Class notes– Feb. 25

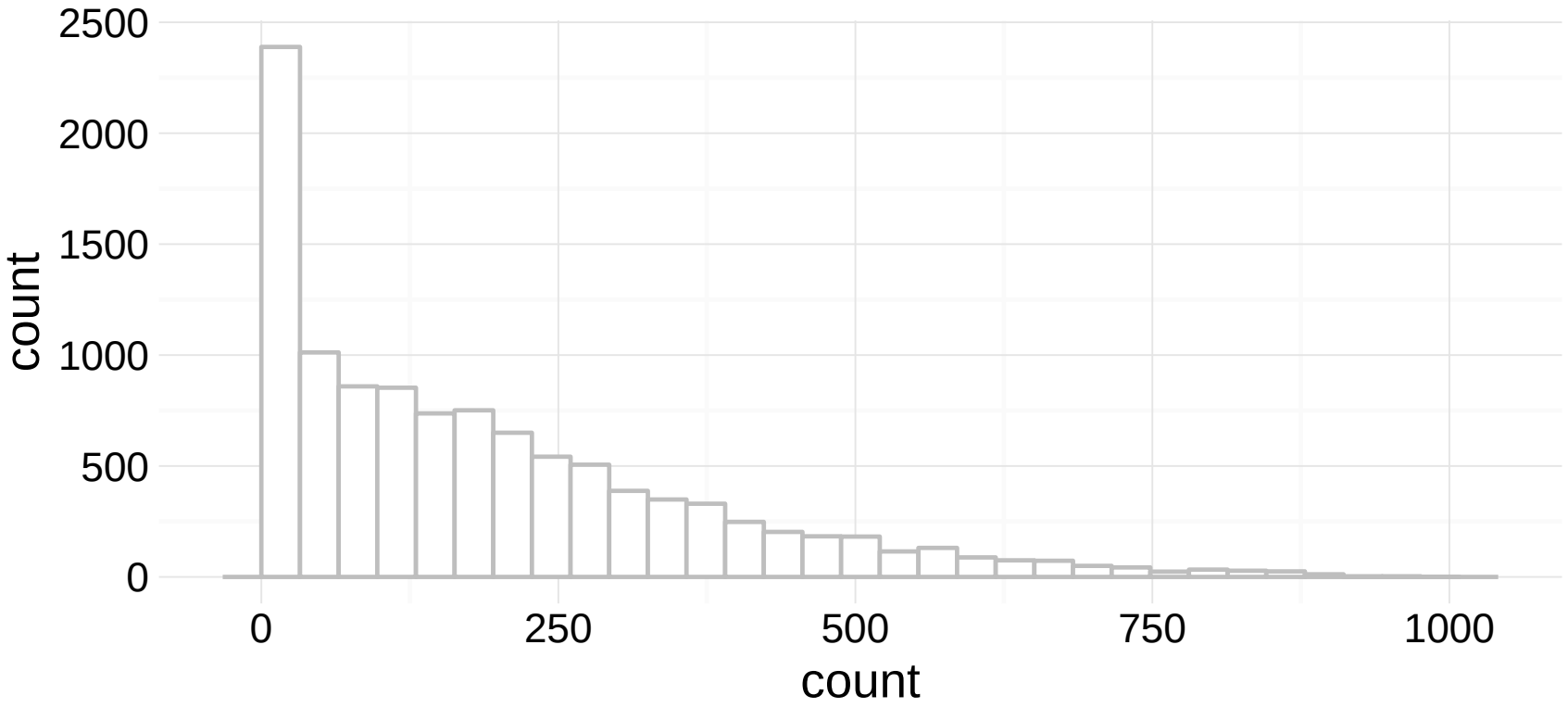
Brooke Anderson

February 25, 2016

Adding datetime features

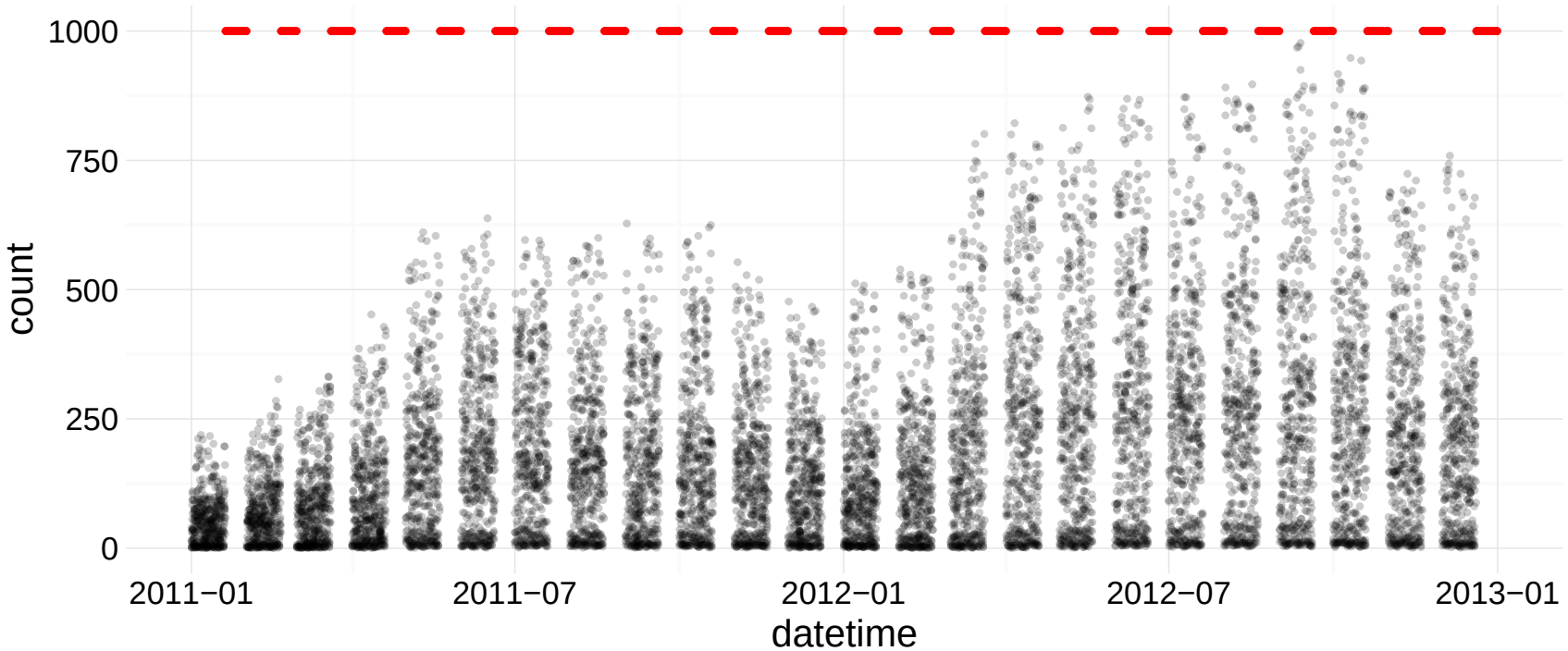
```
train <- read.csv("data/train.csv", as.is = TRUE)
test <- read.csv("data/test.csv", as.is = TRUE)

train <- mutate(train,
                datetime = ymd_hms(datetime),
                year = factor(year(datetime)),
                hour = factor(hour(datetime)),
                month = month(datetime),
                yday = yday(datetime))
```

