Linear regression

Brooke Anderson

February 22, 2016

```
knitr::opts_knit$set(root.dir = "..") # Reset root directory for analysis
library(lubridate) # To help handle dates
library(dplyr) # Data wrangling
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2) # Plotting
library(gridExtra) # To arrange multiple ggplot objects in one graph
```

Read in and clean up the data:

```
train <- read.csv("data/train.csv", as.is = TRUE) # `as.is` so `datetime` comes in as
                                                  # character, not factor
test <- read.csv("data/test.csv", as.is = TRUE)

train <- mutate(train,
                datetime = ymd_hms(datetime),
                year = factor(year(datetime)),
                hour = factor(hour(datetime)),
                month = month(datetime),
                yday = yday(datetime),
                season = factor(season, levels = c(1, 2, 3, 4),
                                labels = c("Spring", "Summer", "Fall", "Winter")),
                workingday = factor(workingday, levels = c(0, 1),
                                    labels = c("Holiday / weekend",
                                               "Working day")))
test  <- mutate(test,
                datetime = ymd_hms(datetime),
                year = factor(year(datetime)),
                hour = factor(hour(datetime)),
                month = month(datetime),
                yday = yday(datetime),
```

```
                season = factor(season, levels = c(1, 2, 3, 4),
                                labels = c("Spring", "Summer", "Fall", "Winter")),
                workingday = factor(workingday, levels = c(0, 1),
                                    labels = c("Holiday / weekend",
                                               "Working day")))
```

Intercept-only model

Fit a linear regression with only an intercept, to get an idea of the performance of a very simple model:

```
mod_0 <- glm(count ~ 1, data = train)
```

Determine the RMSLE in the training data. From Kaggle, the equation is:

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(p_i+1) - log(a_i+1))^2}$$

```
rmsle <- function(train_preds, actual_preds){
  log_p_1 <- log(train_preds + 1)
  log_a_1 <- log(actual_preds + 1)
  sle <- (log_p_1 - log_a_1)^2
  rmsle <- sqrt(mean(sle))
  return(rmsle)
}
```

```
train_preds <- predict(mod_0)
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 1.569198
```

I'll also write out predictions to submit to Kaggle:

```
write_test_preds <- function(test_preds, mod_name){
  out_file <- data.frame(datetime = as.character(test$datetime),
                         count = test_preds)
  out_name <- paste0("test_predictions/", mod_name, ".csv")
  write.csv(out_file, file = out_name, row.names = FALSE)
}
```

```
test_preds <- predict(mod_0, newdata = test)
write_test_preds(test_preds, mod_name = "intercept_only")
```

On Kaggle, this model resulted in a RMSLE of 1.58456.

Linear regression

Fit a linear regression based on hour of the day, working day, and season (all modeled as factors):

```
mod_1 <- glm(count ~ hour*workingday*season, # `*` means fit interaction + main effects
             data = train)
```

```
train_preds <- predict(mod_1)
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.5079531
```

The RMSLE is now a third of it's value with the intercept-only model. I also tried it on Kaggle:

```
test_preds <- predict(mod_1, newdata = test)
write_test_preds(test_preds, mod_name = "hour_workday_season")
```

On Kaggle, this had an RMSLE of 0.57524.

GLM adding weather and temperature

Next, I kept the interaction between `hour` and `workingday` and added in `weather` and `temp` (naively not adjusting for the fact that I probably have some collinearity). At first, I was having some problems with this because I was getting negative predictions, which couldn't be handled by the `rmsle` function, so I fit the model using `family = quasipoisson`. This will (among other things) prevent any predictions from being below zero.

```
mod_2 <- glm(count ~ hour*workingday*season + factor(weather) + temp,
             data = train, family = quasipoisson)
```

```
train_preds <- predict(mod_2, type = "response")
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.442596
```

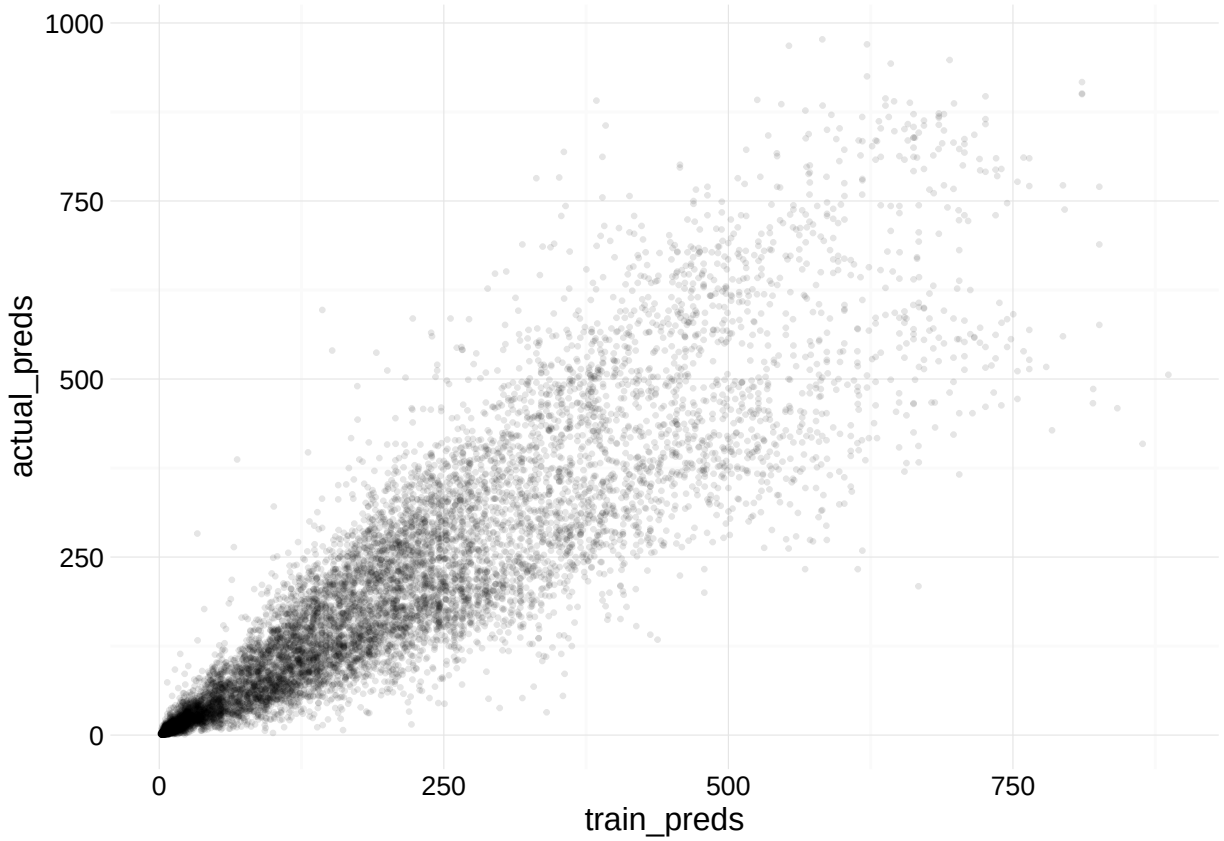
The RMSLE continues to improve for the training data. I also submitted to Kaggle:

```
test_preds <- predict(mod_2, newdata = test, type = "response")
write_test_preds(test_preds, mod_name = "hour_workday_season_weather_temp")
```

The RMSLE on the Leaderboard testing data was 0.48868.

