Homework 11 - Modeling

Sarah Cannon

Due: 11/26/24

Question 1 (3 points)

> **Pick any model we've made this week (either in class or on the homework). Create a model card for this model, following the outline/examples given in last week's reading, available at https://arxiv.org/pdf/1810.03993 (https://arxiv.org/pdf/1810.03993). You should be sure to include all relevant sections of the model card. Note that you do not need to format your model card in the way that was done in the examples in the paper, just listing the sections with sentences and/or lists and/or images for each is fine.**

Model Card: California COVID-19 Daily Cases Spline Model

Model Details

- **Developer**: Nolan Windham in Academic/Research context
- **Model Type**: Natural cubic spline regression model with 10 degrees of freedom
- **Training Algorithm**: Linear regression using natural splines (`ns()` function in R)
- **Parameters**:
  - Degrees of freedom: 10
  - Response variable: new_daily_cases
  - Predictor variable: day_number

Intended Use

- **Primary Intended Uses**:
  - Modeling and visualizing trends in daily COVID-19 cases in California
  - Understanding the overall pattern of case counts over time
  - Smoothing daily fluctuations to reveal underlying trends
- **Primary Intended Users**:
  - Public health researchers
  - Policy makers
  - Student
- **Out-of-scope Use Cases**:
  - Precise daily case predictions
  - Future case forecasting
  - Risk assessment

Factors

- Temporal patterns (day of week effects)
- Testing availability and policies
- Reporting delays

Metrics

- Visual fit to data
- Residual patterns
- Balance between underfitting and overfitting

Evaluation Data

- **Dataset**: California Department of Public Health COVID-19 cases data
- **Timeframe**: February 1, 2020 to May 30, 2023
- **Preprocessing**:
    - Calculation of daily new cases from cumulative cases
    - Removal of missing values
    - Creation of sequential day numbers

Training Data

- **Source**: Same as evaluation data (full dataset used for training)
- **Size**: 1,215 daily observations
- **Distribution**: Covers entire pandemic period including multiple waves

Quantitative Analyses

- **Model Fit**: Captures major trends while smoothing daily fluctuations
- **Limitations**: May not capture very sharp changes in case counts

Ethical Considerations

- Model should not be used for individual medical decisions
- Results may be influenced by testing availability and reporting practices
- Potential for misinterpretation if model limitations are not understood

Caveats and Recommendations

- Model is descriptive rather than predictive
- Users should consider reporting delays and changes in testing practices
- 10 degrees of freedom chosen to balance detail and overfitting
- Should be used alongside other data sources for decision-making

---

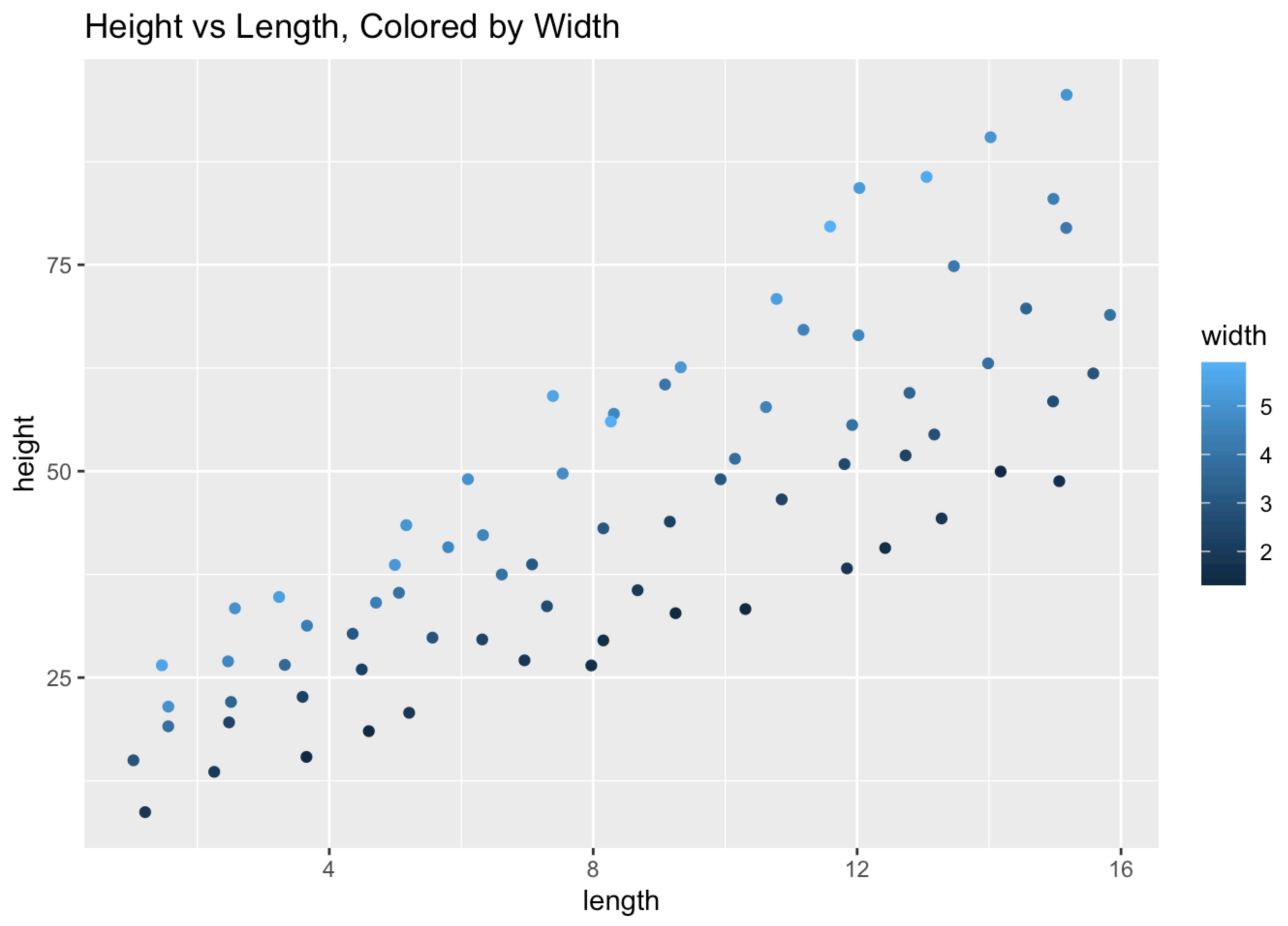
Question 2 (21 points)

a. (1 points) **Import the “lwh.csv” data set. Plot your data with length on the x-axis, height on the y-axis, and width as color.**

```
lwh <- read_csv("lwh.csv")
```

```
## Rows: 75 Columns: 3
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (3): length, width, height
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
ggplot(lwh) +
  geom_point(aes(x = length, y = height, color = width)) +
  labs(title = "Height vs Length, Colored by Width")
```



b. (2 points) **Use the lm function to create a model for this data set, using length and width to predict height, without allowing any interactions between length and width. What is the equation you get for height in terms of length and width?**

```
model <- lm(height ~ length + width, data = lwh)
model
```

```
##
## Call:
## lm(formula = height ~ length + width, data = lwh)
##
## Coefficients:
## (Intercept)       length        width
##     -14.201        3.851        7.716
```

The equation is: height = -14.201 + 3.851 length + 7.716 width

c. (3 points) **If you know the width is 1, what is the equation of your model for height in terms of length? If you know the width is 5, what is the equation of your model for height in terms of length?**

If width is 1, the equation is: height = -6.485 + 3.851 length

If width is 5, the equation is: height = 24.379 + 3.851 length

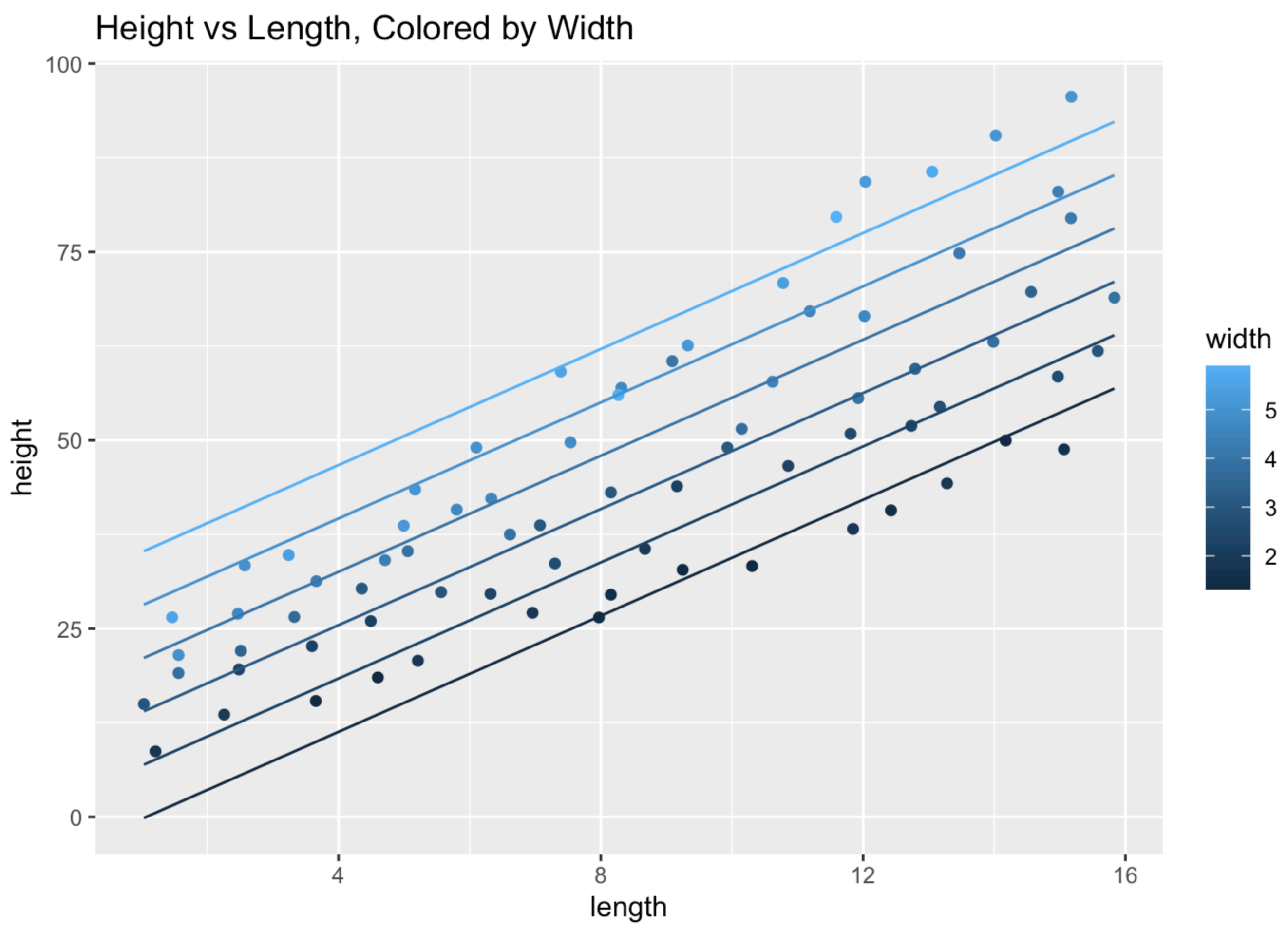
d. (3 points) **Make a data grid that only has six different values for length and six different values for width in it. Your grid should have 36 rows in total**

```
grid <- lwh %>%
  data_grid(
    length = seq_range(length, n = 6),
    width = seq_range(width, n = 6)
  )
grid
```

```
## # A tibble: 36 × 2
##    length width
##     <dbl> <dbl>
##  1   1.03  1.31
##  2   1.03  2.23
##  3   1.03  3.14
##  4   1.03  4.06
##  5   1.03  4.98
##  6   1.03  5.90
##  7   3.99  1.31
##  8   3.99  2.23
##  9   3.99  3.14
## 10   3.99  4.06
## # ℹ 26 more rows
```

e. (3 points) **Add predictions from your model onto the data grid you made in the previous part, and add to the ggplot you made in part (a) six different lines, one for each of the six different models you get relating height to length for the six different widths.**

```
grid %>%
  add_predictions(model) %>%
  ggplot() +
  geom_point(data = lwh, aes(x = length, y = height, color = width)) +
  geom_line(aes(x = length, y = pred, color = width, group = width)) +
  labs(title = "Height vs Length, Colored by Width",)
```

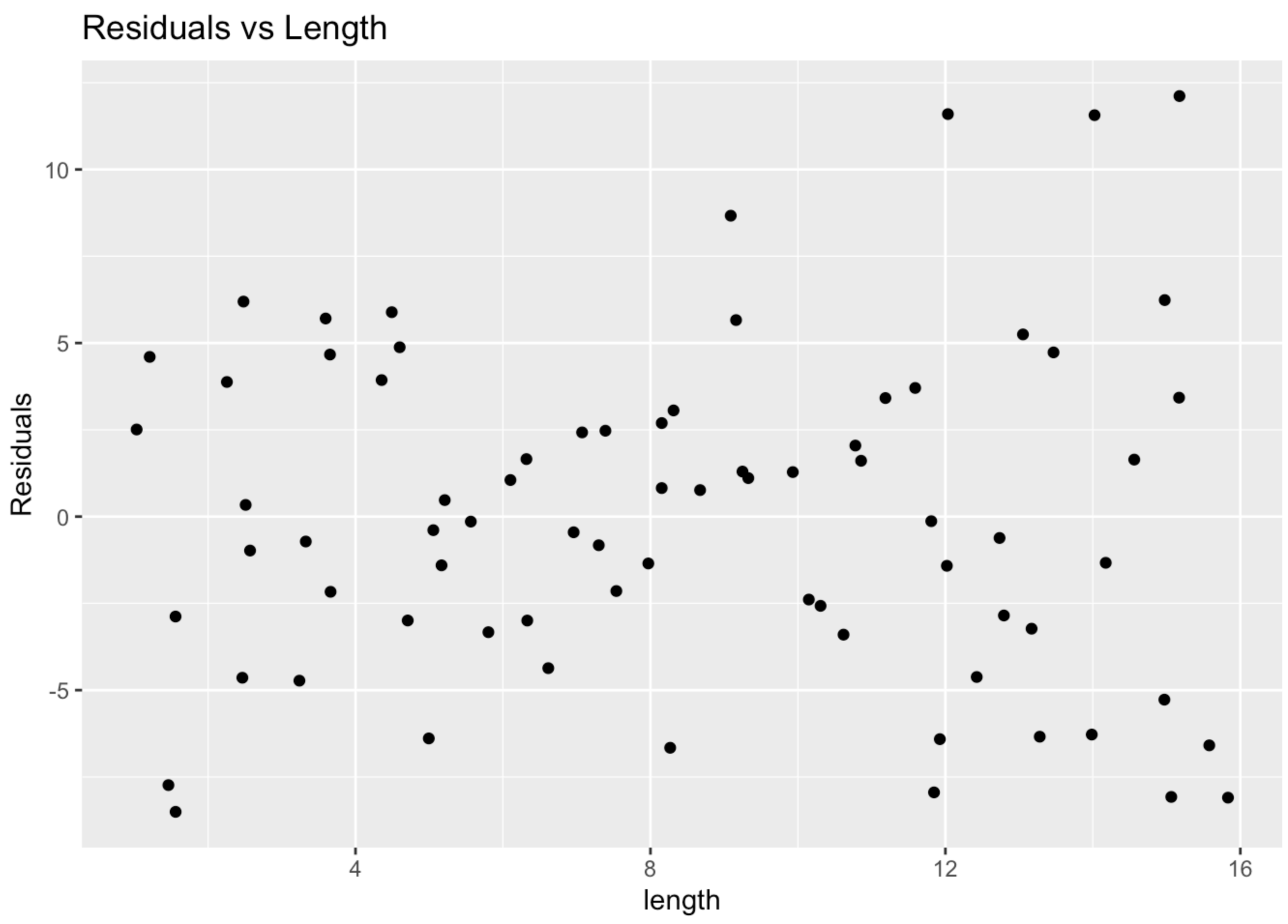


> f. (2 points) **What do you observe about the slopes of the six lines you drew in the previous part? Explain why this is the case.**

The slopes of all six lines are identical. This is because we used a model without interactions between length and width. In this type of model, width only affects the intercept (vertical position) of each line, not its slope.

