# Feature Engineering and Variable Selection

Brooke Anderson

March 1, 2016

# Data wrangling

This feature engineering took some data wrangling. For that, I found the packages `dplyr`, `tidyr`, and `lubridate` very helpful. You might want to check out RStudio's Data Wrangling Cheatsheet.

# Lagged weather variables

There are breaks in the time series, so you need to group everything by continuous time groups before doing that lagging. Otherwise, you would end up with a case where you're saying that the temperature the last hour was really the last hour of a day several days ago, before a gap in the time series.

# Lagged weather variables

This code isn't terribly elegant, but it's one way to get those groups (`time_group`). First, calculate the difference between each `datetime` and that for the previous observation:

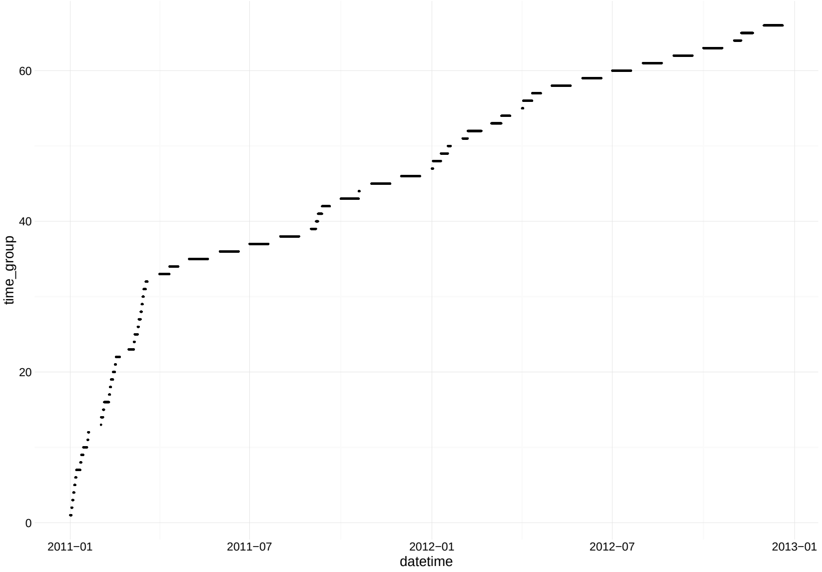
```
train <- mutate(train,
                time_diff = difftime(datetime,
                                     lag(datetime)),
                time_group = NA)
```

# Lagged weather variables

Then loop through and assign `time_group` numbers. Everytime you hit an observation over an hour after the last one, increment the `group_num`:

```
group_num <- 1
for(i in 1:nrow(train)){
  if(train$time_diff[i] > dhours(1) &
     !is.na(train$time_diff[i])){
    group_num <- group_num + 1
  }
  train$time_group[i] <- group_num
}
```

# Lagged weather variables



# Lagged weather variables

Then group by this new variable before you do lagged weather values:

```
train <- group_by(train, time_group) %>%
  mutate(temp1 = lag(temp, 1),
         temp2 = lag(temp, 2),
         temp3 = lag(temp, 3),
         temp4 = lag(temp, 4),
         temp5 = lag(temp, 5),
         weather1 = lag(weather, 1),
         weather2 = lag(weather, 2),
         weather3 = lag(weather, 3),
         weather4 = lag(weather, 4),
         weather5 = lag(weather, 5)) %>%
  ungroup() %>%
  select(-time_diff)
```

# Lagged weather variables

There are missing values for these lagged variables (you can't get lags that reach before the first hour in the time group):

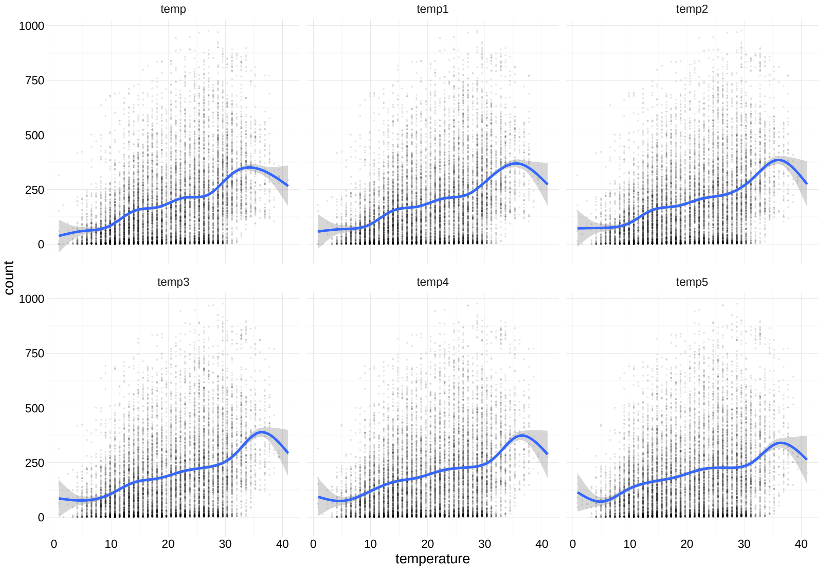
```
## Source: local data frame [6 x 2]
##
##     lag num_missing
##   (chr)       (int)
## 1     0           0
## 2     1          66
## 3     2         132
## 4     3         198
## 5     4         264
## 6     5         329
```

