L15 Model Building

Data Science I (STAT 301-1)

Shay Lebovitz

Contents

Overview

The goal of this lab is for students to explore and develop modeling techniques and tools to explore data. The lab's focus is on real data, specifically on how to progressively build up a model to aid in our understanding of a select few datasets (`ggplot2::diamonds` and `nycflight13::flights`).

Datasets

We will be utilizing the familiar `diamonds` and `flight` datasets from the `ggplot2` and `nycflights` packages, respectively.

Exercises

Please complete the following exercises. Be sure that your solutions are clearly indicated and that the document is neatly formatted.

```r
# Loading package(s)
library(tidyverse)
library(nycflights13)
library(hexbin)
library(modelr)
library(lubridate)
```

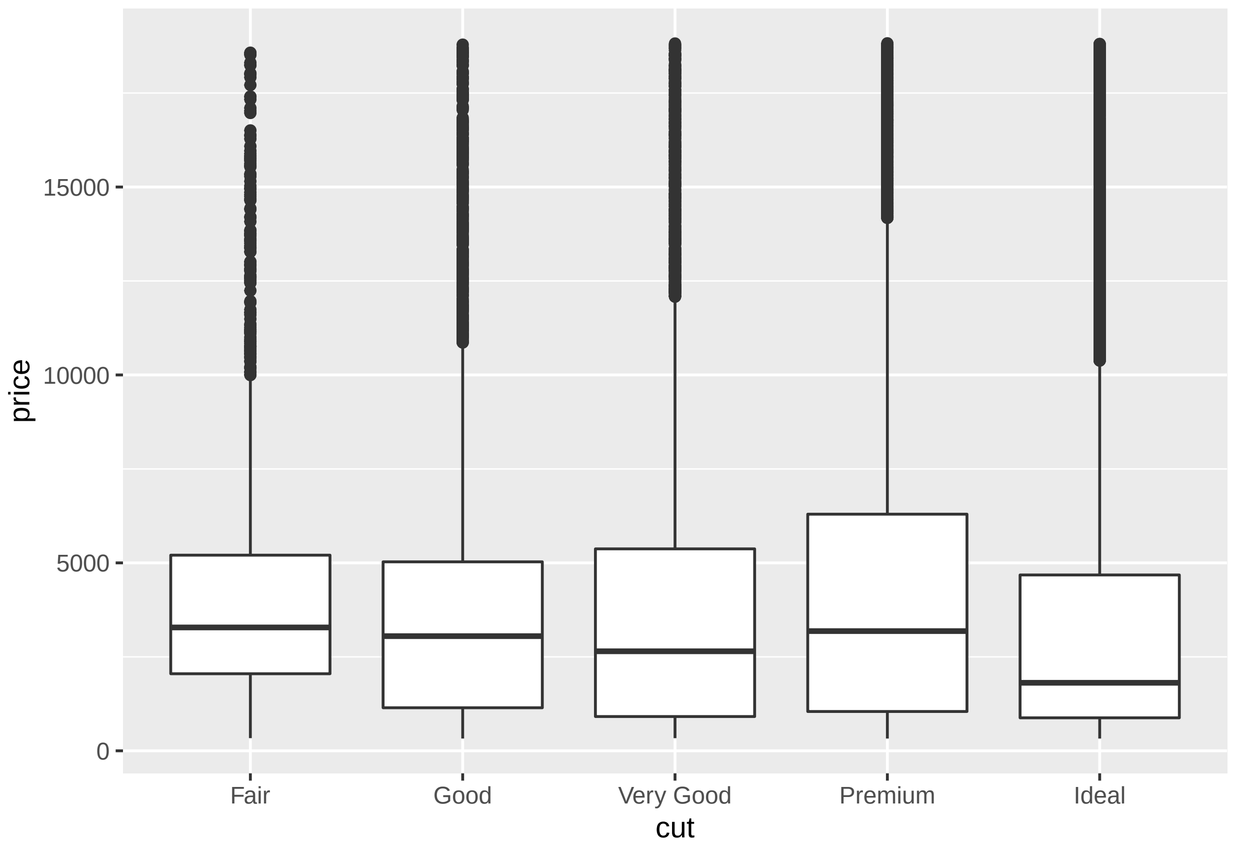
Load Packages

Exercise 1

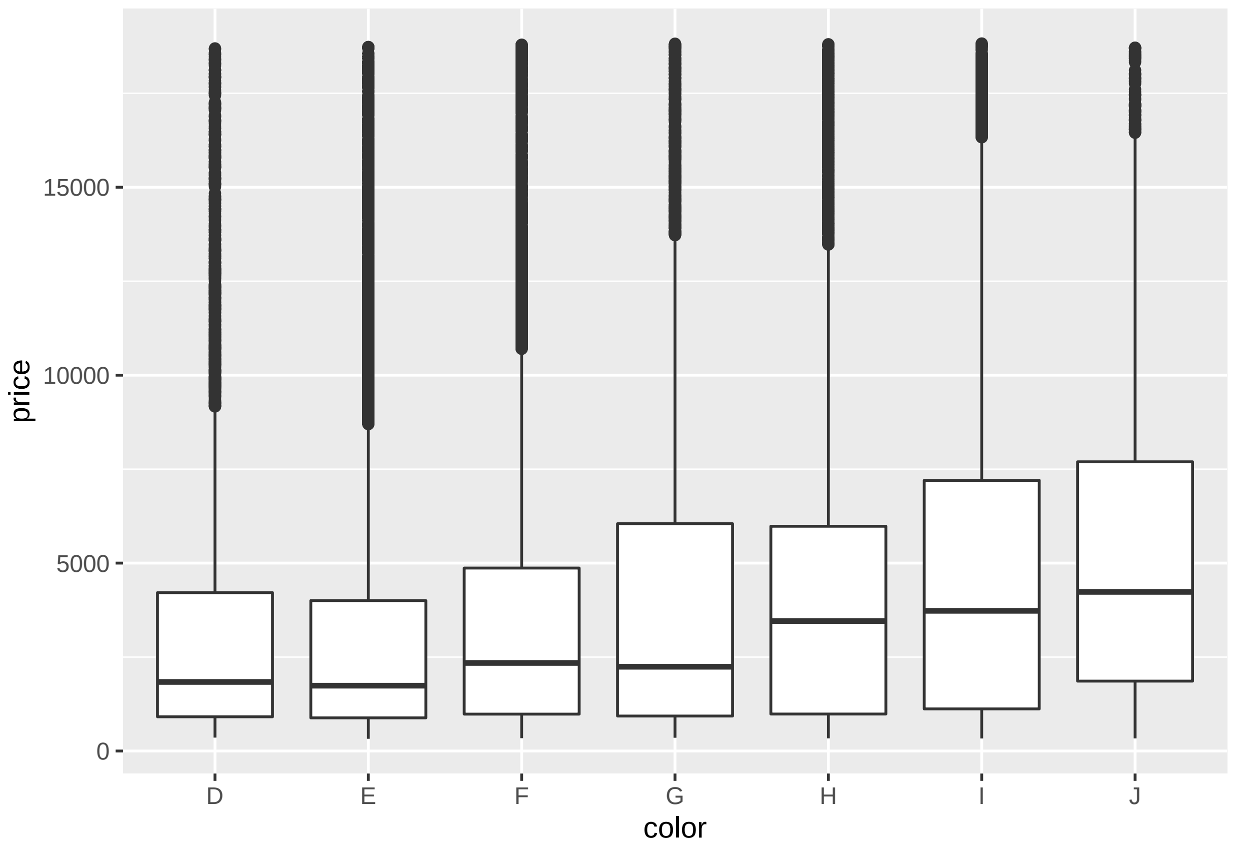
Work through and transcribe (do not copy and paste – rewrite) the analyses detailed in section 24.2 Why are low quality diamonds more expensive?. Consider adding your own comments/notes and using your own naming conventions.

It is strange that higher quality diamonds tend to cost less than lower quality diamonds, as shown in these plots.

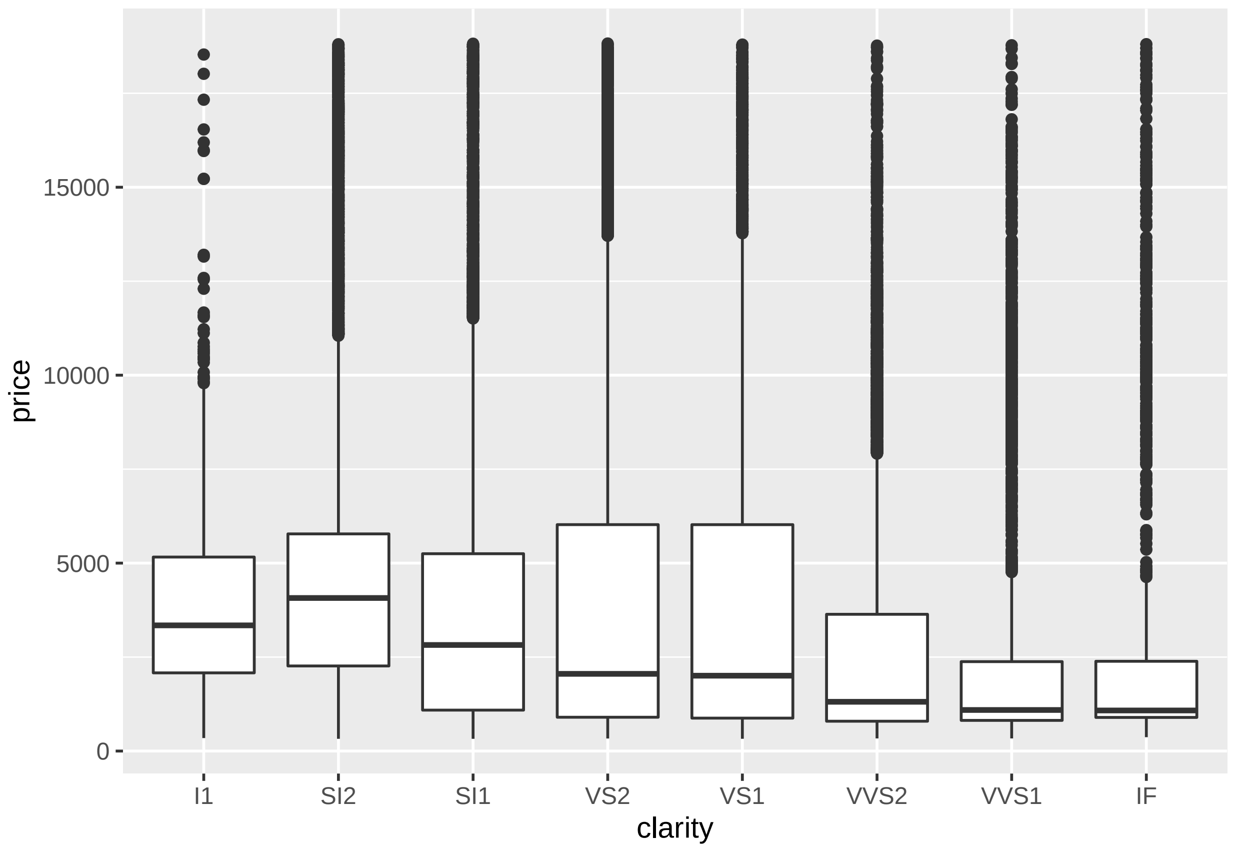
```
ggplot(diamonds, aes(cut, price)) + geom_boxplot()
```