Distribution of response variable

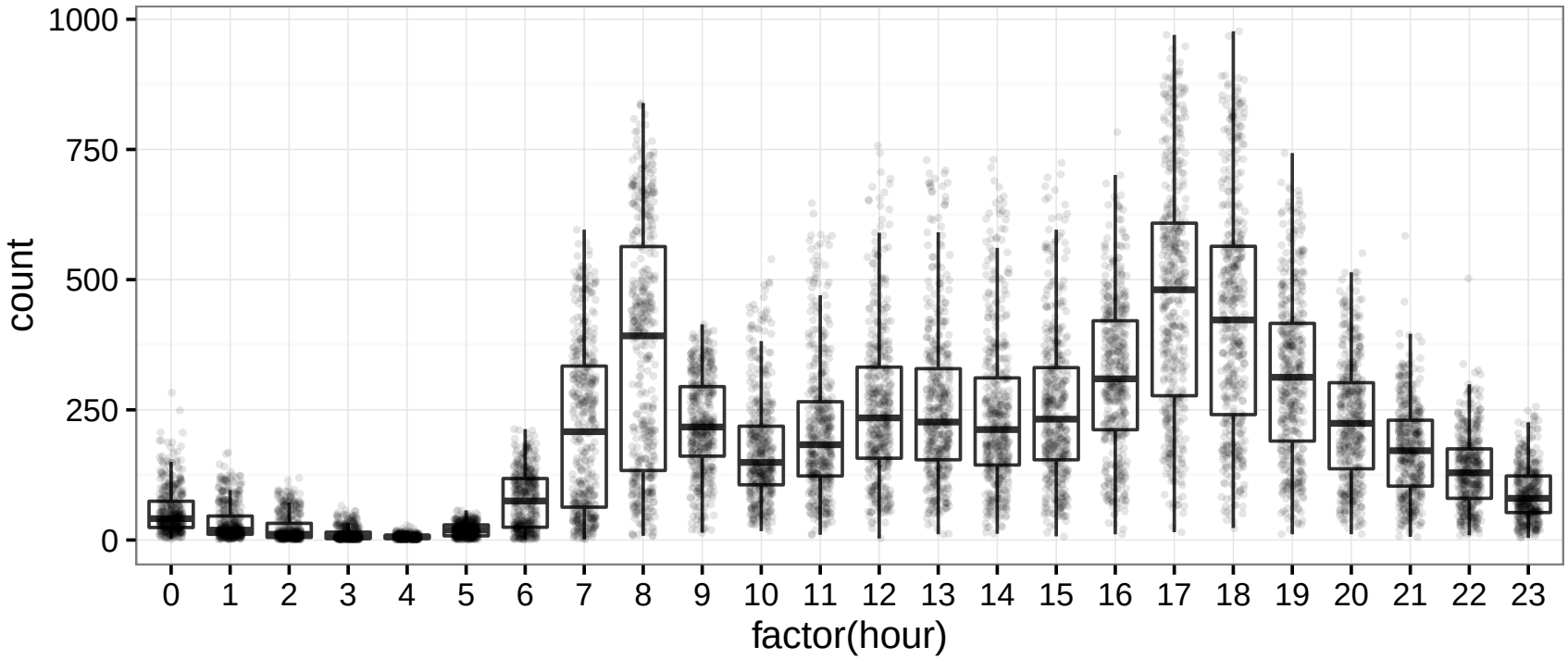


Exploring patterns in bike use by time

Number of bikes in use by times. Red points show observation times of testing data.



Exploring patterns in bike use by time



Exploring patterns in bike use by time

RMSLE

From Kaggle:

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

RMSLE

```
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)
}
```

Intercept-only model

```
mod_0 <- glm(count ~ 1, data = train)
train_preds <- predict(mod_0)
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

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

Function to write Kaggle submission

```
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)
}
```

Linear regression

Fit a linear regression based on hour of the day, working day, and season (all modeled as factors). On Kaggle: 0.57524.

```
mod_1 <- glm(count ~ hour*workingday*season,
             data = train)
train_preds <- predict(mod_1)
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

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

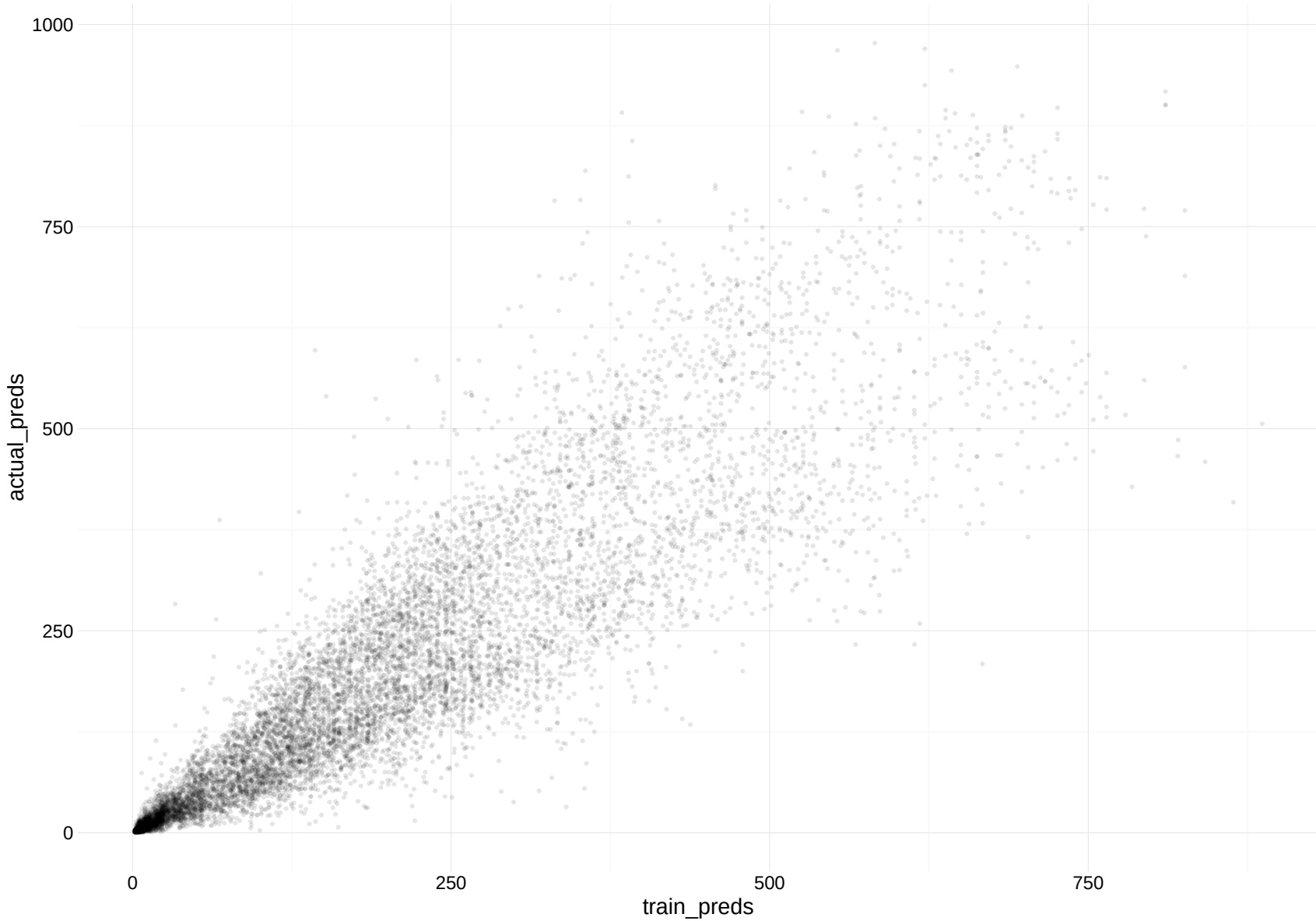
Linear regression

Adding `weather` and `temperature`– on Kaggle: 0.48868.