# Lagged weather variables

Correlations between lagged temperature variables:

```
##       temp temp1 temp2 temp3 temp4 temp5
## temp  1.00  0.99  0.98  0.96  0.94  0.92
## temp1 0.99  1.00  0.99  0.98  0.96  0.94
## temp2 0.98  0.99  1.00  0.99  0.98  0.96
## temp3 0.96  0.98  0.99  1.00  0.99  0.98
## temp4 0.94  0.96  0.98  0.99  1.00  0.99
## temp5 0.92  0.94  0.96  0.98  0.99  1.00
```

# Lagged weather variables



# By-day variables

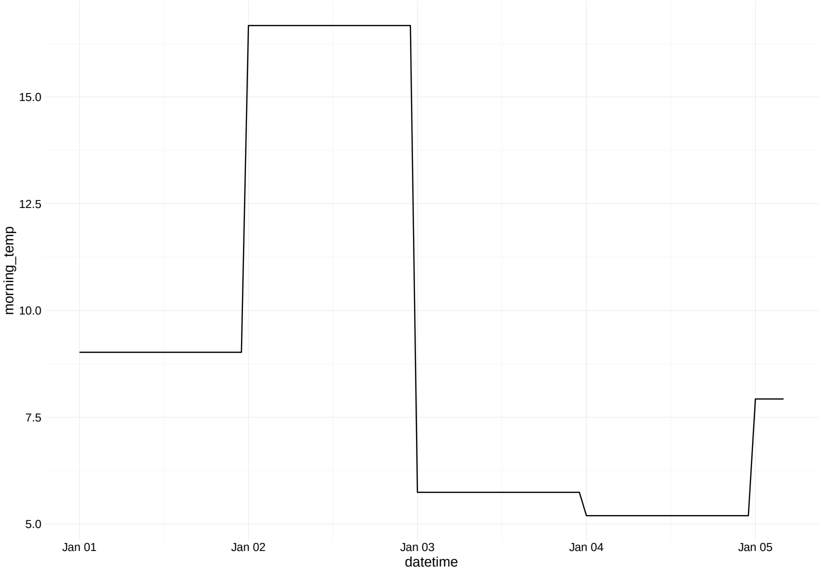
Next, create some variables by day. For example, was there one or more hours of bad weather during the day?

# By-day variables

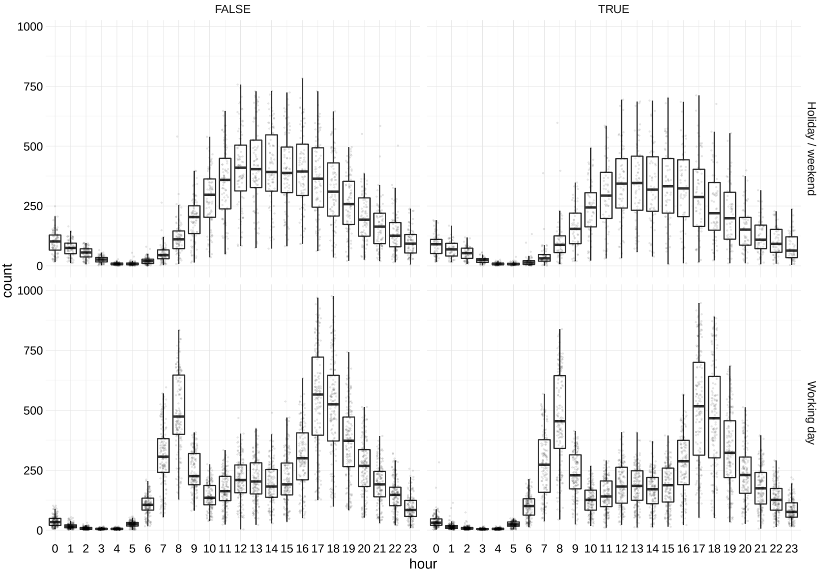
```
train <- mutate(train,
                day = format(datetime, "%Y%m%d")) %>%
  group_by(day) %>%
  mutate(mean_daily_temp = mean(temp),
         morning = hour %in% c("6", "7", "8"),
         morning_temp = mean(temp[morning]),
         hours_bad_weather = sum(weather != "Clear"),
         any_bad_weather = hours_bad_weather > 0,
         morning_bad_weather = sum(weather[morning] !=
                                       "Clear") > 0) %>%
  select(-morning) %>%
  ungroup()
```

# By-day variables

You can see that these values are constant within a day:



# By-day variables



# Feature creation

The dataframe now has 33 predictors (although many of these are very correlated with each other). Once you add in all the interactions, it will have even (a lot) more.

