Regression analysis Case-Study

Martin Hanewald

1 August 2018

1 Abstract

This analysis tries to predict the hourly demand of a bikeshare provider in Washington D.C. The dataset has been obtained from https://www.kaggle.com/c/bike-sharing-demand (https://www.kaggle.com/c/bike-sharing-demand).

From kaggle description:

> Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able rent a bike from a one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world. The data generated by these systems makes them attractive for researchers because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded. Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city. In this competition, participants are asked to combine historical usage patterns with weather data in order to forecast bike rental demand in the Capital Bikeshare program in Washington, D.C.

We first conduct an exploratory analysis towards the relationships between the predictor variables and the count variable.

Then we try out different models to evaluate the best fit.

2 Configuration and libraries

```
library(tidyverse)
library(caret)
library(lubridate)
```

3 Loading data

```
rawdat <- read_csv('data/bikeshare_train.csv')
rawdat
```

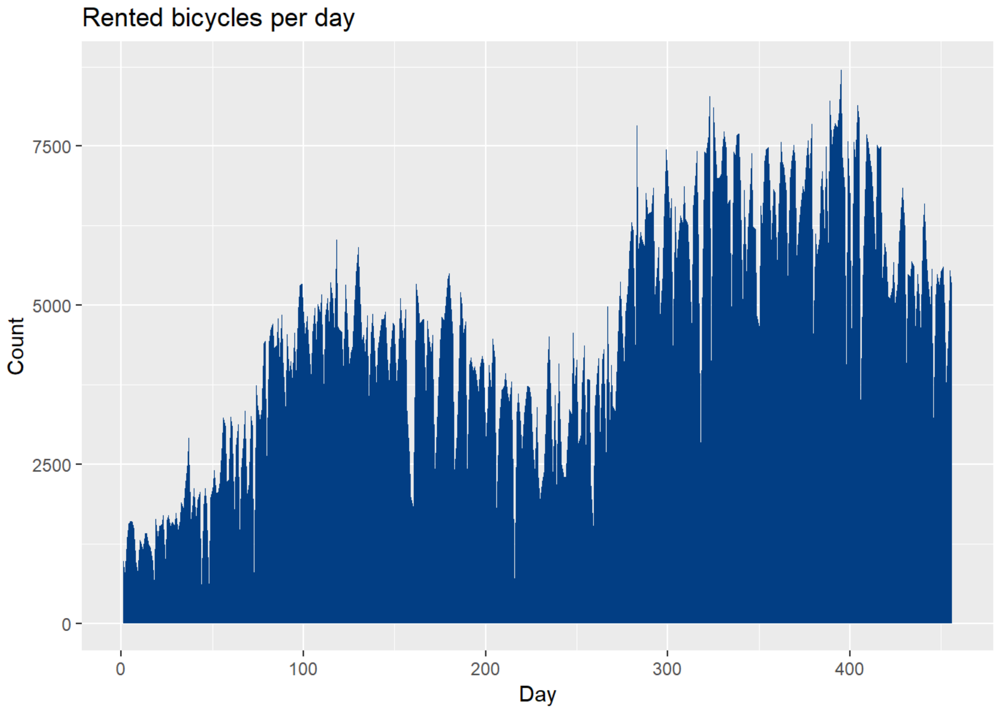
datetime <S3: POSIXct>	season <int>	holiday <int>	workingday <int>	weather <int>	temp <dbl>	atemp <dbl>	humidity <int>	windspeed <dbl>	casual <int>
2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	3
2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8
2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5
2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3
2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0

datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual ▸
<S3: POSIXct>	<int>	<int>	<int>	<int>	<dbl>	<dbl>	<int>	<dbl>	<int>
2011-01-01 05:00:00	1	0	0	2	9.84	12.880	75	6.0032	0
2011-01-01 06:00:00	1	0	0	1	9.02	13.635	80	0.0000	2
2011-01-01 07:00:00	1	0	0	1	8.20	12.880	86	0.0000	1
2011-01-01 08:00:00	1	0	0	1	9.84	14.395	75	0.0000	1
2011-01-01 09:00:00	1	0	0	1	13.12	17.425	76	0.0000	8

1-10 of 10,000 rows | 1-10 of 12 columns Previous **1** 2 3 4 5 6 ... 1000 Next

The data consists of 10.886 rows of hourly data. The variable count describes the number of bicycles rented. Aggregated on a daily basis we get the following timeseries.

```
rawdat %>% mutate(day = date(datetime)) %>%
    group_by(day) %>%
    summarise(count = sum(count)) %>%
    ggplot(aes(x=row_number(day), y = count)) + geom_area(fill = qpal[2]) +
    labs(x = "Day", y = 'Count', title='Rented bicycles per day')
```



4 Preprocessing

4.1 Feature Engineering

Some engineered features like season, holiday and workingday have already been included in the dataset. But we want to add variables for **year, month and hour**.

Secondly the variable **count** is a typical Poisson distributed variable. In order to also work with models, which are not Poisson-compatible we add a log-transformed variable as well.