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

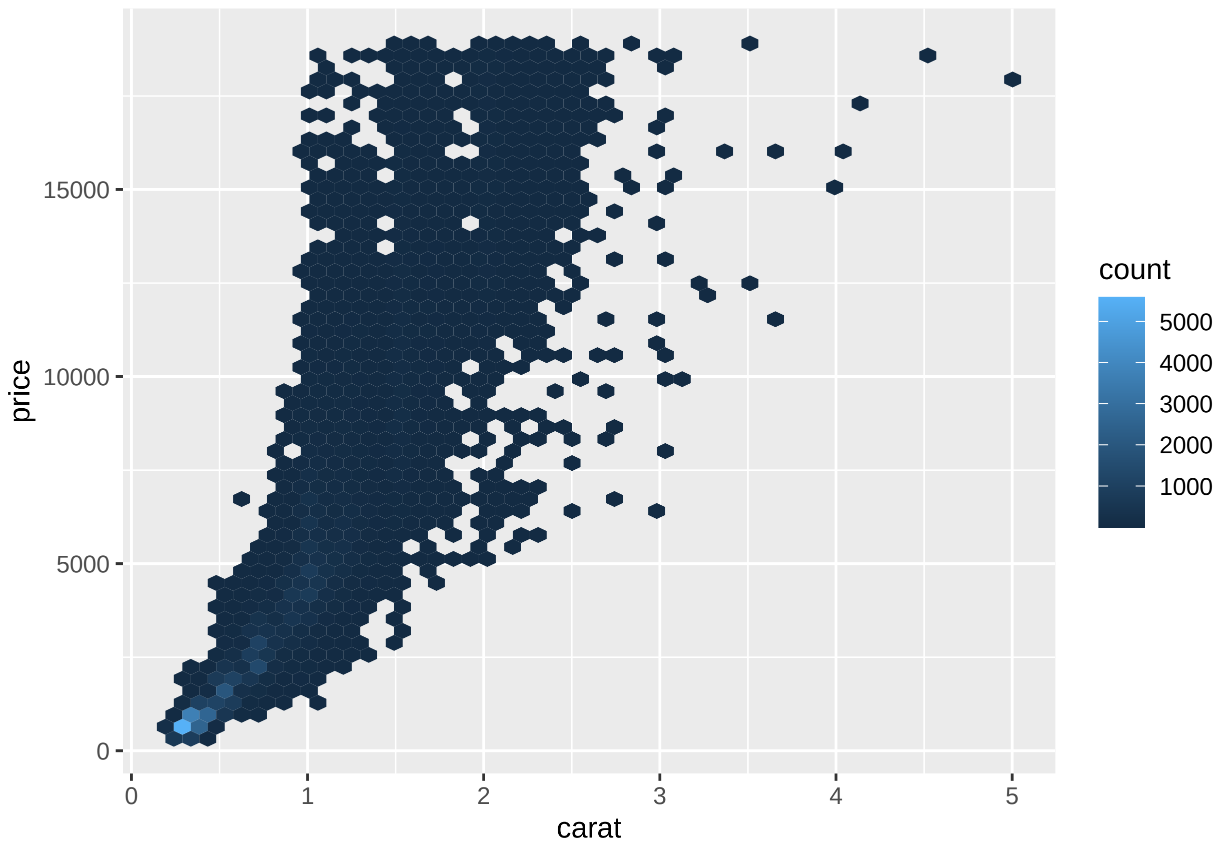


```
ggplot(diamonds, aes(clarity, price)) + geom_boxplot()
```



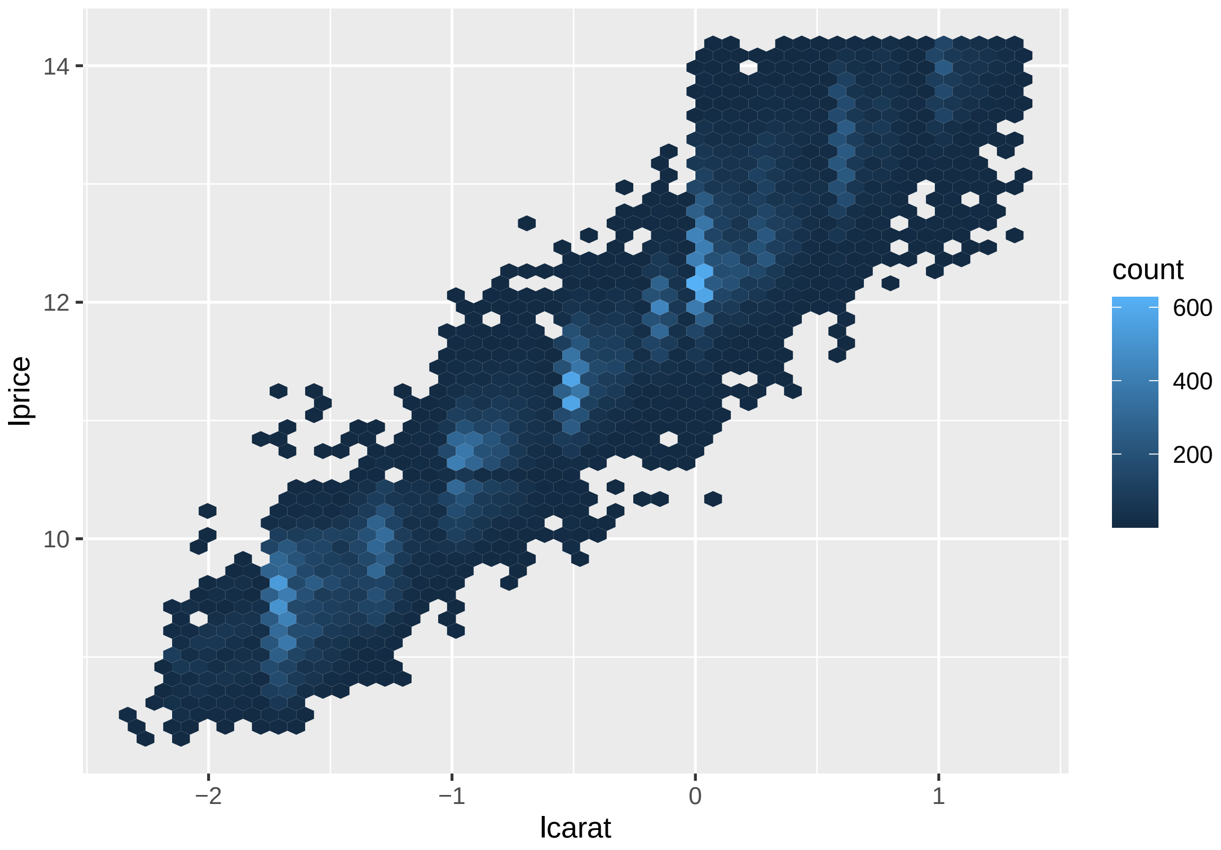
The reason for this is that the weight of the diamond is the most important factor in determining its price.

```
ggplot(diamonds, aes(carat, price)) +
  geom_hex(bins = 50)
```



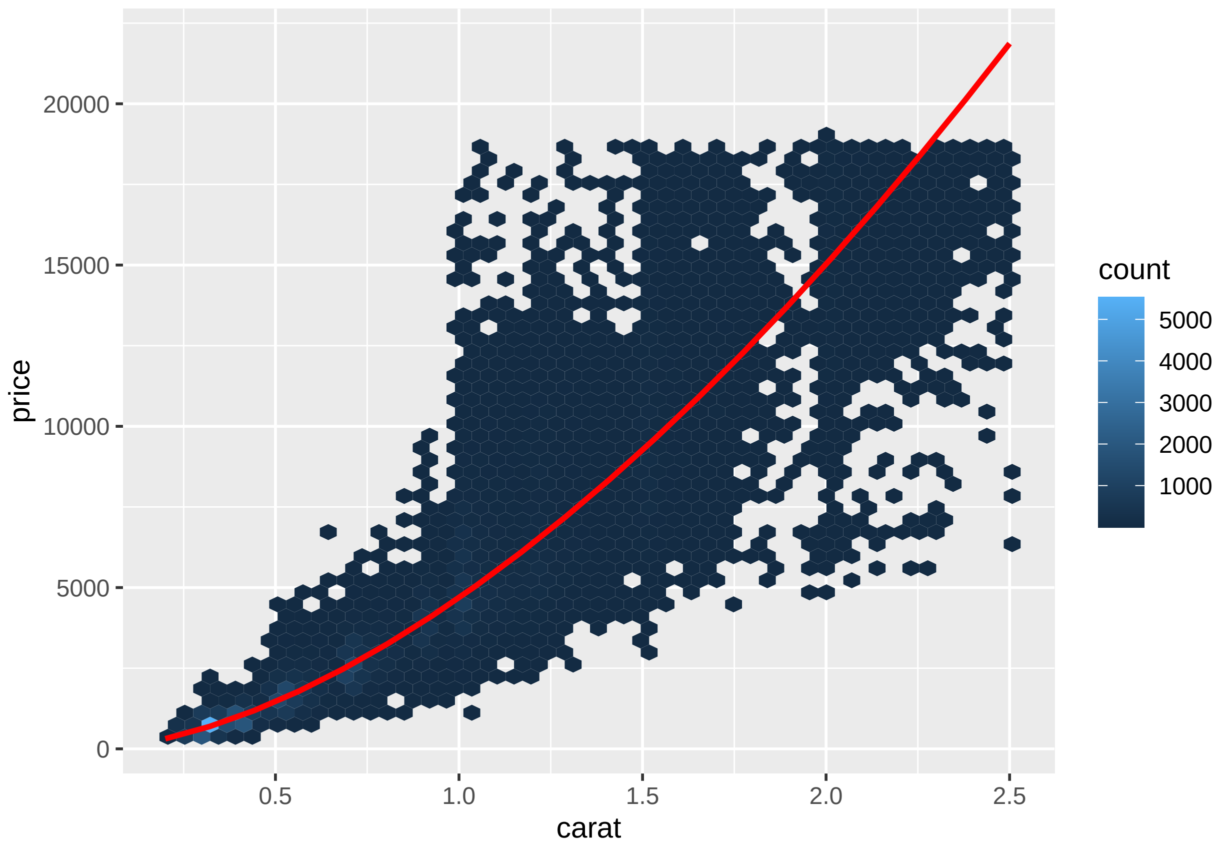
This plot clearly shows a relationship between diamond weight and price, with higher weight diamonds tending to cost more. To determine what other factors determine `price`, we need to factor out the effect of `carat`. First, we only look at diamonds with `carat` less than 2.5 and log the prices.

```
diamonds2 <- diamonds %>%
  filter(carat <= 2.5) %>%
  mutate(lprice = log2(price), lcarat = log2(carat))

ggplot(diamonds2, aes(lcarat, lprice)) +
  geom_hex(bins = 50)
```