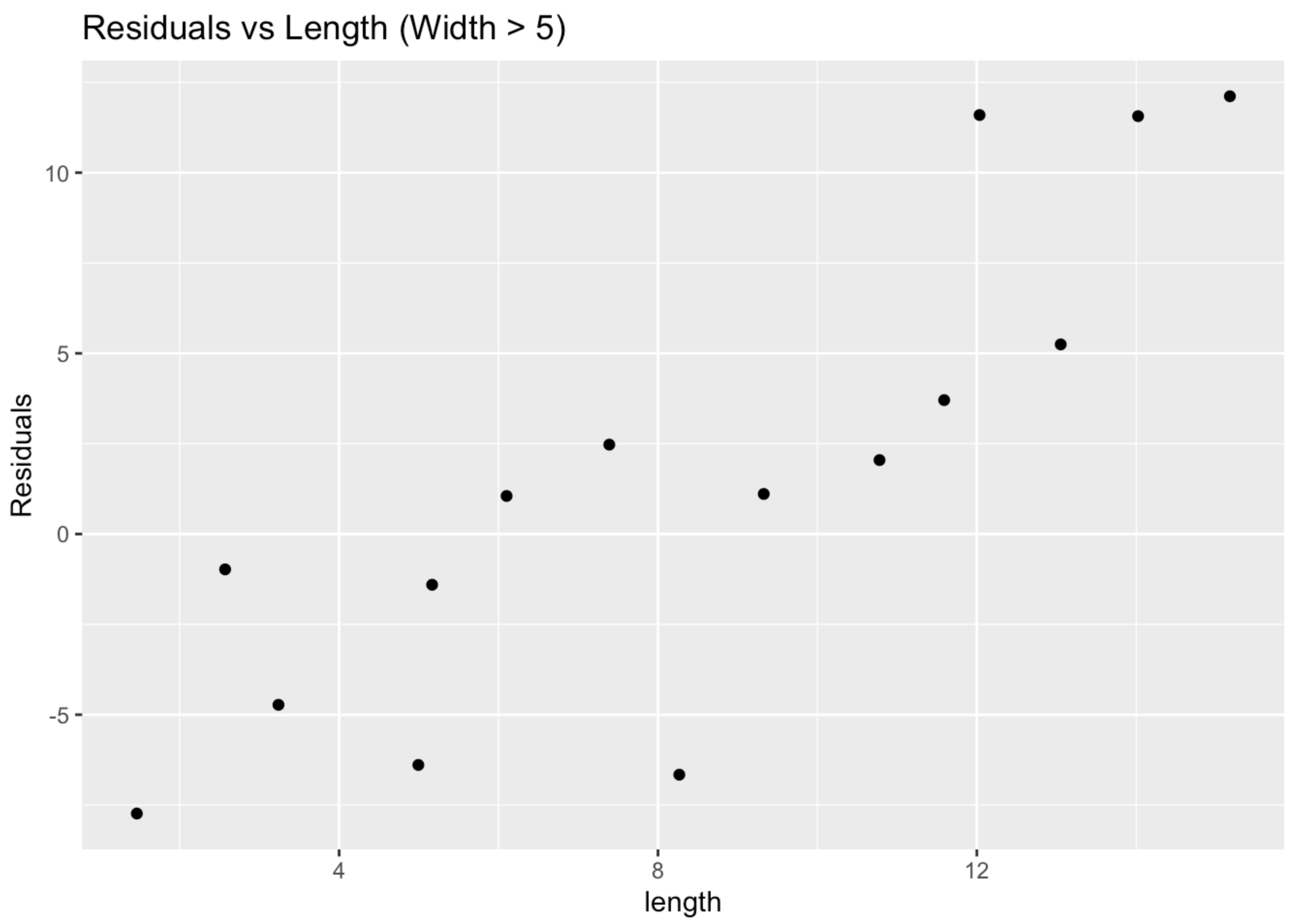
> g. (2 points) **Add residuals on to the original data set, and plot these residuals with length on the x-axis. You do not need to incorporate width into your plot in any way.**

```
lwh %>%
  add_residuals(model) %>%
  ggplot(aes(x = length, y = resid)) +
  geom_point() +
  labs(title = "Residuals vs Length",
       y = "Residuals")
```

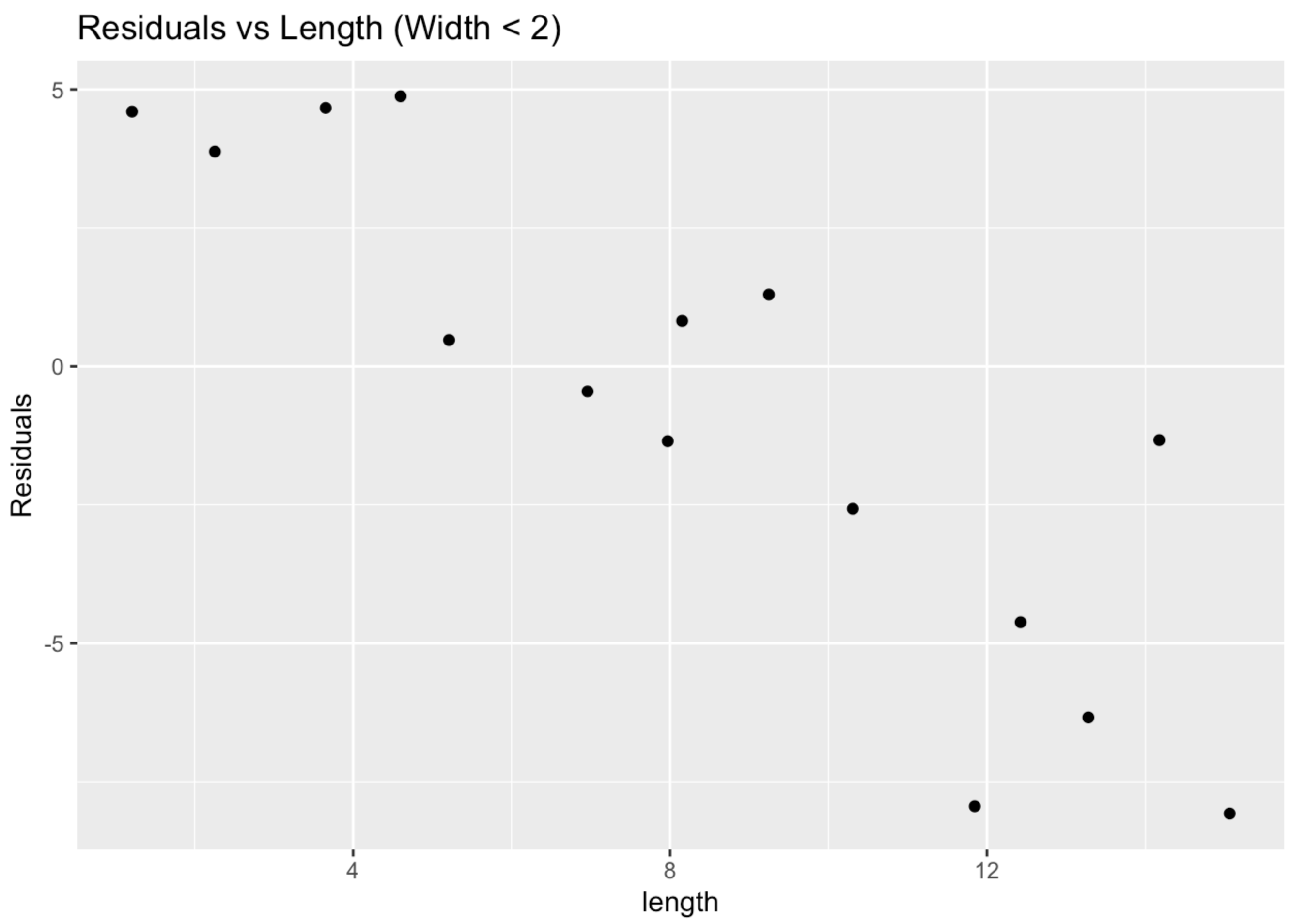


h. (3 points) **Filter the data set (with residuals added) that you made in the previous part to keep rows where the width is greater than 5, and then plot the residuals again, just for the rows where width is greater than 5. Do the same for data points when the width is less than two.**

```
lwh %>%
  add_residuals(model) %>%
  filter(width > 5) %>%
  ggplot(aes(x = length, y = resid)) +
  geom_point() +
  labs(title = "Residuals vs Length (Width > 5)",
       y = "Residuals")
```



```
lwh %>%
  add_residuals(model) %>%
  filter(width < 2) %>%
  ggplot(aes(x = length, y = resid)) +
  geom_point() +
  labs(title = "Residuals vs Length (Width < 2)",
       y = "Residuals")
```



i. (2 points) **Based on your answers to the previous two parts, do you think this is a good model? Explain.**

This is not a good model because the residuals show clear linear trends in opposite directions for different width ranges, suggesting we need a model that includes interactions between length and width.

Question 3 (10 points)

a. (2 points) **Use the lm function to create a model for the same data set as the previous question, using length and width to predict height, *allowing interactions between length and width*. What is the equation you get for height in terms of length and width?**

```
model_int <- lm(height ~ length * width, data = lwh)
model_int
```

```
##
## Call:
## lm(formula = height ~ length * width, data = lwh)
##
## Coefficients:
##  (Intercept)       length        width  length:width
##        3.599        1.730        2.687         0.601
```

height = 3.599 + 1.730 length + 2.687 width + 0.601 length width

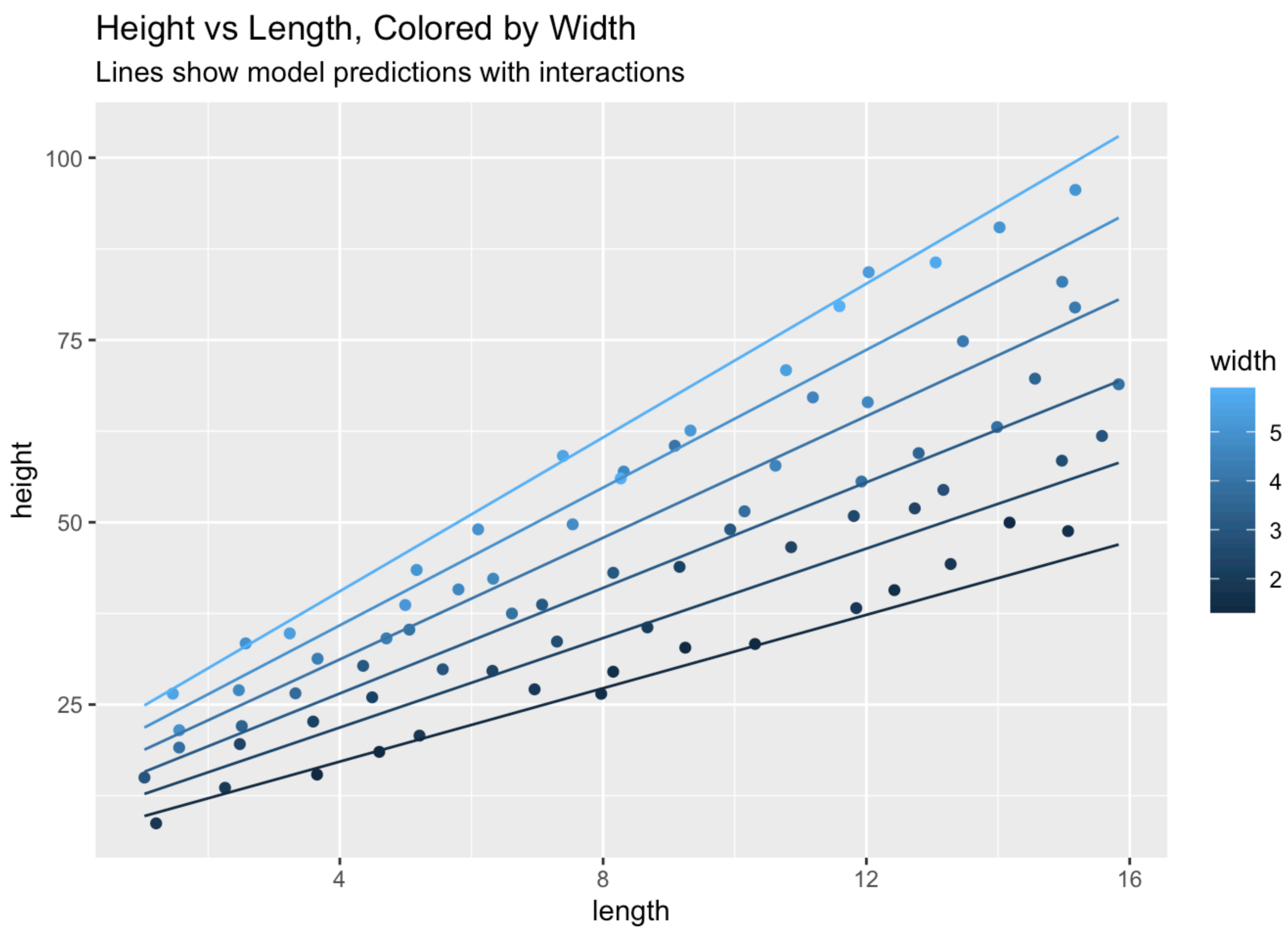
> b. (2 points) **If you know the width is 1, what is the equation of your model for height in terms of length? What about when width is 5?**