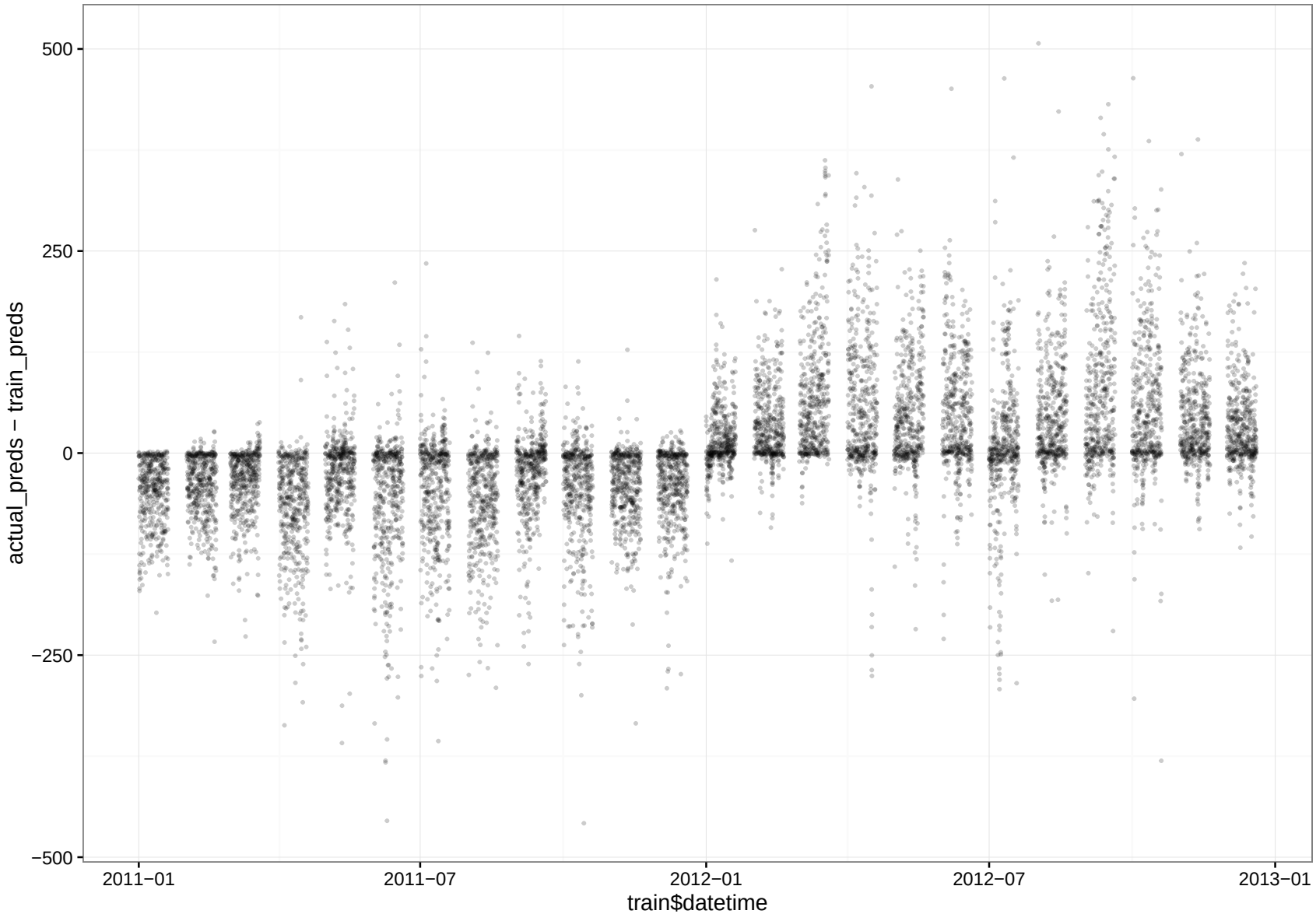
```
mod_2 <- glm(count ~ hour*workingday*season +
                  weather + temp,
                data = train, family = quasipoisson)
train_preds <- predict(mod_2, type = "response")
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

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

Linear regression



Linear regression



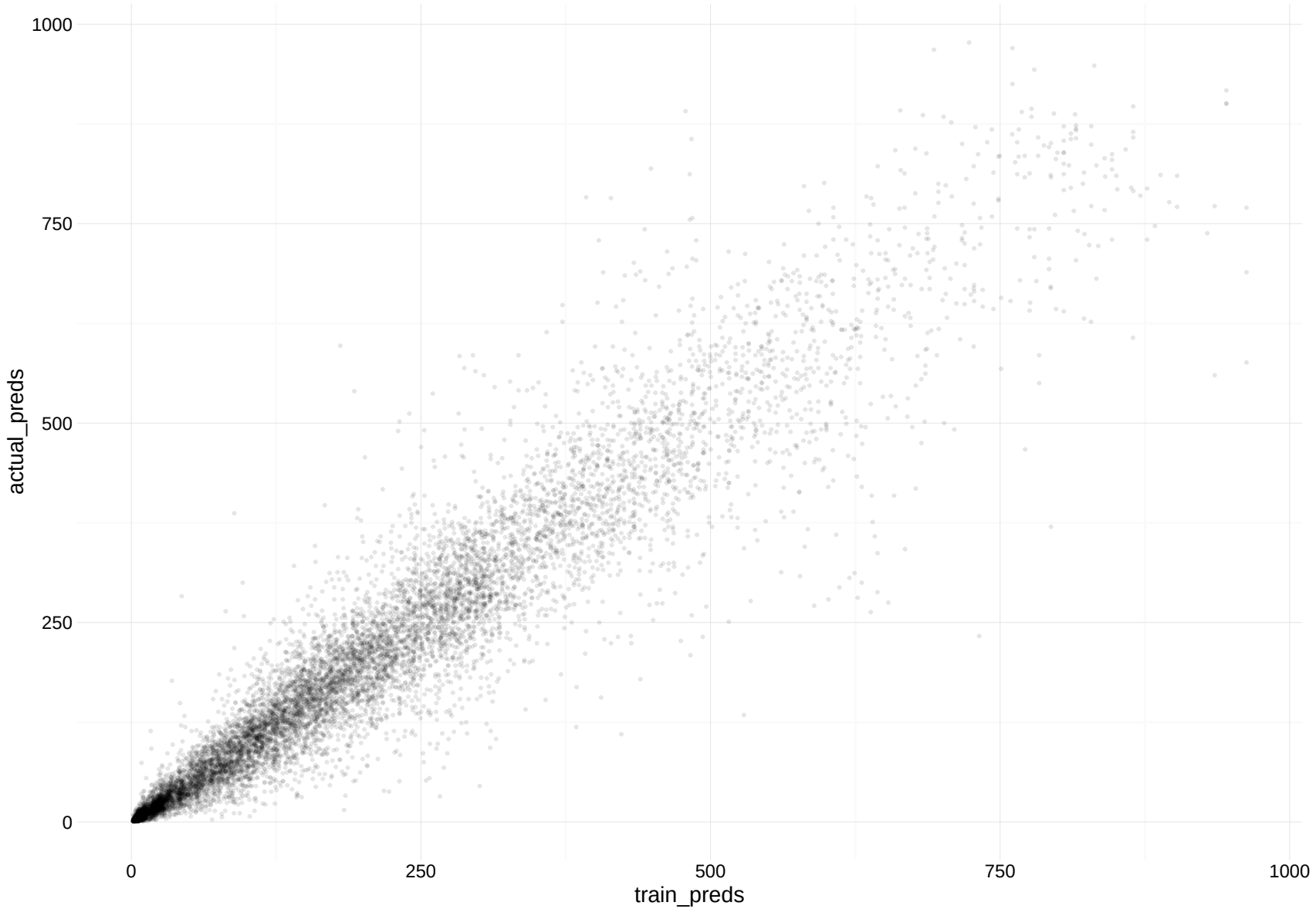