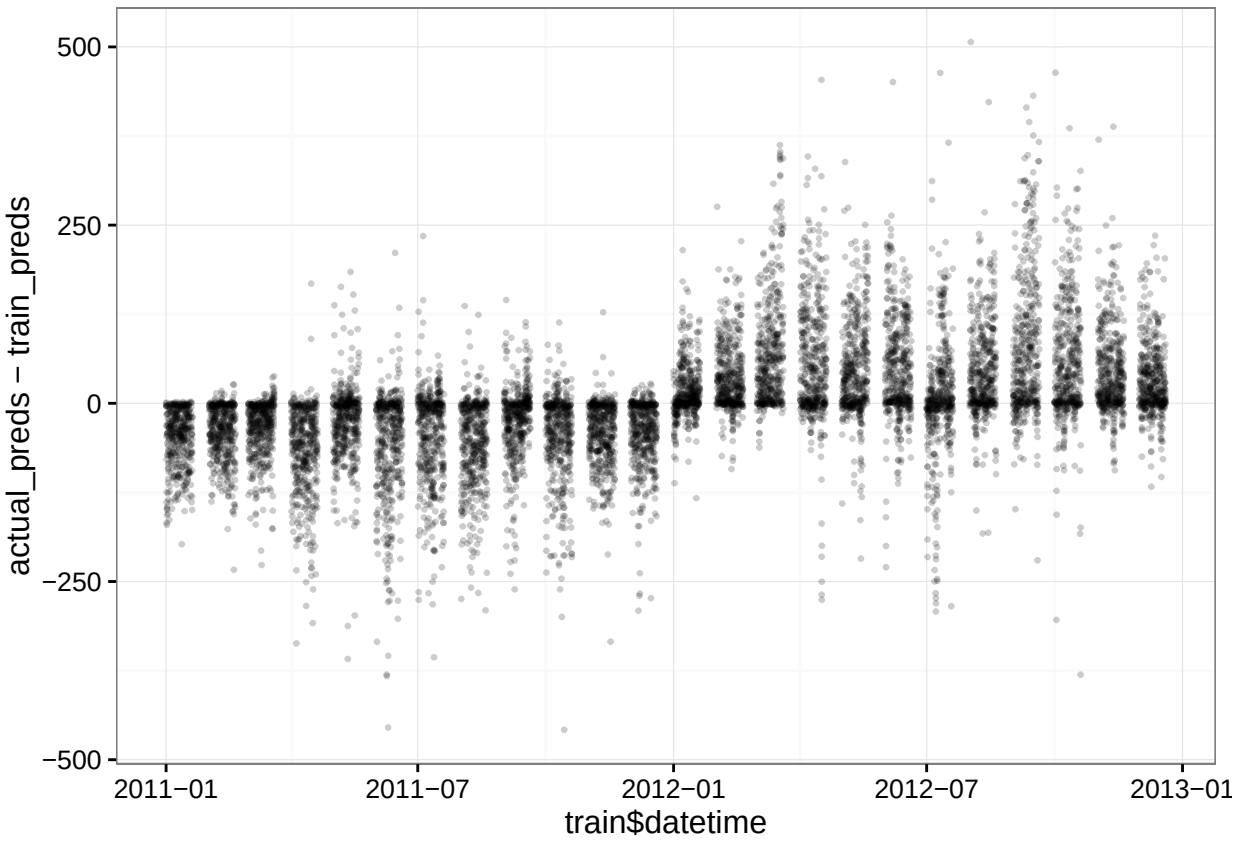
I looked at predictions versus actual values, and there seem to be two groupings in this graph, one where the model overpredicts and one where it underpredicts:

```
qplot(train_preds, actual_preds, alpha = I(0.1), size = I(0.5)) +
  theme_minimal()
```



I also looked at model residuals by time. It looks like the model needs to include the year of the observation:

```
qplot(x = train$datetime, y = actual_preds - train_preds,
      alpha = I(0.2), size = I(0.5)) +
  theme_bw()
```



Adding in year as a factor

Based on the residuals by time for the previous model, `year` may be a useful predictor:

```
mod_3 <- glm(count ~ year*hour*workingday*season + factor(weather) + temp,
             data = train, family = quasipoisson)
```

```
train_preds <- predict(mod_3, type = "response")
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.3491855
```

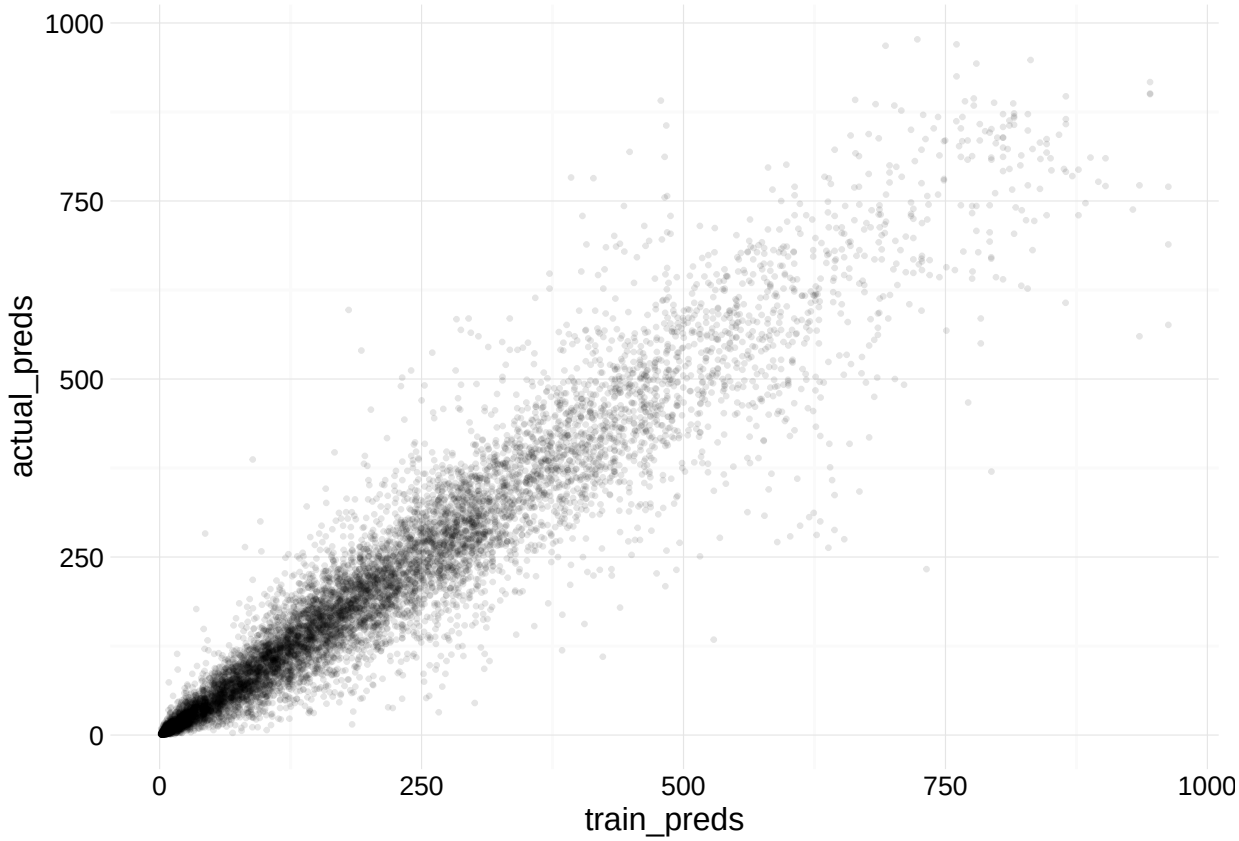
The RMSLE improves quite a bit for the training data with adding the year. I also submitted to Kaggle:

```
test_preds <- predict(mod_3, newdata = test, type = "response")
write_test_preds(test_preds, mod_name = "year_hour_workday_season_weather_temp")
```