When width = 1: height = 3.599 + 1.730 length + 2.687 (1) + 0.601 length (1) = 6.286 + 2.331 length

When width = 5: height = 3.599 + 1.730 length + 2.687 (5) + 0.601 length (5) = 17.034 + 4.735 length

> c. (3 points) **Add to the ggplot you made in part (a) of Question 2 six different lines, corresponding to six different values of width evenly spaced from your smallest width to your largest width. What do you observe about the slopes of the six lines? Explain why this is the case.**

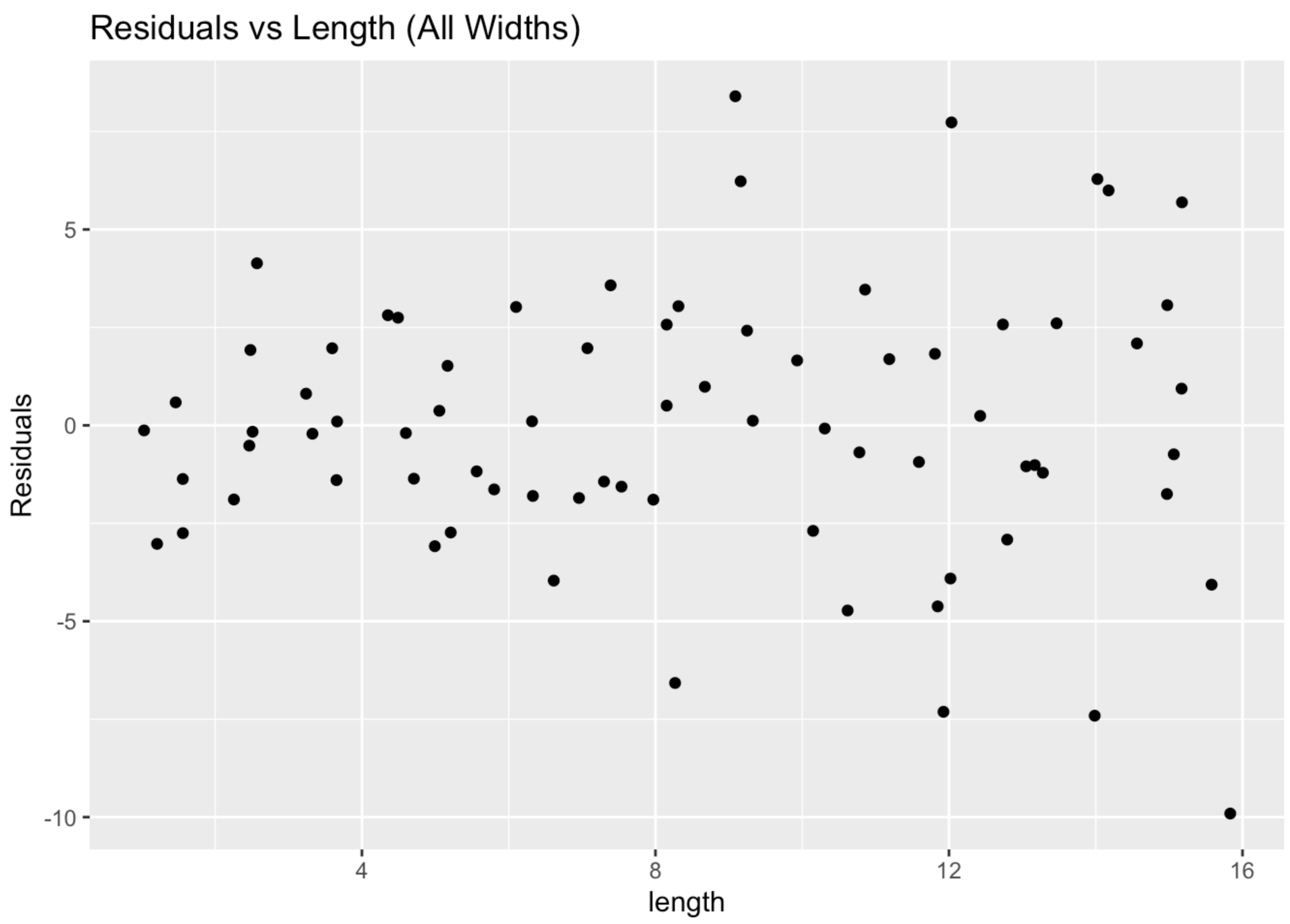
```
grid %>%
  add_predictions(model_int) %>%
  ggplot() +
  geom_point(data = lwh, aes(x = length, y = height, color = width)) +
  geom_line(aes(x = length, y = pred, color = width, group = width)) +
  labs(title = "Height vs Length, Colored by Width",
       subtitle = "Lines show model predictions with interactions")
```



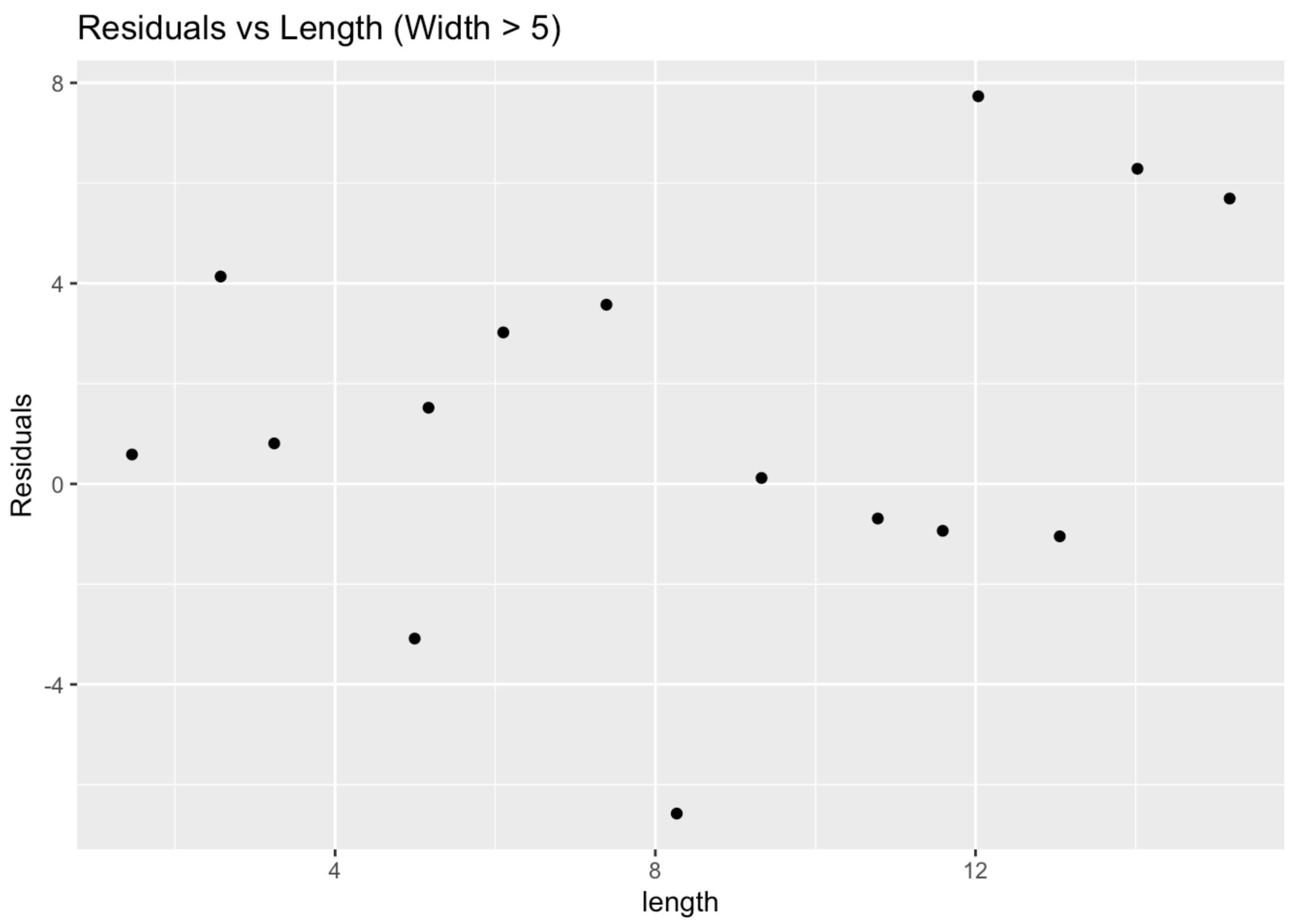
The slopes of the six lines are different and increase as width increases, creating a fan-like pattern. This is because we used a model with interactions between length and width. The interaction term means that the effect of length on height increases by 0.601 units for each unit increase in width.

> d. (3 points) **Plot the residuals for all widths, and then plot the residuals when the width is greater than 5, and when the width is less than 2. What observations do you make about them, and how does this model with interactions compare to your model without interactions in Question 2?**

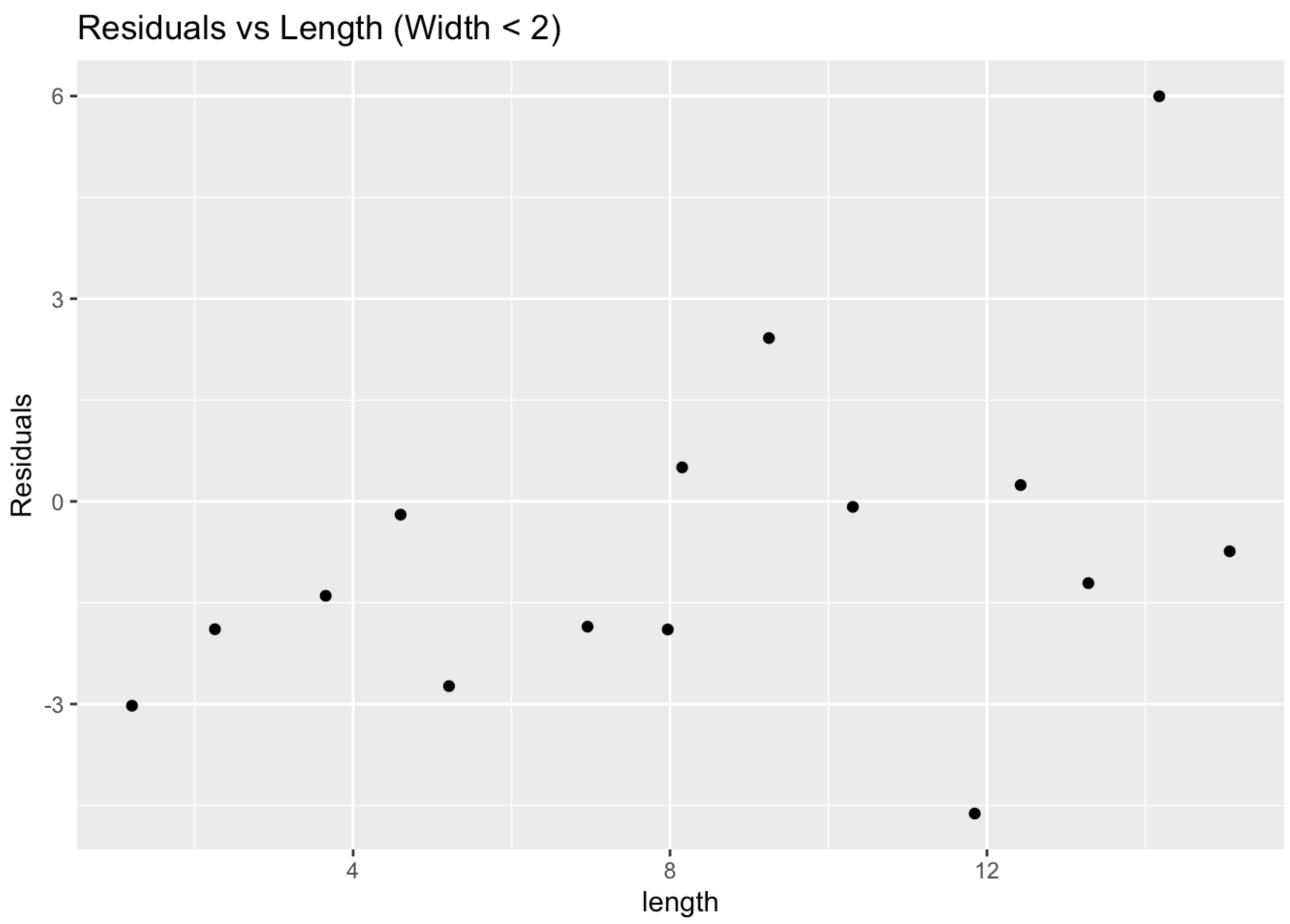
```
lwh %>%
  add_residuals(model_int) %>%
  ggplot(aes(x = length, y = resid)) +
  geom_point() +
  labs(title = "Residuals vs Length (All Widths)",
       y = "Residuals")
```



```
lwh %>%
  add_residuals(model_int) %>%
  filter(width > 5) %>%
  ggplot(aes(x = length, y = resid)) +
  geom_point() +
  labs(title = "Residuals vs Length (Width > 5)",
       y = "Residuals")
```



```
lwh %>%
  add_residuals(model_int) %>%
  filter(width < 2) %>%
  ggplot(aes(x = length, y = resid)) +
  geom_point() +
  labs(title = "Residuals vs Length (Width < 2)",
       y = "Residuals")
```



The residual plots show this interaction model is better than our previous model because For both width ranges (> 5 and < 2), the residuals are more consistently spread around zero without clear linear trends While there’s some spreading pattern in the overall residuals, the subgroup residuals (width > 5 and width < 2) show more stable variance The residuals are smaller in magnitude compared to the previous model, particularly for width < 2

Question 4 (18 points)

> **This question returns to the new daily covid cases data set from the last homework. However, to make things clearer we’ll only look at the first 100 days of data.**

```
c <- read_csv("https://raw.githubusercontent.com/datadesk/california-coronavirus-data/re
fs/heads/master/cdph-state-cases-deaths.csv") %>%
  select(date, confirmed_cases) %>%
  arrange(date) %>%
  mutate(new_daily_cases = confirmed_cases - lag(confirmed_cases)) %>%
  filter(date >="2020-03-01") %>%
  mutate(day_number = row_number()) %>%
  filter(day_number <= 100) %>%
  mutate(day_of_week = factor(weekdays(date),
                  levels = c("Monday", "Tuesday", "Wednesday", "Thursday",
                  "Friday", "Saturday", "Sunday") ))
```