Linear regression

Based on the residuals by time for the previous model, year may be a useful predictor– on Kaggle: 0.43228.

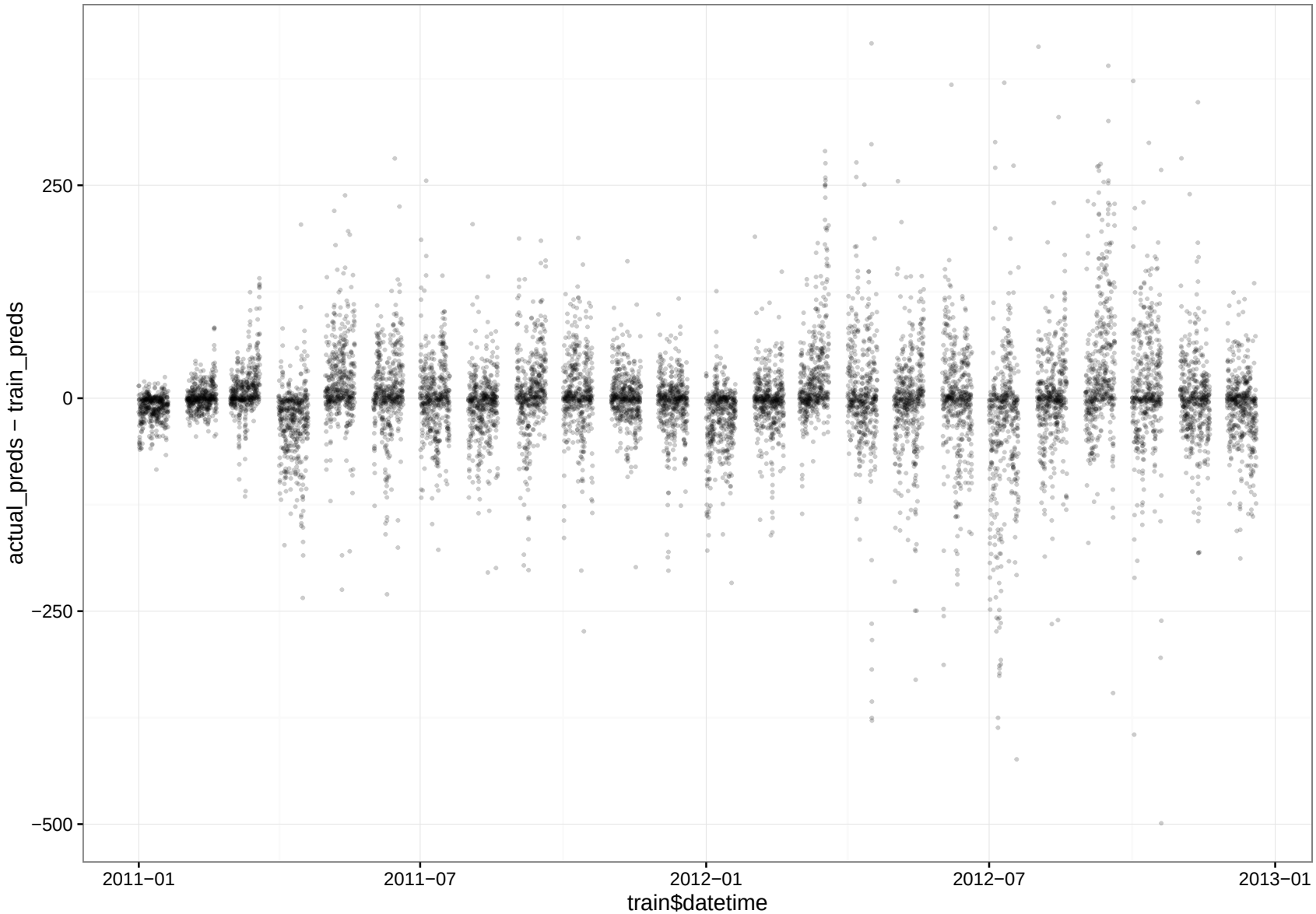
```
mod_3 <- glm(count ~ year*hour*workingday*season +
                weather + temp,
              data = train, family = quasipoisson)
train_preds <- predict(mod_3, type = "response")
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