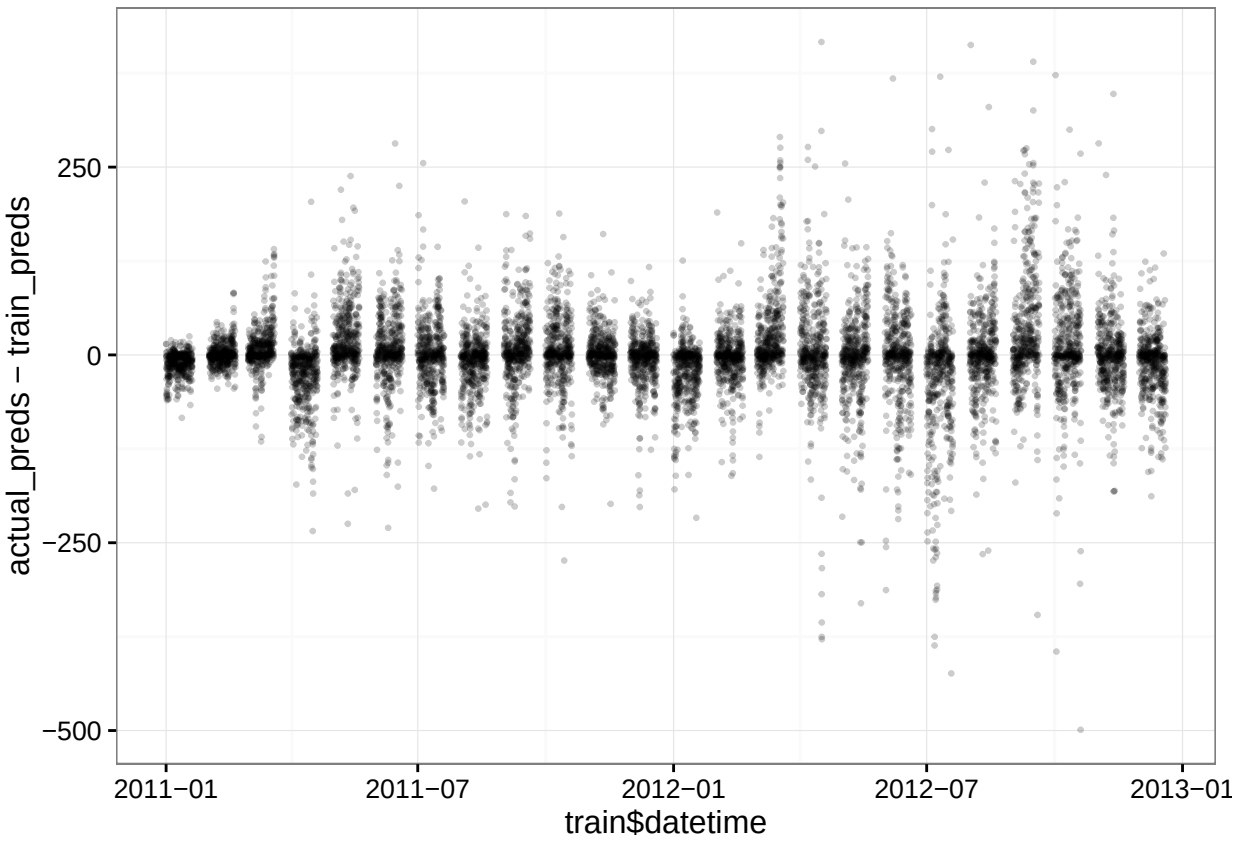
This submission scored 0.43228.

The residual plots look much better for this model, too:

```
qplot(train_preds, actual_preds, alpha = I(0.1), size = I(0.5)) +
  theme_minimal()
```

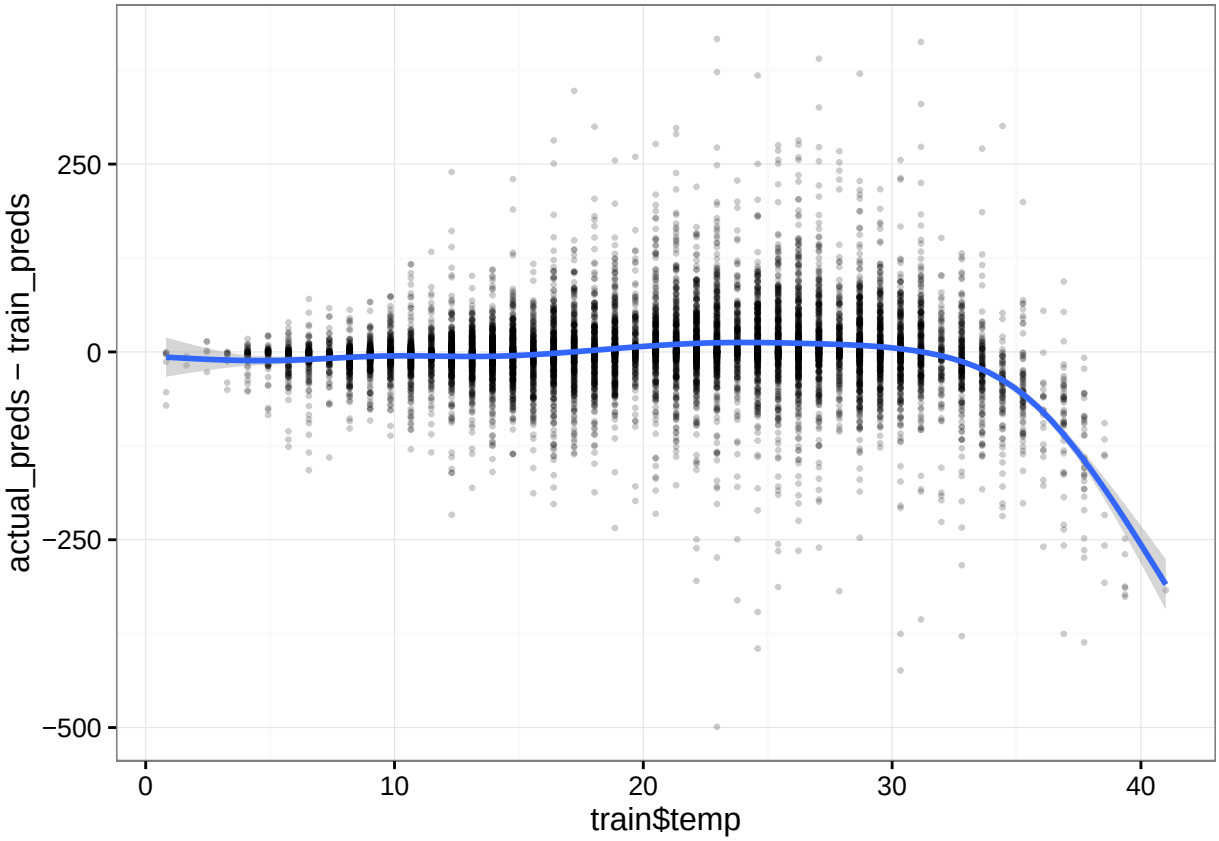



```
qplot(x = train$datetime, y = actual_preds - train_preds,
      alpha = I(0.2), size = I(0.5)) +
  theme_bw()
```