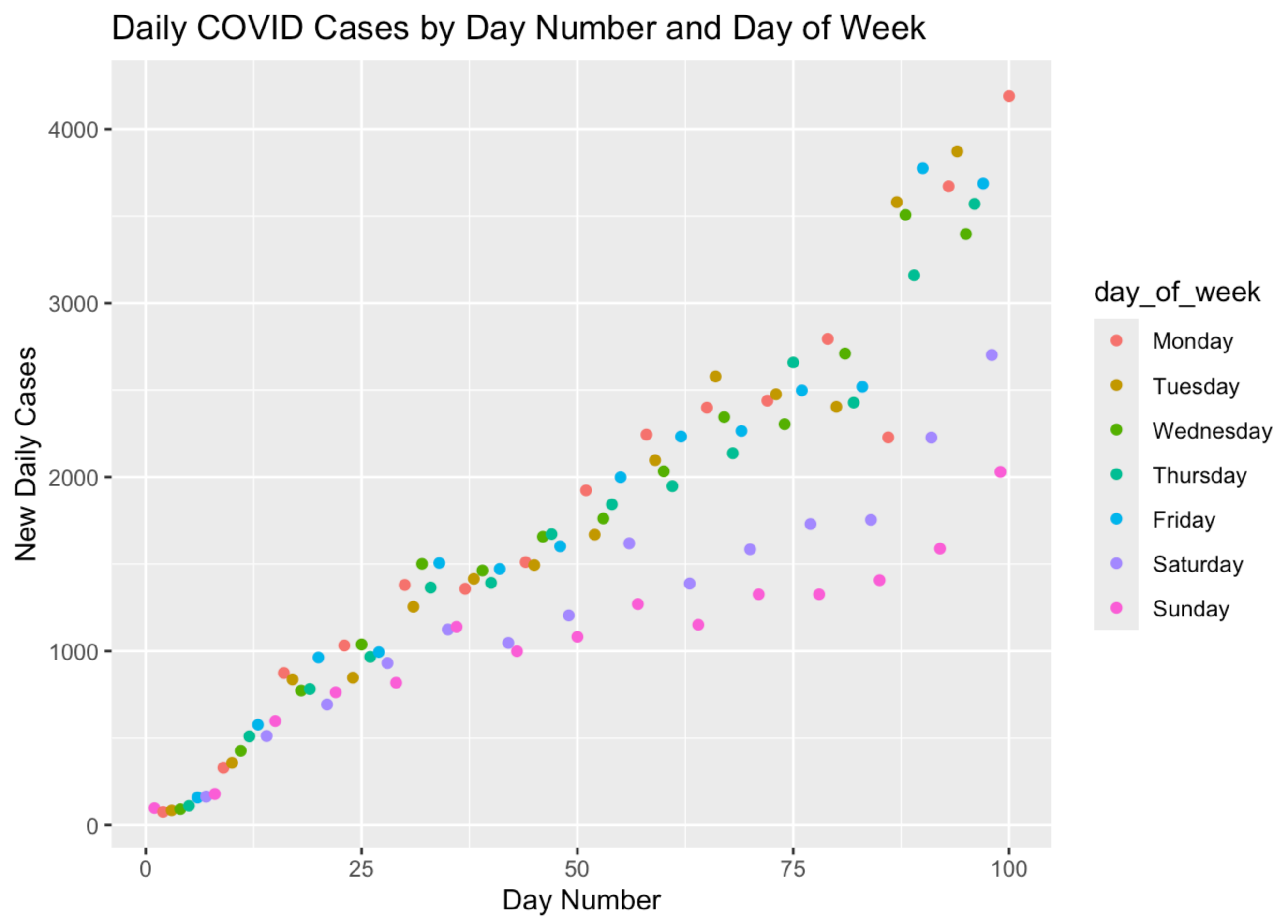
```
## Rows: 1215 Columns: 5
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl  (4): confirmed_cases, probable_cases, confirmed_and_probable_cases, c...
## date (1): date
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
c
```

```
## # A tibble: 100 × 5
##    date       confirmed_cases new_daily_cases day_number day_of_week
##    <date>               <dbl>           <dbl>      <int> <fct>
##  1 2020-03-01             567              98          1 Sunday
##  2 2020-03-02             643              76          2 Monday
##  3 2020-03-03             728              85          3 Tuesday
##  4 2020-03-04             820              92          4 Wednesday
##  5 2020-03-05             931             111          5 Thursday
##  6 2020-03-06            1090             159          6 Friday
##  7 2020-03-07            1254             164          7 Saturday
##  8 2020-03-08            1433             179          8 Sunday
##  9 2020-03-09            1763             330          9 Monday
## 10 2020-03-10            2121             358         10 Tuesday
## # ℹ 90 more rows
```

a. (2 points). **Let’s see if we can get a better model by incorporating both day_number and day_of_week into our model (last homework, we tried these factors separately). Before creating a model, make a plot that shows the relationship between day_number, day_of_week, and new_daily_cases. What do you observe?**

```
ggplot(c, aes(x = day_number, y = new_daily_cases, color = day_of_week)) +
  geom_point() +
  labs(title = "Daily COVID Cases by Day Number and Day of Week",
       x = "Day Number",
       y = "New Daily Cases")
```



The data shows an overall positive trend with day number, but there’s a notable weekly pattern where Saturday and Sunday consistently show lower case counts compared to weekdays, particularly after day 25. While all days show some decrease around this time, the weekend drop is more pronounced.

b. (3 points) **Create a linear model for the number of new_daily_cases in terms of both day_number and day_of_week that does not allow for interactions between the day_number and day_of_week. What is the equation of your model?**

```
model_no_int <- lm(new_daily_cases ~ day_number + day_of_week, data = c)
model_no_int
```

```
##
## Call:
## lm(formula = new_daily_cases ~ day_number + day_of_week, data = c)
##
## Coefficients:
##         (Intercept)           day_number    day_of_weekTuesday
##              362.34                30.08                -38.10
## day_of_weekWednesday   day_of_weekThursday     day_of_weekFriday
##              -65.18               -128.48                -36.78
##  day_of_weekSaturday     day_of_weekSunday
##             -607.44               -814.92
```

new_daily_cases = 362.34 + 30.08 day_number + -38.10 (if Tuesday) + -65.18 (if Wednesday) + -128.48 (if Thursday) + -36.78 (if Friday) + -607.44 (if Saturday) + -814.92 (if Sunday) Where the day_of_week variables are 1 if it's that day, 0 otherwise, and Monday is the intercept.

> c. (3 points) **Create a linear model for the number of new_daily_cases in terms of both day_number and day_of_week that does allow for interactions between the day_number and day_of_week. What is the equation of your model?**

```
model_int <- lm(new_daily_cases ~ day_number * day_of_week, data = c)
model_int
```

```
##
## Call:
## lm(formula = new_daily_cases ~ day_number * day_of_week, data = c)
##
## Coefficients:
##                    (Intercept)                       day_number
##                       110.2616                          35.0276
##             day_of_weekTuesday             day_of_weekWednesday
##                      -139.2207                         -20.6587
##            day_of_weekThursday                day_of_weekFriday
##                       -45.3250                         -47.1143
##            day_of_weekSaturday                day_of_weekSunday
##                        58.0022                         173.5480
##   day_number:day_of_weekTuesday  day_number:day_of_weekWednesday
##                         2.3398                          -0.7497
##  day_number:day_of_weekThursday     day_number:day_of_weekFriday
##                        -1.5977                           0.1527
##  day_number:day_of_weekSaturday     day_number:day_of_weekSunday
##                       -12.8162                         -19.6704
```

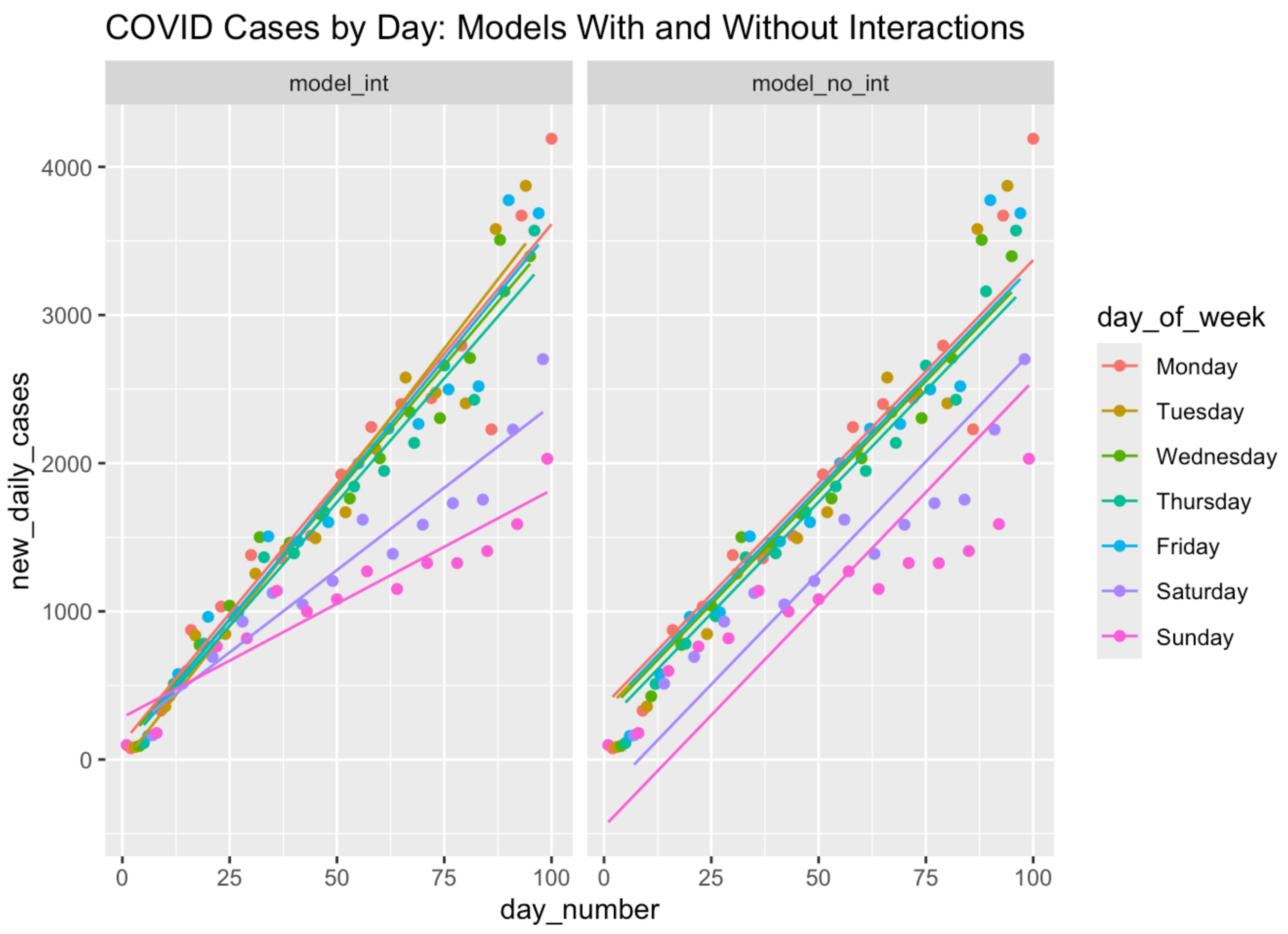
new_daily_cases = 110.2616 + 35.0276 day_number + (-139.2207 + 2.3398 day_number) (if Tuesday) + (-20.6587 - 0.7497 day_number) (if Wednesday) + (-45.3250 - 1.5977 day_number) (if Thursday) + (-47.1143 + 0.1527 day_number) (if Friday) + (58.0022 - 12.8162 day_number) (if Saturday) + (173.5480 - 19.6704 day_number) (if Sunday)

d. (2 points) **Use an appropriate function to add to your original data set predictions from *both* of the models you created in the two previous parts; you should do this using a single function.**

```
c <- c %>%
  gather_predictions(model_no_int, model_int)
```

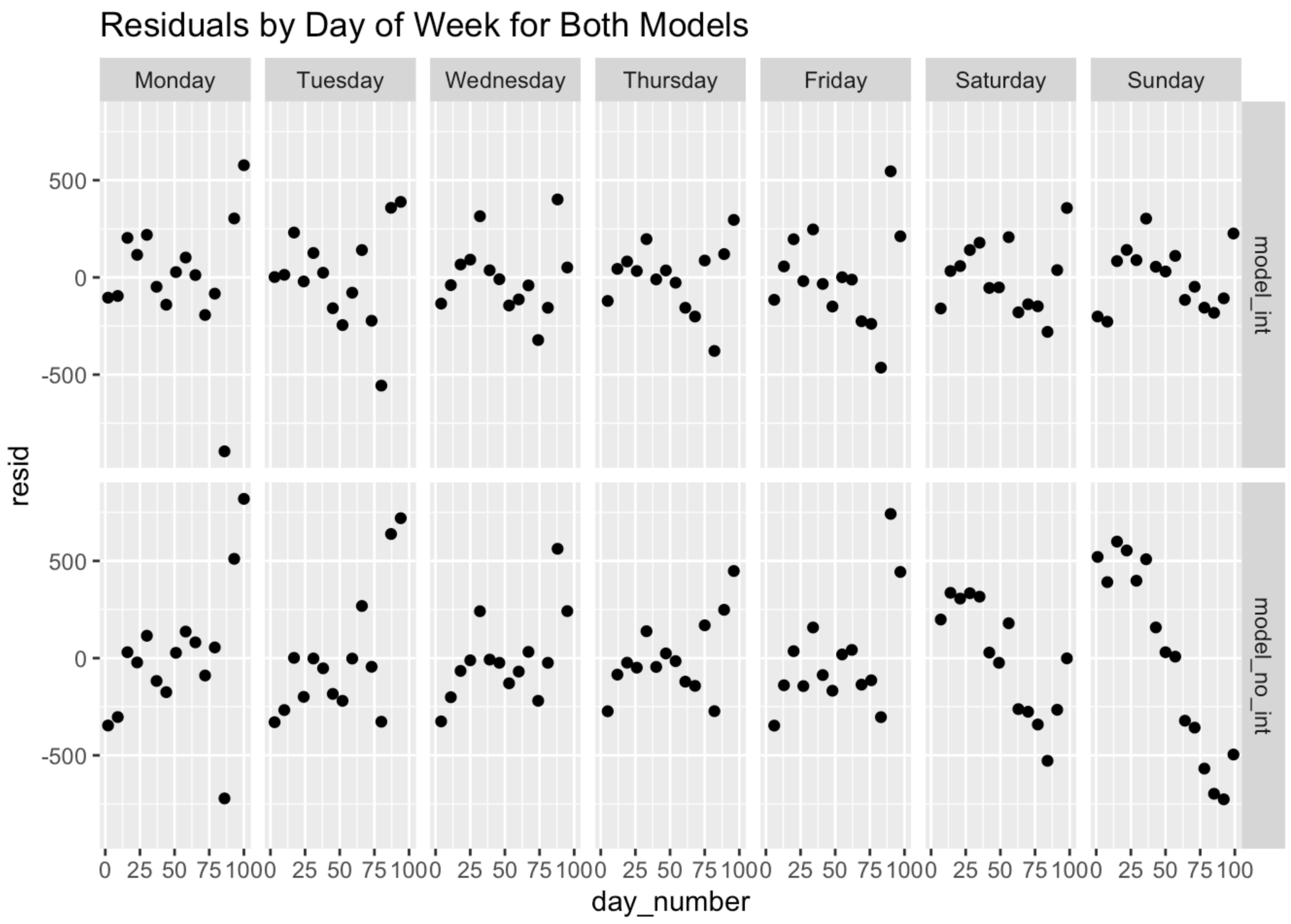
e. (3 points) **Make a plot with two facets, one for each of your models from parts (b) and (c). Each facet should have the original data points (with day_number on the x-axis, new_daily_cases on the y-axis, and colored according to day_of_week) with the lines from your model, one for each day_of_week, drawn on top of it.**

```
ggplot(c, aes(x = day_number, y = new_daily_cases, color = day_of_week)) +
  geom_point() +
  geom_line(aes(y = pred, group = day_of_week)) +
  facet_wrap(~model) +
  labs(title = "COVID Cases by Day: Models With and Without Interactions")
```