Furthermore all categorical variables need to be transformed to R's **factor** variable format.

```
dat <- rawdat %>%
    mutate(year = year(datetime),
           month = month(datetime),
           hour = hour(datetime)) %>%
    mutate(logcount = log(count + 1)) %>%
    mutate_at(vars(season, holiday, workingday,
                  weather, year, month, hour), as.factor)

dat
```

datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual ▸
<S3: POSIXct>	<fctr>	<fctr>	<fctr>	<fctr>	<dbl>	<dbl>	<int>	<dbl>	<int>

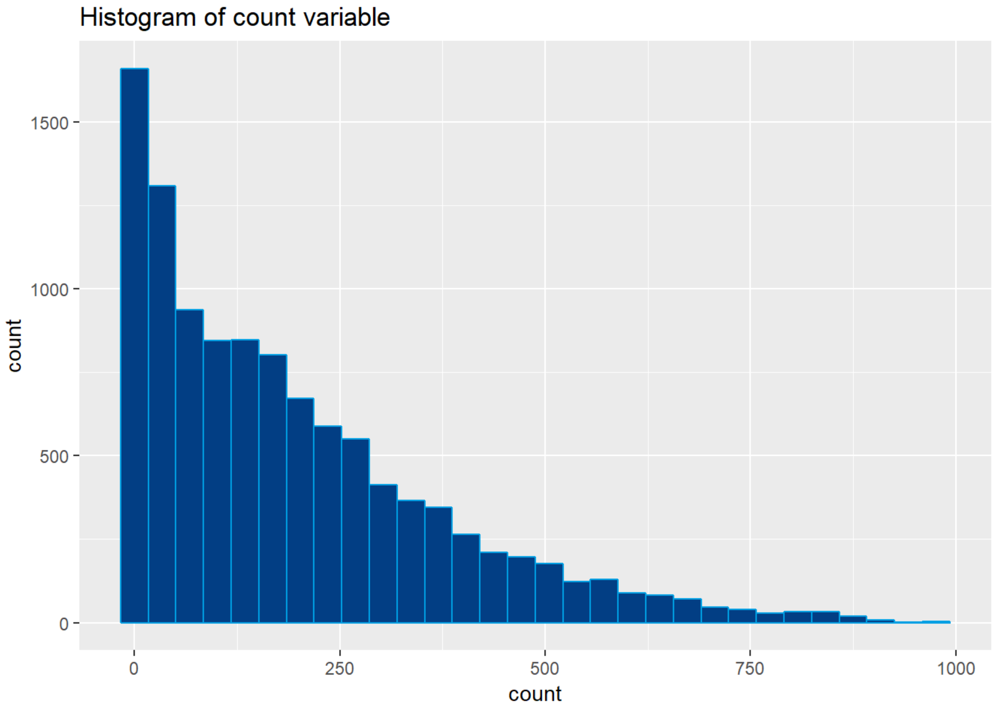
datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual
<S3: POSIXct>	<fctr>	<fctr>	<fctr>	<fctr>	<dbl>	<dbl>	<int>	<dbl>	<int>
2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	3
2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8
2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5
2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3
2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0
2011-01-01 05:00:00	1	0	0	2	9.84	12.880	75	6.0032	0
2011-01-01 06:00:00	1	0	0	1	9.02	13.635	80	0.0000	2
2011-01-01 07:00:00	1	0	0	1	8.20	12.880	86	0.0000	1
2011-01-01 08:00:00	1	0	0	1	9.84	14.395	75	0.0000	1
2011-01-01 09:00:00	1	0	0	1	13.12	17.425	76	0.0000	8