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

Linear regression

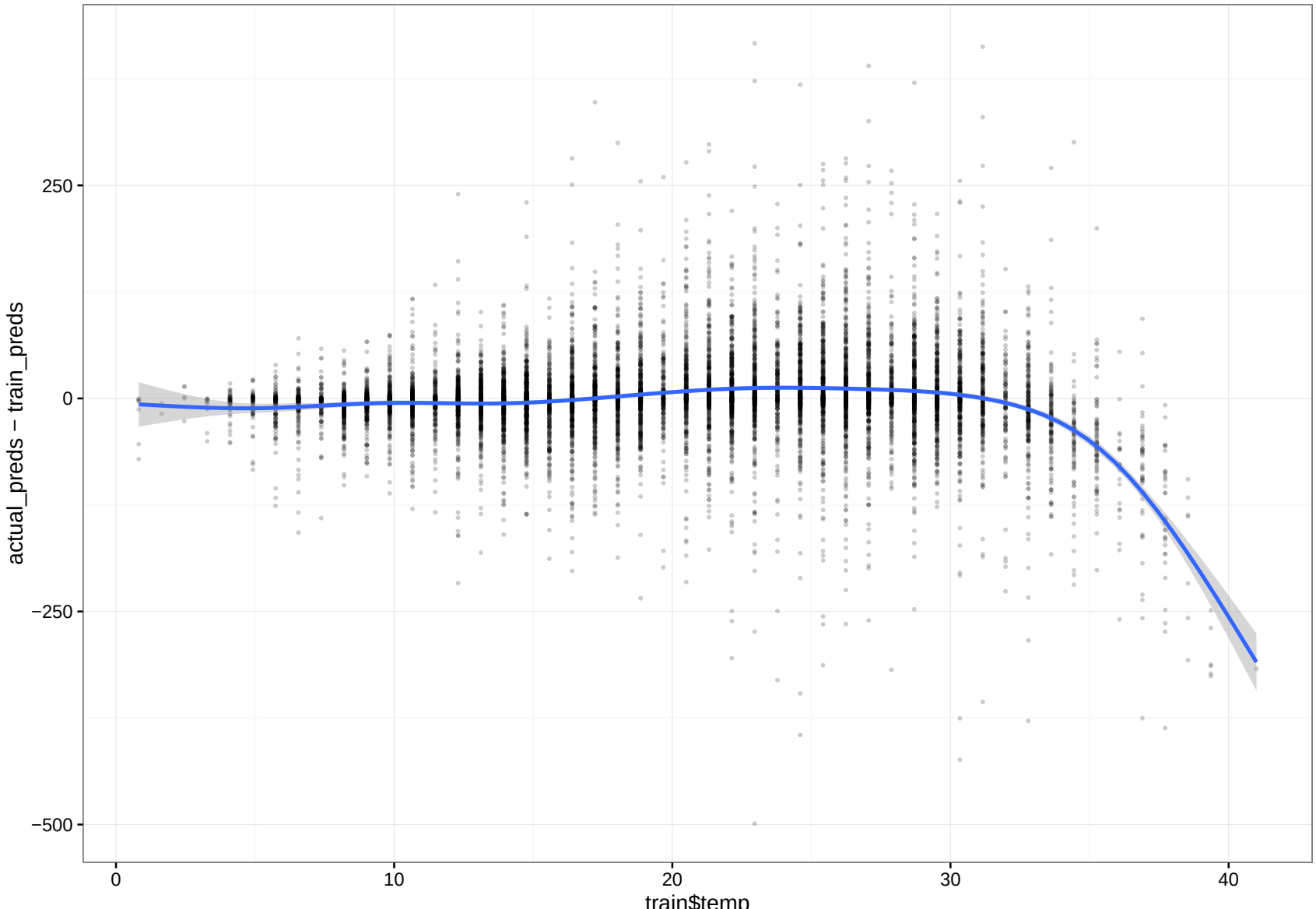


Linear regression



Linear regression

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



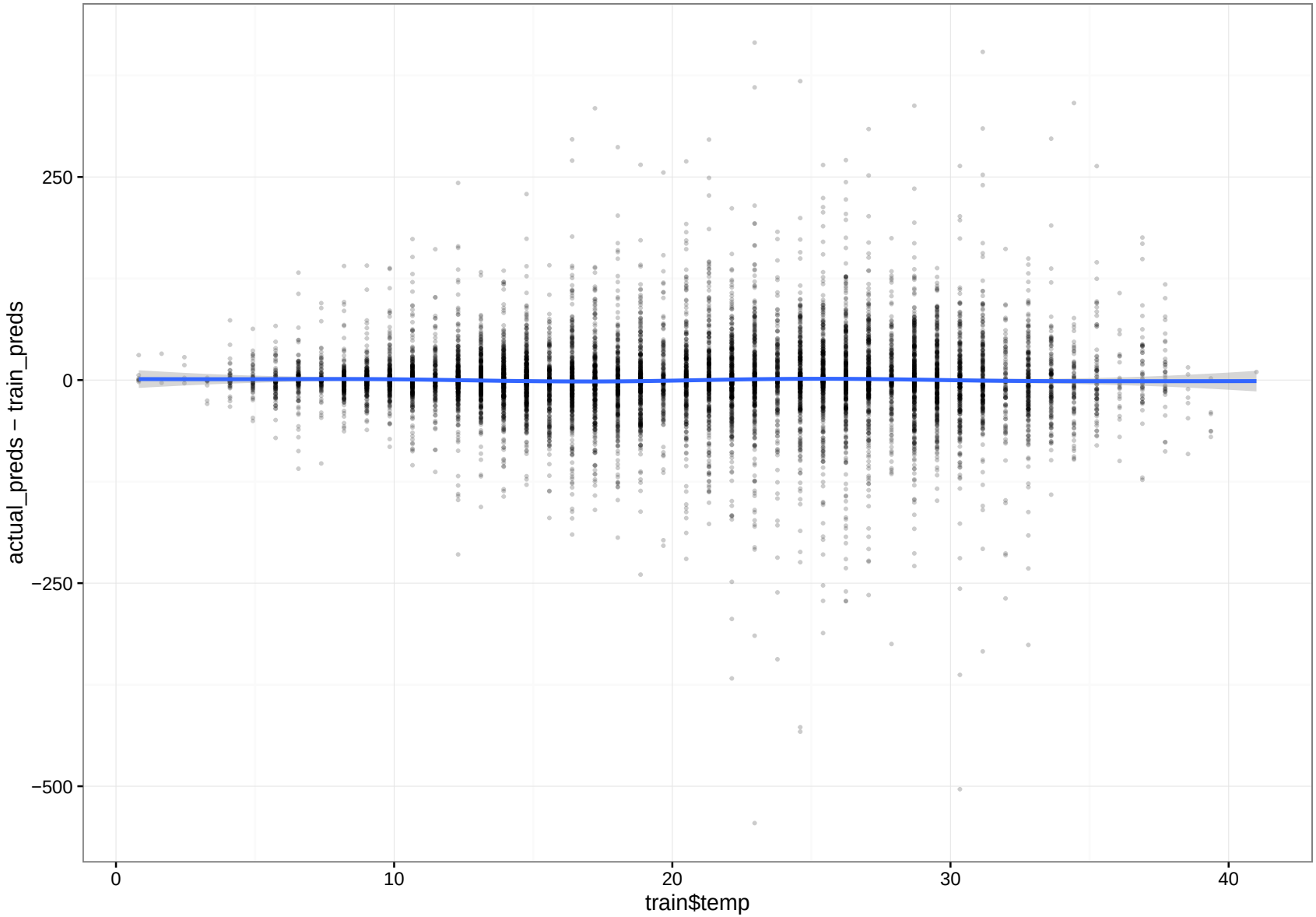
Linear regression

Add spline with knots at high temperature– on Kaggle: 0.41048.

```
library(splines)
mod_4 <- glm(count ~ year*hour*workingday*season +
                 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
```

Linear regression



Tuning with caret

You first must create a function with your loss function. It needs to input a dataframe with the columns `pred` and `obs` and output a numeric vector with a name.

```
rmsle_fun <- function(data, lev = NULL,
                          model = NULL, ...){
  log_p_1 <- log(data$pred + 1)
  log_a_1 <- log(data$obs + 1)
  sle <- (log_p_1 - log_a_1)^2
  rmsle <- sqrt(mean(sle))
  names(rmsle) <- "rmsle"
  return(rmsle)
}
```

Tuning with caret

Next, specity to use that function in a `trainControl` object using the `summaryFunction` argument. You'll reference this argument when you `train`.

```
fitControl <- trainControl(method = "cv",
                           number = 5,
                           summaryFunction = rmsle_fun)
```

