More Models

Dominguez Center for Data Science Workshop

2025-04-16

Load packages

```r
# For data wrangling and plotting
library(tidyverse)

# For loading the meadowfoam dataset
library(Sleuth3)

# For fitting classification trees
library(rpart)

# For creating charts of trees
library(rpart.plot)
```

Plan for today

- Cover lingering questions.
- Introduce a few more models:
  - Logistic regression
  - Classification trees
- Remember that you can either fill in the "more_modeling.qmd" file or follow along with the "more_modeling_key.qmd" file.
  - Both documents have code related to topics we covered in previous weeks already filled in.

Questions Round-Up

But what does all of the output of `summary()` mean?

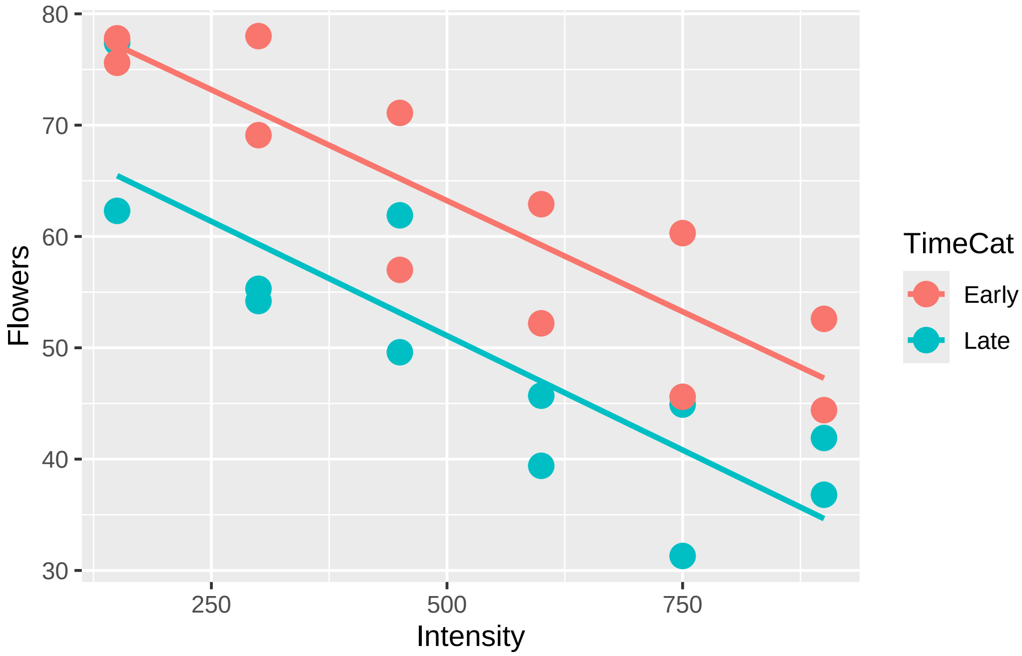
Let's return to the Meadowfoam flowers example from last time and build a model to predict the number of flowers on a plant based on the timing and intensity of the lighting.