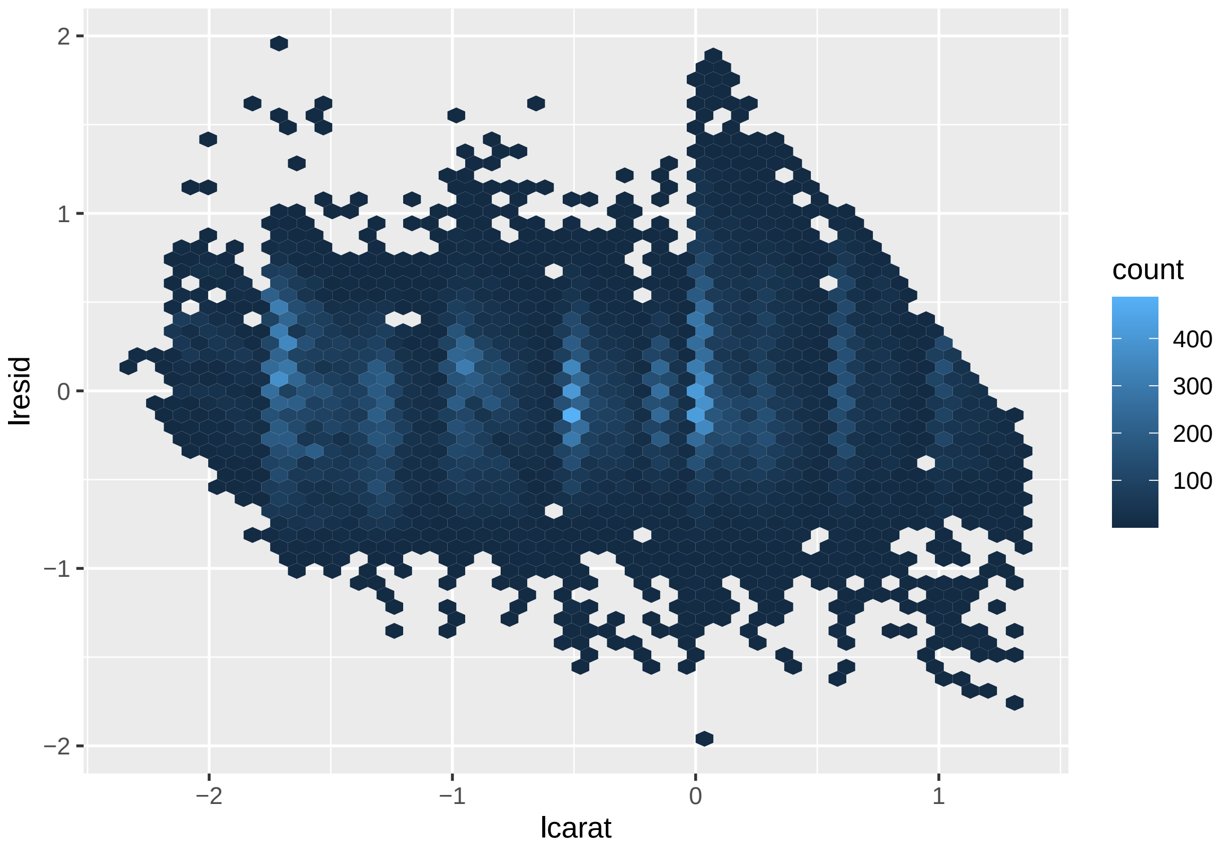
This linearizes the plot so that we can more easily see the trend. Next, we can create a linear model of the logged price on logged carat.

```
mod_diamond <- lm(lprice ~ lcarat, data = diamonds2)

grid <- diamonds2 %>%
  data_grid(carat = seq_range(carat, 20)) %>%
  mutate(lcarat = log2(carat)) %>%
  add_predictions(mod_diamond, "lprice") %>%
  mutate(price = 2 ^ lprice)

ggplot(diamonds2, aes(carat, price)) +
  geom_hex(bins = 50) +
  geom_line(data = grid, colour = "red", size = 1)
```



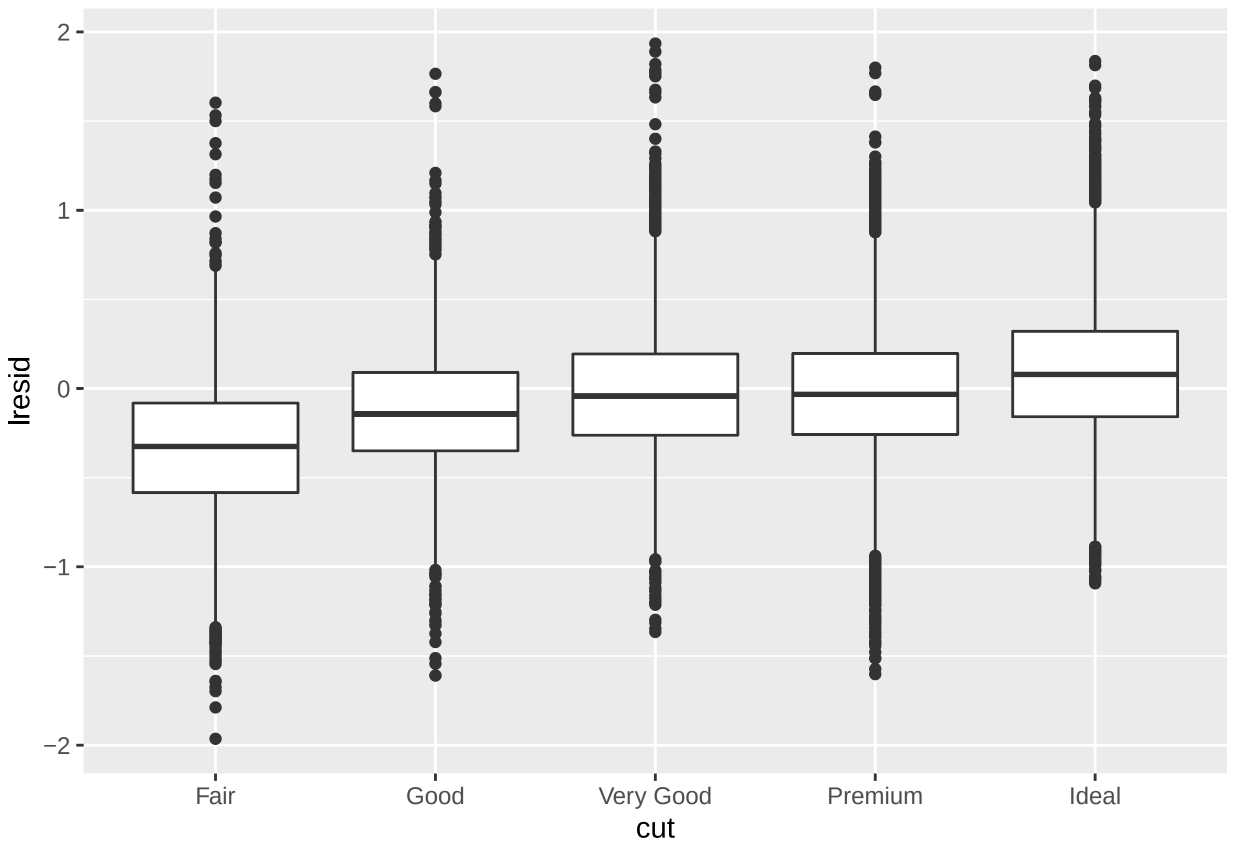
This shows the model after being 'unlogged' so that it shows the original relationship. It shows that the majority of the diamonds above 2.0 carats are below the predicted line, showing they are 'underpriced'. To verify we removed the strong linear pattern, we can look at the residuals.

```
diamonds2 <- diamonds2 %>%
  add_residuals(mod_diamond, "lresid")

ggplot(diamonds2, aes(lcarat, lresid)) +
  geom_hex(bins = 50)
```

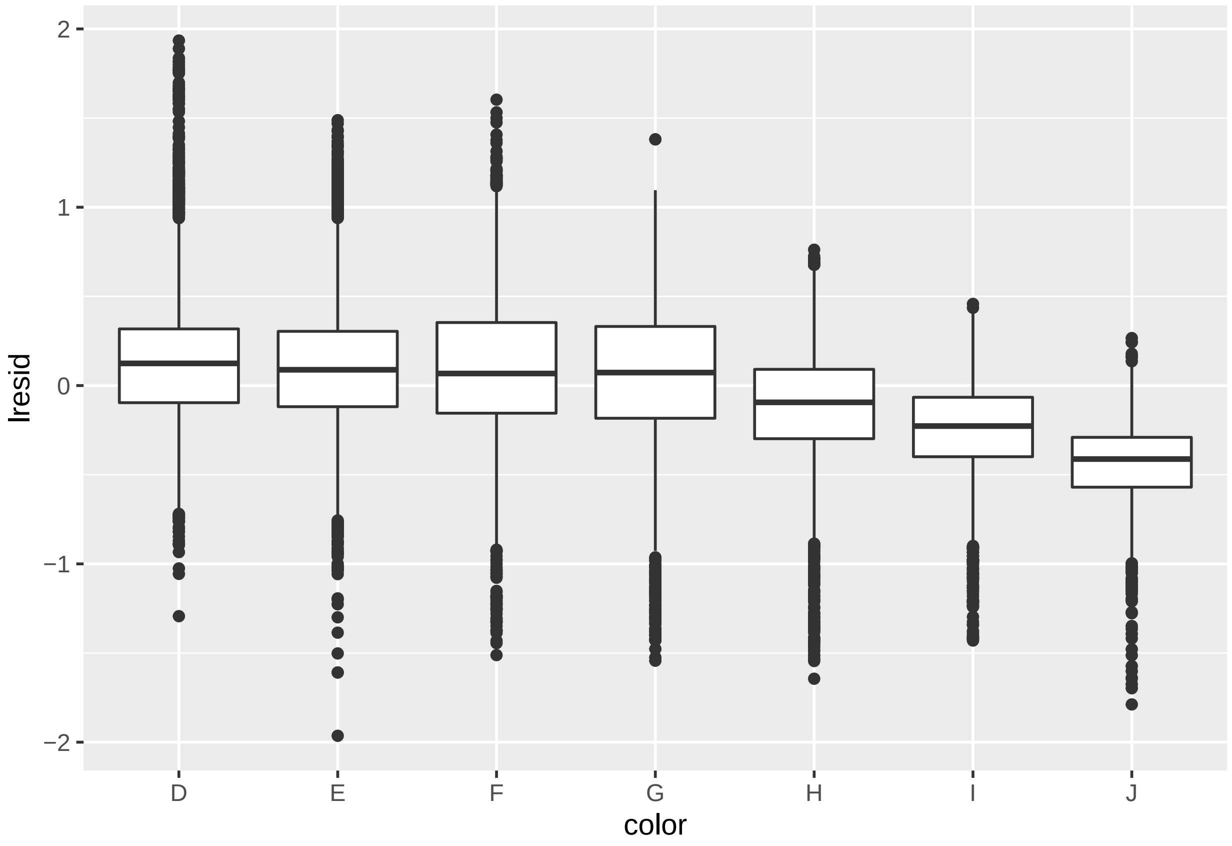


We see no pattern or relationship here, which is good. Next, we can recheck how diamond quality relates to price, using the updated `diamonds2` data.