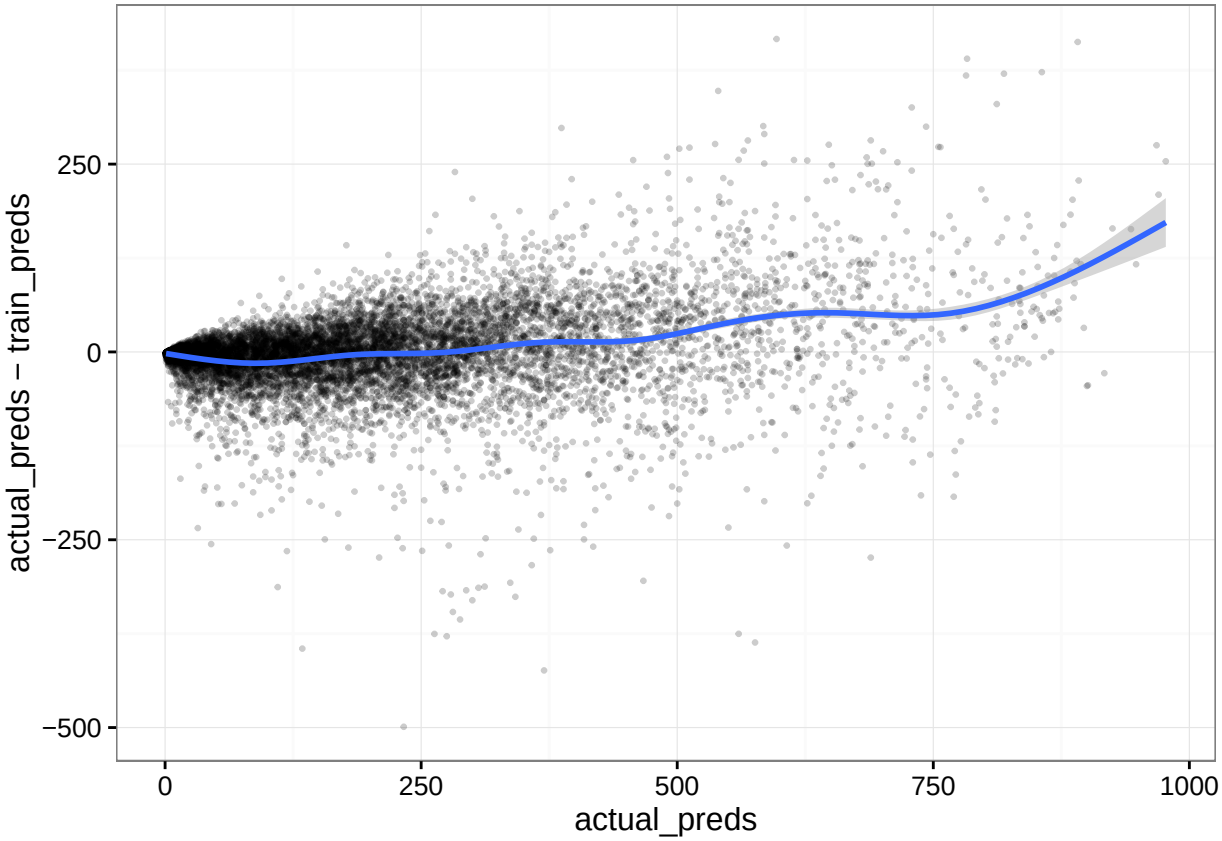
Looking a bit more at the residuals, it may help to include a non-linear function of temperature as a predictor:

```
qplot(x = train$temp, y = actual_preds - train_preds,
      alpha = I(0.2), size = I(0.5)) +
  geom_smooth() +
  theme_bw()
```



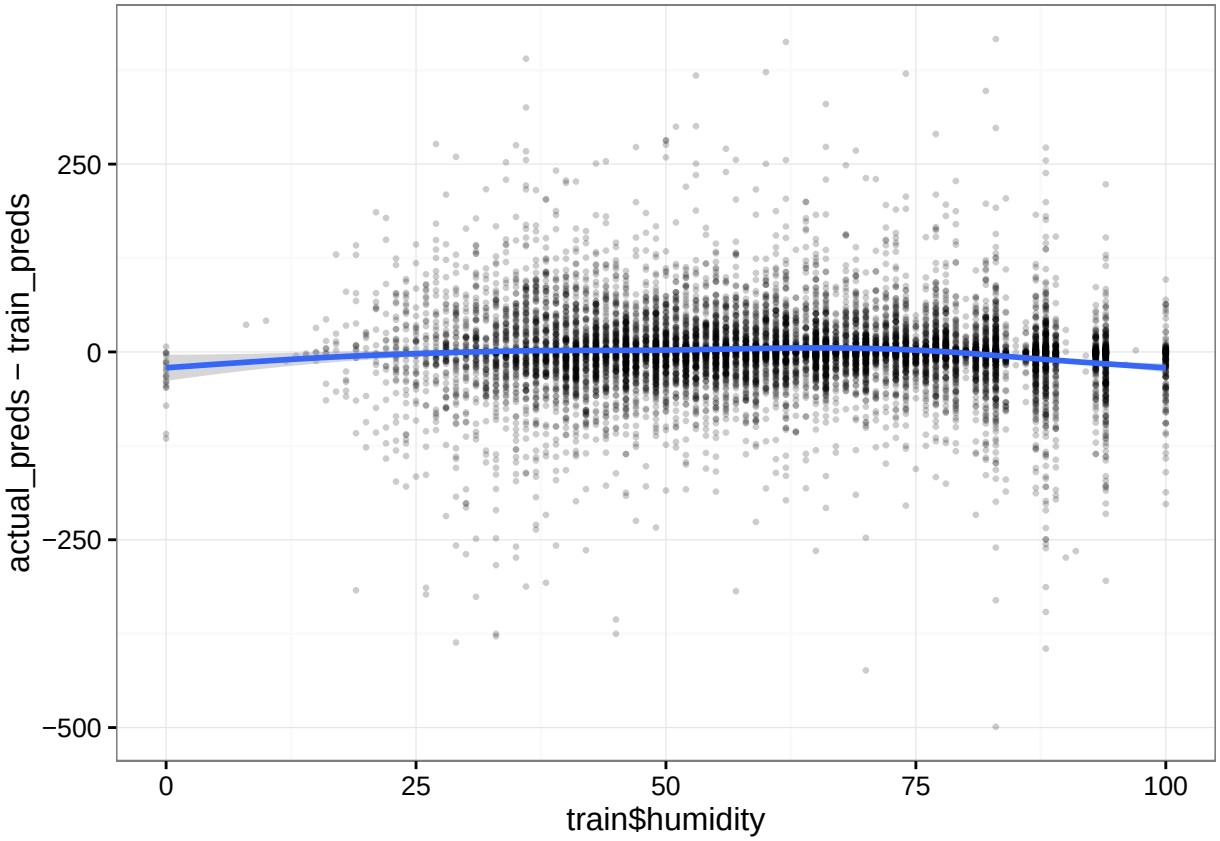
It looks like the model consistently tends to underpredict when actual counts are very high:

```
qplot(x = actual_preds, y = actual_preds - train_preds,
      alpha = I(0.2), size = I(0.5)) +
  geom_smooth() +
  theme_bw()
```