In the code below, I fit two models: one with an interaction term and one without. Let's explore and interpret the `summary()` output.

```
# Load the data
data(case0901)

# Recode the timing variable
case0901 <- case0901 %>%
  mutate(TimeCat = case_match(Time,
                              1 ~ "Late",
                              2 ~ "Early"
    ))

# Visualize the data
ggplot(data = case0901,
       mapping = aes(x = Intensity,
                     y = Flowers,
                     color = TimeCat)) +
  geom_point(size = 4) +
  geom_smooth(method = lm, se = FALSE)
```




```
# Build a model without an interaction term
modFlowers <- lm(Flowers ~ Intensity + TimeCat, data = case0901)
summary(modFlowers)
```

```
Call:
lm(formula = Flowers ~ Intensity + TimeCat, data = case0901)

Residuals:
   Min     1Q Median     3Q    Max 
-9.652 -4.139 -1.558  5.632 12.165 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  83.464167   3.273772  25.495  < 2e-16 ***
Intensity    -0.040471   0.005132  -7.886 1.04e-07 ***
TimeCatLate -12.158333   2.629557  -4.624 0.000146 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.441 on 21 degrees of freedom
Multiple R-squared:  0.7992,	Adjusted R-squared:   0.78 
F-statistic: 41.78 on 2 and 21 DF,  p-value: 4.786e-08
```

```r
# Build a model with an interaction term
modFlowers_interact <- lm(Flowers ~ Intensity * TimeCat, data = case0901)
summary(modFlowers_interact)
```

```
Call:
lm(formula = Flowers ~ Intensity * TimeCat, data = case0901)

Residuals:
   Min     1Q Median     3Q    Max 
-9.516 -4.276 -1.422  5.473 11.938 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)           83.146667   4.343305  19.144 2.49e-14 ***
Intensity             -0.039867   0.007435  -5.362 3.01e-05 ***
TimeCatLate          -11.523333   6.142360  -1.876   0.0753 .  
Intensity:TimeCatLate  -0.001210   0.010515  -0.115   0.9096    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.598 on 20 degrees of freedom
Multiple R-squared:  0.7993,	Adjusted R-squared:  0.7692 
F-statistic: 26.55 on 3 and 20 DF,  p-value: 3.549e-07
```

And, how do I start using a local installation of RStudio?

- First download all of your projects from Posit as I will delete the workspace at the end of May.
- You can find directions here for downloading a local installation of RStudio.

- Bring any questions or issues with the transition to my `R` office hours on May 14th 1-3pm.

Any other questions?

Logistic Regression

Consider this model when:

- Response variable (y): categorical (binary)

- Explanatory variable (x): quantitative and/or categorical

Come back to this data example:

> "The social contract of Halloween is simple: Provide adequate treats to costumed masses, or be prepared for late-night tricks from those dissatisfied with your offer. To help you avoid that type of vengeance, and to help you make good decisions at the supermarket this weekend, we wanted to figure out what Halloween candy people most prefer. So we devised an experiment: Pit dozens of fun-sized candy varietals against one another, and let the wisdom of the crowd decide which one was best." – Walt Hickey

> "While we don't know who exactly voted, we do know this: 8,371 different IP addresses voted on about 269,000 randomly generated matchups.2 So, not a scientific survey or anything, but a good sample of what candy people like."


```r
# Load the data
candy <- read_csv("https://raw.githubusercontent.com/fivethirtyeight/data/master/candy-power-
  mutate(pricepercent = pricepercent*100)

# Look at the variables
glimpse(candy)
```

```
Rows: 85
Columns: 13
$ competitorname   <chr> "100 Grand", "3 Musketeers", "One dime", "One quarter~
$ chocolate        <dbl> 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,~
$ fruity           <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,~
```