1-10 of 10,000 rows | 1-10 of 16 columns Previous **1** 2 3 4 5 6 ... 1000 Next

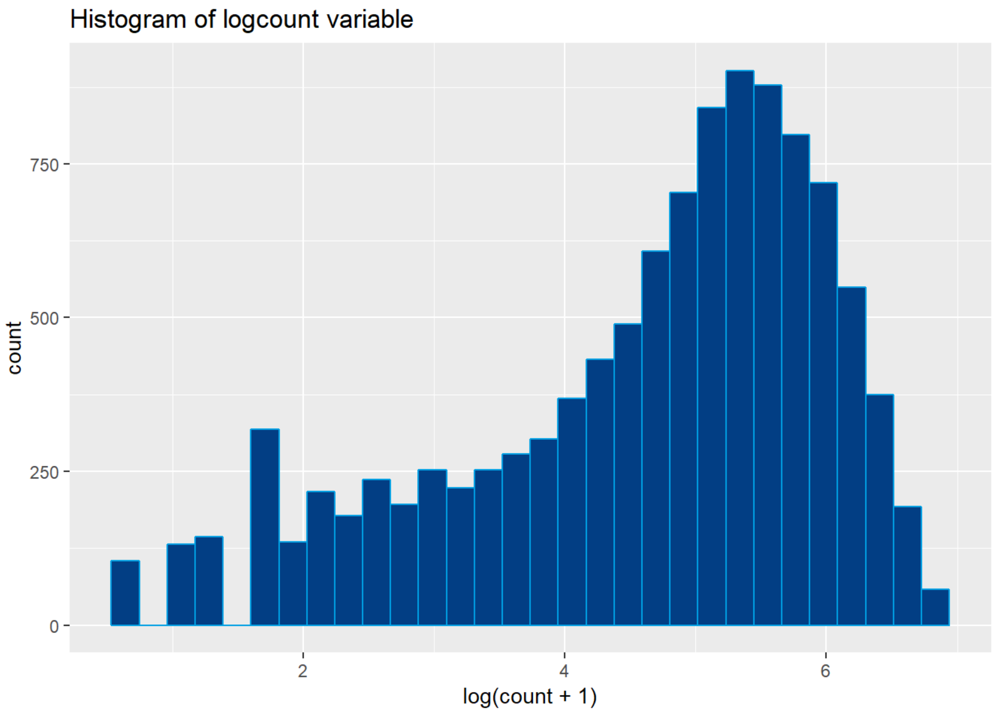
```
str(dat)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    10886 obs. of  16 variables:
##  $ datetime  : POSIXct, format: "2011-01-01 00:00:00" "2011-01-01 01:00:00" "2011-01-01 02:00:00" ...
##  $ season    : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ holiday   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ workingday: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ weather   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 2 1 1 1 1 ...
##  $ temp      : num  9.84 9.02 9.02 9.84 9.84 ...
##  $ atemp     : num  14.4 13.6 13.6 14.4 14.4 ...
##  $ humidity  : int  81 80 80 75 75 75 80 86 75 76 ...
##  $ windspeed : num  0 0 0 0 0 ...
##  $ casual    : int  3 8 5 3 0 0 2 1 1 8 ...
##  $ registered: int  13 32 27 10 1 1 0 2 7 6 ...
##  $ count     : int  16 40 32 13 1 1 2 3 8 14 ...
##  $ year      : Factor w/ 2 levels "2011","2012": 1 1 1 1 1 1 1 1 1 1 ...
##  $ month     : Factor w/ 12 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ hour      : Factor w/ 24 levels "0","1","2","3",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ logcount  : num  2.833 3.714 3.497 2.639 0.693 ...
```

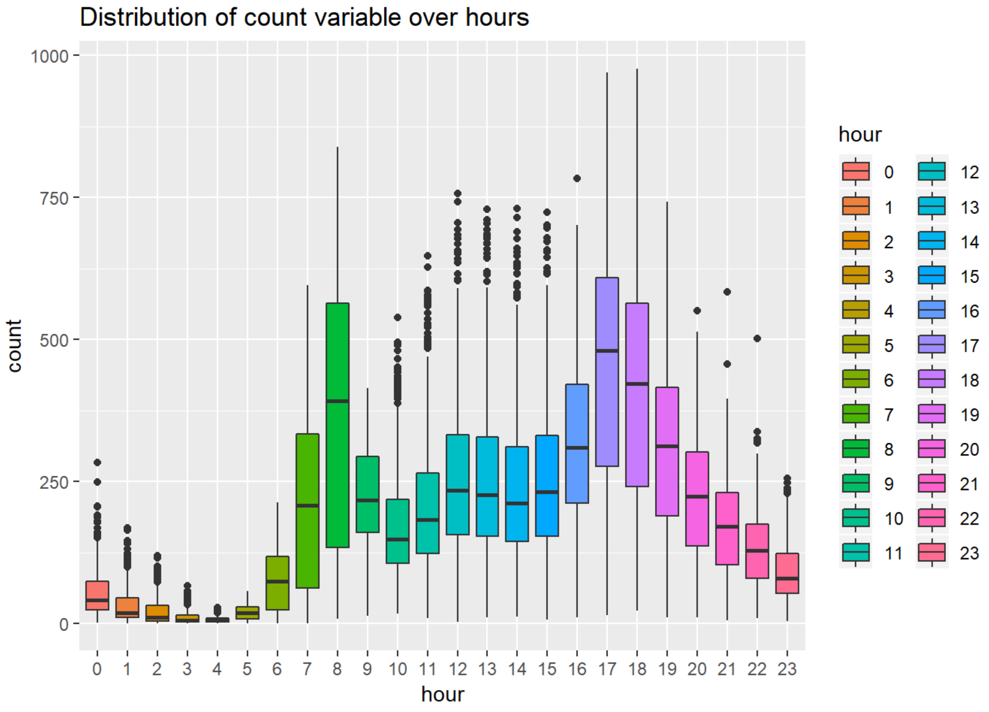
```
dat %>%
    ggplot(aes(x = count)) + geom_histogram(color=qpal[1], fill=qpal[2]) + labs(title='Histogram of count variable')
```