f. (3 points) **Plot the residuals of both models using facets, one for each day_of_week. You are welcome to do all of these facets in one plot like we did in one class, or split it up into separate plots for each model if that makes it easier for you to see the patterns of the residuals.**

```
c %>%
  group_by(model) %>%
  mutate(resid = new_daily_cases - pred) %>%
  ggplot(aes(x = day_number, y = resid)) +
  geom_point() +
  facet_grid(model ~ day_of_week) +
  labs(title = "Residuals by Day of Week for Both Models")
```



g. (2 points) **Look at your plots in the previous two parts. Which model do you think is better and why?**

The model with interactions (model_int) is better because its residuals are more consistent across all days of the week, showing less systematic patterns It better captures the different trends for weekdays vs weekends

Question 5 (14 points)

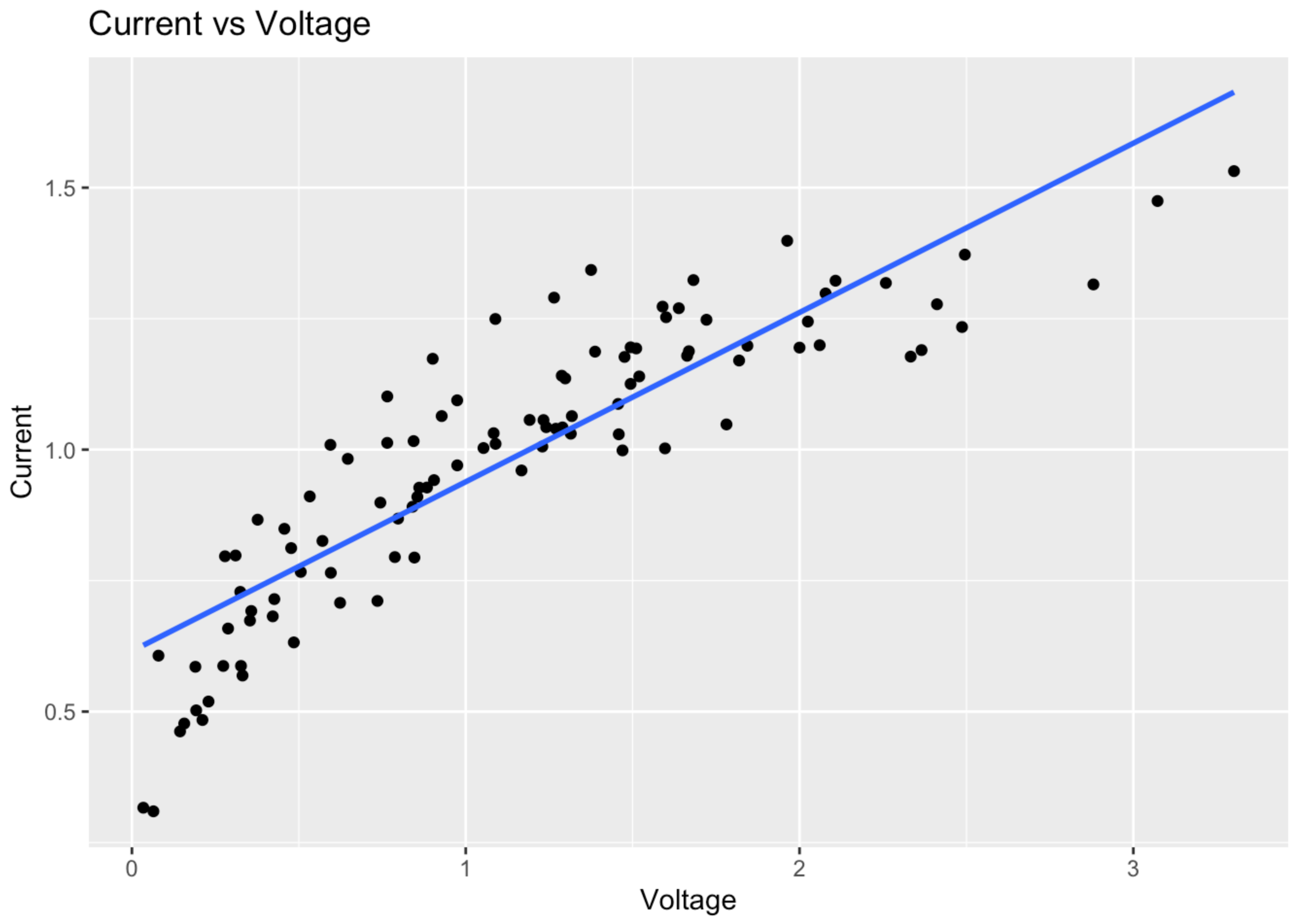
Consider the following data set, that shows (fictional) voltage and current measurements of several samples of a semiconductor material. You want to try to predict, for this material, what voltage is required to achieve a certain current (that is, predict current based on voltage).

```
# You don't need to know how this code works
set.seed(1)
vc <- tibble(voltage = abs(-0.1 + rnorm(100, 1)),
                                       current = voltage^(1/3) + rnorm(100, mean = 0, sd
= 0.1))
```

a. (2 points) **Plot the data (with voltage on the x-axis and current on the y-axis, because we're trying to predict current based on voltage), and attempt to model it with a line. Using your plot, explain why a line isn't a particularly good model for this data.**

```
ggplot(vc, aes(x = voltage, y = current)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Current vs Voltage",
       x = "Voltage",
       y = "Current")
```

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



A line isn’t a particularly good model for this data because the relationship between voltage and current shows clear curvature

b. (3 points) **You know, based on your work with semiconductor materials, that a good family of models to try to predict current based on voltage might be polynomial models:**

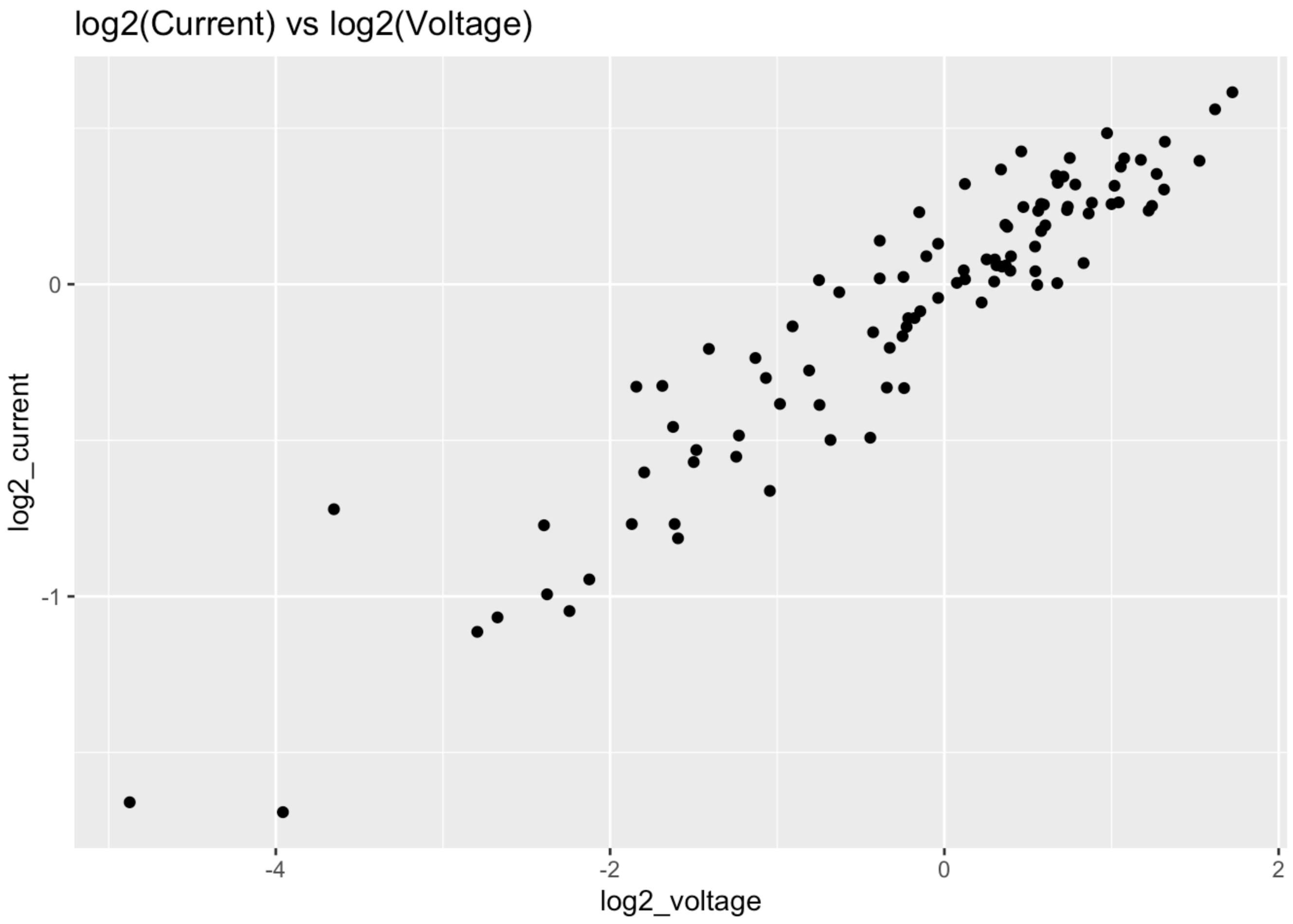
$$current = a1 * voltage^(a2)$$

where we expect a2 to be small, like maybe around 1/2 or 1/3. Taking the log of both sides and simplifying terms using logarithm rules (just like in class), this means we’re looking for the model:

$$log2(current) = log2(a1) + a2 * log2(voltage)$$

Create columns in the vc data set for log2(voltage) and log2(current), and make a plot showing the relationship between log(voltage) and log2(current). Does the relationship appear to be linear?

```
vc <- vc %>%
  mutate(log2_voltage = log2(voltage),
         log2_current = log2(current))

ggplot(vc, aes(x = log2_voltage, y = log2_current)) +
  geom_point() +
  labs(title = "log2(Current) vs log2(Voltage)")
```



c. (3 points) **Create a linear model predicting log2(current) from log2(voltage). What are the coefficients of this linear model, and what is your equation for log2(current) in terms of log2(voltage)?**

```
log_model <- lm(log2_current ~ log2_voltage, data = vc)
log_model
```

```
##
## Call:
## lm(formula = log2_current ~ log2_voltage, data = vc)
##
## Coefficients:
##  (Intercept)  log2_voltage
##     -0.01321       0.34344
```

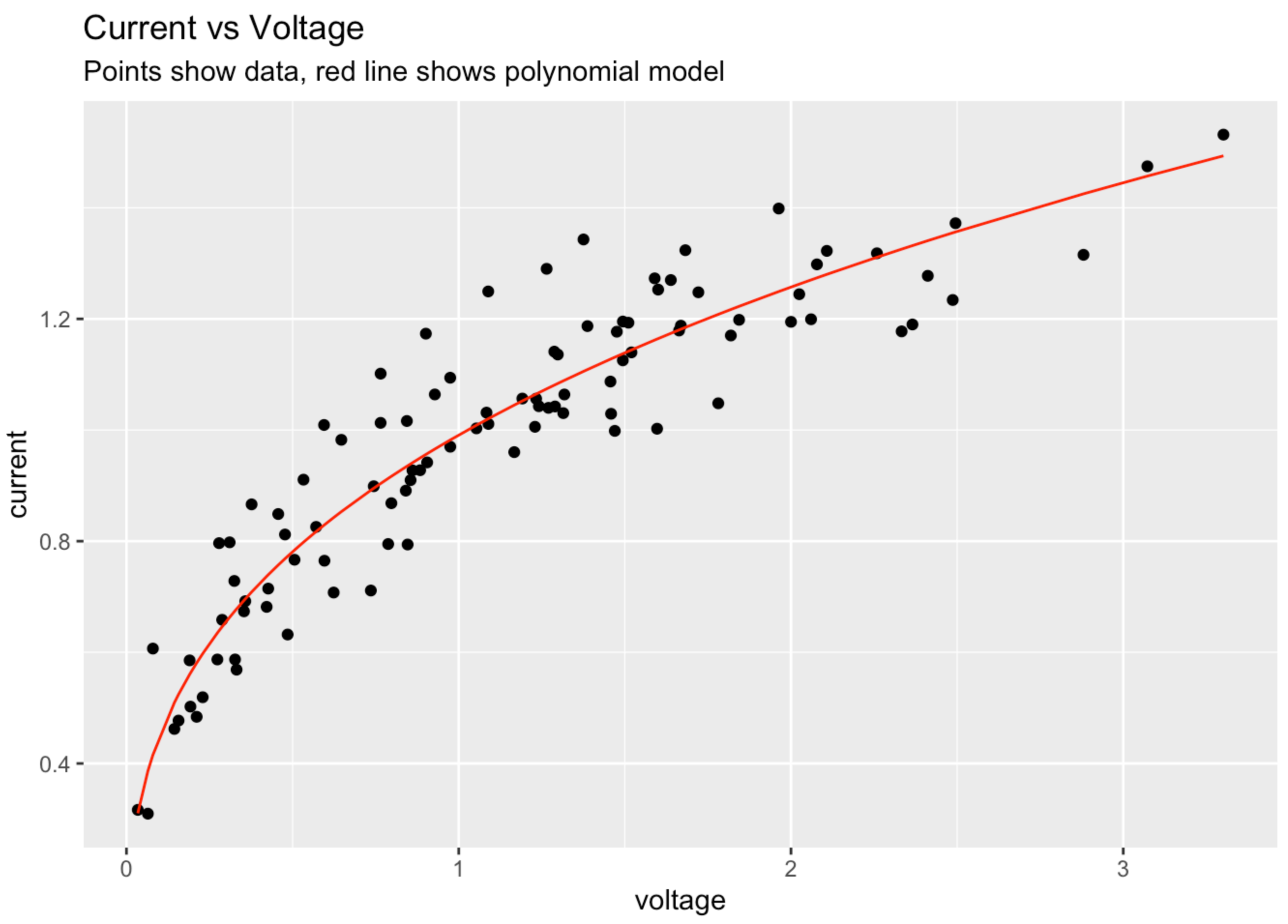
log2(current) = -0.01321 + 0.34344 log2(voltage)