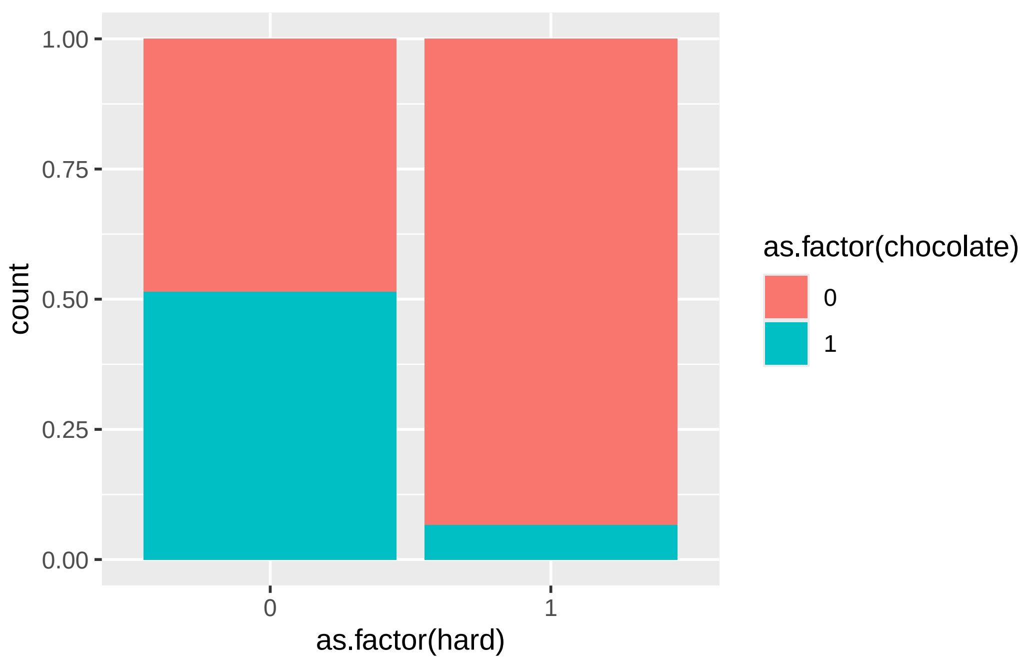
```
$ caramel          <dbl> 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,~
$ peanutyalmondy   <dbl> 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ nougat           <dbl> 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,~
$ crispedricewafer <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ hard             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,~
$ bar              <dbl> 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,~
$ pluribus         <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1,~
$ sugarpercent     <dbl> 0.732, 0.604, 0.011, 0.011, 0.906, 0.465, 0.604, 0.31~
$ pricepercent     <dbl> 86.0, 51.1, 11.6, 51.1, 51.1, 76.7, 76.7, 51.1, 32.5,~
$ winpercent       <dbl> 66.97173, 67.60294, 32.26109, 46.11650, 52.34146, 50.~
```

Question of interest: Does this candy have chocolate in it??

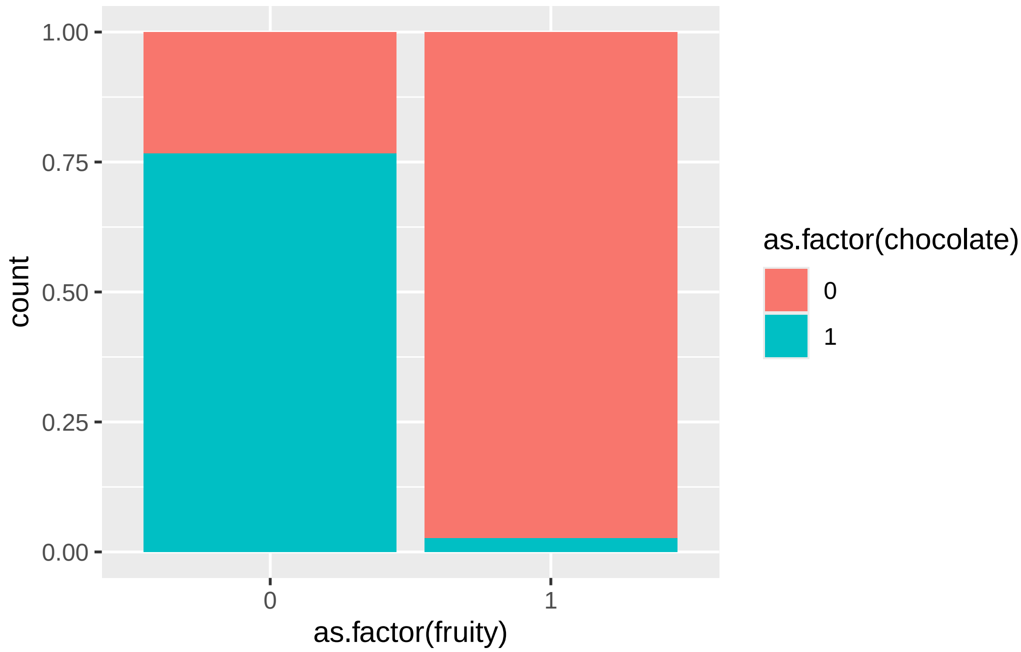
Follow-up question: Which variables in our dataset do you think would be good predictors of whether or not a candy will have chocolate in it?