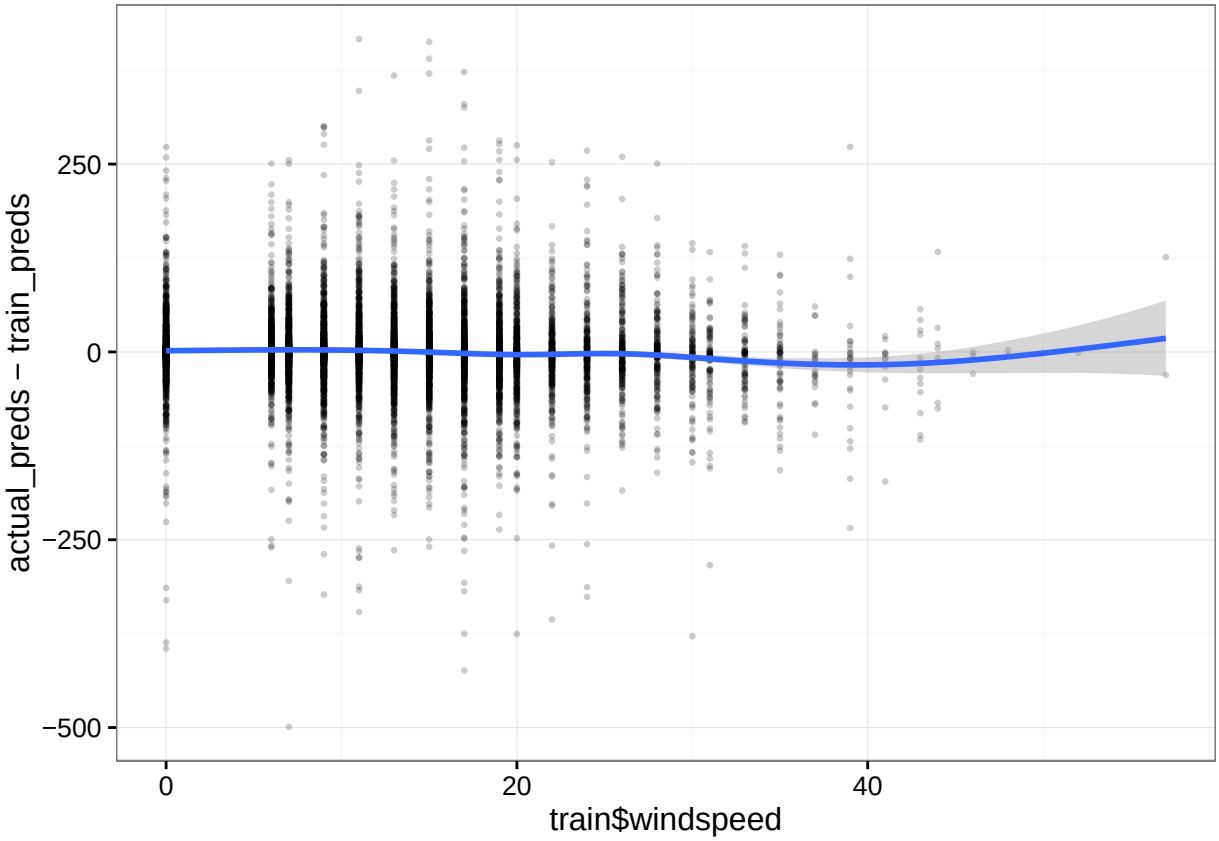


I'm not sure that humidity or windspeed will help much:

```
qplot(x = train$humidity, y = actual_preds - train_preds,
      alpha = I(0.2), size = I(0.5)) +
  geom_smooth() +
  theme_bw()
```




```
qplot(x = train$windspeed, y = actual_preds - train_preds,
      alpha = I(0.2), size = I(0.5)) +
  geom_smooth() +
  theme_bw()
```



Non-linear function for temperature

Based on the residuals by temperature for the previous model, it may be possible to improve the model by using a non-linear function of `temp`. I did this using a natural cubic spline, using the `ns` function from the `splines` package. I places the knots at the higher temperatures where there seemed to be a pattern with residuals before:

```
library(splines)
mod_4 <- glm(count ~ year*hour*workingday*season + factor(weather) +
               ns(temp, knots = c(30, 35)),
             data = train, family = quasipoisson)
```

```
train_preds <- predict(mod_4, type = "response")
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.3399749
```

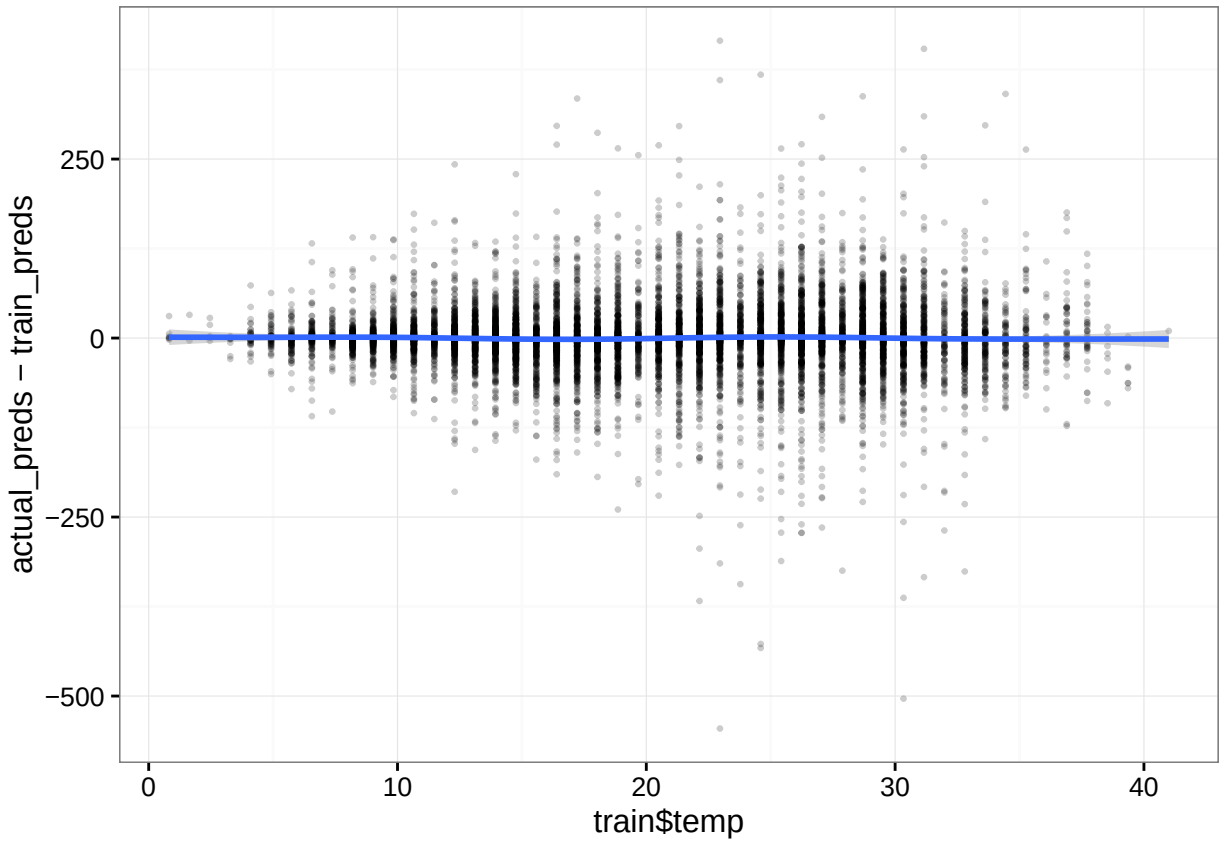
The RMSLE improves some for this model. I also submitted to Kaggle:

```
test_preds <- predict(mod_4, newdata = test, type = "response")
write_test_preds(test_preds, mod_name = "year_hour_workday_season_weather_ns_temp")
```

This submission scored 0.41048.

This helped with the pattern in the residuals and temperature:

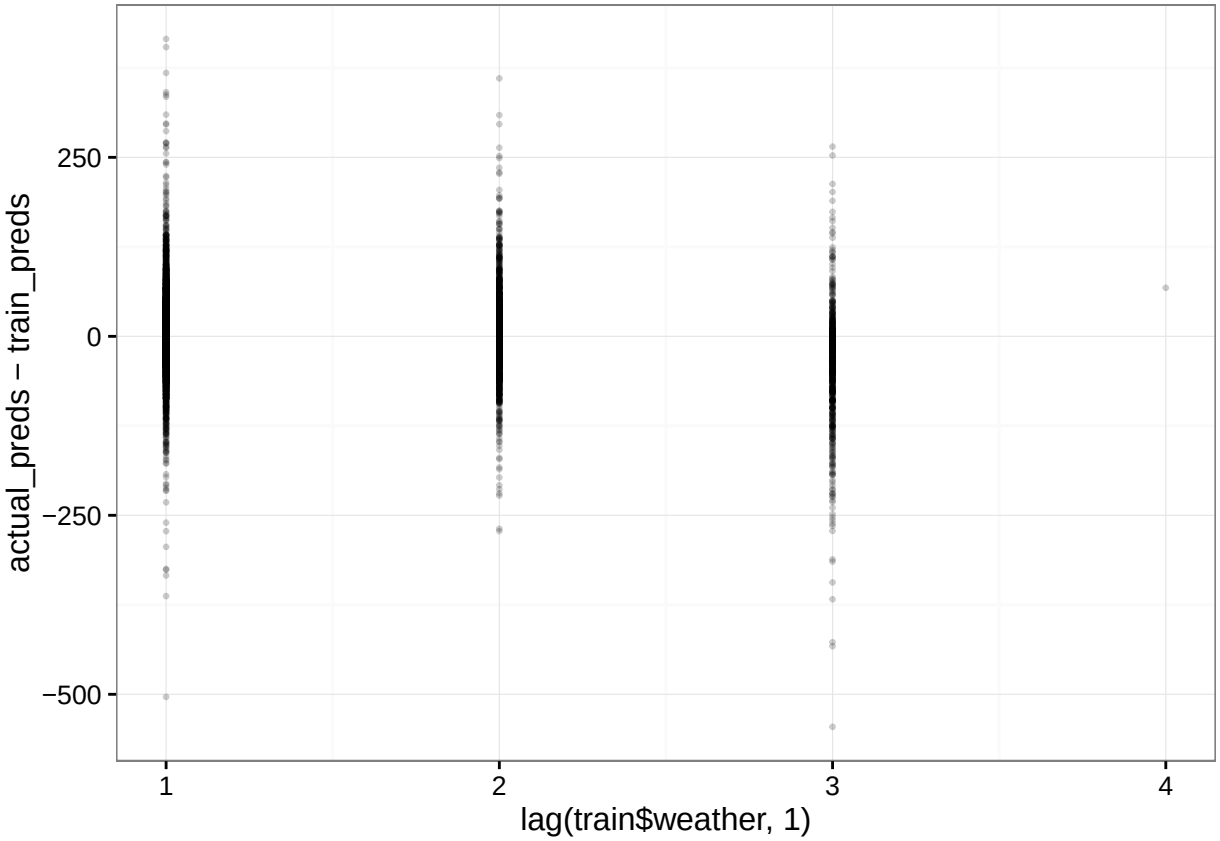
```
qplot(x = train$temp, y = actual_preds - train_preds,
      alpha = I(0.2), size = I(0.5)) +
  geom_smooth() +
  theme_bw()
```



