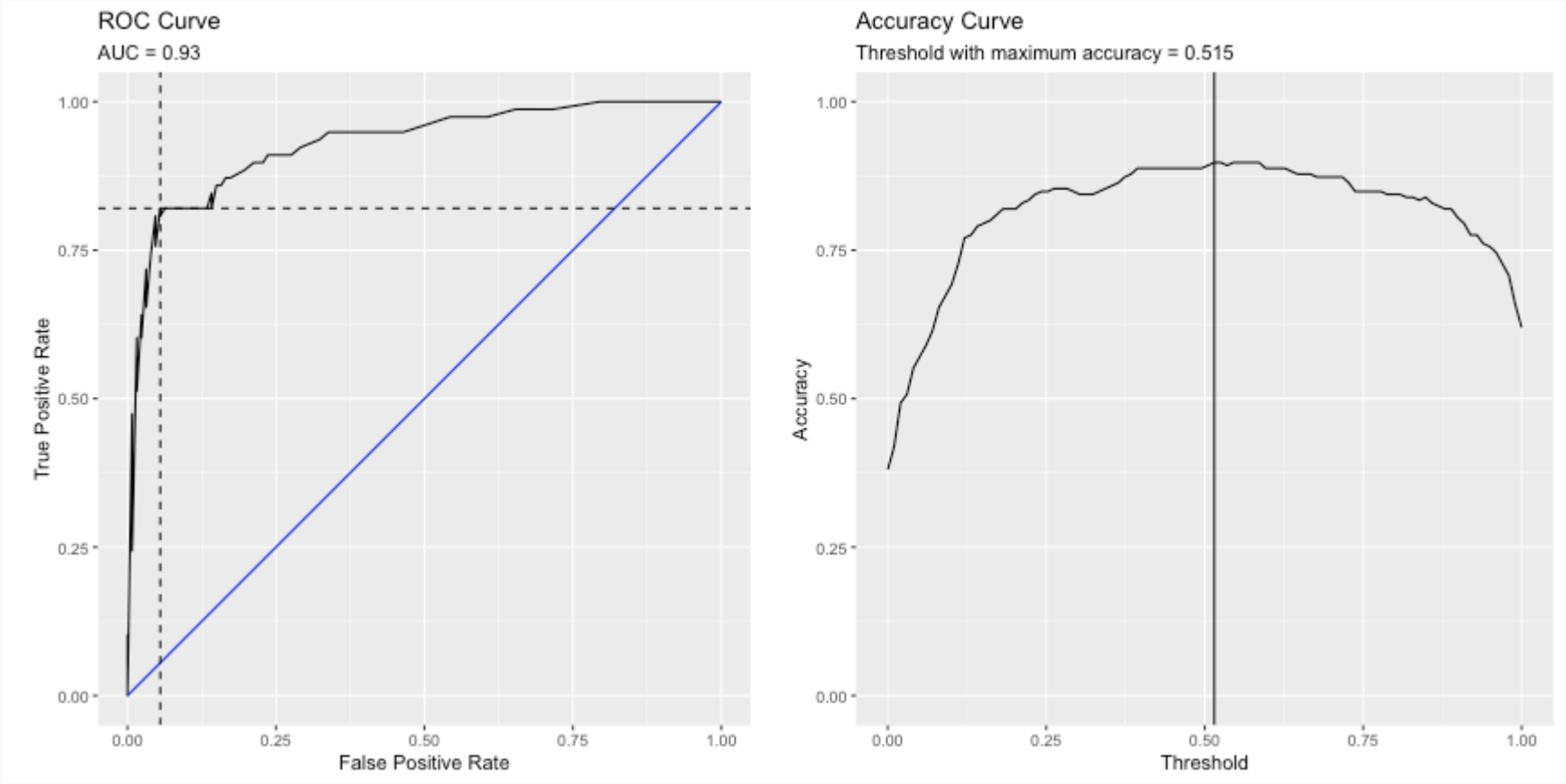
```
## AUC = 0.93
## Cost of false-positive = 1
## Cost of false-negative = 1
## Threshold with minimum cost = 0.515
```

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

