Week 9 - R demo (complete)

Load penguins data

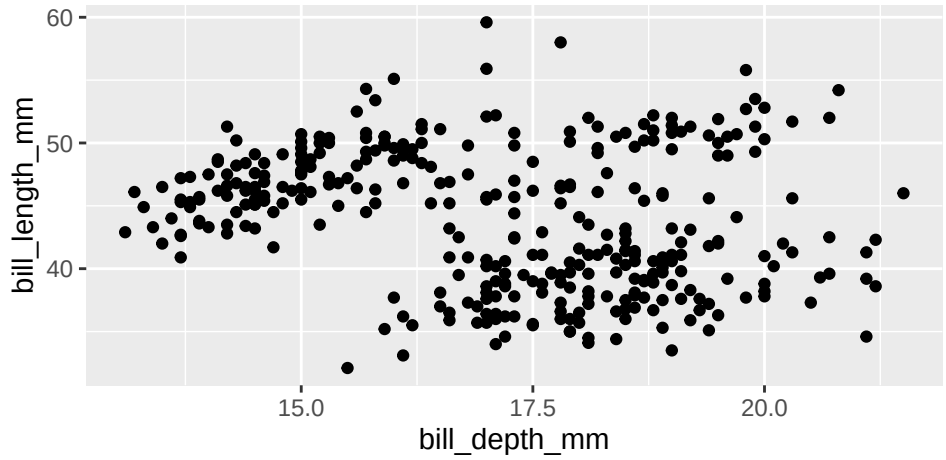
```
# Let's take a first look at the data
glimpse(penguins);
```

```
## Rows: 344
## Columns: 8
## $ species           <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, A...
## $ island            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torge...
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34....
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18....
## $ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, ...
## $ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 347...
## $ sex               <fct> male, female, female, NA, female, male, female, m...
## $ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2...
```

For this demo, we'll be trying to predict the length of a penguin's bill using a linear regression model.

```
# First step let's look at the association between bill depth and bill length
penguins %>% ggplot(aes(x=bill_depth_mm, y=bill_length_mm)) +
  geom_point()
```

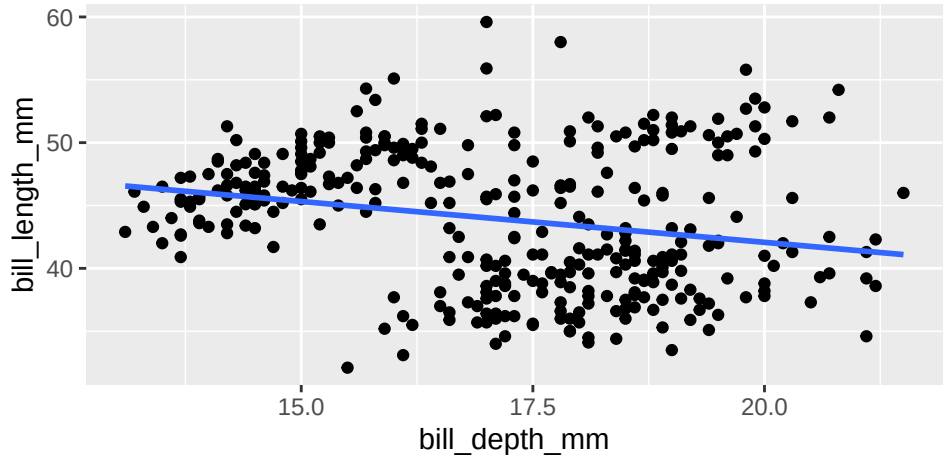
```
## Warning: Removed 2 rows containing missing values (geom_point).
```




```
# Let's get rid of the warning
penguins_clean <- penguins %>% filter(!is.na(bill_depth_mm) & !is.na(bill_length_mm))

# Now we can fit a linear regression model to start exploring this association more deeply
penguins_clean %>% ggplot(aes(x=bill_depth_mm, y=bill_length_mm)) +
  geom_point() +
  geom_smooth(method="lm", se=FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
model1 <- lm(bill_length_mm ~ bill_depth_mm, data=penguins_clean)
summary(model1)$coefficients
```

```
##                 Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)   55.0673698  2.5159514 21.887295 6.914549e-67
## bill_depth_mm -0.6498356  0.1457327 -4.459093 1.119662e-05
```