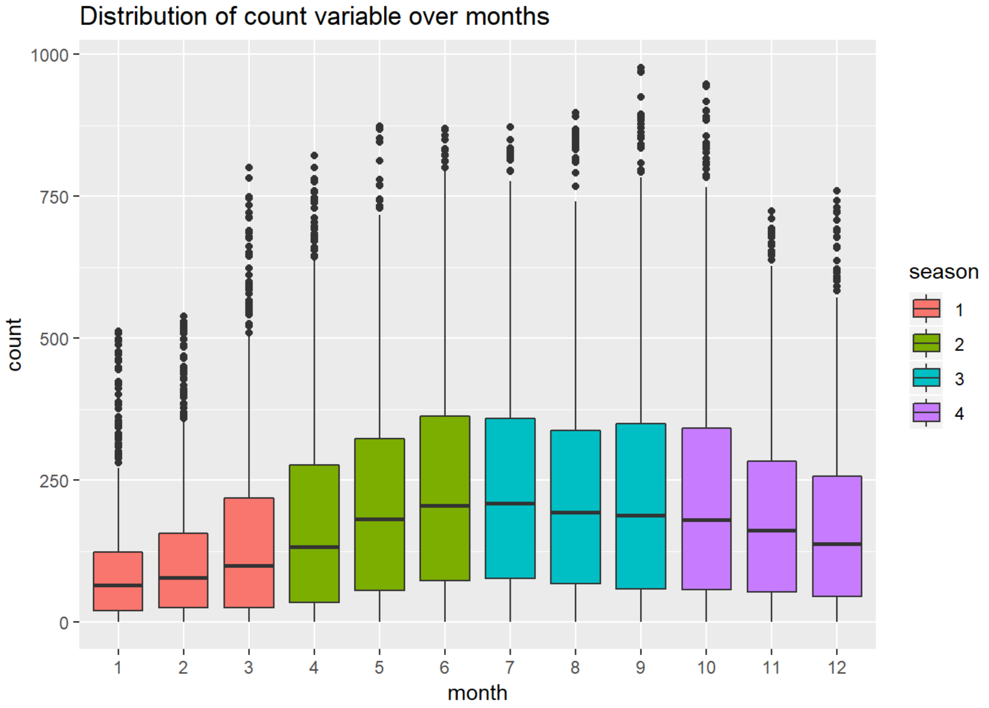
```
dat %>%
    ggplot(aes(x = log(count+1))) + geom_histogram(color=qpal[1], fill=qpal[2]) + labs(title='Histogram of logcount
 variable')
```



```
dat %>% ggplot(aes(x=hour, y=count, fill=hour)) + geom_boxplot()+
    labs(title='Distribution of count variable over hours')
```



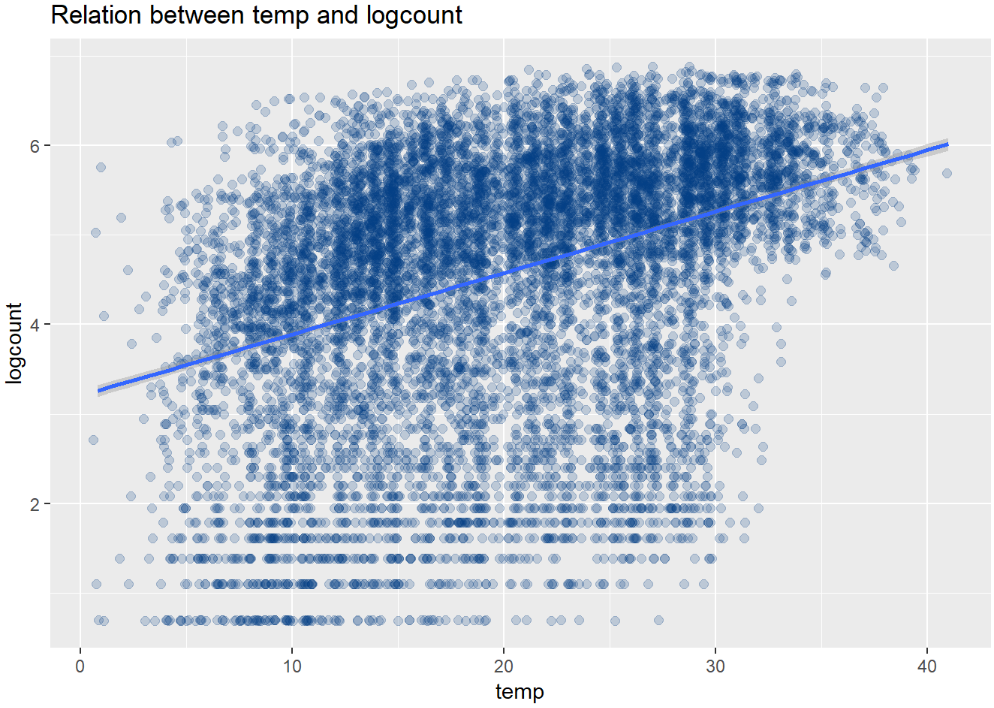
```
dat %>% ggplot(aes(x=month, y=count, fill=season)) + geom_boxplot() +
    labs(title='Distribution of count variable over months')
```