> d. (3 points) **Starting with the linear model you made in the previous part for log2(current) in terms of log2(voltage), work backwards to get an equation for current in terms of voltage. That is, your answer should look like current = a1 * voltage^(a2), but with the values a1 and a2 replaced by numbers. What does your equation for current in terms of voltage become?**

current = 0.9909 voltage^(0.34344)

> e. (3 points) **Add a column to your original data set for the model you came up with in the previous part (that is, add a column giving a prediction for current, not a prediction for log(current)), and draw this model on top of your original data. Does it look like a good model? Explain why or why not.**

```
vc <- vc %>%
  mutate(pred_current = 2^(-0.01321) * voltage^(0.34344))

ggplot(vc, aes(x = voltage, y = current)) +
  geom_point() +
  geom_line(aes(y = pred_current), color = "red") +
  labs(title = "Current vs Voltage",
       subtitle = "Points show data, red line shows polynomial model")
```



The polynomial model appears to be a good fit for the data because: The line (model predictions) follows the curve of the data points closely There are no systematic deviations like we saw with the linear model The model captures the non-linear relationship between voltage and current

Question 6 (9 points)

> **Sometimes, like in the previous question, you have some contextual information that helps you guess at what class of models makes sense. Sometimes, you don't have that information!**

> a. (1 points) **Import the data set "sim8.csv", which has an x column and a y column. Add a log2(x) and a log2(y) column**

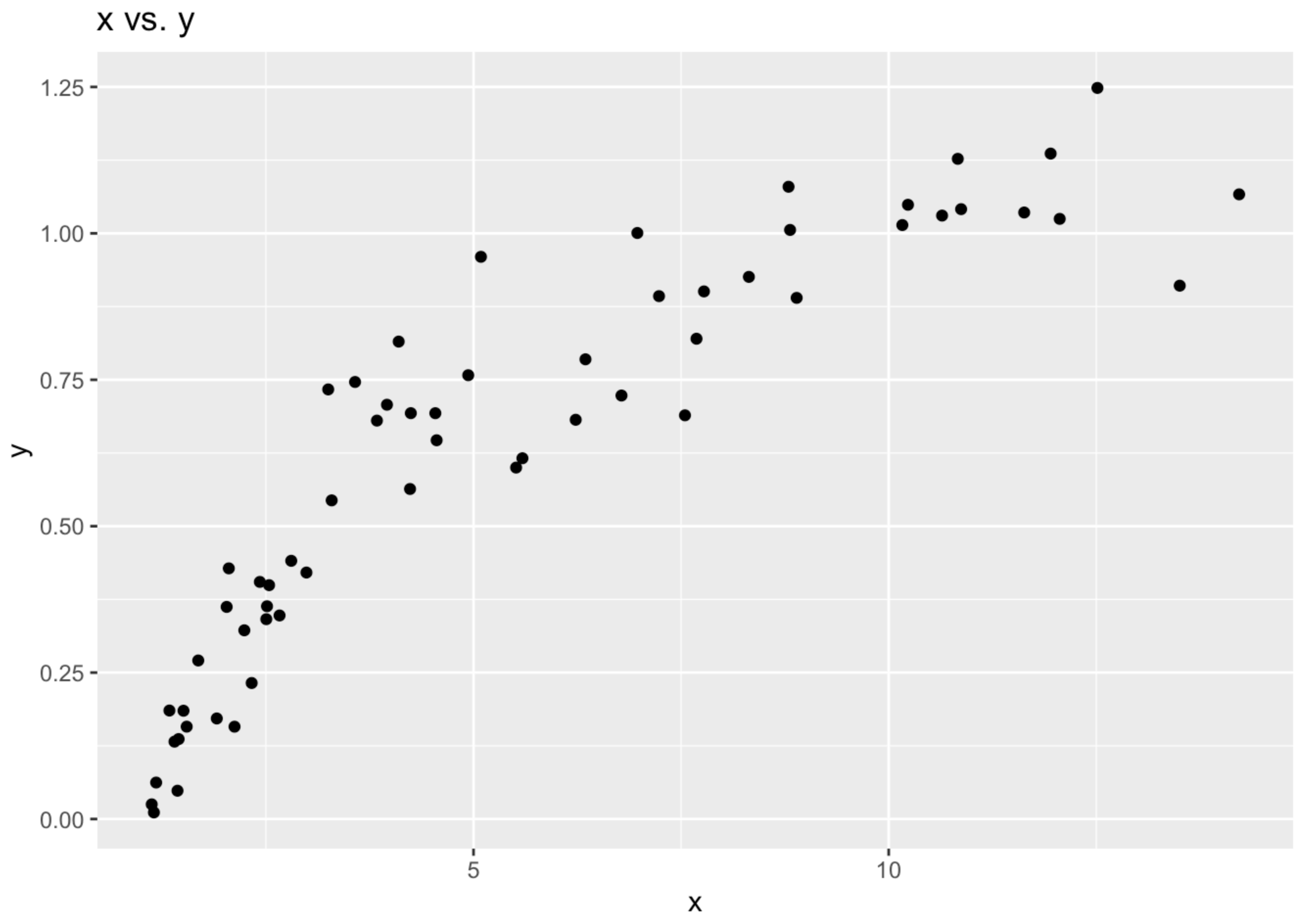
```
sim8 <- read_csv("sim8.csv")
```

```
## Rows: 60 Columns: 2
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (2): x, y
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

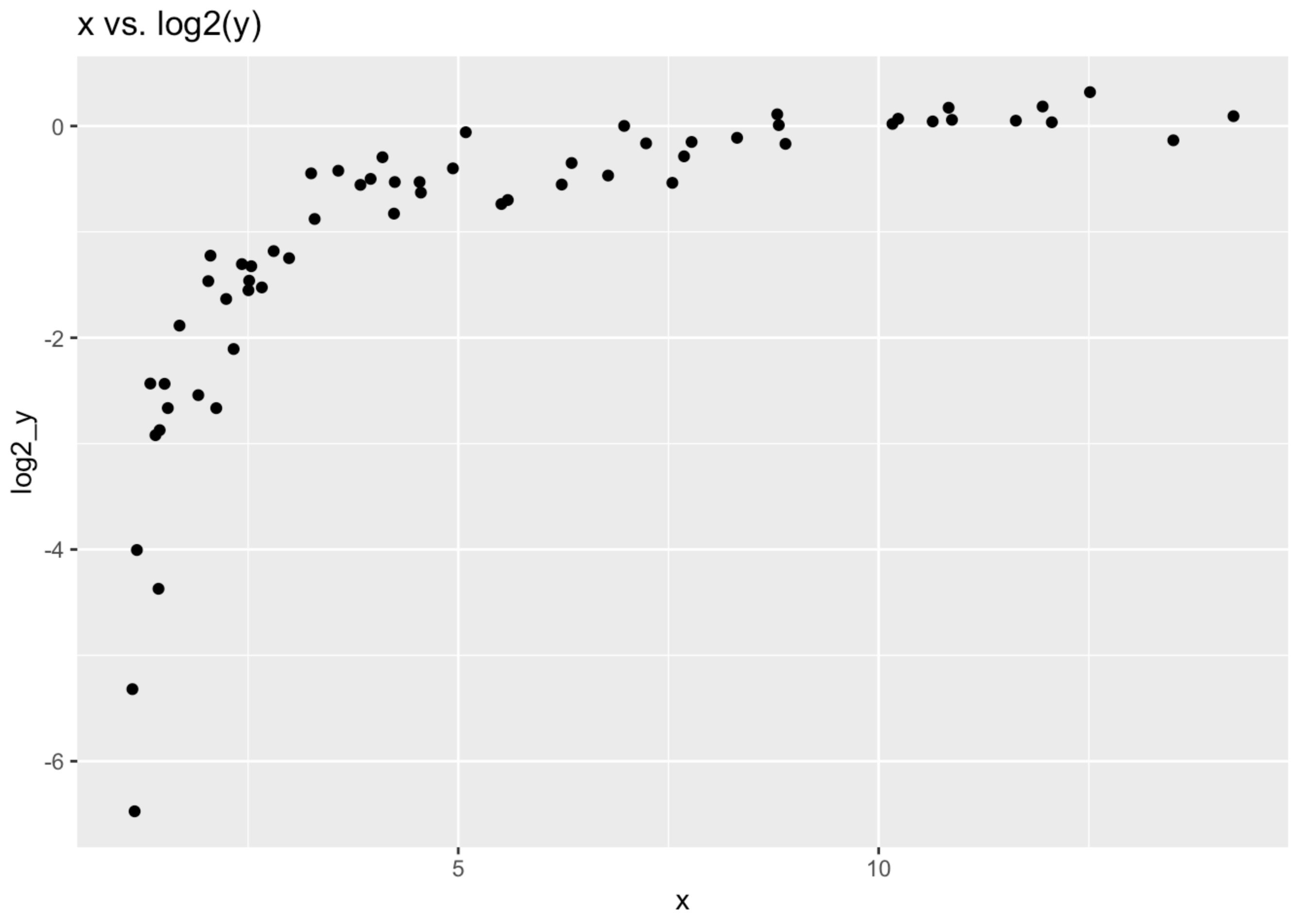
```
sim8 <- sim8 %>%
  mutate(log2_x = log2(x),
         log2_y = log2(y))
```

b. (4 points) **Make 4 plots for this data: x vs. y, x vs. log2(y), log2(x) vs. y, and log2(x) vs. log2(y). (Note the easiest way to do this is to call the ggplot function 4 separate times). Which plot looks most linear?**

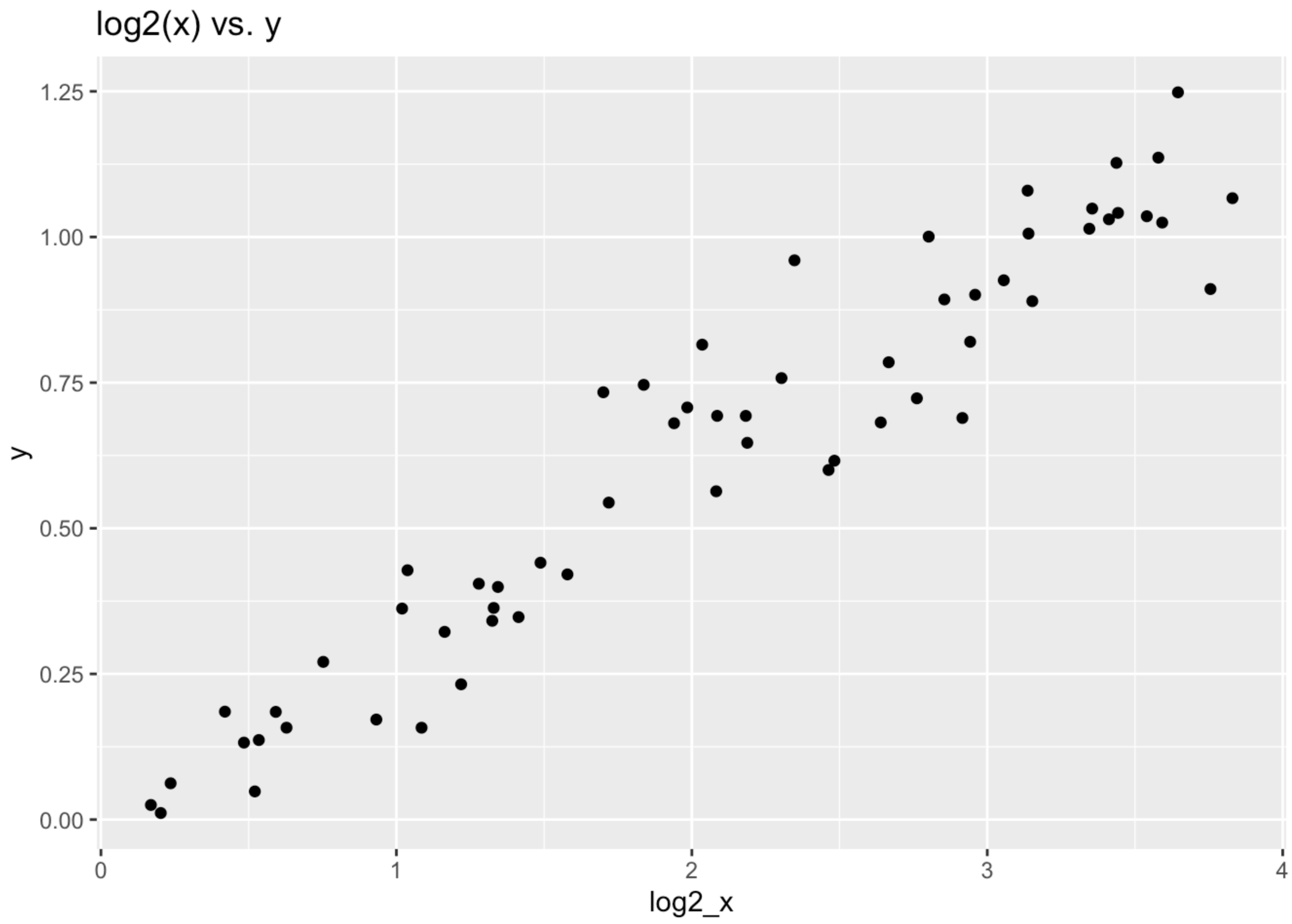
```
# x vs. y
p1 <- ggplot(sim8, aes(x, y)) + geom_point() + labs(title = "x vs. y")