Visualize the relationships between the variables

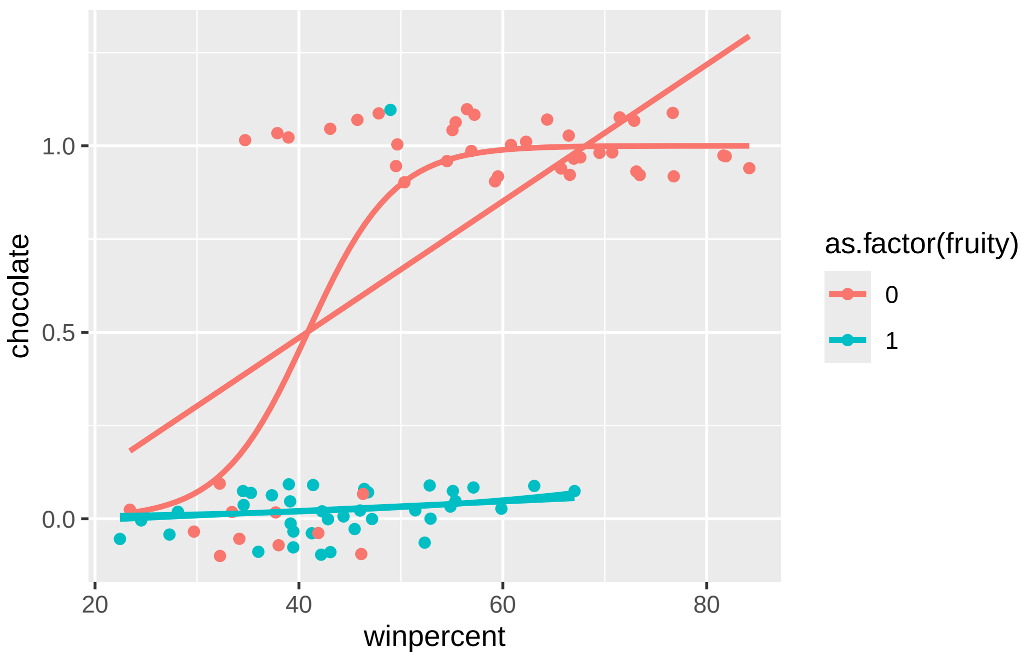
```
# Potential graphs
ggplot(data = candy, mapping = aes(x = as.factor(hard),
                                   fill = as.factor(chocolate))) +
  geom_bar(position = "fill")
```



```r
ggplot(data = candy, mapping = aes(x = as.factor(fruity),
                                   fill = as.factor(chocolate))) +
  geom_bar(position = "fill")
```




```r
ggplot(data = candy, mapping = aes(x = winpercent,
                                   y = chocolate,
                                   color = as.factor(fruity))) +
  geom_jitter(width = 0, height = 0.1) +
  geom_smooth(method = "lm", se = FALSE) +
  geom_smooth(method = "glm", method.args = list(family = "binomial"),
              se = FALSE)
```



Build a logistic regression model

```
# Linear model
mod_lin <- lm(chocolate ~ as.factor(fruity) + winpercent, data = candy)
summary(mod_lin)
```

```
Call:
lm(formula = chocolate ~ as.factor(fruity) + winpercent, data = candy)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.63922 -0.14170  0.01374  0.17507  0.90544 

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)        -0.010511   0.129843  -0.081    0.936    
as.factor(fruity)1 -0.582356   0.065592  -8.879 1.26e-13 ***
winpercent          0.014034   0.002229   6.295 1.44e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.278 on 82 degrees of freedom
Multiple R-squared:  0.6967,	Adjusted R-squared:  0.6893 
F-statistic: 94.18 on 2 and 82 DF,  p-value: < 2.2e-16
```

```r
# Logistic model
mod_log <- glm(chocolate ~ as.factor(fruity) + winpercent, data = candy,
               family = "binomial")
summary(mod_log)
```

```

Call:
glm(formula = chocolate ~ as.factor(fruity) + winpercent, family = "binomial", 
    data = candy)

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)    
(Intercept)        -7.62478    2.40682  -3.168 0.001535 ** 
as.factor(fruity)1 -5.79549    1.48680  -3.898  9.7e-05 ***
winpercent          0.18799    0.05552   3.386 0.000709 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 116.407  on 84  degrees of freedom
Residual deviance:  32.104  on 82  degrees of freedom
AIC: 38.104

Number of Fisher Scoring iterations: 7
```