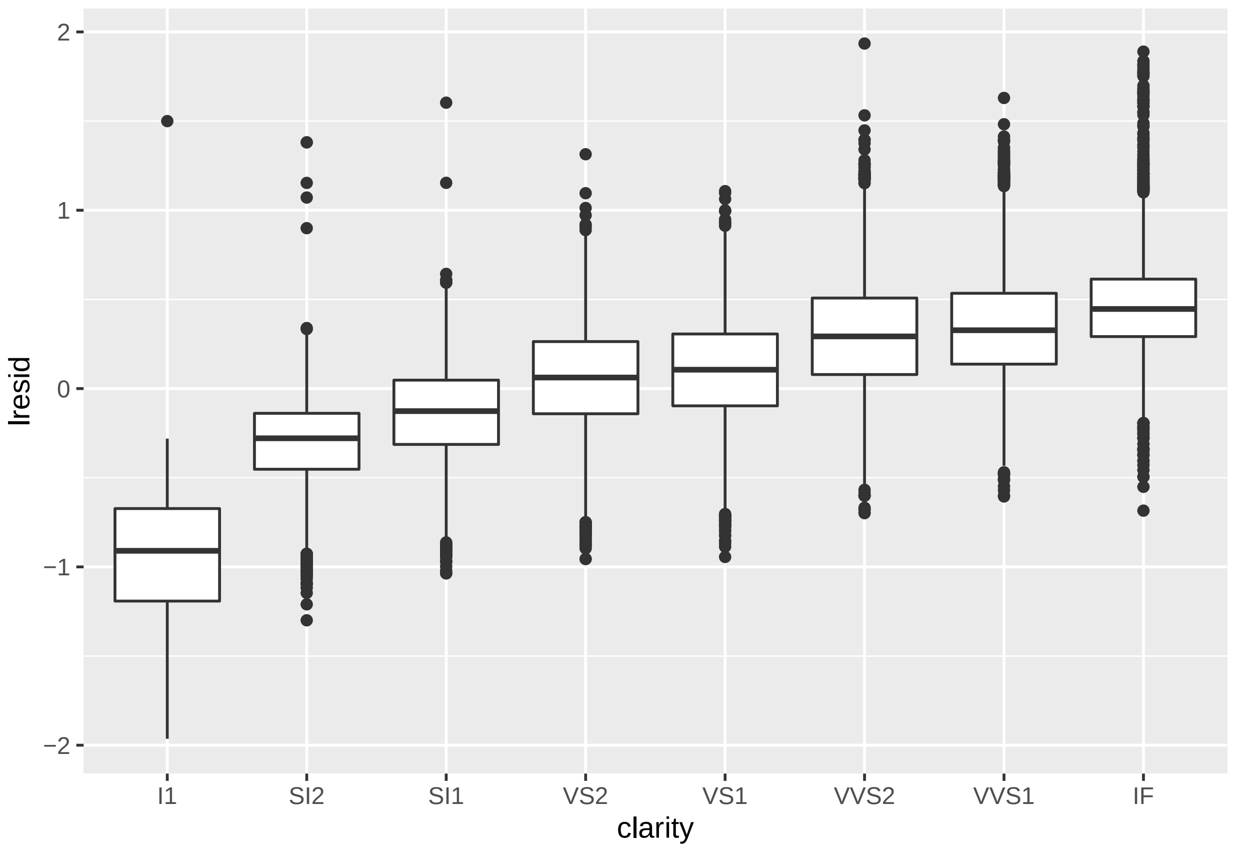
```
ggplot(diamonds2, aes(cut, lresid)) + geom_boxplot()
```



```
ggplot(diamonds2, aes(color, lresid)) + geom_boxplot()
```



```
ggplot(diamonds2, aes(clarity, lresid)) + geom_boxplot()
```



Here we clearly see that price does infact go up with increasing cut, color, and clarity quality.You can think of the y-axis as the expected difference between the price when factoring out weight vs. when not. Thus, a residual of -1 indicates that the logged price was one unit lower than a prediction based solely on weight.

Exercise 2 (Website: 24.2.3 Ex. 2)

If `log(price) = a_0 + a_1 * log(carat)`, what does that model say about the relationship between `price` and `carat`? **Take `log(price) = a_0 + a_1 * log(carat)` and exponentiate both sides. Then we get `price = e^(a_0 + a_1) * carat`, which is a linear relationship. This tells us that `ln(price)` scales linearly with `ln(carat)`, and thus `price` scales exponentially with `carat`.**

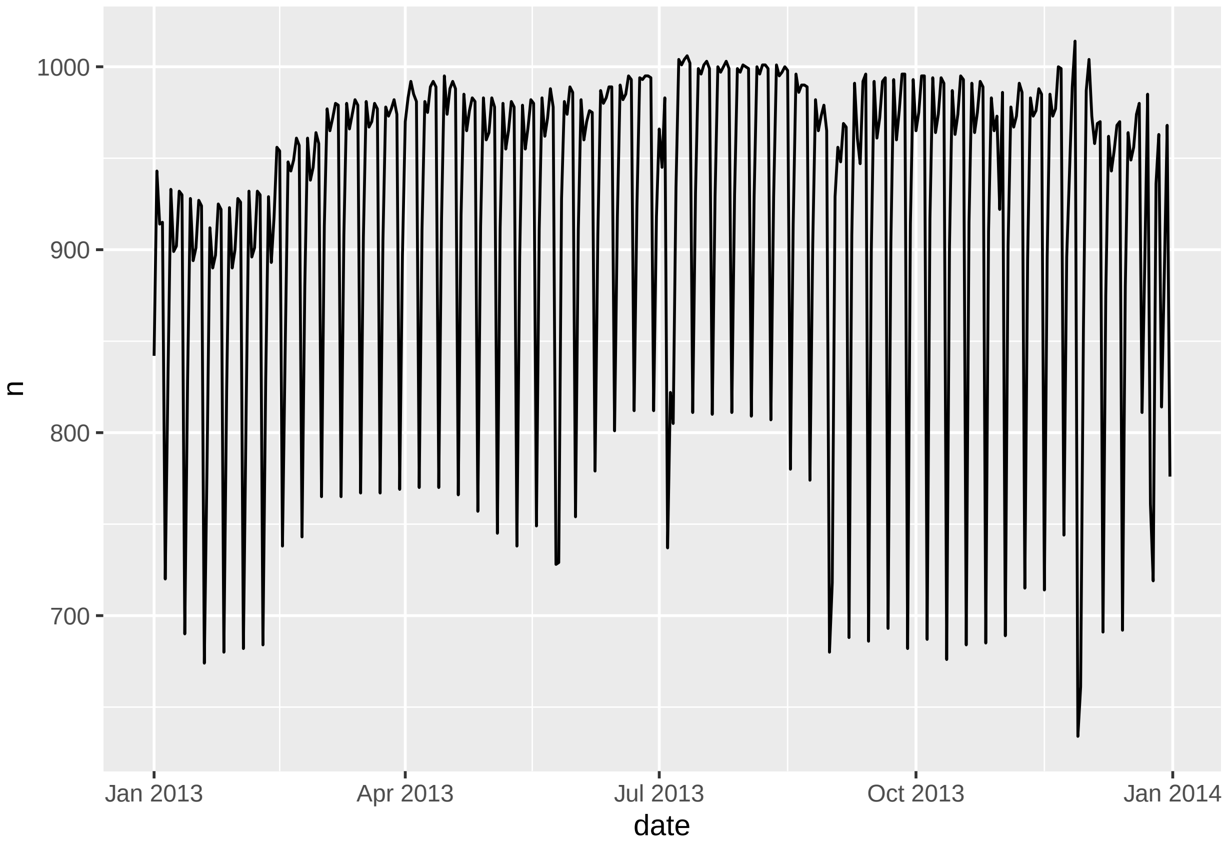
Exercise 3

Work through and transcribe (do not copy and paste – rewrite) the analysis detailed in section 24.3 What affects the number of daily flights? Consider adding your own comments/notes and using your own naming conventions. **Start by visualizing the number of flights that leave NYC airports every day.**

```
daily <- flights %>%
  mutate(date = make_date(year, month, day)) %>%
  group_by(date) %>%
  summarise(n = n())
daily
```

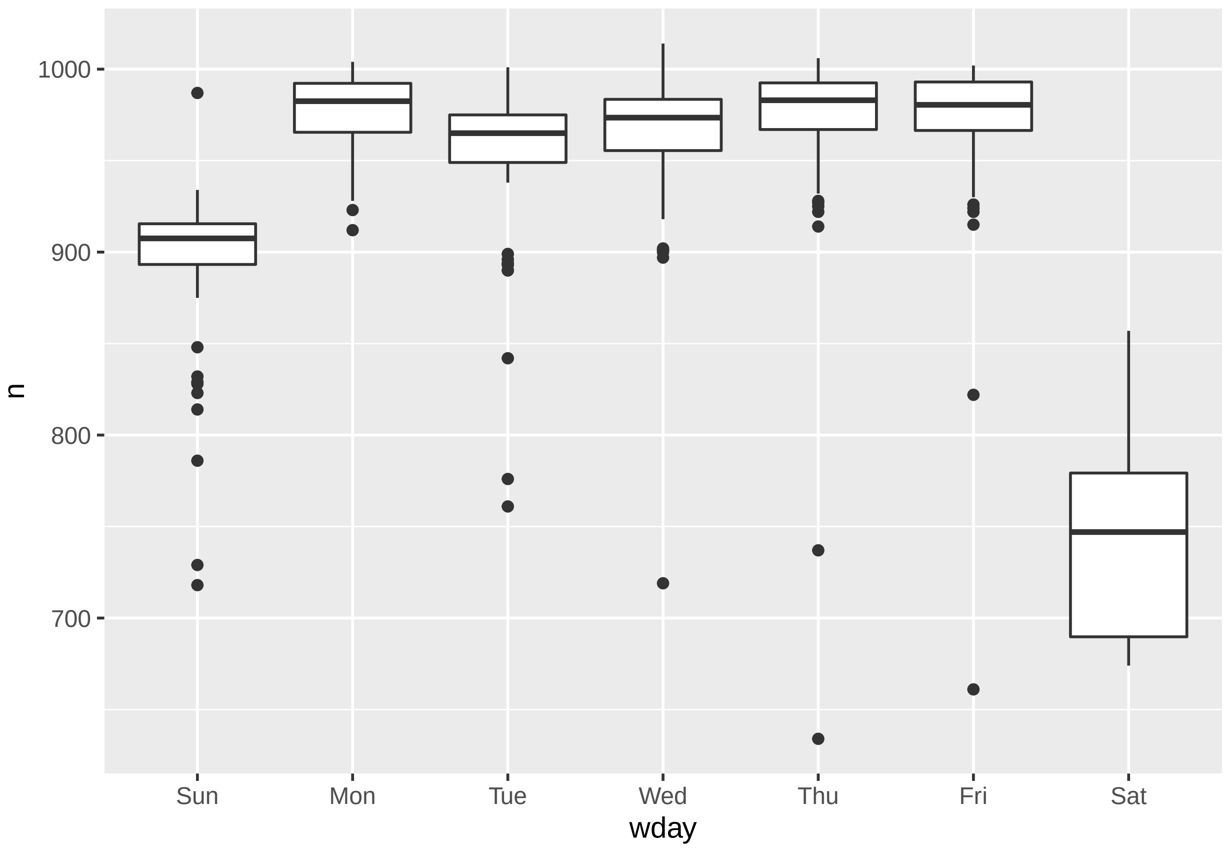
```
## # A tibble: 365 x 2
##    date           n
##    <date>     <int>
##    1 2013-01-01   842
##    2 2013-01-02   943
##    3 2013-01-03   914
##    4 2013-01-04   915
##    5 2013-01-05   720
##    6 2013-01-06   832
##    7 2013-01-07   933
##    8 2013-01-08   899
##    9 2013-01-09   902
##   10 2013-01-10   932
## # ... with 355 more rows
```

```
ggplot(daily, aes(date, n)) +
  geom_line()
```