Tuning a k-NN model

```
set.seed(825)
mod_1 <- train(count ~ temp + hour + workingday + year,
               data = train,
               method = "knn",
               trControl = fitControl,
               metric = "rmsle",
               maximize = FALSE,
               preProcess = c("center", "scale",
                              "spatialSign"),
               tuneLength = 5)

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

```
## [1] 0.4080661
```

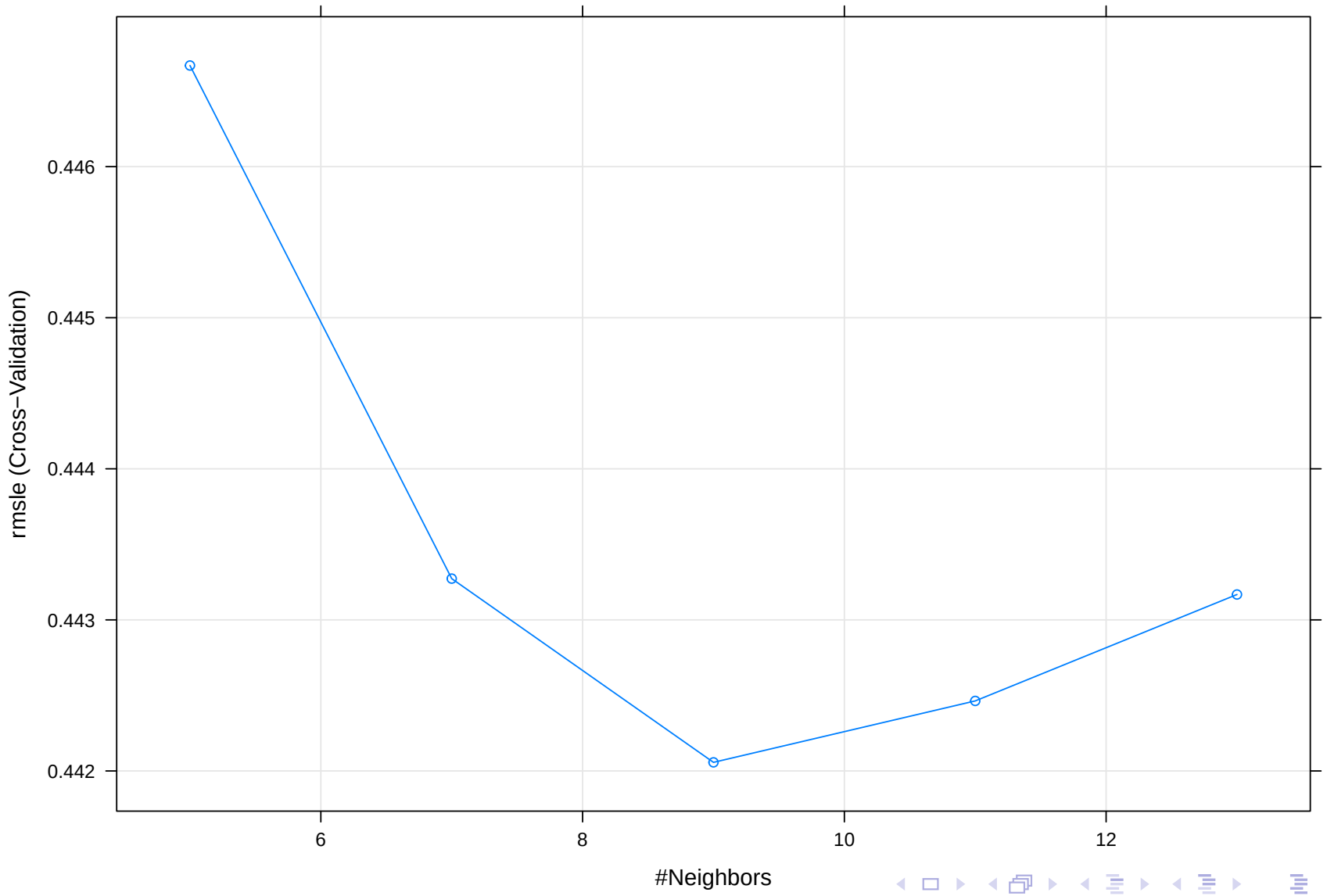
Tuning a k-NN model

```
mod_1
```

```
## k-Nearest Neighbors
##
## 10886 samples
##    15 predictor
##
## Pre-processing: centered, scaled, spatial sign transform
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 8709, 8708, 8710, 8708, 8709
##
## Resampling results across tuning parameters:
##
##   k   rmsle      rmsle SD
##    5  0.4466694  0.005523422
##    7  0.4432731  0.007505331
##    9  0.4420570  0.008668263
```

Tuning a k-NN model

```
plot(mod_1)
```



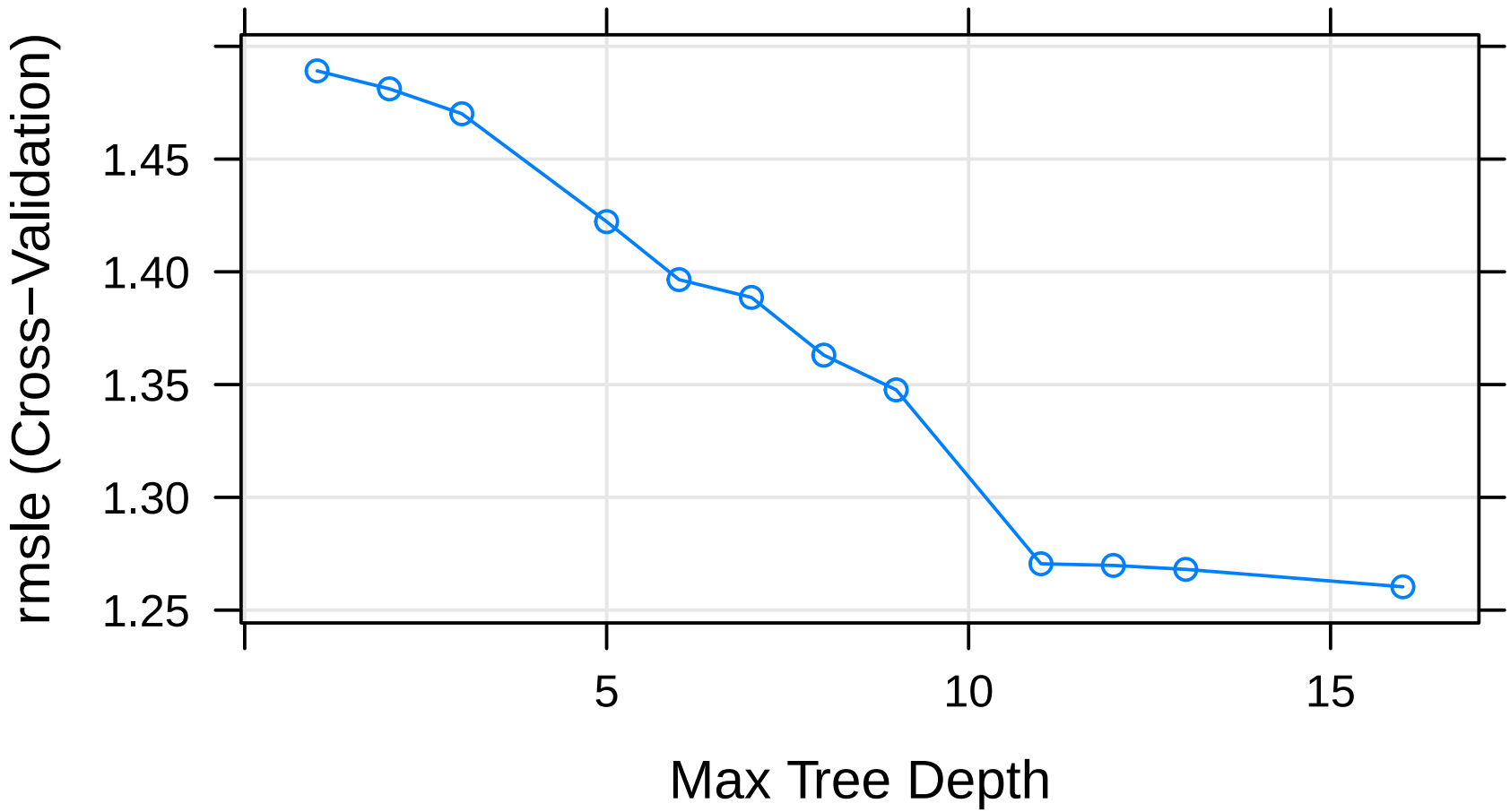
Tuning a regression tree model