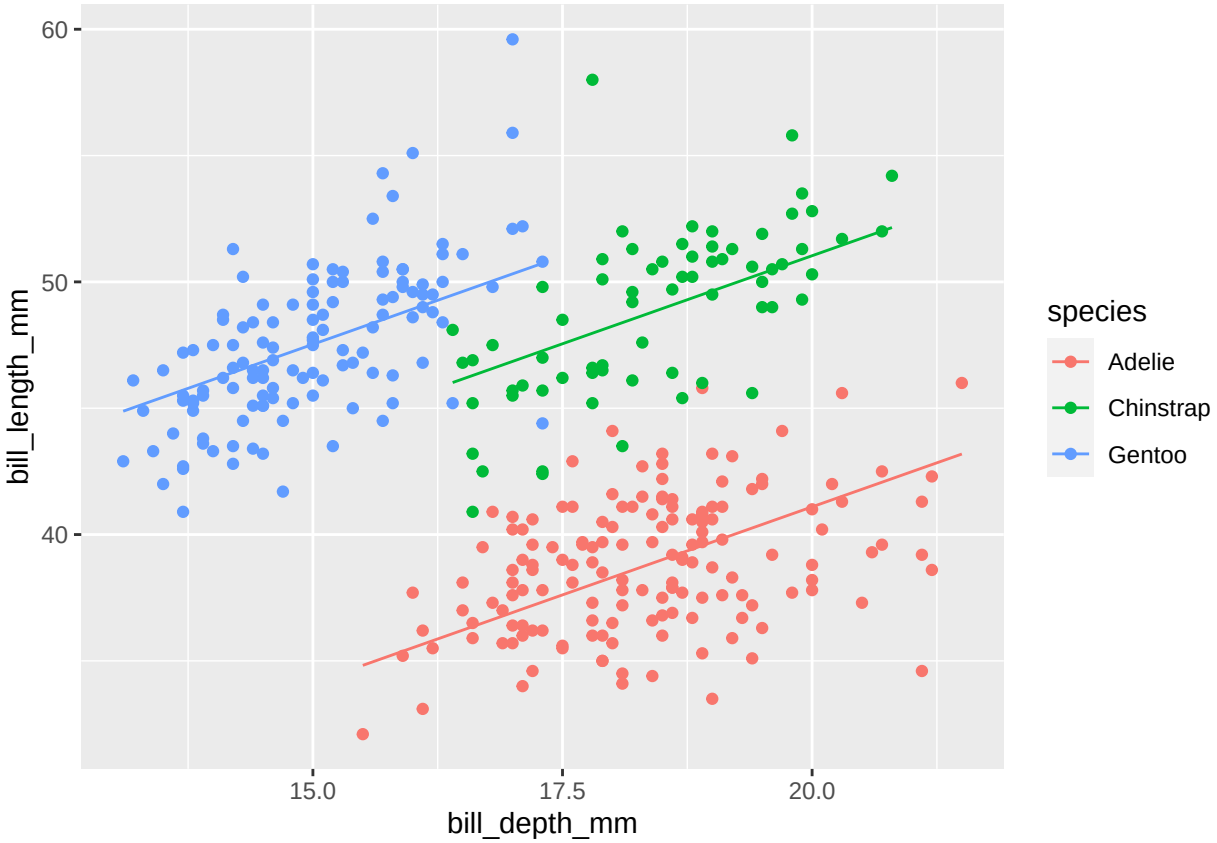
Do you think this fitted linear model is effectively representing the association between bill depth and bill length? What other variable could we add to this model?

```
#glimpse(penguins_clean)

# New variable to add to the model: species
model2 <- lm(bill_length_mm ~ bill_depth_mm + species, data=penguins_clean)
summary(model2)$coefficients
```

```
##                    Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)      13.216370  2.2475424  5.880365 9.834290e-09
## bill_depth_mm     1.394011  0.1219961 11.426679 8.661124e-26
## speciesChinstrap  9.938955  0.3677851 27.023812 1.810788e-86
## speciesGentoo    13.403279  0.5118140 26.187795 2.421386e-83
```

```
# Here Adelie is the baseline because it doesn't appear in the summary table
# If we have a categorical variable with M levels, we'll need to create M-1 indicator variables to repr
# xi2 = I(penguin i is Chinstrap)
# xi3 = I(penguin i is Gentoo)
# So Adelies will have x2=x3=0; Chinstrap will have x2=1 and x3=0 and Gentoo will have x2=0 and x3=1
library(broom)
penguins_clean %>% ggplot(aes(x=bill_depth_mm, y=bill_length_mm, color=species)) +
  geom_point() +
  geom_line(data=augment(model2), aes(y=.fitted))
```