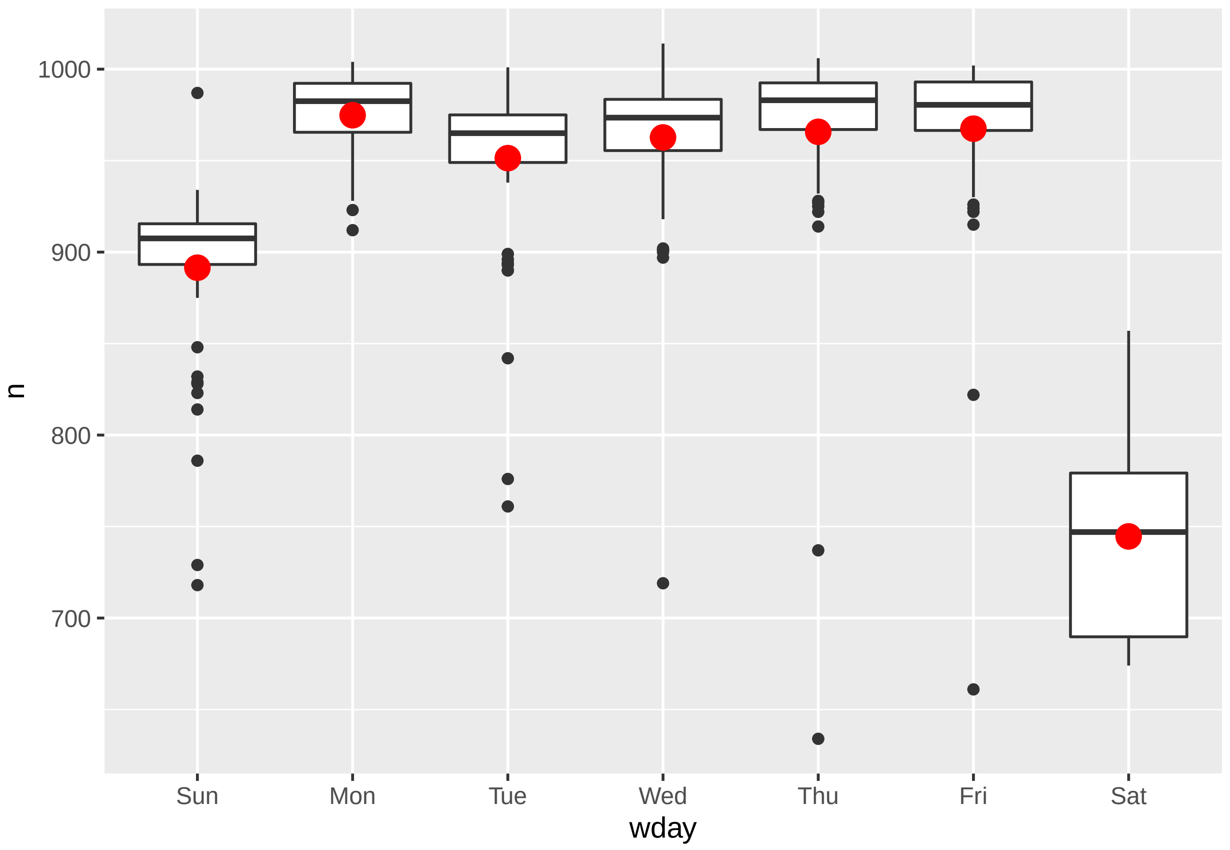
We see that some days of the week have drastically lower flights than the rest. To examine this lets look at the amount of flights that leave per day of the week.

```
daily <- daily %>%
  mutate(wday = wday(date, label = TRUE))
ggplot(daily, aes(wday, n)) +
  geom_boxplot()
```



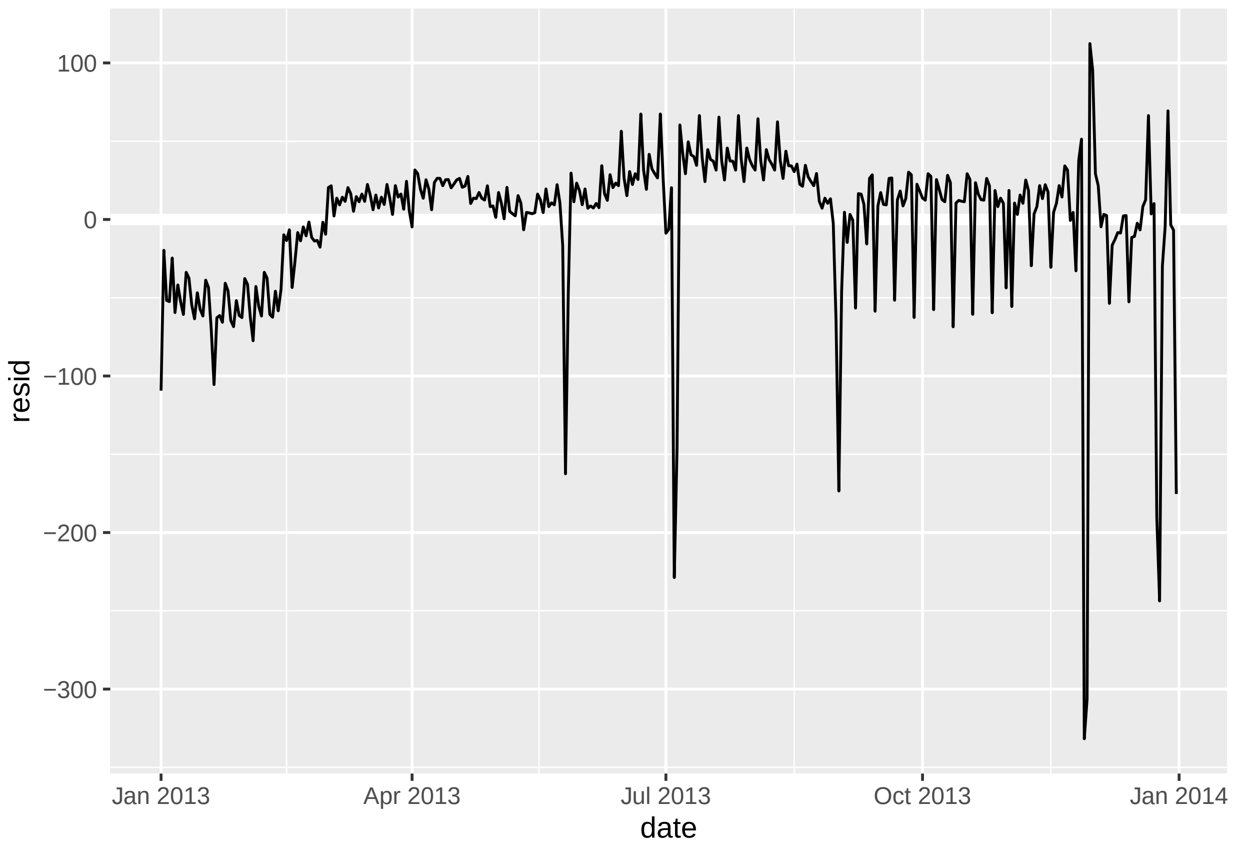
There are slighly fewer flights on Sundays, and much fewer on Saturdays. We want to remove this factor from our model. The first step is to create the model and plot the predictions.

```
mod <- lm(n ~ wday, data = daily)

grid <- daily %>%
  data_grid(wday) %>%
  add_predictions(mod, "n")

ggplot(daily, aes(wday, n)) +
  geom_boxplot() +
  geom_point(data = grid, colour = "red", size = 4)
```



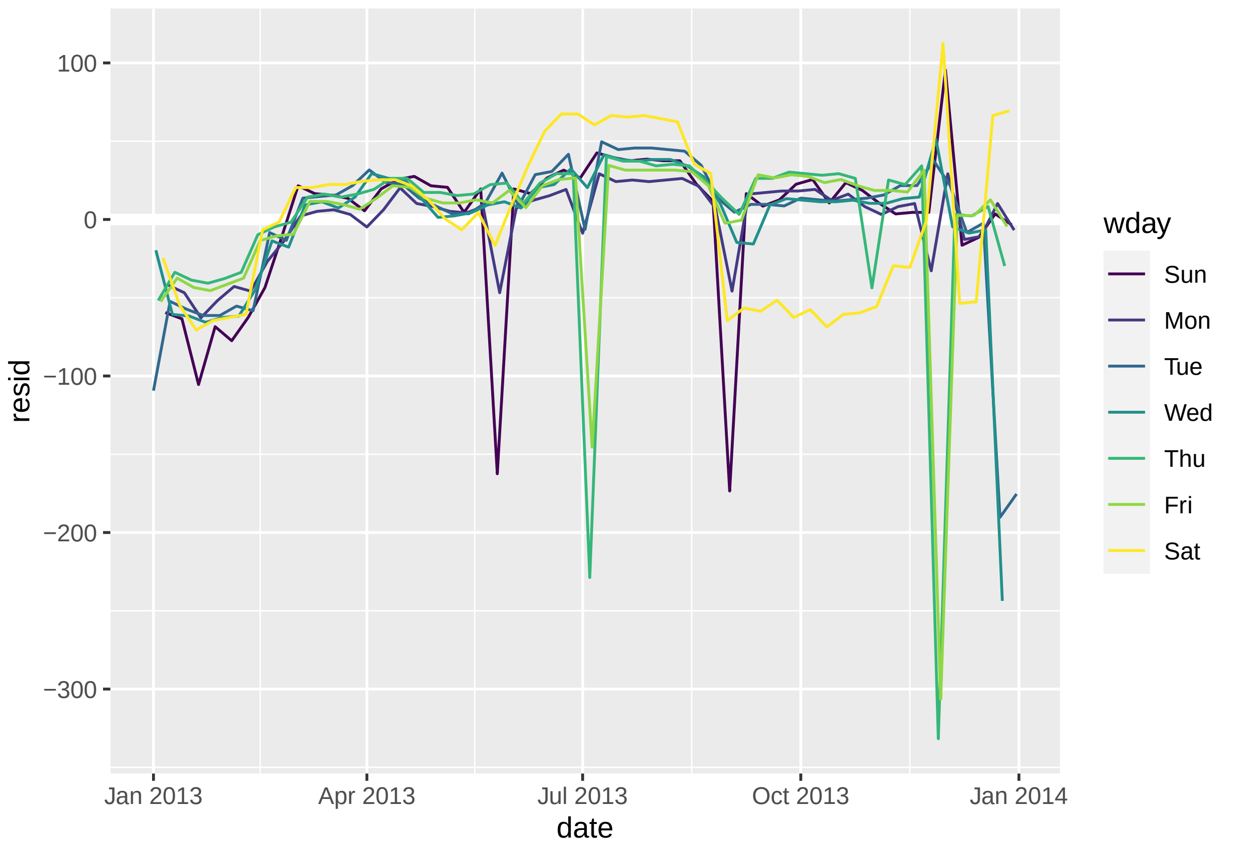
Next, let's look at the residuals.

```
daily <- daily %>%
  add_residuals(mod)
daily %>%
  ggplot(aes(date, resid)) +
  geom_ref_line(h = 0) +
  geom_line()
```