p2 <- ggplot(sim8, aes(x, log2_y)) + geom_point() + labs(title = "x vs. log2(y)")

p3 <- ggplot(sim8, aes(log2_x, y)) + geom_point() + labs(title = "log2(x) vs. y")

p4 <- ggplot(sim8, aes(log2_x, log2_y)) + geom_point() + labs(title = "log2(x) vs. log2
(y)")

p1
```



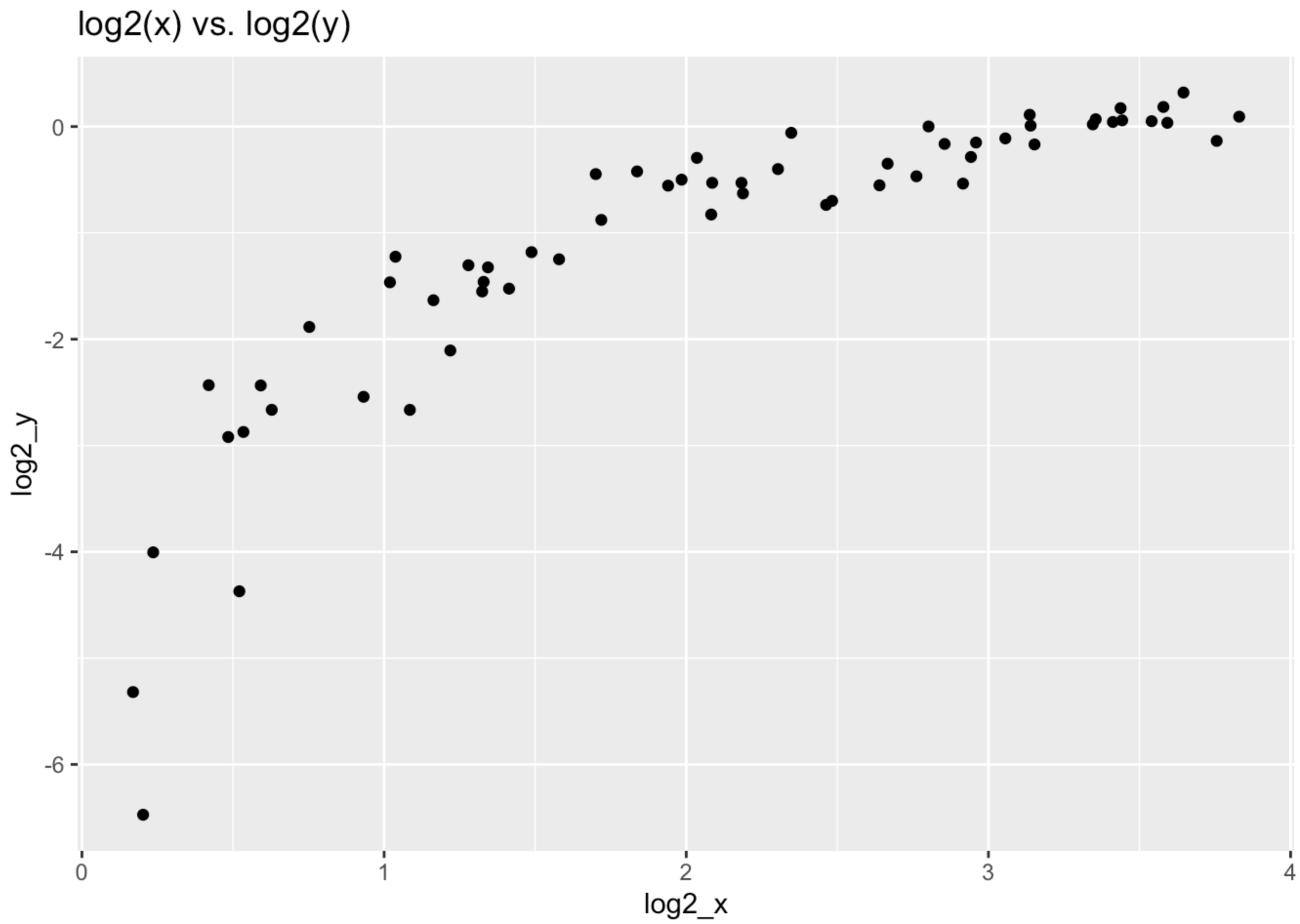
```
p2
```



```
p3
```



```
p4
```



The plot of log2(x) vs. y looks most linear, suggesting a logarithmic model is appropriate.

c. (4 points) **Using your answer from the previous question, determine what linear model you should make, and make this linear model. Write out the equation of your linear model. What value does your model predict for y when x is 4?**

```
log_model <- lm(y ~ log2_x, data = sim8)
log_model
```

```
##
## Call:
## lm(formula = y ~ log2_x, data = sim8)
##
## Coefficients:
## (Intercept)       log2_x
##   -0.006647     0.305009
```

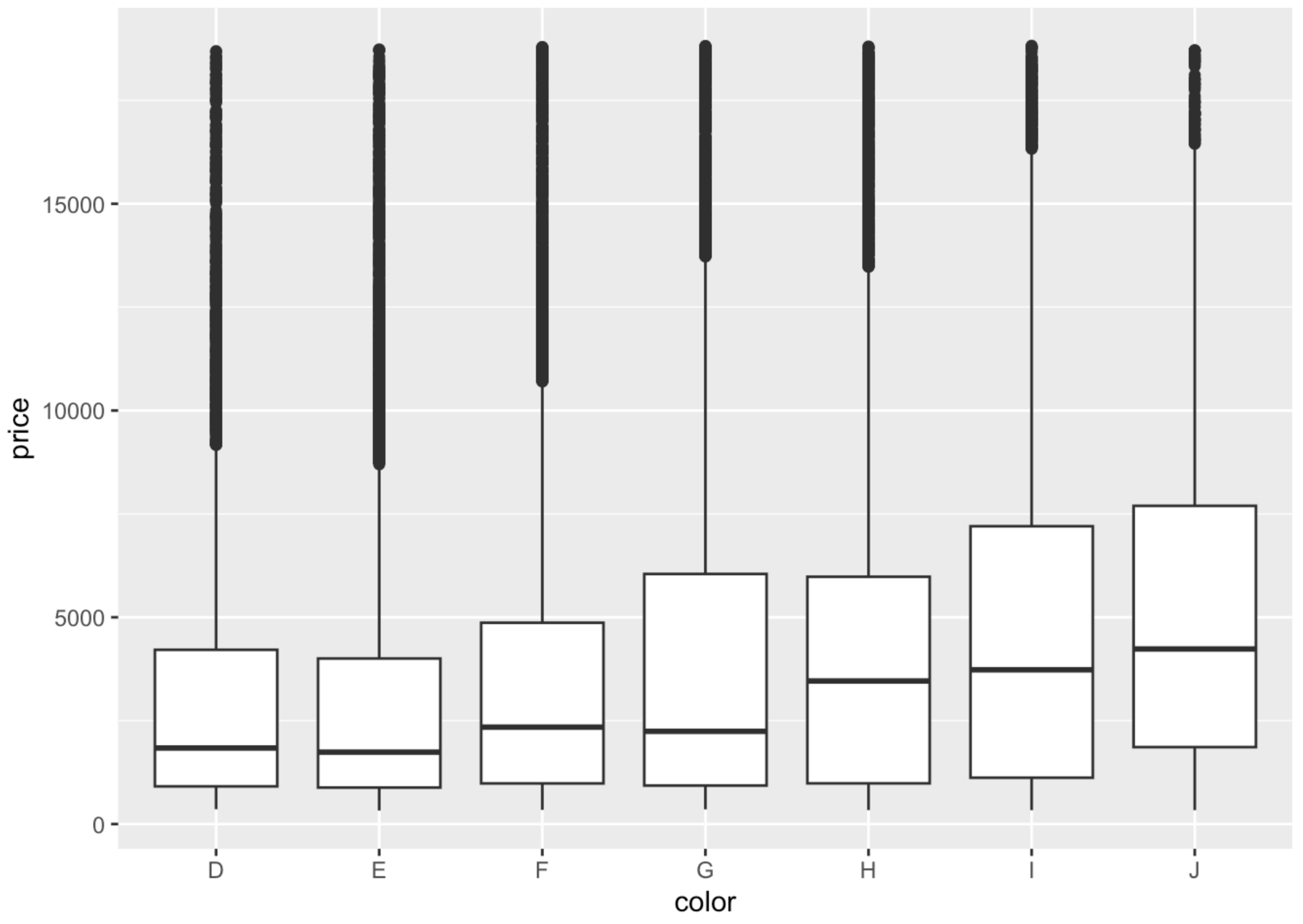
y = -0.006647 + 0.305009 log2(x)

when x = 4: y = 0.603371

Question 7 (5 points)

> **When you look at the diamonds data set, it seems like diamonds with better colors have lower prices (letters closer to the beginning of the alphabet are better color ratings):**

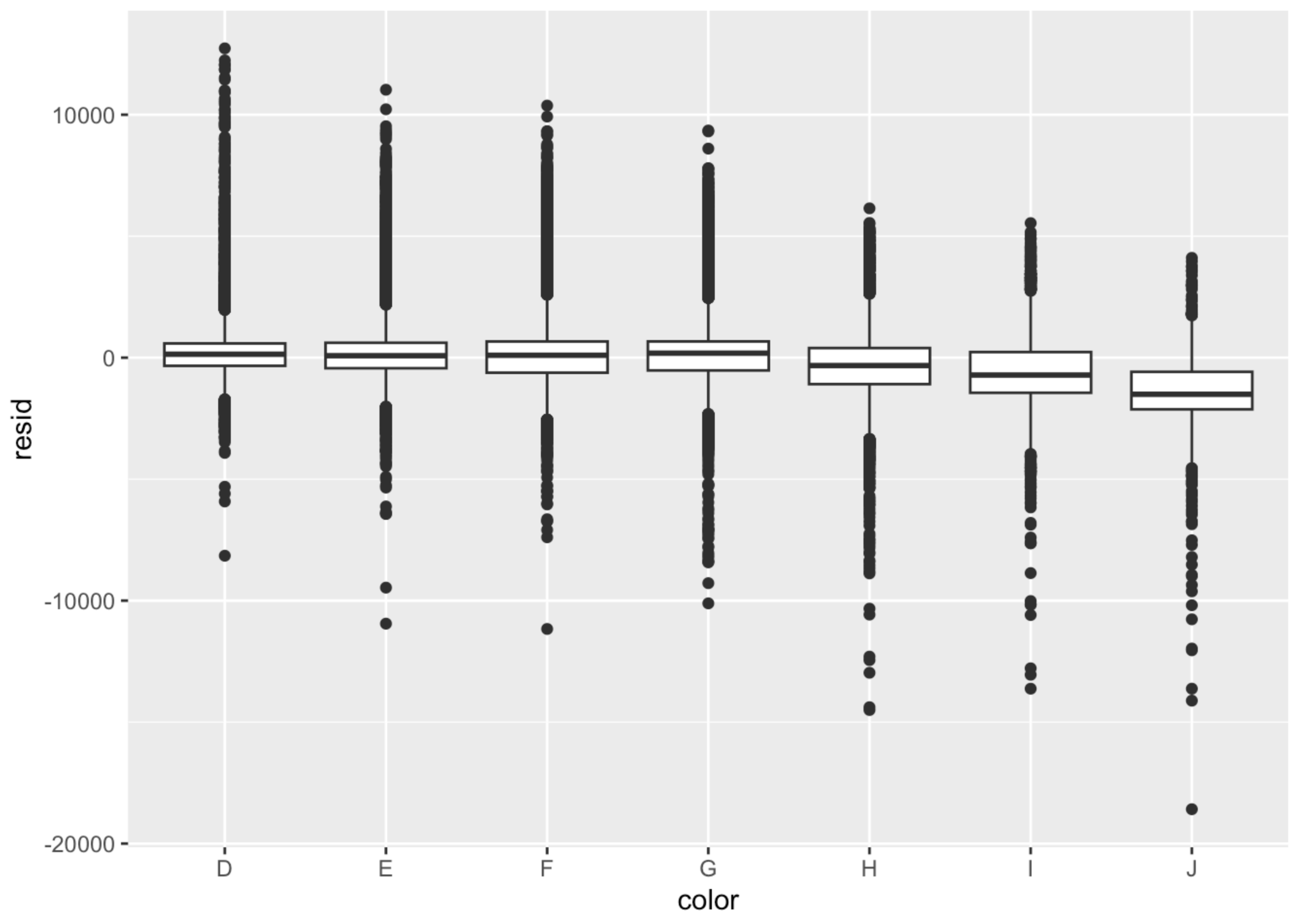
```
ggplot(diamonds) + geom_boxplot(aes(color, price))
```



>

However, size (carat) might be a confounding variable here. When you look at what part of a diamond's price isn't explained by size, what do you observe about the relationship between color and price?