4.2 Exploratory analysis

In general we have the strong assumption, that the weather influences the count variable. We check this assumption with scatterplots.

```
dat %>%
    ggplot(aes(x = temp, y = logcount)) +
    geom_jitter(alpha = .2, color=qpal[2], size = 2) +
    geom_smooth(method='lm') +
    labs(title='Relation between temp and logcount')
```

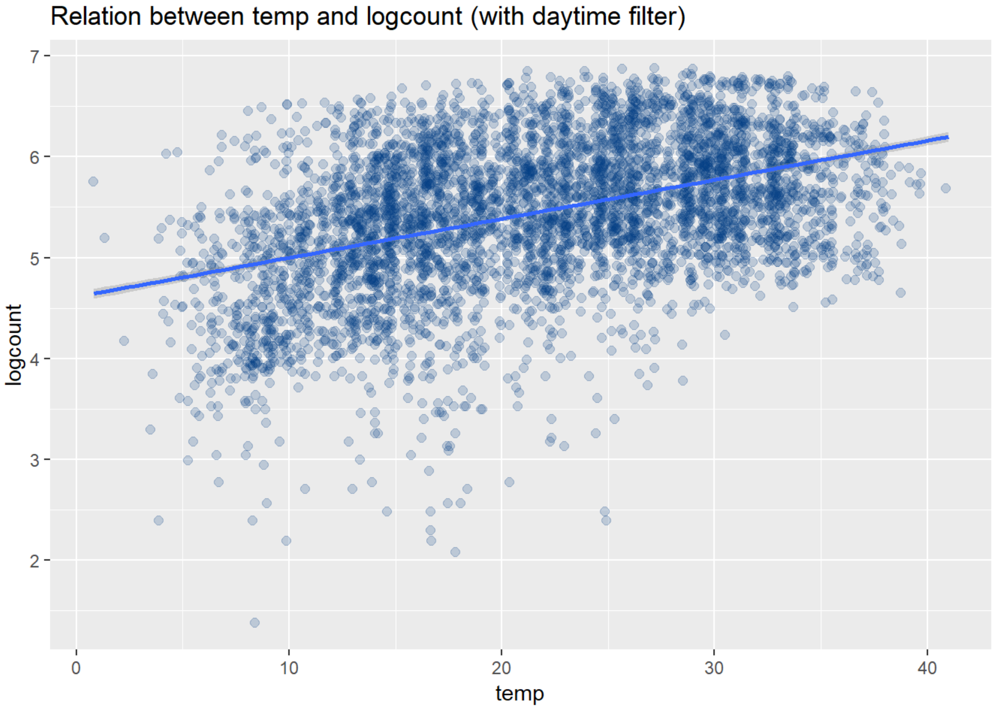


Although we see a positive relationship between **logcount** and **temp** we see a weird pattern in the lower range, where the relation seems to break.

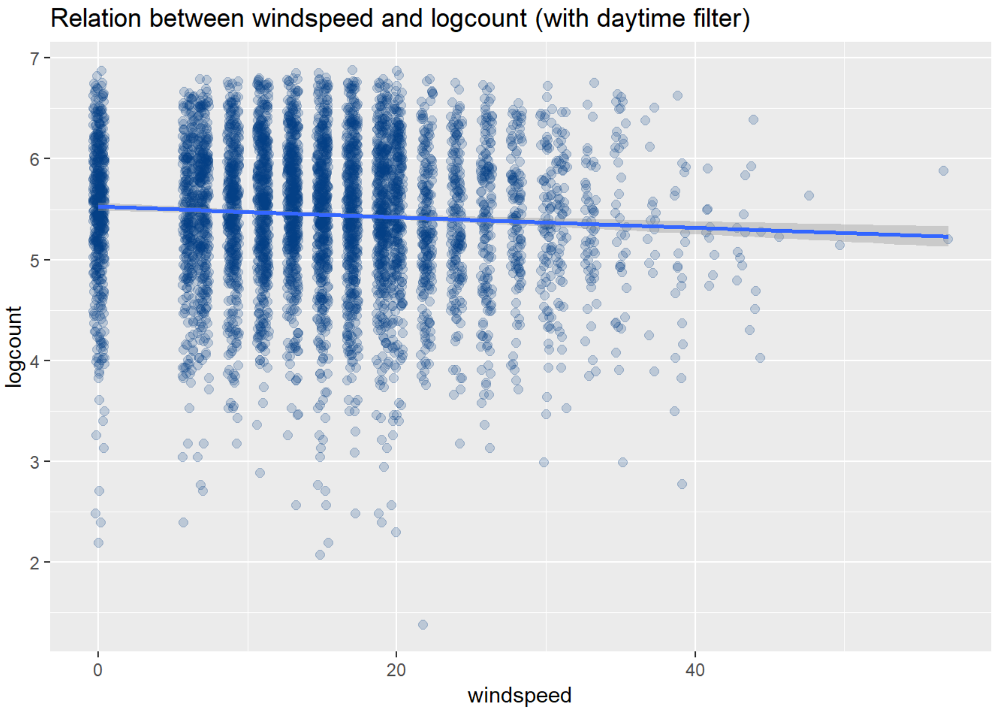
This is due to our failure to account for day and nighttime. Naturally the bike rentals during 3 am in the morning will not go up, even when it is warmer than usual.