We see that the residuals get larger past June. I'll recreate the graph but draw a different line for each day of the week.

```
ggplot(daily, aes(date, resid, colour = wday)) +
  geom_ref_line(h = 0) +
  geom_line()
```



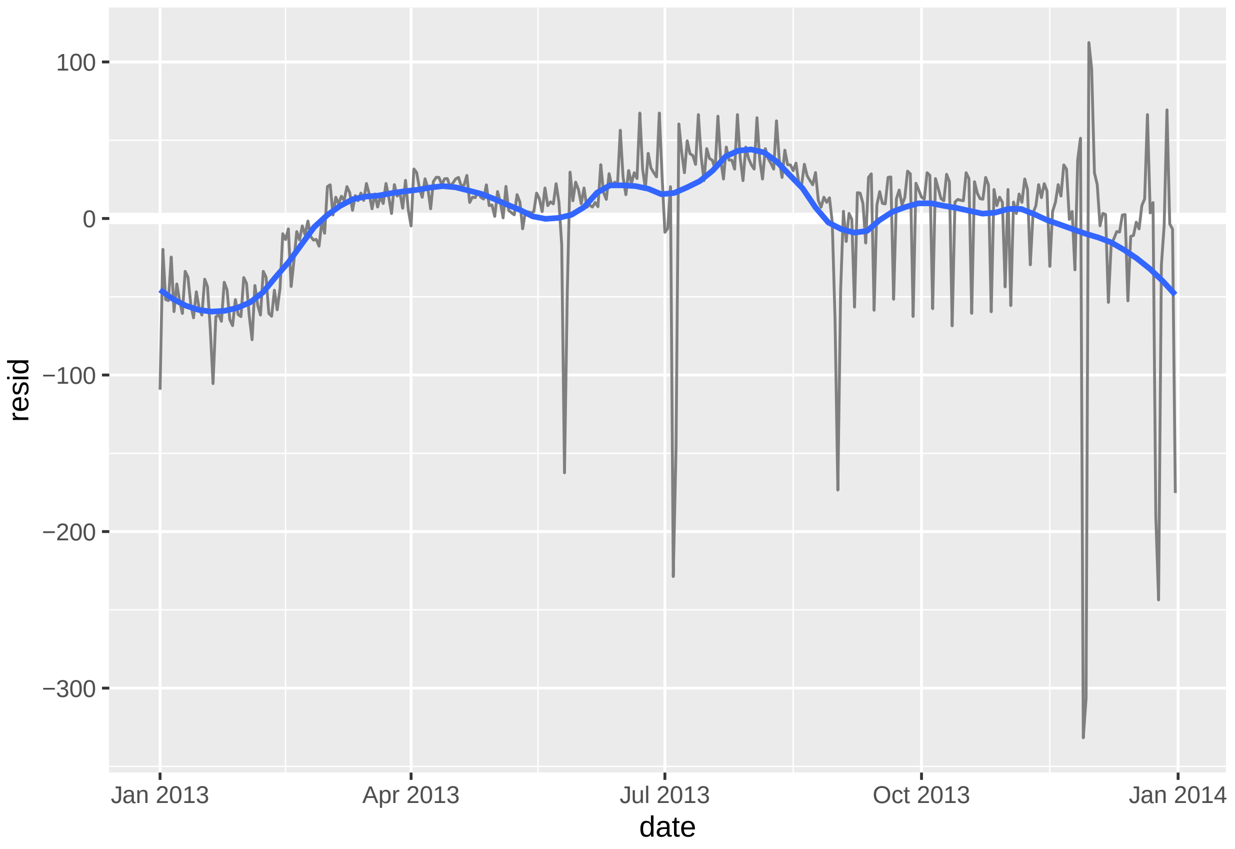
From this plot, we see that in Fanuary and February, we overestimate Saturday flights, in summer we overestimate Saturday flights, and in fall we underestimte Saturday flights. Late in the year, there is high variability of residuals for Saturday flights.

```
daily %>%
  filter(resid < -100)
```

```
## # A tibble: 11 x 4
##    date           n wday  resid
##    <date>     <int> <ord> <dbl>
##  1 2013-01-01   842 Tue   -109.
##  2 2013-01-20   786 Sun   -105.
##  3 2013-05-26   729 Sun   -162.
##  4 2013-07-04   737 Thu   -229.
##  5 2013-07-05   822 Fri   -145.
##  6 2013-09-01   718 Sun   -173.
##  7 2013-11-28   634 Thu   -332.
##  8 2013-11-29   661 Fri   -306.
##  9 2013-12-24   761 Tue   -190.
## 10 2013-12-25   719 Wed   -244.
## 11 2013-12-31   776 Tue   -175.
```

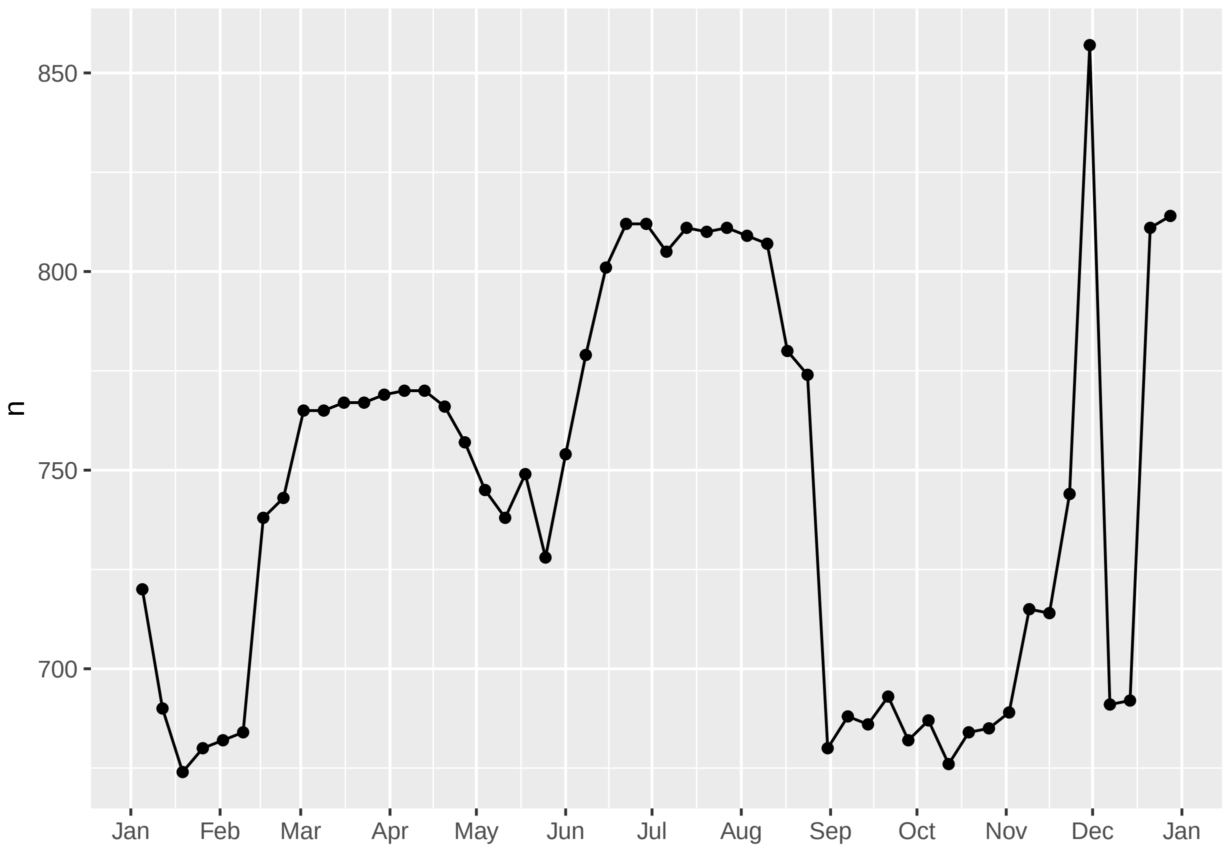
The days with far fewer flights than predicted are typically US holidays like New Years, Fourth of July, Thanksgiving, and Christmas.

```
daily %>%
  ggplot(aes(date, resid)) +
  geom_ref_line(h = 0) +
  geom_line(colour = "grey50") +
  geom_smooth(se = FALSE, span = 0.20)
```



This shows the smoothed overall trend for the year. We see more flights than expected in summer and less than expected in winter. Next, we want to look just at Saturday flights over the span of the year.