Prediction

You can use the `predict()` function in the same way as we did for linear regression.

```r
new_cases <- data.frame(fruity = 0, winpercent = 50)
predict(mod_log, newdata = new_cases, type = "response")
```

```
        1 
0.8550401 
```

Classification Trees

Consider this model when:

- Response variable (y): categorical
- Explanatory variable (x): quantitative and/or categorical

This is a more flexible model fit than logistic regression.

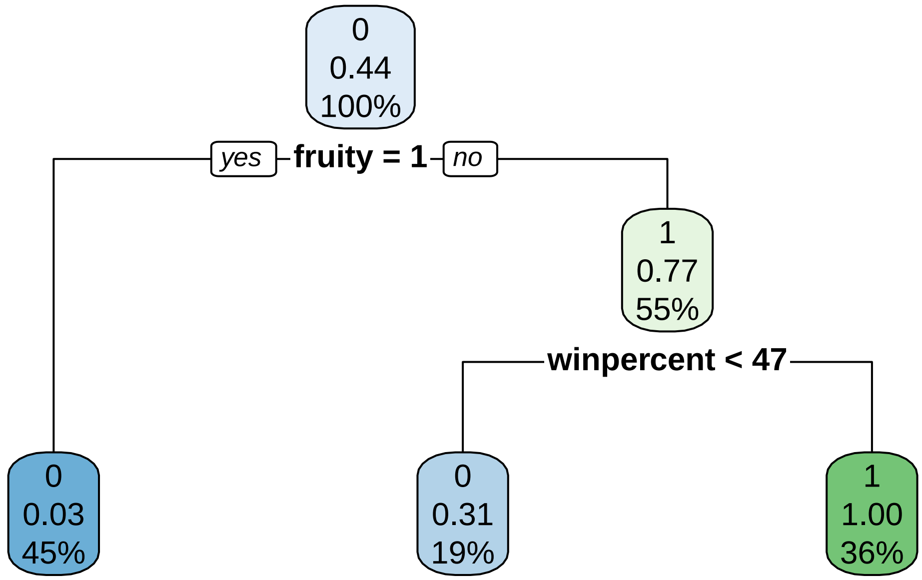
```
# Fit the model
tree <- rpart(chocolate ~ . - competitorname, data = candy, method = "class")
tree
```

```
n= 85

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 85 37 0 (0.56470588 0.43529412)
  2) fruity>=0.5 38  1 0 (0.97368421 0.02631579) *
  3) fruity< 0.5 47 11 1 (0.23404255 0.76595745)
    6) winpercent< 47.06318 16  5 0 (0.68750000 0.31250000) *
    7) winpercent>=47.06318 31  0 1 (0.00000000 1.00000000) *
```

```
rpart.plot(tree)
```



Which model is better: the logistic regression model or the classification tree?

Three metrics:

- Accuracy: Proportion of the guesses were correct
- Sensitivity: Among the chocolate candies, proportion of times that the model identified the candy has chocolate
- Specificity: Among the non-chocolate candies, proportion of times that the model identified the candy doesn't have chocolate