Therefore we apply a filter to only look at hours from 8am to 20pm.

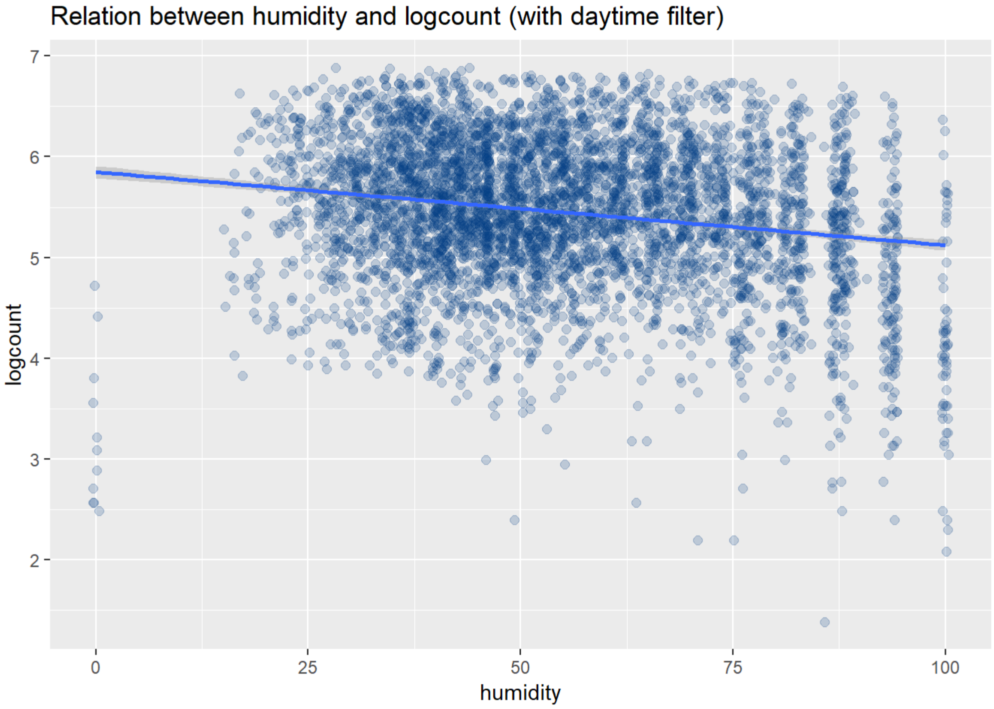
```
dat %>% filter(as.numeric(hour) > 8, as.numeric(hour) < 20) %>%
    ggplot(aes(x = temp, y = logcount)) +
    geom_jitter(alpha = .2, color=qpal[2], size = 2) +
    geom_smooth(method='lm') +
    labs(title='Relation between temp and logcount (with daytime filter)')
```



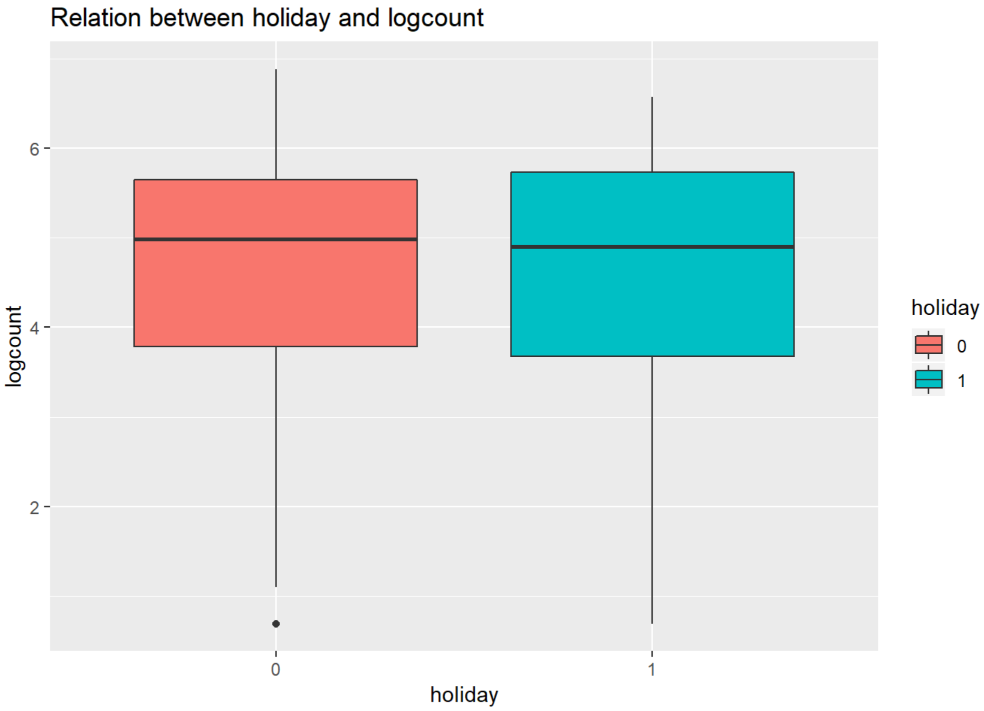
```
dat %>% filter(as.numeric(hour) > 8, as.numeric(hour) < 20) %>%
    ggplot(aes(x = windspeed, y = logcount)) +
    geom_jitter(alpha = .2, color=qpal[2], size = 2) +
    geom_smooth(method='lm') +
    labs(title='Relation between windspeed and logcount (with daytime filter)')
```