It looks like there might be a relationship with weather lagged by one day, particularly for the "3" category of weather:

```
qplot(x = lag(train$weather, 1), y = actual_preds - train_preds,
      alpha = I(0.2), size = I(0.5)) +
  theme_bw()
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



Adding weather at lag 1

Based on the residuals by temperature for the previous model, it may be possible to improve the model by using a non-linear function of `temp`. I did this using a natural cubic spline, using the `ns` function from the `splines` package. I places the knots at the higher temperatures where there seemed to be a pattern with residuals before:

```
library(splines)
mod_5 <- glm(count ~ year*hour*workingday*season +
               factor(weather)*factor(lag(weather, 1)) +
               ns(temp, knots = c(30, 35)),
             data = train, family = quasipoisson)
```

```
train_preds <- c(predict(mod_4, type = "response")[1], # Need to use last model to
                 predict(mod_5, type = "response"))    # predict first observation
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.3266834
```

The RMSLE improves some for this model. I also submitted to Kaggle:

```
test_preds <- predict(mod_5, newdata = test, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

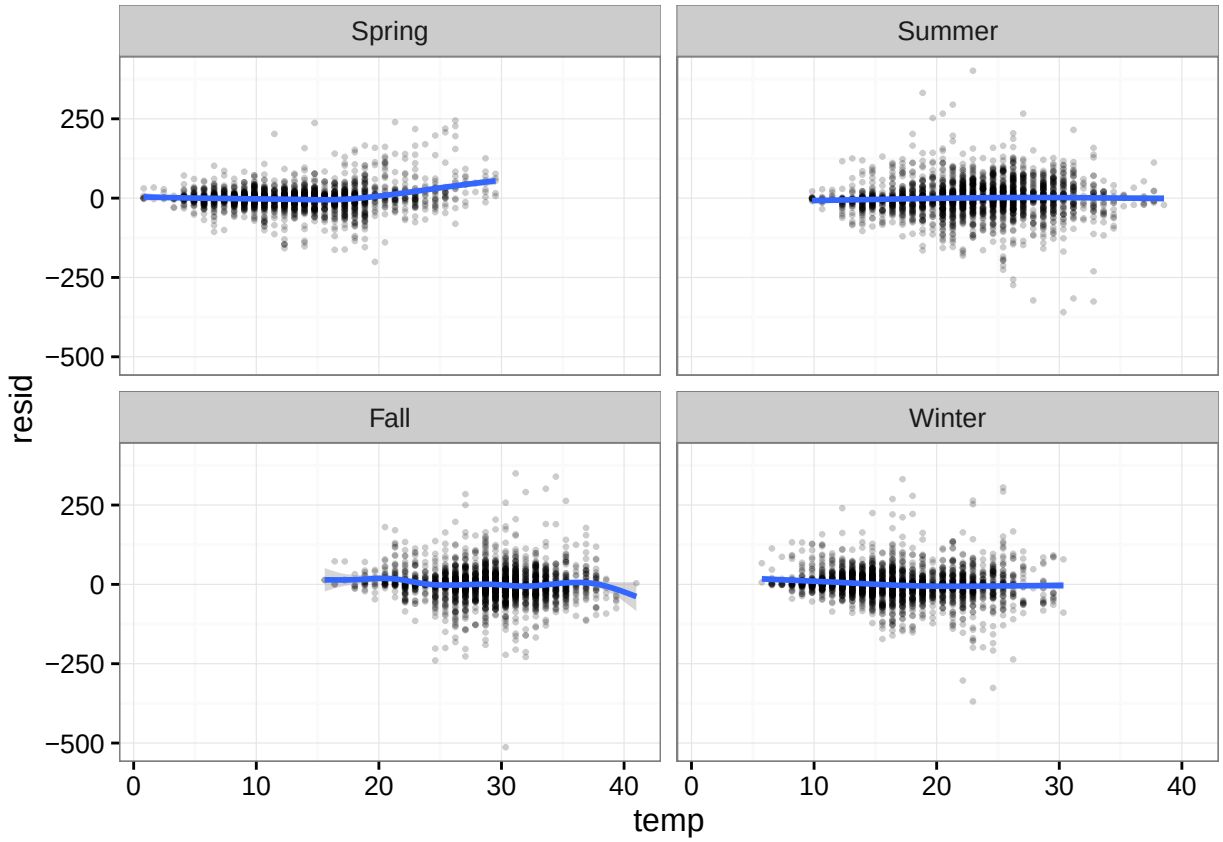
```
test_preds <- c(predict(mod_4, newdata = test, type = "response")[1],
                test_preds[2:length(test_preds)])
write_test_preds(test_preds, mod_name = "year_hour_workday_season_lag_weather_ns_temp")
```

This submission scored 0.39710.

It looks like there are still some patterns between weather and residuals for the winter:

```
to_plot <- data.frame(temp = train$temp,
                      resid = actual_preds - train_preds,
                      season = train$season)

ggplot(to_plot, aes(x = temp, y = resid)) +
  geom_point(alpha = 0.2, size = 0.5) +
  geom_smooth() +
  facet_wrap(~ season, ncol = 2) +
  theme_bw()
```



Adding extra temperature spline for winter

Based on the residuals by temperature for the previous model, it may be possible to improve the model by using a non-linear function of `temp`. I did this using a natural cubic spline, using the `ns` function from the

splines package. I places the knots at the higher temperatures where there seemed to be a pattern with residuals before:

```
mod_6 <- glm(count ~ year*hour*workingday*season +
               factor(weather) + factor(lag(weather, 1)) +
               ns(temp, knots = c(30, 35)) +
               I(season == "Winter"):ns(temp, knots = c(20, 25)),
             data = train, family = quasipoisson)
```

```
train_preds <- c(predict(mod_4, type = "response")[1], # Need to use last model to
                 predict(mod_6, type = "response"))    # predict first observation
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.326231
```

This improves RMSLE, but just a tiny bit. I also submitted to Kaggle:

```
test_preds <- predict(mod_6, newdata = test, type = "response")
```