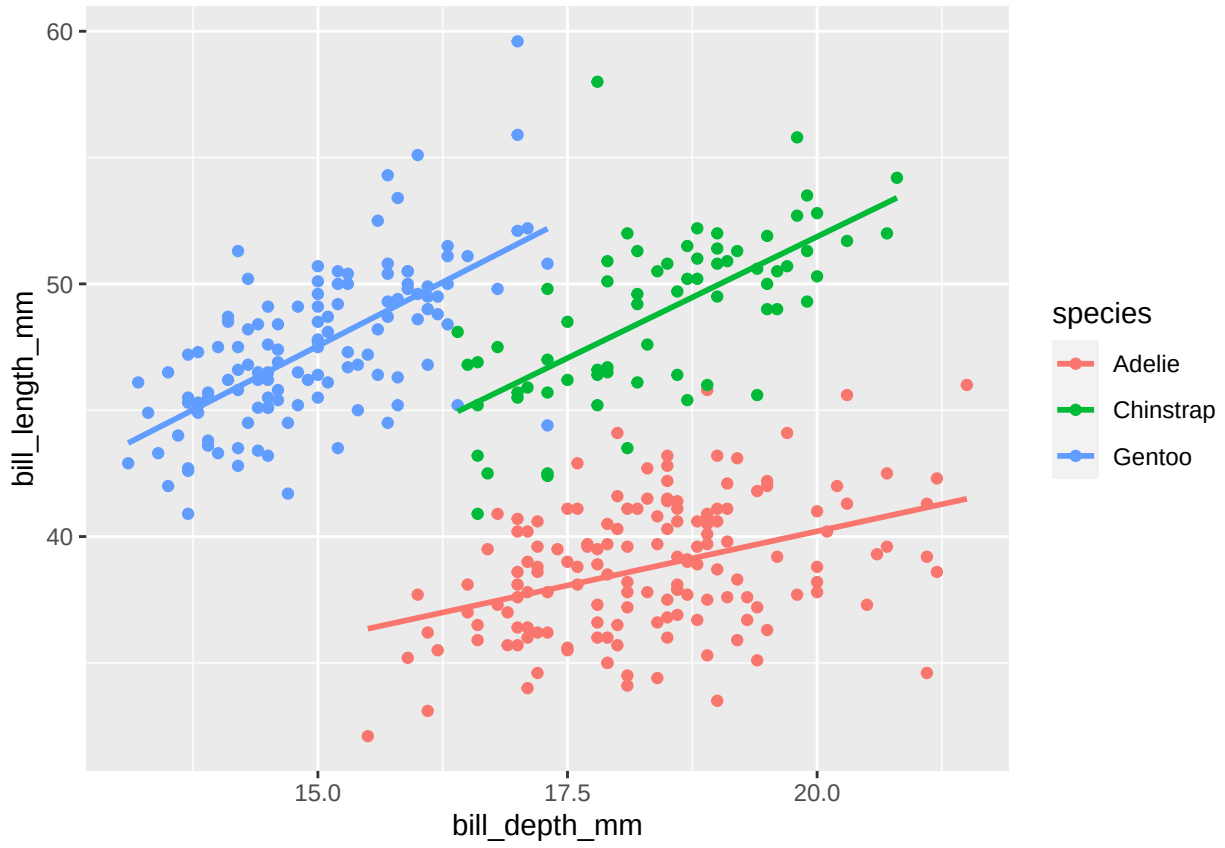



```
# What about an interaction term?
model3 <- lm(bill_length_mm ~ bill_depth_mm * species, data=penguins_clean)
summary(model3)$coefficients
```

```
##                                   Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)                     23.0680794  3.0165065  7.647283 2.180145e-13
## bill_depth_mm                    0.8570263  0.1640619  5.223797 3.081780e-07
## speciesChinstrap                -9.6401718  5.7154193 -1.686695 9.259009e-02
## speciesGentoo                   -5.8385785  4.5352593 -1.287375 1.988497e-01
## bill_depth_mm:speciesChinstrap   1.0650576  0.3100169  3.435483 6.656531e-04
## bill_depth_mm:speciesGentoo      1.1637417  0.2789196  4.172319 3.843496e-05
```

```
penguins_clean %>% ggplot(aes(x=bill_depth_mm, y=bill_length_mm, color=species)) +
  geom_point() +
  geom_smooth(method="lm", se=FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Comparing the prediction accuracy of multiple models