Let's draw a confusion matrix on the board.

```
# How do we access the predictions?
mod_log$fitted.values
```

```
           1            2            3            4            5            6 
0.9930717593 0.9938421950 0.1736410990 0.7397432702 0.0271030505 0.8629517717 
           7            8            9           10           11           12 
0.9558307979 0.0383285038 0.3824594355 0.0009757615 0.4260742583 0.0012931995 
          13           14           15           16           17           18 
0.0001492352 0.0041786848 0.0024674304 0.0048688710 0.0023433814 0.0097036752 
          19           20           21           22           23           24 
0.0640219308 0.2308960271 0.0228591193 0.0041063273 0.9418718857 0.9834396396 
          25           26           27           28           29           30 
```

```
0.9523403317 0.9709987112 0.0002937128 0.9581869379 0.9988951129 0.0035421407
          31           32           33           34           35           36
0.0023239271 0.0300759597 0.9970112146 0.9925387186 0.0090548161 0.9385829224
          37           38           39           40           41           42
0.9978001523 0.9782278139 0.9887154063 0.7968476718 0.9324921076 0.0467840884
          43           44           45           46           47           48
0.9965734663 0.9923926640 0.0001009495 0.0024610736 0.7462081297 0.9956682326
          49           50           51           52           53           54
0.3697286677 0.0034607785 0.0016602131 0.9995759405 0.9997254840 0.9979343096
          55           56           57           58           59           60
0.9977110625 0.0011282190 0.9912437059 0.1149850852 0.0046552731 0.2502249957
          61           62           63           64           65           66
0.1735126952 0.0447294856 0.3769790426 0.0083795584 0.9988752643 0.9725108551
          67           68           69           70           71           72
0.1028017327 0.0296109068 0.3062080104 0.0009870617 0.2076949961 0.1728309046
          73           74           75           76           77           78
0.0002515951 0.0428208726 0.0145996040 0.6157899065 0.7257656829 0.8467783120
          79           80           81           82           83           84
0.0104339048 0.9995577649 0.0075915868 0.0022683073 0.0061928604 0.5628620667
          85
0.8435957161
```

```
predict(tree, type = "class")
```

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
 1  1  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
 0  1  1  0  0  0  1  1  0  1  1  1  1  1  1  0  1  1  0  0  0  1  0  0  0  1
53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78
 1  1  1  0  1  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  1
79 80 81 82 83 84 85
 0  1  0  0  0  0  1
Levels: 0 1
```

```
# Add predictions
candy <- candy %>%
  mutate(log_model_guess = if_else(mod_log$fitted.values >= 0.5, 1, 0),
         tree_model_guess = predict(tree, type = "class"))

# Confusion matrix
count(candy, chocolate, log_model_guess)
```

```
# A tibble: 4 x 3
  chocolate log_model_guess     n
      <dbl>           <dbl> <int>
1         0               0    45
2         0               1     3
3         1               0     4
4         1               1    33
```

```
count(candy, chocolate, tree_model_guess)
```

```
# A tibble: 3 x 3
  chocolate tree_model_guess     n
      <dbl> <fct>            <int>
1         0 0                   48
2         1 0                    6
3         1 1                   31
```

```
# Accuracy
mean(candy$chocolate == candy$log_model_guess)
```

```
[1] 0.9176471
```

```
mean(candy$chocolate == candy$tree_model_guess)
```

```
[1] 0.9294118
```

```
# Sensitivity: Prob can identify that a candy has chocolate among the chocolate candies
candy %>%
  filter(chocolate == 1) %>%
  summarize(mean(log_model_guess == 1), mean(tree_model_guess == 1))
```

```
# A tibble: 1 x 2
  `mean(log_model_guess == 1)` `mean(tree_model_guess == 1)`
                         <dbl>                         <dbl>
1                        0.892                         0.838
```

```
# Specificity: Prob can identify that a candy doesn't have chocolate among candies without cl
candy %>%
  filter(chocolate == 0) %>%
  summarize(mean(log_model_guess == 0), mean(tree_model_guess == 0))
```

```
# A tibble: 1 x 2
  `mean(log_model_guess == 0)` `mean(tree_model_guess == 0)`
                         <dbl>                          <dbl>
1                        0.938                              1
```

Resources related to modeling

We have just scratched the surface of building and interpreting models in `R`. Here are some additional resources on modeling:

- Chapters 5, 6, and 10 of ModernDive
- For a more machine learning approach, check out Tidy Modeling with R or Introduction to Statistical Learning in R
- For more advanced modeling, check out Beyond Multiple Linear Regression
- For bayesian model, check out Bayes Rules!

Homework

- Download materials from Posit Cloud.
- Go back over all the materials we have covered and try to apply to your own work.
- Bring questions and ideas to my `R` office hours on May 14th 1-3pm.