# Creating test and validation sets

Split your training data into test and validation sets, stratifying by time group:

```
sample_size <- length(unique(train$time_group)) * (2 / 3)
train_groups <- sample(unique(train$time_group),
                        size = sample_size)
my_train_split <- train$time_group %in% train_groups

my_train <- filter(train, my_train_split) %>%
  select(-casual, -registered, -datetime, -day)
my_test <- filter(train, !my_train_split) %>%
  select(-casual, -registered, -datetime, -day)
```

# Forward stepwise selection

```
library(leaps) # For variable selection
regfit_for <- regsubsets(count ~ season * workingday *
                           hour * year +
                           holiday + weather + temp +
                           atemp + humidity + windspeed +
                           month + yday + temp1 + temp2 +
                           temp3 + temp4 + temp5 +
                           weather1 + weather2 +
                           weather3 + weather4 +
                           weather5 + mean_daily_temp +
                           morning_temp +
                           hours_bad_weather +
                           any_bad_weather +
                           morning_bad_weather,
                         data = my_train,
                         method = "forward", nvmax = 418)
```

# Forward stepwise selection

Here are the first ten predictors from that process:

```r
names(coef(regfit_for, 10))
```

```
##  [1] "(Intercept)"
##  [2] "year2012"
##  [3] "temp"
##  [4] "mean_daily_temp"
##  [5] "workingdayWorking day:year2012"
##  [6] "workingdayWorking day:hour7:year2012"
##  [7] "workingdayWorking day:hour8:year2012"
##  [8] "workingdayWorking day:hour9:year2012"
##  [9] "workingdayWorking day:hour17:year2012"
## [10] "workingdayWorking day:hour18:year2012"
## [11] "workingdayWorking day:hour19:year2012"
```

# Test on validation set

To pick the best number of variables, test the models on the validation set. First, create a model matrix for the test data:

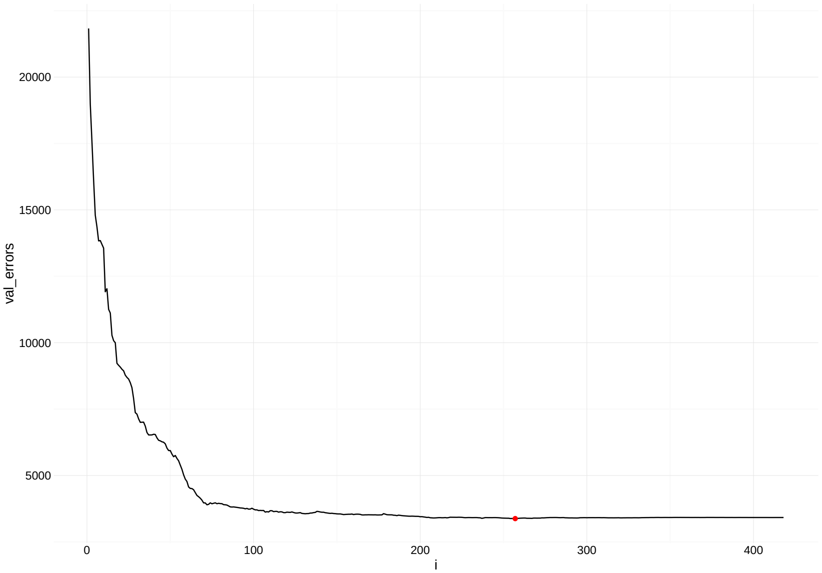
```
test_mat <- model.matrix(count ~ season * workingday *
                            hour * year +
                            holiday + weather + temp +
                            atemp + humidity + windspeed +
                            month + yday + temp1 + temp2 +
                            temp3 + temp4 + temp5 +
                            weather1 + weather2 +
                            weather3 + weather4 +
                            weather5 + mean_daily_temp +
                            morning_temp +
                            hours_bad_weather +
                            any_bad_weather +
                            morning_bad_weather,
                          data = my_test)
```

# Test on validation set

Then test on the validation test set:

```
val_errors <- rep(NA, 418)
val_rmsle <- rep(NA, 418)
actual <- my_test$count[complete.cases(my_test)]
for(i in 1:418){
  coef_i <- coef(regfit_for, id = i)
  pred <- test_mat[ , names(coef_i)] %*% coef_i
  pred[pred < 0] <- 0
  val_errors[i] <- mean((actual - pred)^2)
  val_rmsle[i] <- rmsle(pred, actual)
}
```

# Test on validation set



# Test on validation set