```
# Set up
set.seed(17);
n <- nrow(penguins_clean)
training_indices <- sample(1:n, size=round(0.8*n))

penguins_clean <- penguins_clean %>% rowid_to_column() # adds a new ID column

# Create training dataset
train <- penguins_clean %>% filter(rowid %in% training_indices)
y_train <- train$bill_length_mm;

# Testing dataset includes all observations NOT in the training data
test <- penguins_clean %>% filter(!rowid %in% training_indices)
y_test <- test$bill_length_mm;

# Fit models to training data
modA_train <- lm(bill_length_mm ~ bill_depth_mm,           data = train)
modB_train <- lm(bill_length_mm ~ bill_depth_mm + species, data = train)
modC_train <- lm(bill_length_mm ~ bill_depth_mm * species, data = train)


# Make predictions for testing data using training model
yhat_modA_test <- predict(modA_train, newdata = test)
```

```r
yhat_modB_test <- predict(modB_train, newdata = test)
yhat_modC_test <- predict(modC_train, newdata = test)


# Make predictions for training data using training model
yhat_modA_train <- predict(modA_train, newdata = train)
yhat_modB_train <- predict(modB_train, newdata = train)
yhat_modC_train <- predict(modC_train, newdata = train)


# Calculate RMSE for testing data
modA_test_RMSE <- sqrt(sum((y_test - yhat_modA_test)^2) / nrow(test))
modB_test_RMSE <- sqrt(sum((y_test - yhat_modB_test)^2) / nrow(test))
modC_test_RMSE <- sqrt(sum((y_test - yhat_modC_test)^2) / nrow(test))

# Calculate RMSE for training data
modA_train_RMSE <- sqrt(sum((y_train - yhat_modA_train)^2) / nrow(train))
modB_train_RMSE <- sqrt(sum((y_train - yhat_modB_train)^2) / nrow(train))
modC_train_RMSE <- sqrt(sum((y_train - yhat_modC_train)^2) / nrow(train))

mytable <- tibble(Model = c("A","B","C"),
       RMSE_testdata = c(modA_test_RMSE, modB_test_RMSE, modC_test_RMSE),
       RMSE_traindata = c(modA_train_RMSE, modB_train_RMSE, modC_train_RMSE),
       ratio_of_RMSEs = RMSE_testdata / RMSE_traindata)

library(knitr)
knitr::kable(mytable)
```

Model	RMSE_testdata	RMSE_traindata	ratio_of_RMSEs
A	4.883778	5.397941	0.9047483
B	2.156709	2.583299	0.8348664
C	2.221110	2.476753	0.8967830

What does it mean if the RMSE based on test data is *smaller* than the RMSE based on the training data?

- We know that if it is *much larger*, then that suggests that the model may be overfitting the training data, which makes it less effective at generalizing the pattern to new observations
- However, because of the random split between training/testing datasets, it's possible that just by chance we end up with a model that happens to have smaller RMSE with the testing data than with the training data!
- If this happens, than certainly we don't have evidence that the model is overfitting, and in fact, this would suggest that the model is generalizing well to new observations it hasn't seen before (i.e. the predictions it makes for new observations are as accurate or more accurate as what we would expect based on what we saw with the training data).
- There was a question on Piazza asking if this was called "underfitting" (i.e. the opposite of overfitting) - a good hypothesis, but *NO*. "Underfitting" means something different (and, just like overfitting, it is not a good thing).

```r
# Can we use sample_n() to create training and testing data

train2 <- penguins_clean %>% sample_n(size=0.8*n)
test2 <- penguins_clean %>% filter(!(rowid %in% train2$rowid))
```

```r
model4 <- lm(bill_length_mm ~ bill_depth_mm + flipper_length_mm, data=penguins_clean)
summary(model4)$coefficients
```

```
##                     Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)       -28.1470094  5.5143525 -5.104318 5.544113e-07
## bill_depth_mm       0.6210276  0.1354274  4.585686 6.375054e-06
## flipper_length_mm   0.3056891  0.0190191 16.072742 1.309269e-43
```

```r
data2 <- tibble(bill_depth_mm=15, species="Adelie")
predict(model2, newdata=data2)
```

```
##        1
## 34.12653
```