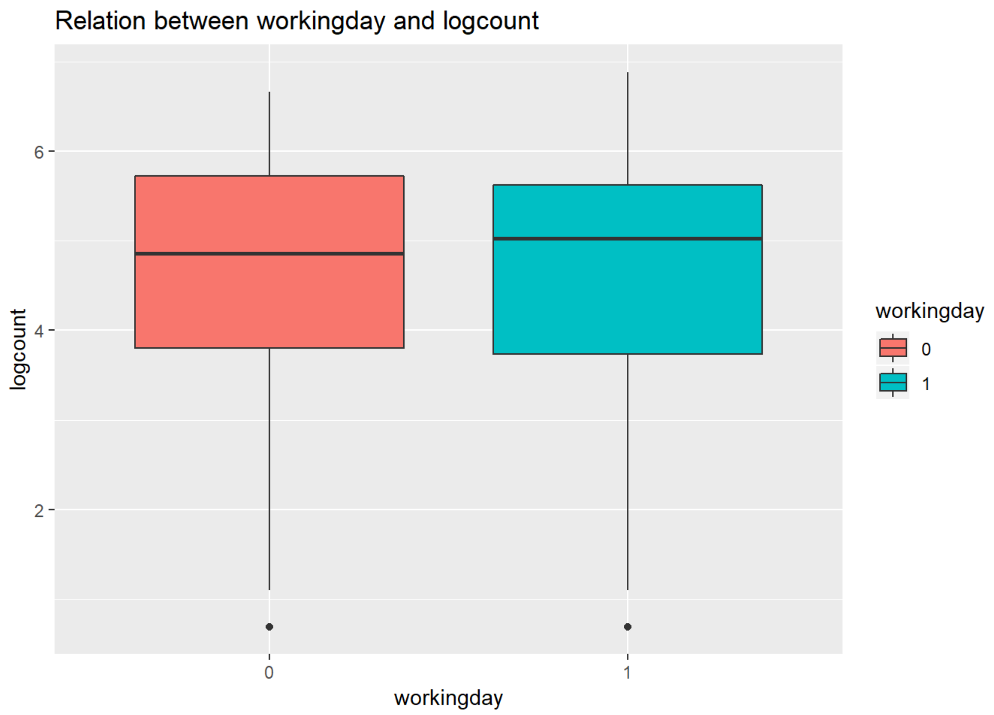
```
dat %>% filter(as.numeric(hour) > 8, as.numeric(hour) < 20) %>%
    ggplot(aes(x = humidity, y = logcount)) +
    geom_jitter(alpha = .2, color=qpal[2], size = 2) +
    geom_smooth(method='lm') +
    labs(title='Relation between humidity and logcount (with daytime filter)')
```



```
dat %>%
    ggplot(aes(fill = holiday, y = logcount, x = holiday)) + geom_boxplot() +
    labs(title = 'Relation between holiday and logcount')
```



```
dat %>%
    ggplot(aes(fill = workingday, y = logcount, x = workingday)) + geom_boxplot() +
    labs(title = 'Relation between workingday and logcount')
```



4.3 Splitting

We split our dataset with the 80/20 rule into training and testset by random sampling. This is done stratisfied on the **hour** variable, to retain an evenly distributed number of datasets per hour in both sets.