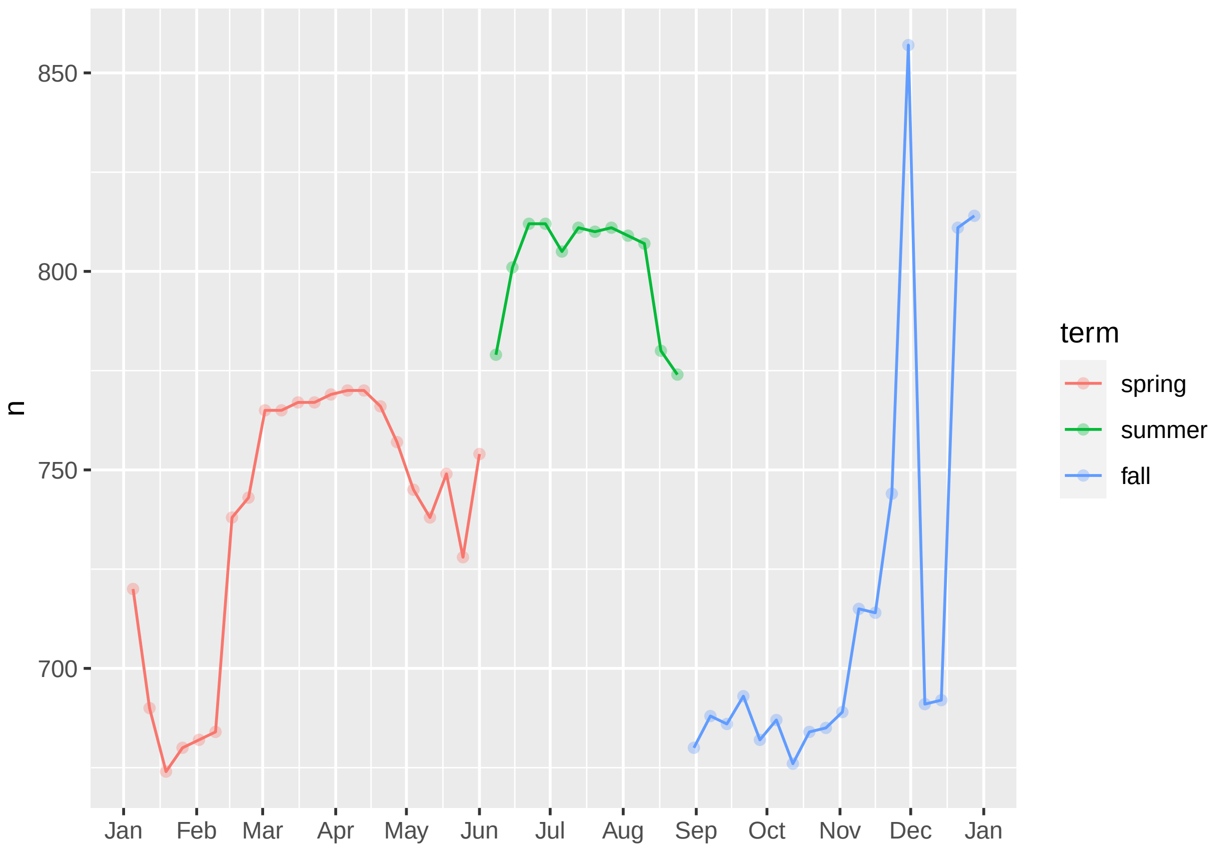
```
daily %>%
  filter(wday == "Sat") %>%
  ggplot(aes(date, n)) +
  geom_point() +
  geom_line() +
  scale_x_date(NULL, date_breaks = "1 month", date_labels = "%b")
```



It looks like summer holiday has more Saturday flights, as days of the week don't matter as much in summer.

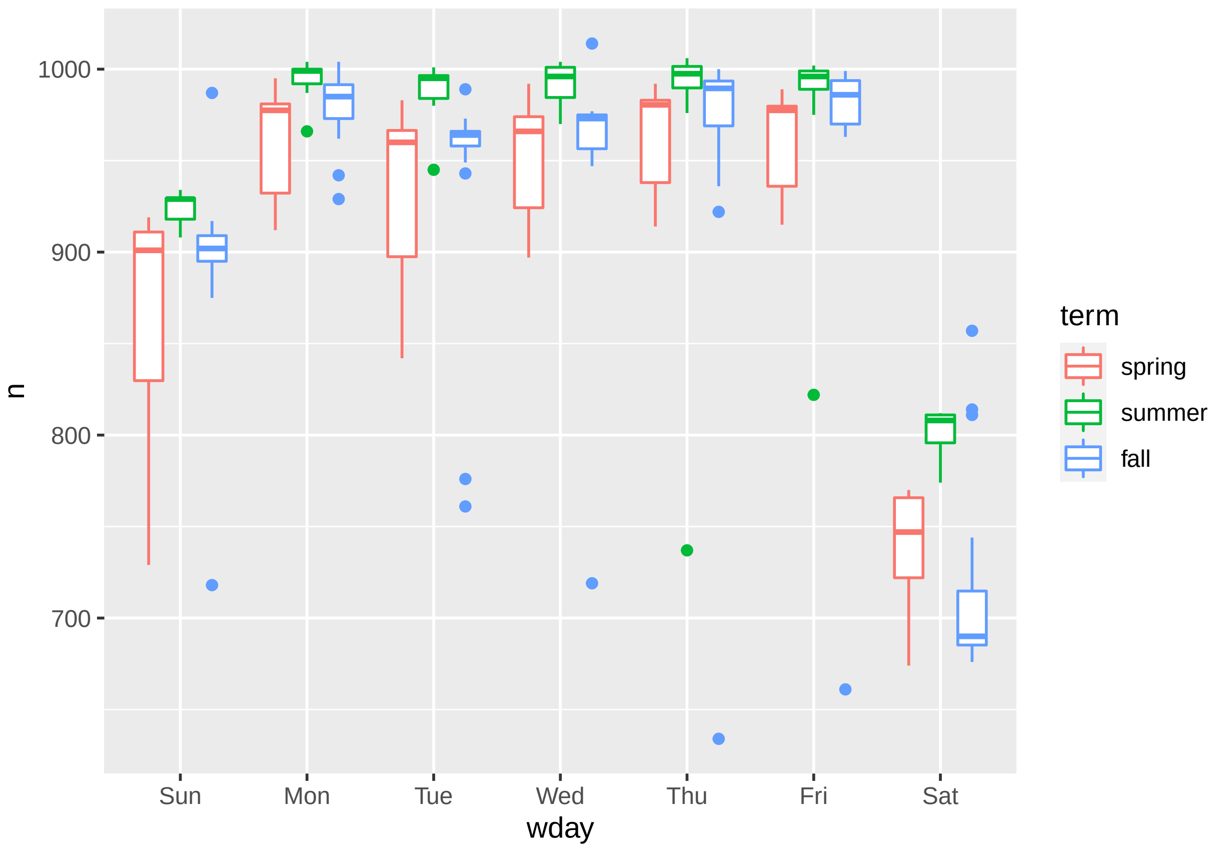
We can split this graph up into three sections based on school sections:

```
term <- function(date) {
  cut(date,
      breaks = ymd(20130101, 20130605, 20130825, 20140101),
      labels = c("spring", "summer", "fall")
  )
}

daily <- daily %>%
  mutate(term = term(date))

daily %>%
  filter(wday == "Sat") %>%
  ggplot(aes(date, n, colour = term)) +
  geom_point(alpha = 1/3) +
  geom_line() +
  scale_x_date(NULL, date_breaks = "1 month", date_labels = "%b")
```



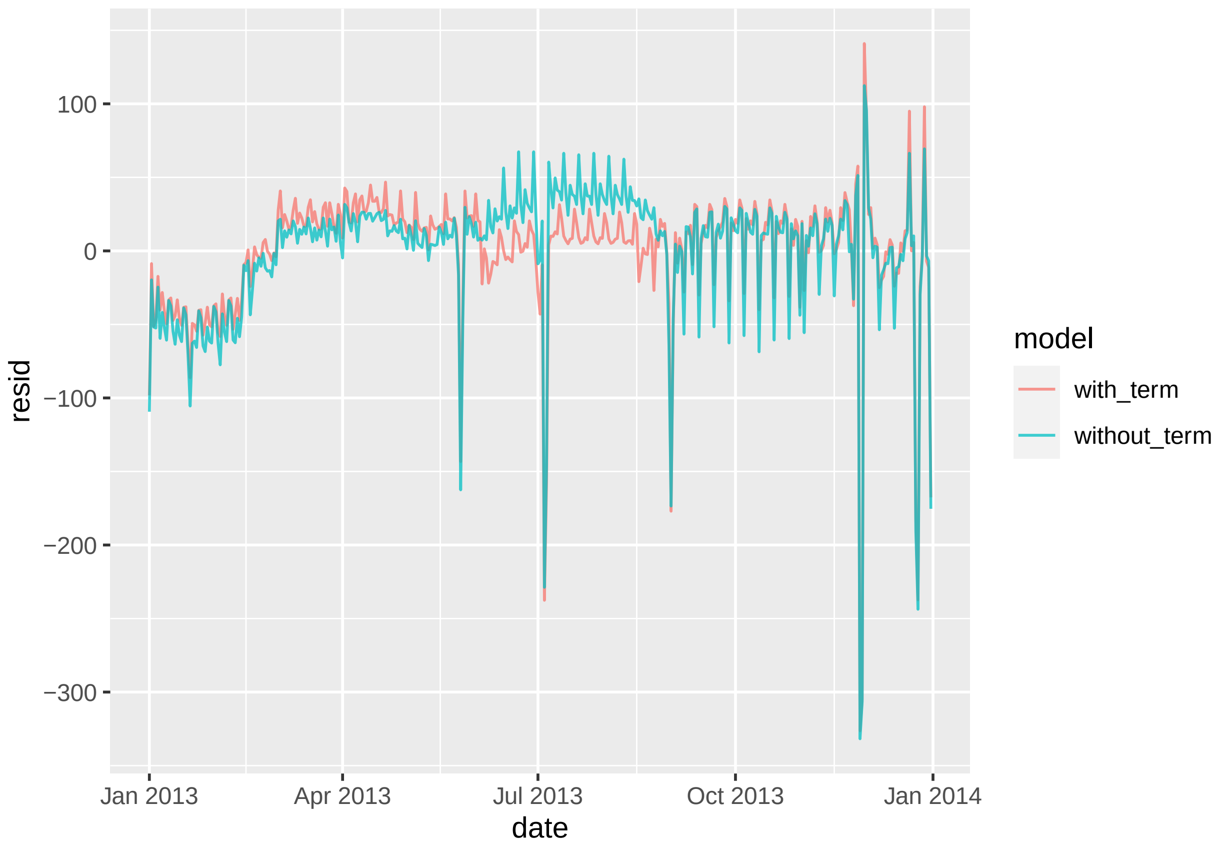
We can do the same for the other days of the week.

```
daily %>%
  ggplot(aes(wday, n, colour = term)) +
  geom_boxplot()
```