```
set.seed(825)
mod_2 <- train(count ~ season + holiday + workingday +
                 weather + temp + atemp + humidity +
                 windspeed + year + hour +
                 month + yday,
               data = train,
               method = "rpart2",
               trControl = fitControl,
               metric = "rmsle",
               maximize = FALSE,
               tuneLength = 12)
```

```
## Loading required package: rpart
```

Tuning a regression tree model

```
plot(mod_2)
```

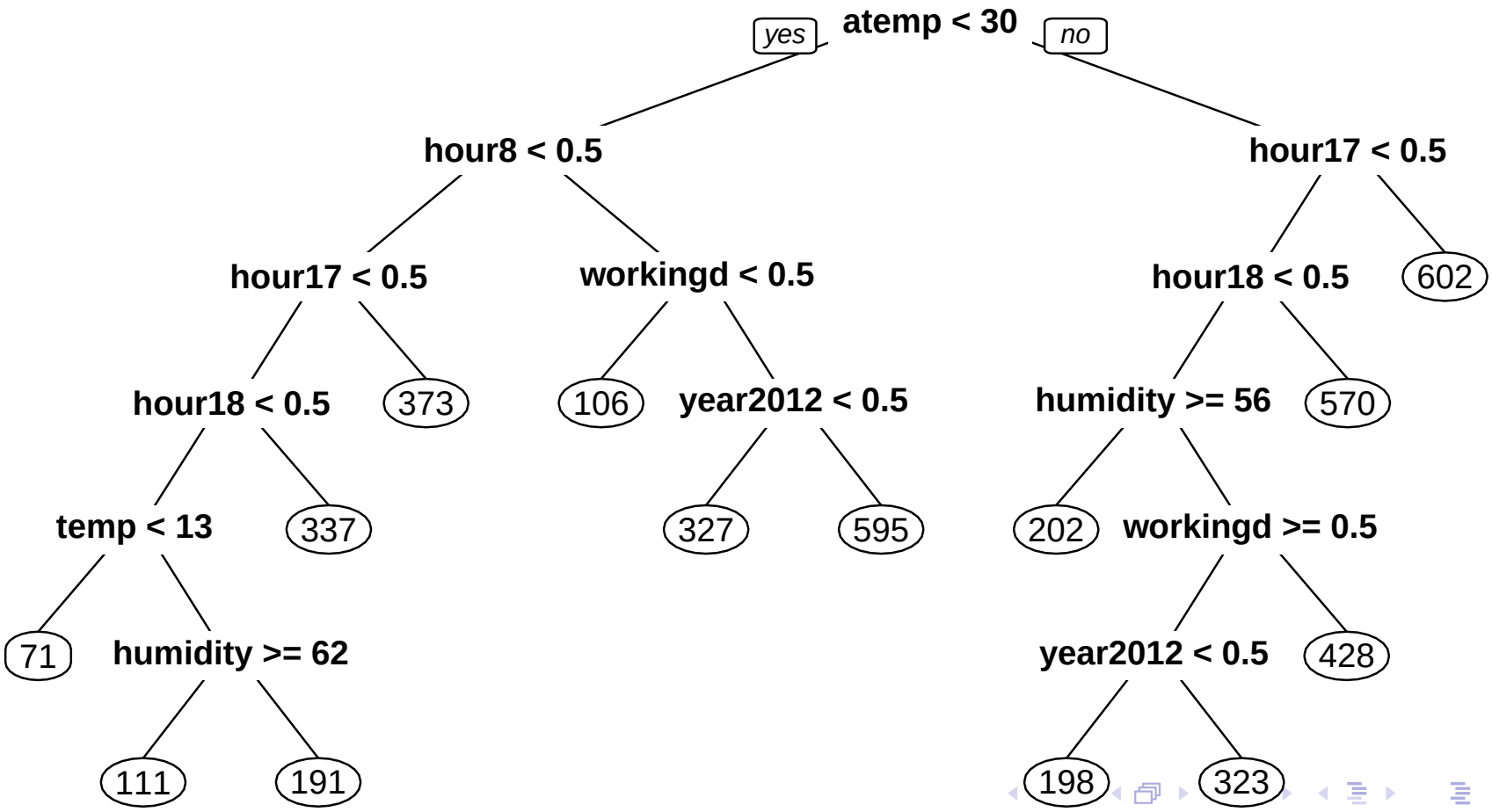


Tuning a regression tree model

```
library(rpart.plot)
```

```
## Loading required package: rpart
```

```
prp(mod_2$finalModel)
```



Tuning a regression tree model

On Kaggle: 1.29878.