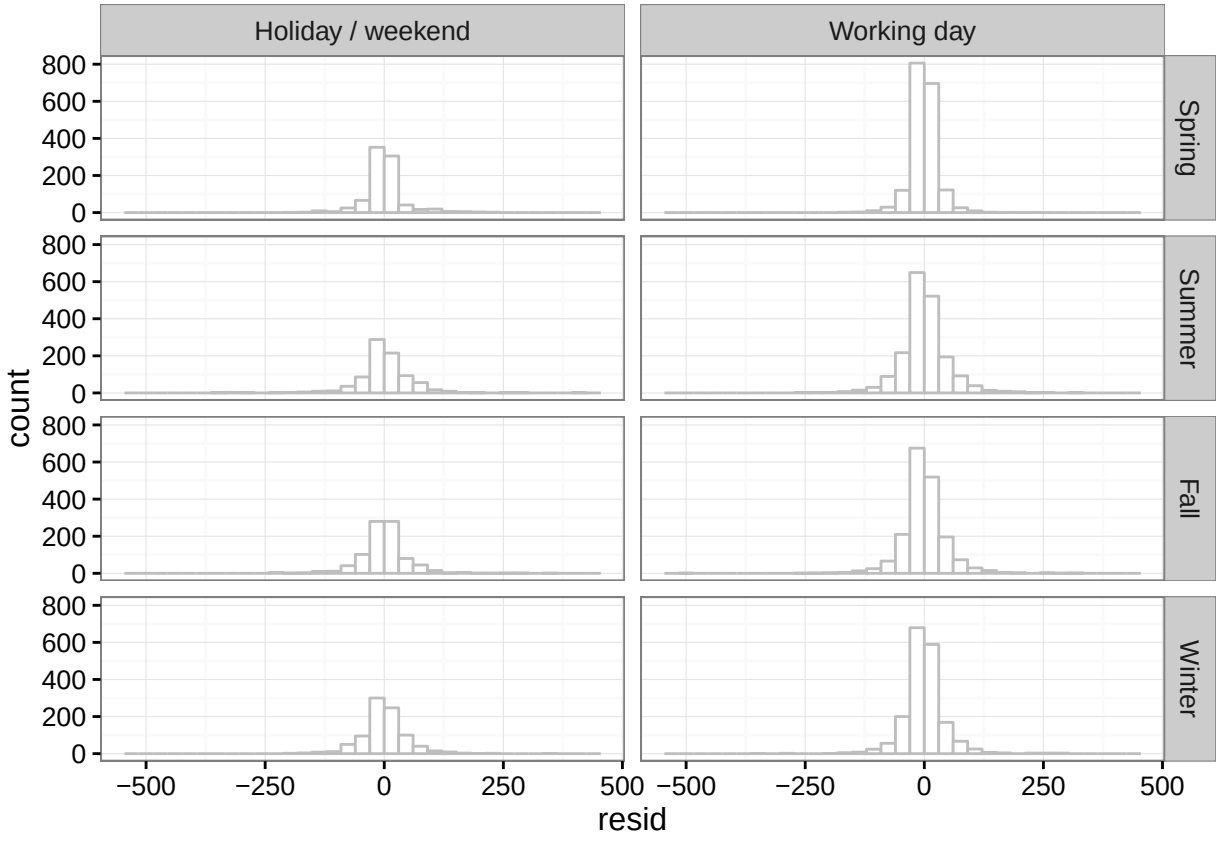
```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
test_preds <- c(predict(mod_4, newdata = test, type = "response")[1],
                test_preds[2:length(test_preds)])
write_test_preds(test_preds, mod_name = "year_hour_workday_season_lag_weather_ns_temp_winter")
```

(It looks like I might be pushing things to try to fit this model.) This submission scored 0.39943, a bit worse than the last model.

```
check_resids <- mutate(train,
                       resid = actual_preds - train_preds) %>%
  arrange(resid)

ggplot(check_resids, aes(x = resid)) +
  geom_histogram(color = "gray", fill = "white") +
  facet_grid(season ~ workingday) +
  theme_bw()
```



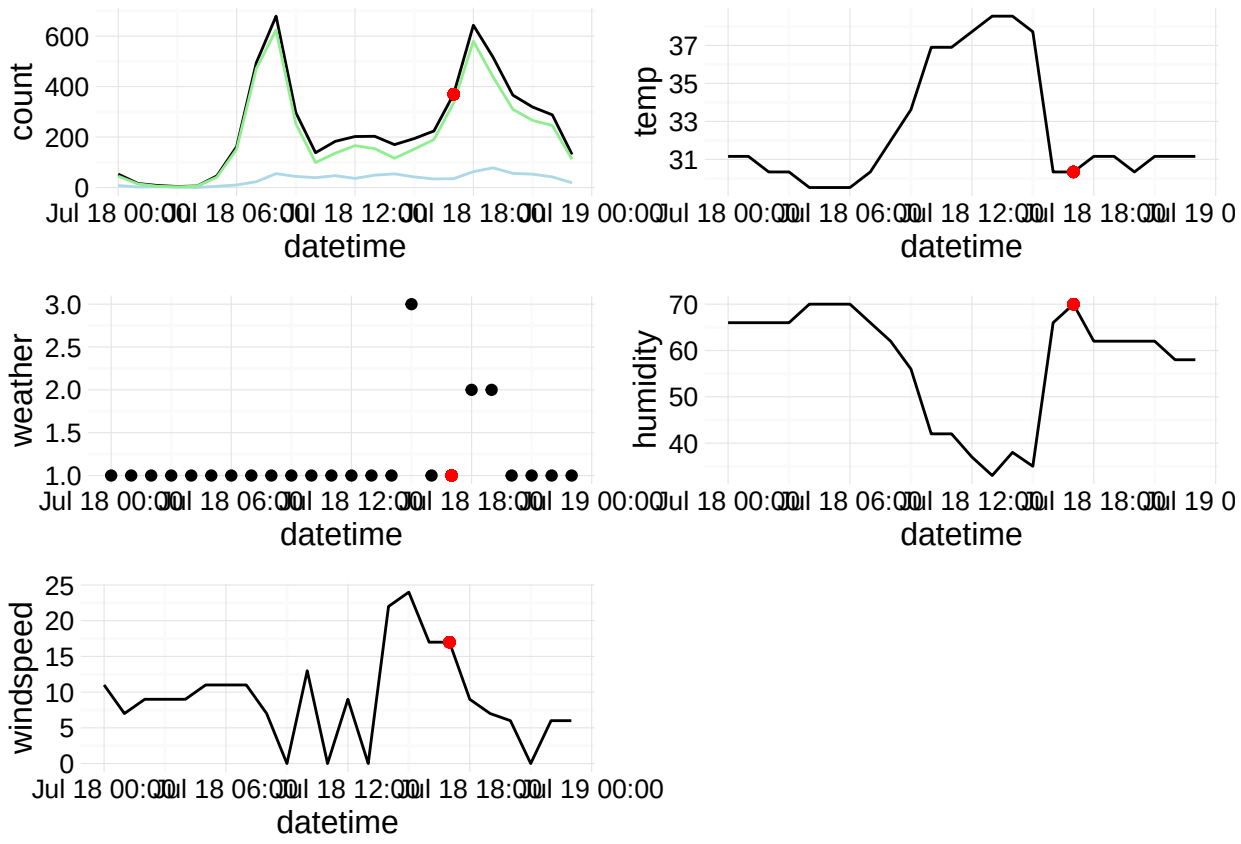

```
check_resids[1:3, ]
```

```
##              datetime season holiday        workingday weather  temp
## 1 2012-07-18 17:00:00   Fall       0       Working day       1 30.34
## 2 2012-04-16 13:00:00 Summer       1 Holiday / weekend       1 30.34
## 3 2012-10-02 08:00:00 Winter       0       Working day       3 24.60
##    atemp humidity windspeed casual registered count year hour month yday
## 1 34.850       70   16.9979     35        335   370 2012   17     7  200
## 2 33.335       45   19.9995     79        184   263 2012   13     4  107
## 3 27.275       88    0.0000      6        128   134 2012    8    10  276
##       resid
## 1 -513.2114
## 2 -352.2824
## 3 -336.9167
```

```
unusual_date <- which.min(actual_preds - train_preds)
unusual_time <- train[unusual_date, "datetime"]
unusual_day <- train[floor_date(train[ , "datetime"], "day")
  == floor_date(train[unusual_date, "datetime"], "day"), ]

a <- ggplot(unusual_day, aes(x = datetime, y = count)) +
     geom_line() +
     geom_line(aes(y = casual), color = "lightblue") +
     geom_line(aes(y = registered), color = "lightgreen") +
     geom_point(aes(x = unusual_time,
                    y = unusual_day[unusual_day$datetime == unusual_time, "count"]),
```