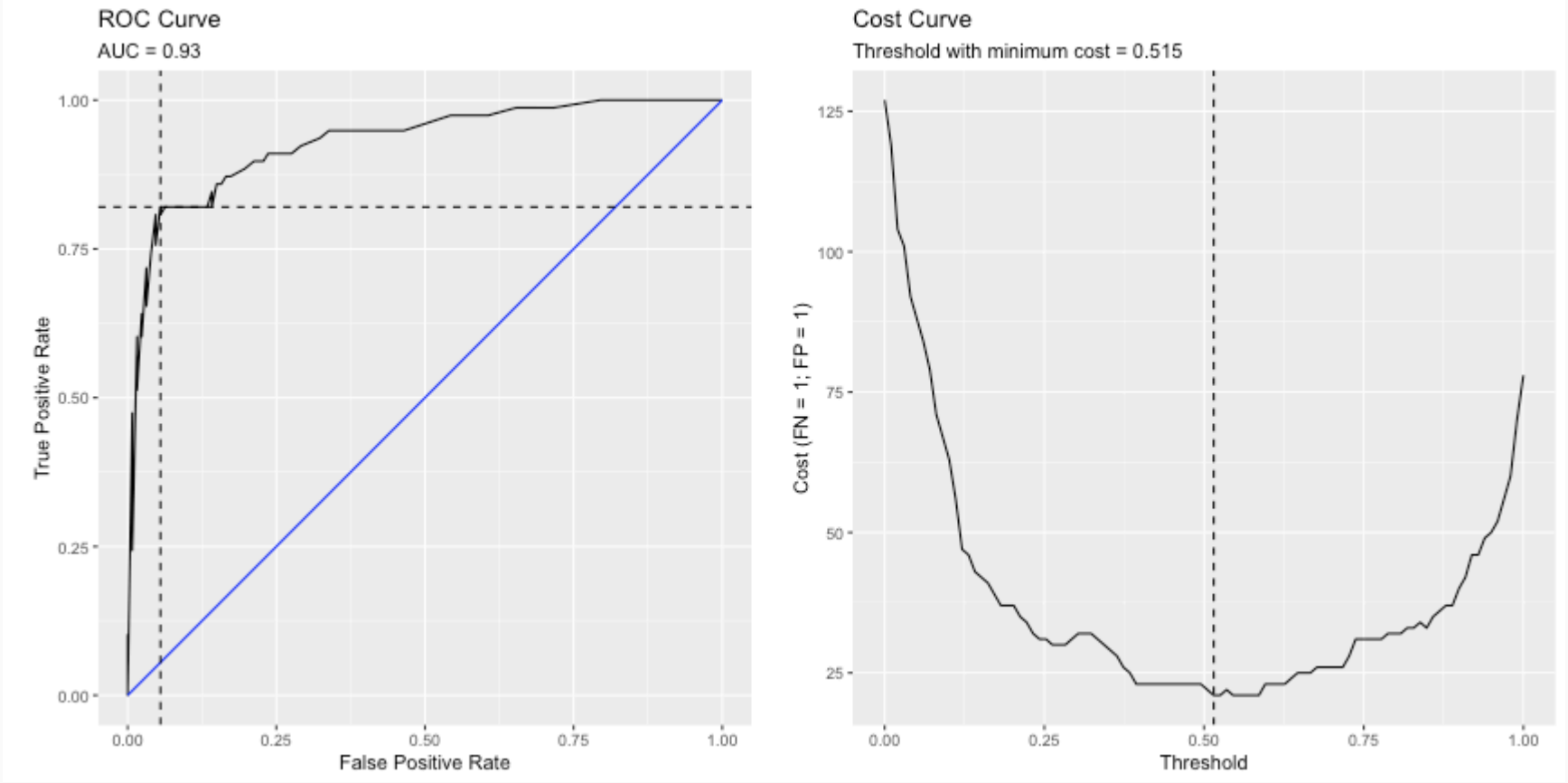
ROC Curve

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



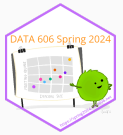
ROC Curve

```
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.