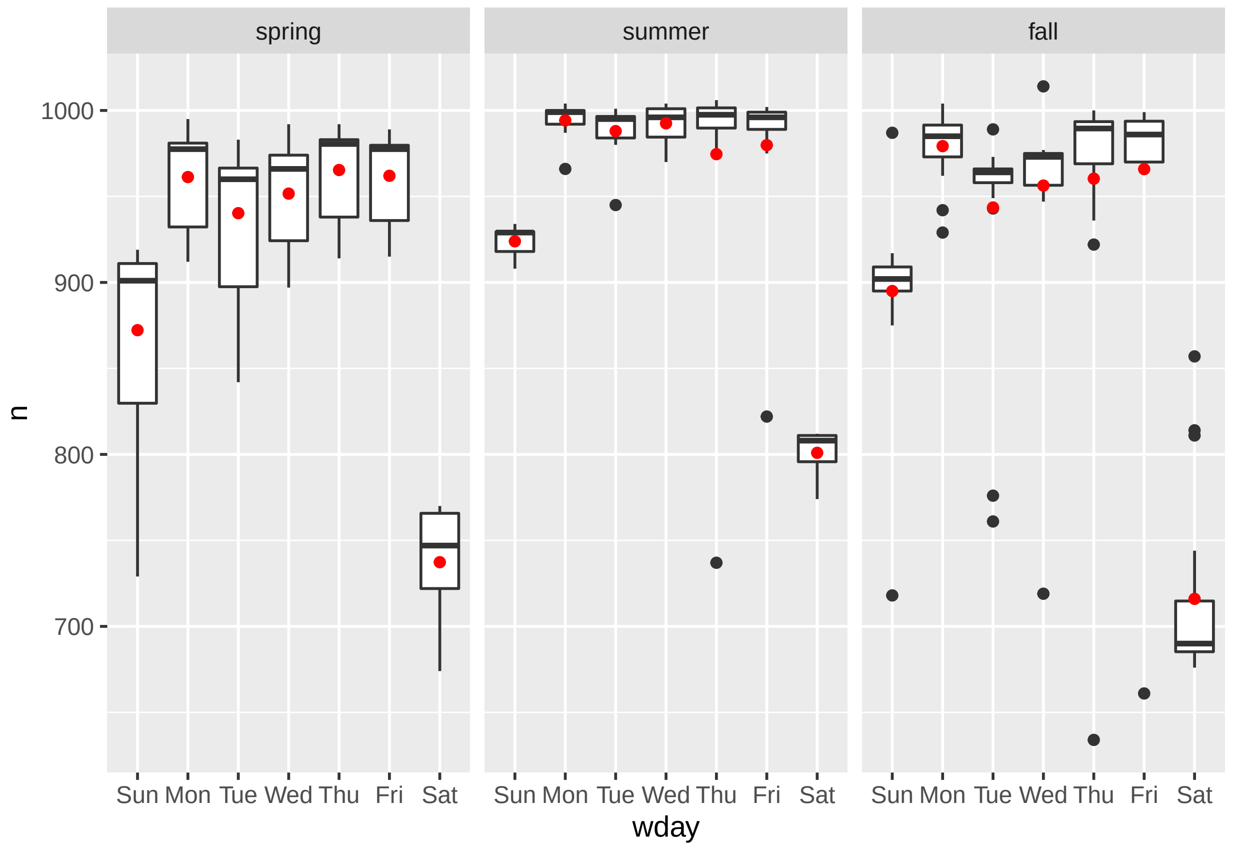
Because there is significant variation across the terms, we want to fit a separate day of the week effect into the model.

```
mod1 <- lm(n ~ wday, data = daily)
mod2 <- lm(n ~ wday * term, data = daily)

daily %>%
  gather_residuals(without_term = mod1, with_term = mod2) %>%
  ggplot(aes(date, resid, colour = model)) +
  geom_line(alpha = 0.75)
```



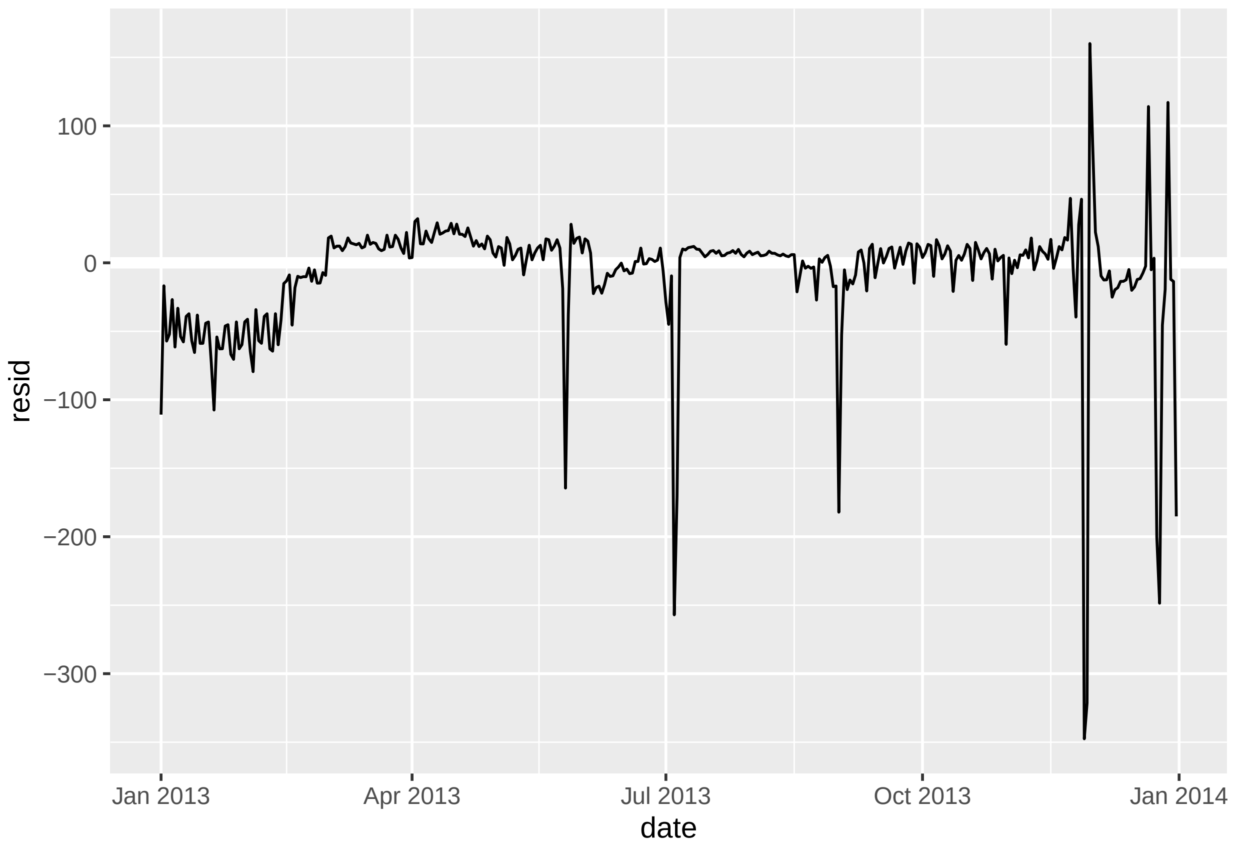
With the term does a lot better than without it, but still not that good. Now lets overlay the updated model predictions on the original data

```
grid <- daily %>%
  data_grid(wday, term) %>%
  add_predictions(mod2, "n")

ggplot(daily, aes(wday, n)) +
  geom_boxplot() +
  geom_point(data = grid, colour = "red") +
  facet_wrap(~ term)
```



We see that the means are quite accurate, but the large outliers are causing problems. To fix this, we'll use the `MASS:rlm()` function.

```r
mod3 <- MASS::rlm(n ~ wday * term, data = daily)

daily %>%
  add_residuals(mod3, "resid") %>%
  ggplot(aes(date, resid)) +
  geom_hline(yintercept = 0, size = 2, colour = "white") +
  geom_line()
```

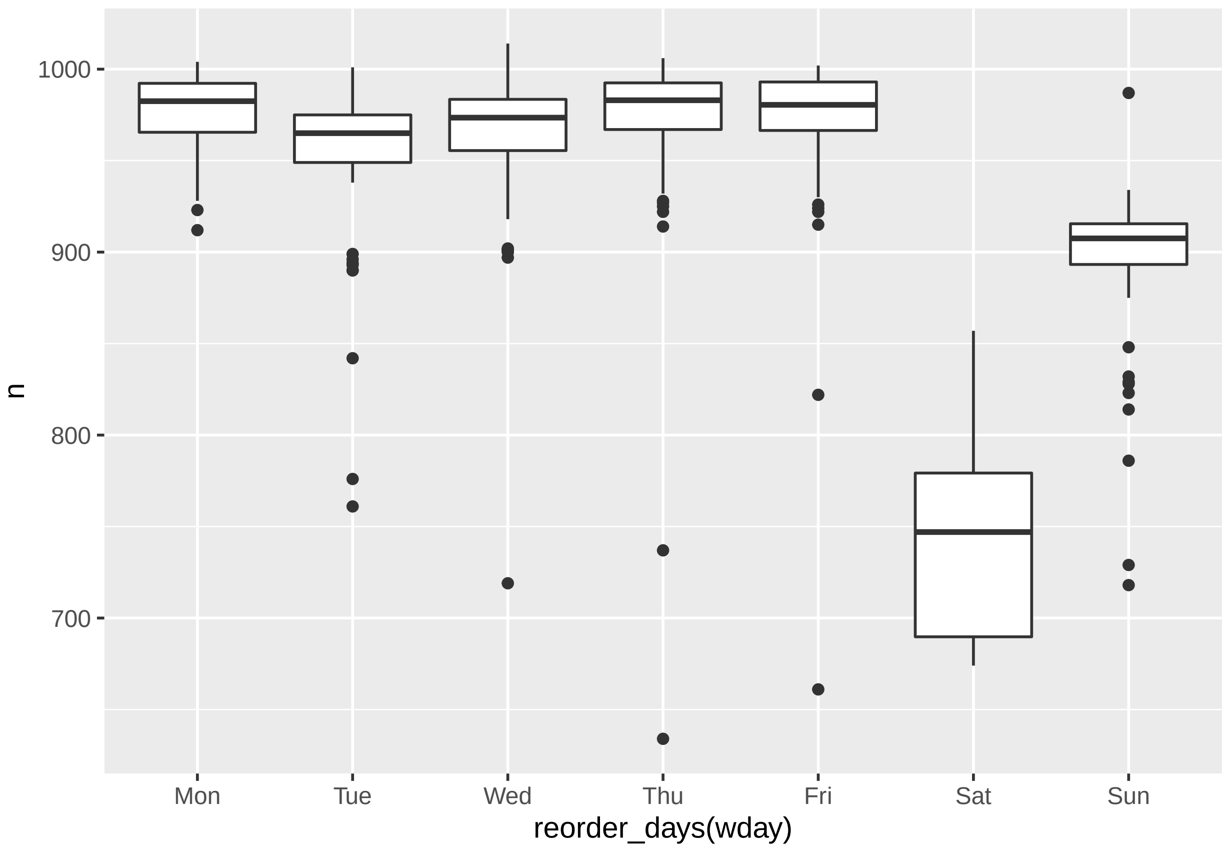


Now, the residuals are much more centered around 0, which means the model is largely accurate. It also clearly shows the major outliers, which are the main holidays like Christmas, New Years, and the Fourth of July.

Exercise 4 (Website: 24.3.5 Ex. 8)

It's a little frustrating that Sunday and Saturday are on separate ends of the plot. Write a function to organize the levels of the factor so that the week starts on Monday.

```
reorder_days <- function(x) {
  fct_relevel(x, levels(x)[-1])
}

daily <- daily %>%
  mutate(wday = wday(date, label = TRUE))
ggplot(daily) +
  geom_boxplot(mapping = aes(reorder_days(wday), n))
```



Optional Exercise 1 (Website: 24.3.5 Ex. 3) — NOT REQUIRED

Create a new variable that splits the `wday` variable into terms, but only for Saturdays; i.e., it should have `Thurs`, `Fri`, etc. for the weekdays and for Sundays, but `Sat-summer`, `Sat-spring`, `Sat-fall`. How does this model compare with the model that contains every combination of `wday` and `term`?

Optional Exercise 2 (Website: 24.3.5 Ex. 5) — NOT REQUIRED

What happens if you fit a day of week effect that varies by month (i.e. `n ~ wday * month`)? Why is this not very helpful?

Optional Exercise 3 (Website: 24.3.5 Ex. 7) — NOT REQUIRED

We hypothesized that people leaving on Sundays are more likely to be business travelers who need to be somewhere on Monday. Explore that hypothesis by seeing how it breaks down based on distance and time: If the hypothesis is true, you'd expect to see more Sunday evening flights to places that are far away.