```
diamond_model <- lm(price ~ carat, data = diamonds)

diamonds %>%
  add_residuals(diamond_model) %>%
  ggplot() +
  geom_boxplot(aes(color, resid))
```



After accounting for carat, there is a slight negative relationship between color and price. Diamonds with better color grades (closer to D) have slightly higher residuals than those with worse grades (closer to J). This suggests the apparent relationship between color and price was largely due to size differences.

Question 8 (10 points)

> **Consider the flights data set.**

```r
library(nycflights13)
set.seed(555555) # Fixes the randomness used throughout the rest of this question, so st
udents all get  the same answers
```

> a. (2 points) **First, remove all flights that do not have an arrival delay. Split the remaining data up into training and test data, where 80% of your data is in the training data set and 20% is in the test data set.**

```
flights_clean <- flights %>%
  filter(!is.na(arr_delay))

train <- flights_clean %>%
  sample_frac(0.8)

test <- anti_join(flights_clean, train)
```

```
## Joining with `by = join_by(year, month, day, dep_time, sched_dep_time,
## dep_delay, arr_time, sched_arr_time, arr_delay, carrier, flight, tailnum,
## origin, dest, air_time, distance, hour, minute, time_hour)`
```

> b. (3 points) **Make a model, using your training data, that attempts to predict arrival delay using departure delay. Compute the mean of the squares of the residuals of this model on the training data and on the test data. Do you think there is overfitting?**

```
model1 <- lm(arr_delay ~ dep_delay, data = train)

train_mse <- train %>%
  add_residuals(model1) %>%
  summarize(mse = mean(resid^2))

test_mse <- test %>%
  add_residuals(model1) %>%
  summarize(mse = mean(resid^2))

train_mse
```

```
## # A tibble: 1 × 1
##     mse
##   <dbl>
## 1  325.
```

```
test_mse
```

```
## # A tibble: 1 × 1
##     mse
##   <dbl>
## 1  326.
```

Since these values are very close, there doesn't appear to be overfitting. An overfitted model would perform notably better on the training data than on the test data.

c. (5 points) **Make a model trying to predict arrival delay using at least three different columns in flights. You can choose which columns to use and whether to use interactions or not, but your mean residuals squared for the training data should be less than for the model in the previous part. Compute the mean squared residual for your model on both the training data and the test data. Do you think there's overfitting in your model?**

```
model2 <- lm(arr_delay ~ dep_delay + distance + air_time, data = train)

train_mse2 <- train %>%
  add_residuals(model2) %>%
  summarize(mse = mean(resid^2))

test_mse2 <- test %>%
  add_residuals(model2) %>%
  summarize(mse = mean(resid^2))

train_mse2
```

```
## # A tibble: 1 × 1
##     mse
##   <dbl>
## 1  244.
```

```
test_mse2
```

```
## # A tibble: 1 × 1
##     mse
##   <dbl>
## 1  247.
```

new model with three variables (dep_delay, distance, and air_time) performs better than the original model: Training MSE: 244 (improved from 325) Test MSE: 247 (improved from 326)

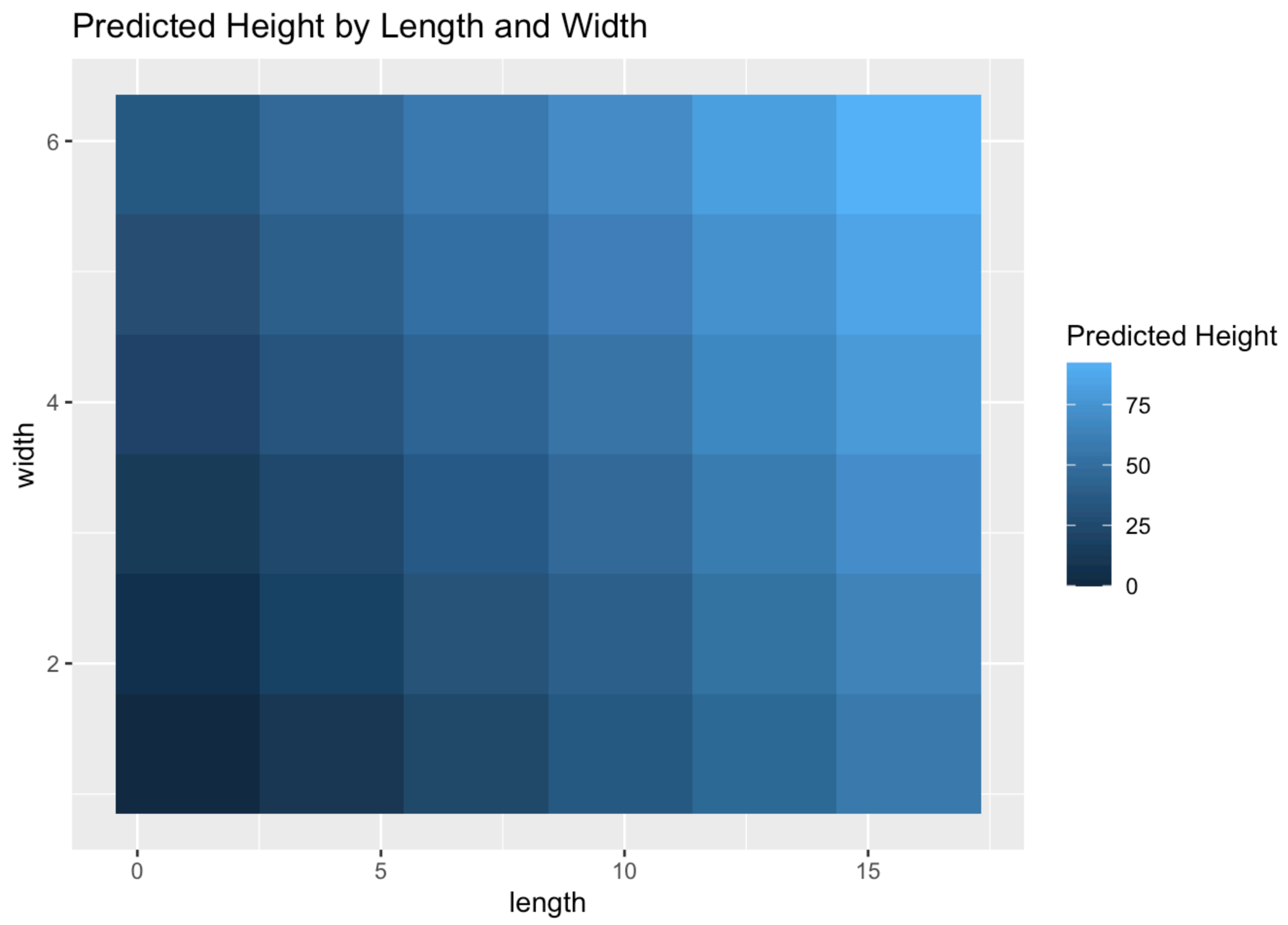
The small difference between training and test MSE suggests there isn't significant overfitting, even though we added more variables to the model.

Question 9 (4 points)

a. **Explain what geom_tile() does, explain when you might want to use it, and make an example plot using geom_tile() and a model different from the one we used in class (the data grid with predictions from Question 2 or 3 might be particularly convenient).**

geom_tile() creates a heatmap-style visualization where rectangles are colored based on a value. It's useful when you want to show the relationship between two continuous variables and a third variable represented by color.

```r
grid %>%
  add_predictions(model) %>%
  ggplot(aes(x = length, y = width, fill = pred)) +
  geom_tile() +
  labs(title = "Predicted Height by Length and Width",
       fill = "Predicted Height")
```

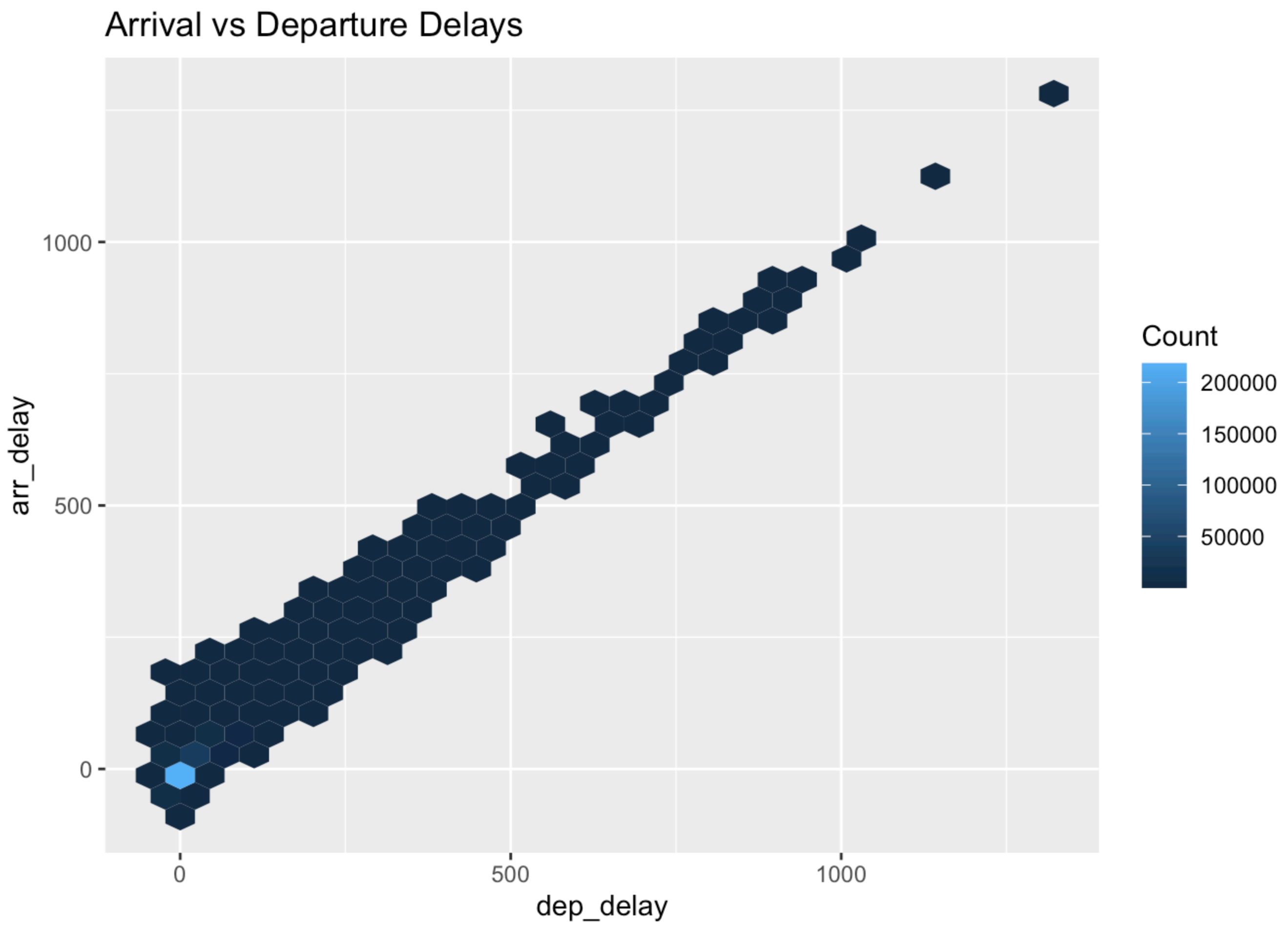


b. **Explain what geom_hex() does, explain when you might want to use it, and make an example plot using geom_hex() and a data set different from the one we used in class (diamonds).**

geom_hex() creates a hexagonal heatmap showing the density of points. It’s useful for large datasets where points would overlap too much in a regular scatter plot.

```
ggplot(flights, aes(x = dep_delay, y = arr_delay)) +
  geom_hex() +
  labs(title = "Arrival vs Departure Delays",
       fill = "Count")
```

```
## Warning: Removed 9430 rows containing non-finite outside the scale range
## (`stat_binhex()`).
```



Question 10 (6 points)

Here is an example tibble to use for this question:

```
example <- tibble(date = rep(c(1:14), each = 2),
                  day_of_week = rep(rep(c("Sunday", "Monday", "Tuesday",
                                        "Wednesday", "Thursday", "Friday",
                                        "Saturday"), times = 2), each = 2),
                  time_of_day = rep(c("Morning", "Afternoon"), times = 14),
                  num_observations = c(38,45,16,32,45,32,45,43,23,38,
                                       75,34,54,28,38,42,16,32,45,32,
                                       24,43,23,45,75,34,54,28))

example
```

```
## # A tibble: 28 × 4
##     date day_of_week time_of_day num_observations
##    <int> <chr>       <chr>                  <dbl>
##  1     1 Sunday      Morning                   38
##  2     1 Sunday      Afternoon                 45
##  3     2 Monday      Morning                   16
##  4     2 Monday      Afternoon                 32
##  5     3 Tuesday     Morning                   45
##  6     3 Tuesday     Afternoon                 32
##  7     4 Wednesday   Morning                   45
##  8     4 Wednesday   Afternoon                 43
##  9     5 Thursday    Morning                   23
## 10     5 Thursday    Afternoon                 38
## # ℹ 18 more rows
```

a. **Explain why the following two commands produce different output:**

```
data_grid(example, day_of_week, time_of_day)
```

```
## # A tibble: 14 × 2
##    day_of_week time_of_day
##    <chr>       <chr>
##  1 Friday      Afternoon
##  2 Friday      Morning
##  3 Monday      Afternoon
##  4 Monday      Morning
##  5 Saturday    Afternoon
##  6 Saturday    Morning
##  7 Sunday      Afternoon
##  8 Sunday      Morning
##  9 Thursday    Afternoon
## 10 Thursday    Morning
## 11 Tuesday     Afternoon
## 12 Tuesday     Morning
## 13 Wednesday   Afternoon
## 14 Wednesday   Morning
```

```
select(example, day_of_week, time_of_day)
```

```
## # A tibble: 28 × 2
##    day_of_week time_of_day
##    <chr>       <chr>
##  1 Sunday      Morning
##  2 Sunday      Afternoon
##  3 Monday      Morning
##  4 Monday      Afternoon
##  5 Tuesday     Morning
##  6 Tuesday     Afternoon
##  7 Wednesday   Morning
##  8 Wednesday   Afternoon
##  9 Thursday    Morning
## 10 Thursday    Afternoon
## # ℹ 18 more rows
```

the first creates a grid with all possible combinations of day_of_week and time_of_day values.

the second shows those two columns from the existing data, including duplicates

b. **Explain why the following two commands produce the same output:**

```
example %>% data_grid(day_of_week, time_of_day)
```

```
## # A tibble: 14 × 2
##    day_of_week time_of_day
##    <chr>       <chr>
##  1 Friday      Afternoon
##  2 Friday      Morning
##  3 Monday      Afternoon
##  4 Monday      Morning
##  5 Saturday    Afternoon
##  6 Saturday    Morning
##  7 Sunday      Afternoon
##  8 Sunday      Morning
##  9 Thursday    Afternoon
## 10 Thursday    Morning
## 11 Tuesday     Afternoon
## 12 Tuesday     Morning
## 13 Wednesday   Afternoon
## 14 Wednesday   Morning
```

```
example %>% distinct(day_of_week, time_of_day) %>% complete(day_of_week, time_of_day)
```

```
## # A tibble: 14 × 2
##    day_of_week time_of_day
##    <chr>       <chr>
##  1 Friday      Afternoon
##  2 Friday      Morning
##  3 Monday      Afternoon
##  4 Monday      Morning
##  5 Saturday    Afternoon
##  6 Saturday    Morning
##  7 Sunday      Afternoon
##  8 Sunday      Morning
##  9 Thursday    Afternoon
## 10 Thursday    Morning
## 11 Tuesday     Afternoon
## 12 Tuesday     Morning
## 13 Wednesday   Afternoon
## 14 Wednesday   Morning
```

the are he same because create a complete grid of all unique combinations of day_of_week and time_of_day

The second achieves the same result by first finding unique combinations and then ensuring all combinations are present

c. **Explain why creating the following data grid would be a bad idea:**

```
data_grid(example, date, day_of_week)
```

```
## # A tibble: 98 × 2
##     date day_of_week
##    <int> <chr>
##  1     1 Friday
##  2     1 Monday
##  3     1 Saturday
##  4     1 Sunday
##  5     1 Thursday
##  6     1 Tuesday
##  7     1 Wednesday
##  8     2 Friday
##  9     2 Monday
## 10     2 Saturday
## # ℹ 88 more rows
```

Bad ideas because there's a one-to-one relationship between date and day_of_week (each date has exactly one day_of_week). This would create unnecessary combinations that don't make sense.