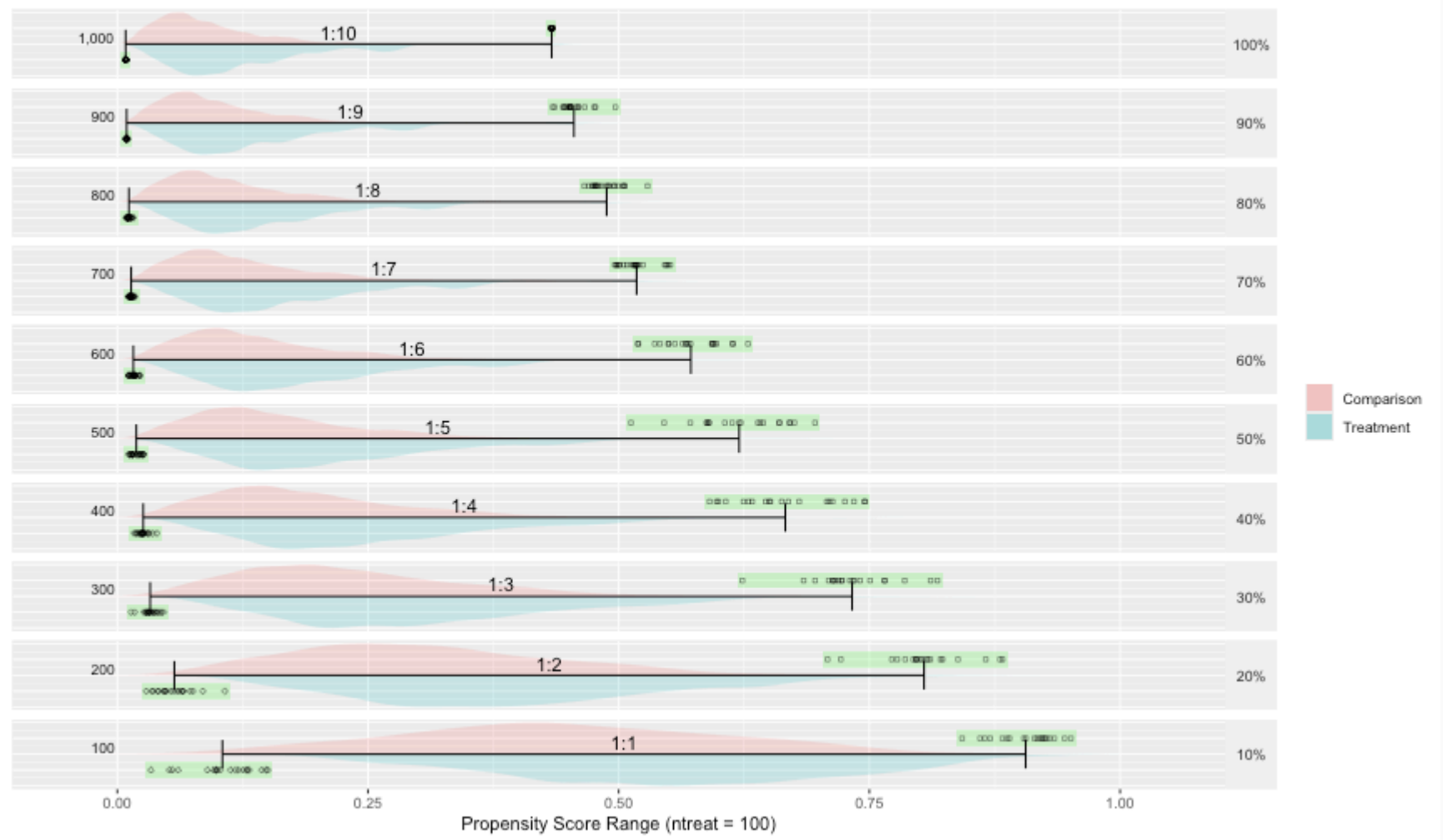
```
test.df2 <- getSimulatedData(
    treat.mean=.6, control.mean=.4)
```

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/U4UXAosdjHorxY919