```
train_preds <- predict(mod_2, newdata = train)
rmsle(train_preds, train$count)
```

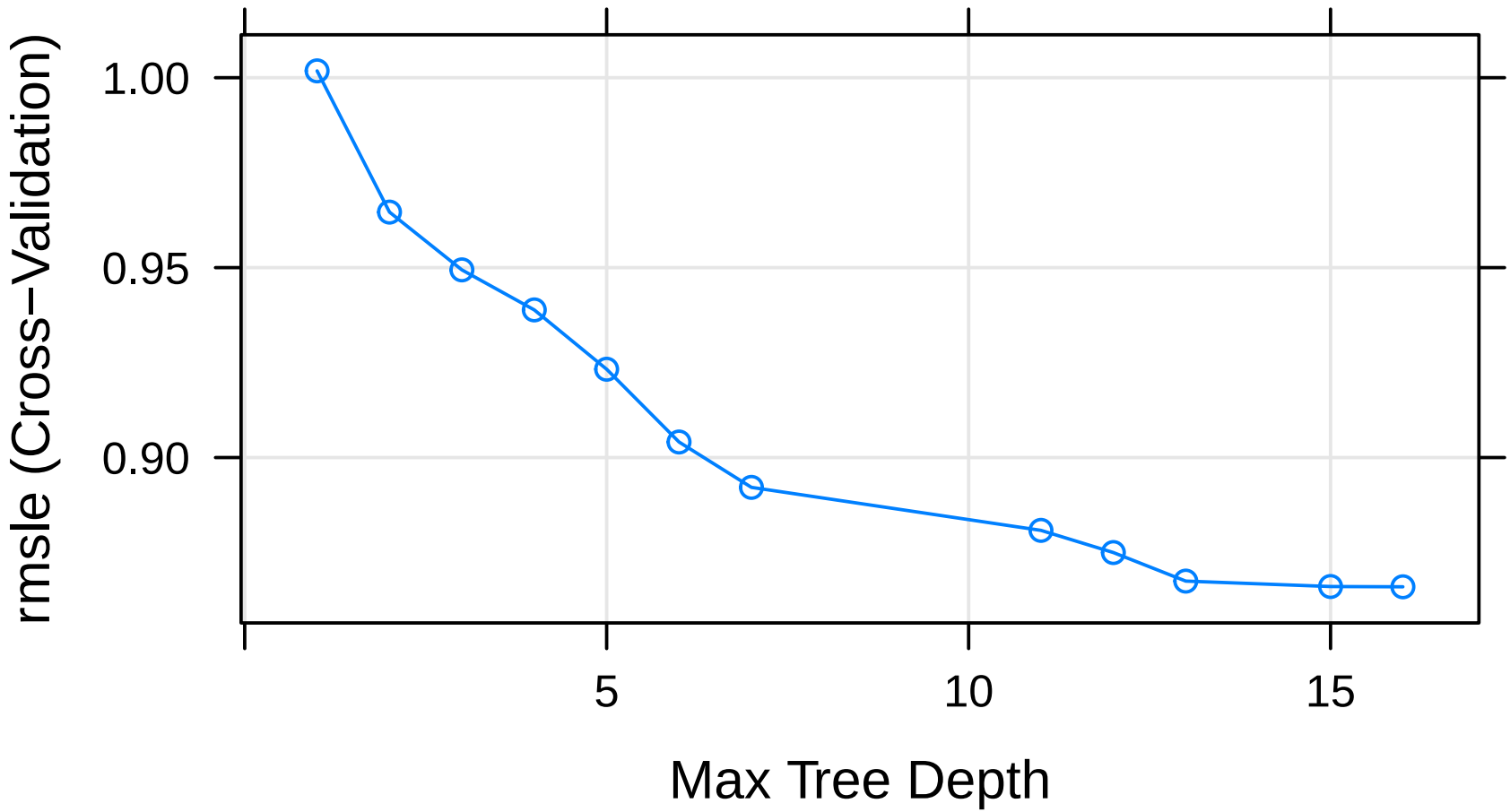
```
## [1] 1.290799
```

Tuning a regression tree model

```
set.seed(825)
mod_3 <- train(count ~ season + holiday + workingday +
                 as.numeric(weather) + temp + atemp +
                 humidity + windspeed + year +
                 as.numeric(hour) + month + yday,
               data = train,
               method = "rpart2",
               trControl = fitControl,
               metric = "rmsle",
               maximize = FALSE,
               tuneLength = 12)
```

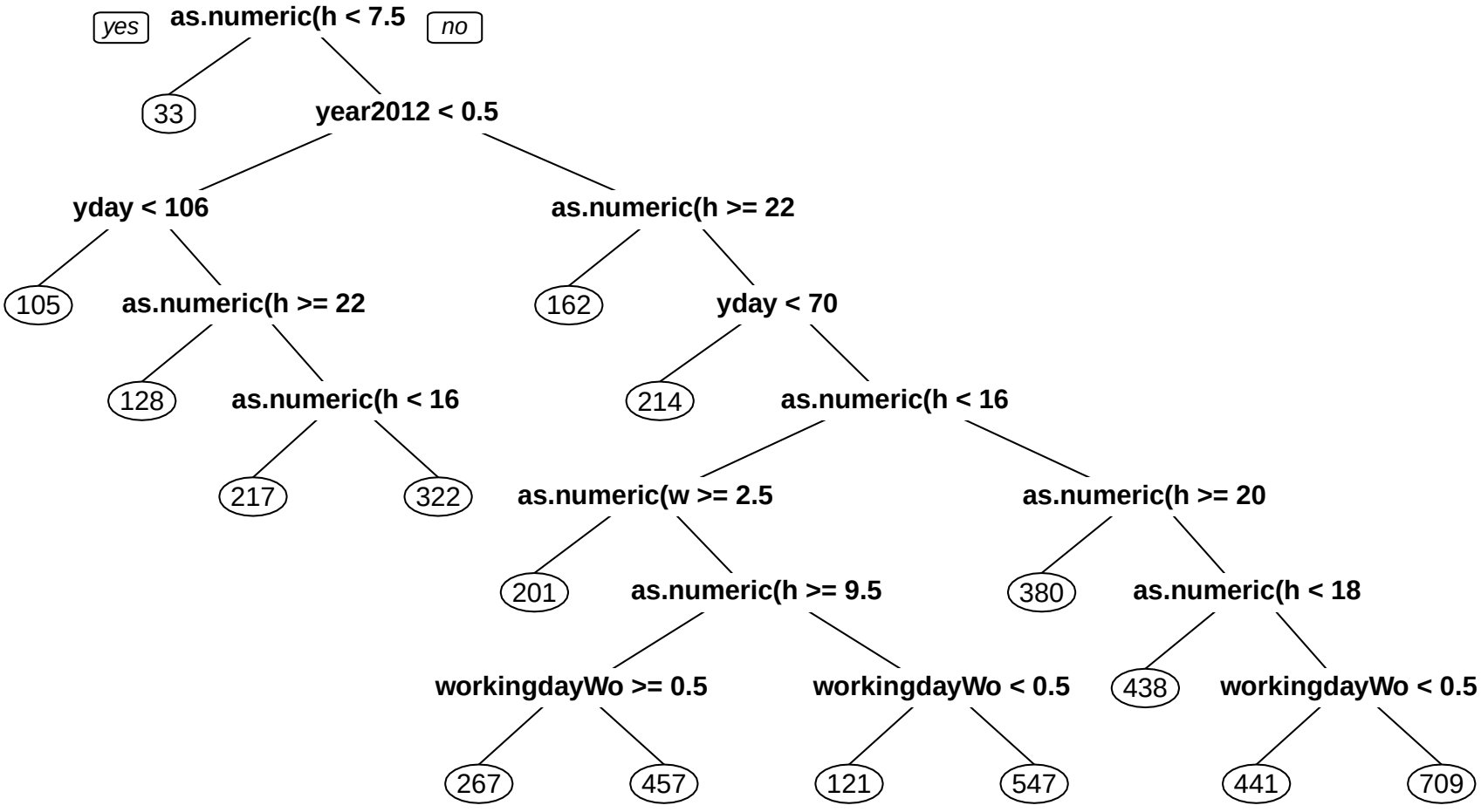
Tuning a regression tree model

```
plot(mod_3)
```



Tuning a regression tree model

```
library(rpart.plot)
prp(mod_3$finalModel)
```



Tuning a regression tree model

```
train_preds <- predict(mod_3, newdata = train)
rmsle(train_preds, train$count)
```

```
## [1] 0.8572327
```