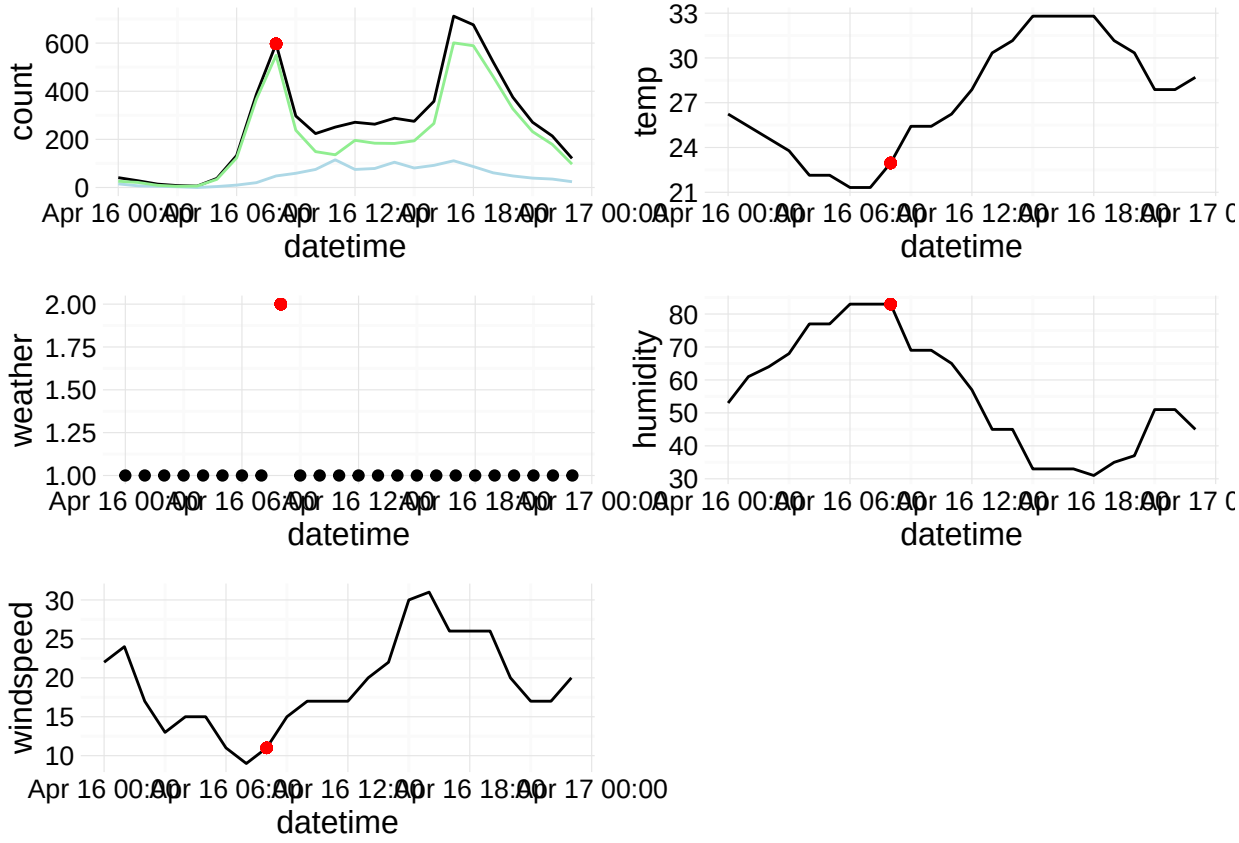
```
                color = "red") +
    theme_minimal()
b <- ggplot(unusual_day, aes(x = datetime, y = temp)) +
  geom_line() +
  geom_point(aes(x = unusual_time,
                 y = unusual_day[unusual_day$datetime == unusual_time, "temp"]),
             color = "red") +
  theme_minimal()
c <- ggplot(unusual_day, aes(x = datetime, y = weather)) +
  geom_point() +
  geom_point(aes(x = unusual_time,
                 y = unusual_day[unusual_day$datetime == unusual_time, "weather"]),
             color = "red") +
  theme_minimal()
d <- ggplot(unusual_day, aes(x = datetime, y = humidity)) +
  geom_line() +
  geom_point(aes(x = unusual_time,
                 y = unusual_day[unusual_day$datetime == unusual_time, "humidity"]),
             color = "red") +
  theme_minimal()
e <- ggplot(unusual_day, aes(x = datetime, y = windspeed)) +
  geom_line() +
  geom_point(aes(x = unusual_time,
                 y = unusual_day[unusual_day$datetime == unusual_time,
                                 "windspeed"]),
             color = "red") +
  theme_minimal()
grid.arrange(a, b, c, d, e, ncol = 2)
```




```
unusual_date <- which.max(actual_preds - train_preds)
unusual_time <- train[unusual_date, "datetime"]
unusual_day <- train[floor_date(train[ , "datetime"], "day")
  == floor_date(train[unusual_date, "datetime"], "day"), ]

a <- ggplot(unusual_day, aes(x = datetime, y = count)) +
     geom_line() +
     geom_line(aes(y = casual), color = "lightblue") +
     geom_line(aes(y = registered), color = "lightgreen") +
     geom_point(aes(x = unusual_time,
                    y = unusual_day[unusual_day$datetime == unusual_time, "count"]),
                color = "red") +
     theme_minimal()
b <- ggplot(unusual_day, aes(x = datetime, y = temp)) +
  geom_line() +
  geom_point(aes(x = unusual_time,
                    y = unusual_day[unusual_day$datetime == unusual_time, "temp"]),
                color = "red") +
  theme_minimal()
c <- ggplot(unusual_day, aes(x = datetime, y = weather)) +
  geom_point() +
  geom_point(aes(x = unusual_time,
                    y = unusual_day[unusual_day$datetime == unusual_time, "weather"]),
                color = "red") +
  theme_minimal()
d <- ggplot(unusual_day, aes(x = datetime, y = humidity)) +
  geom_line() +
```

```
  geom_point(aes(x = unusual_time,
                 y = unusual_day[unusual_day$datetime == unusual_time, "humidity"]),
             color = "red") +
  theme_minimal()
e <- ggplot(unusual_day, aes(x = datetime, y = windspeed)) +
  geom_line() +
  geom_point(aes(x = unusual_time,
                 y = unusual_day[unusual_day$datetime == unusual_time,
                                 "windspeed"]),
             color = "red") +
  theme_minimal()
grid.arrange(a, b, c, d, e, ncol = 2)
```



Adding interaction between current and lag-1 weather

Based on the residuals by temperature for the previous model, it may be possible to improve the model by using a non-linear function of `temp`. I did this using a natural cubic spline, using the `ns` function from the `splines` package. I places the knots at the higher temperatures where there seemed to be a pattern with residuals before:

```
mod_7 <- glm(count ~ year*hour*workingday*season +
               factor(weather)*factor(lag(weather, 1)) +
               ns(temp, knots = c(30, 35)),
             data = train, family = quasipoisson)
```

```
train_preds <- c(predict(mod_4, type = "response")[1], # Need to use last model to
                 predict(mod_7, type = "response"))    # predict first observation
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.3266834
```

The RMSLE improves some for this model. I also submitted to Kaggle:

```
test_preds <- predict(mod_7, newdata = test, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
test_preds <- c(predict(mod_4, newdata = test, type = "response")[1],
                test_preds[2:length(test_preds)])
write_test_preds(test_preds, mod_name = "year_hour_workday_season_lag_weather_int_ns_temp")
```

This submission scored 0.39539.