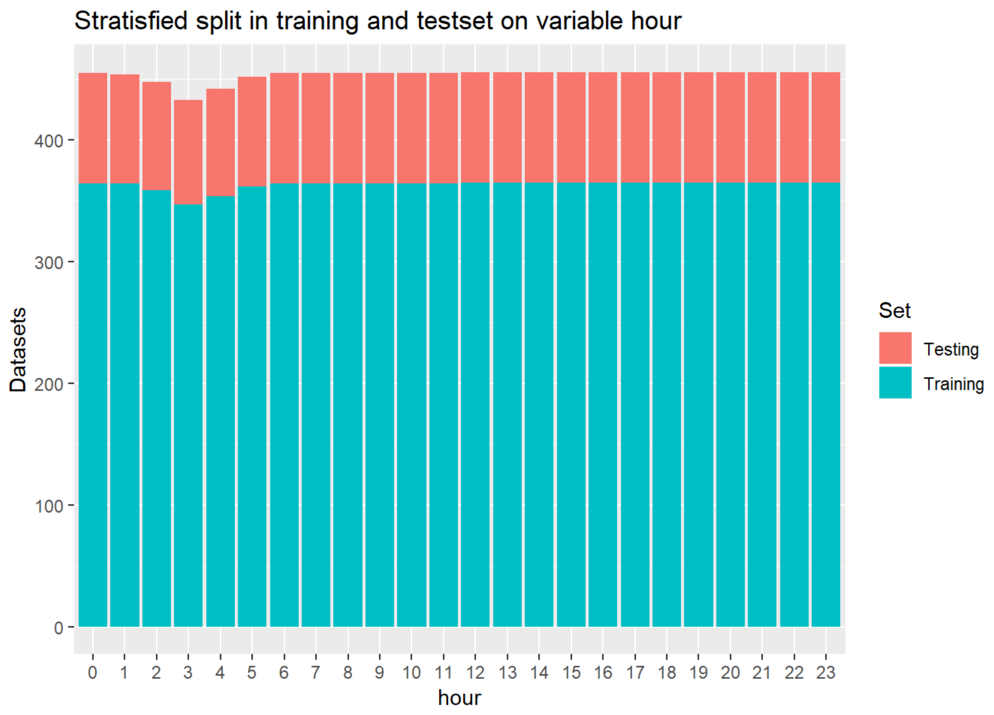
```
modeldat <- dat %>%
    select(count, logcount, year, month, hour, season:windspeed)

intrain <- createDataPartition(modeldat$hour, p = .8, list =F)

training <- modeldat[intrain,]
testing <- modeldat[-intrain,]

training
```

count	logcount	year	month	hour	season	holiday	workingday	weather	temp
<int>	<dbl>	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<dbl>
16	2.8332133	2011	1	0	1	0	0	1	9.84
40	3.7135721	2011	1	1	1	0	0	1	9.02
13	2.6390573	2011	1	3	1	0	0	1	9.84
1	0.6931472	2011	1	4	1	0	0	1	9.84
1	0.6931472	2011	1	5	1	0	0	2	9.84
2	1.0986123	2011	1	6	1	0	0	1	9.02
3	1.3862944	2011	1	7	1	0	0	1	8.20
8	2.1972246	2011	1	8	1	0	0	1	9.84
14	2.7080502	2011	1	9	1	0	0	1	13.12
36	3.6109179	2011	1	10	1	0	0	1	15.58

1-10 of 8,714 rows | 1-10 of 13 columns

Previous **1** 2 3 4 5 6 ... 872 Next

```
trainplot <- modeldat %>% mutate(Set = 'Testing')
trainplot$Set[intrain] <- 'Training'
trainplot %>% ggplot(aes(x=hour, fill=Set)) + geom_bar(stat='count') + labs(y='Datasets', title = 'Stratisfied split
 in training and testset on variable hour')
```



5 Model Training

We activate 10-fold Cross-Validation for the training procedures.

```
trControl <- trainControl(method = 'cv')
fit <- list()
```

5.1 Linear regression

```
fit$lm <- train(logcount ~ . -count,
                     data = training,
                     method = 'lm',
                     trControl = trControl)

fit$lm
```

```
## Linear Regression
##
## 8714 samples
##   12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7842, 7843, 7843, 7843, 7841, 7843, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.5817004  0.8323308  0.4361231
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

5.2 Regression tree

```
fit$rpart <- train(logcount ~ . -count,
                     data = training,
                     method = 'rpart',
                     trControl = trControl,
                     tuneLength=10)

fit$rpart
```

```
## CART
##
## 8714 samples
##   12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7843, 7843, 7844, 7842, 7842, 7843, ...
## Resampling results across tuning parameters:
##
##   cp          RMSE       Rsquared   MAE
##   0.01095069  0.8003807  0.6818636  0.6154020
##   0.01138724  0.8087604  0.6751664  0.6232886
##   0.01511527  0.8264769  0.6610514  0.6406533
##   0.02517244  0.8531976  0.6387889  0.6620359
##   0.05161965  0.8917062  0.6044305  0.6954775
##   0.08101052  0.9774125  0.5254322  0.7608012
##   0.10107544  1.0675567  0.4323570  0.8315109
##   0.10137051  1.1239711  0.3714468  0.8782893
##   0.13915391  1.2383133  0.2379279  0.9750783
##   0.15955655  1.3549408  0.1456860  1.0827115
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.01095069.
```

5.3 Bagged trees

```
fit$treebag <- train(logcount ~ . -count,
                     data = training,
                     method = 'treebag',
                     trControl = trControl)

fit$treebag
```

```
## Bagged CART
##
## 8714 samples
##   12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7843, 7843, 7844, 7842, 7842, 7842, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.7594805  0.7137251  0.5827056
```

5.4 Boosted trees

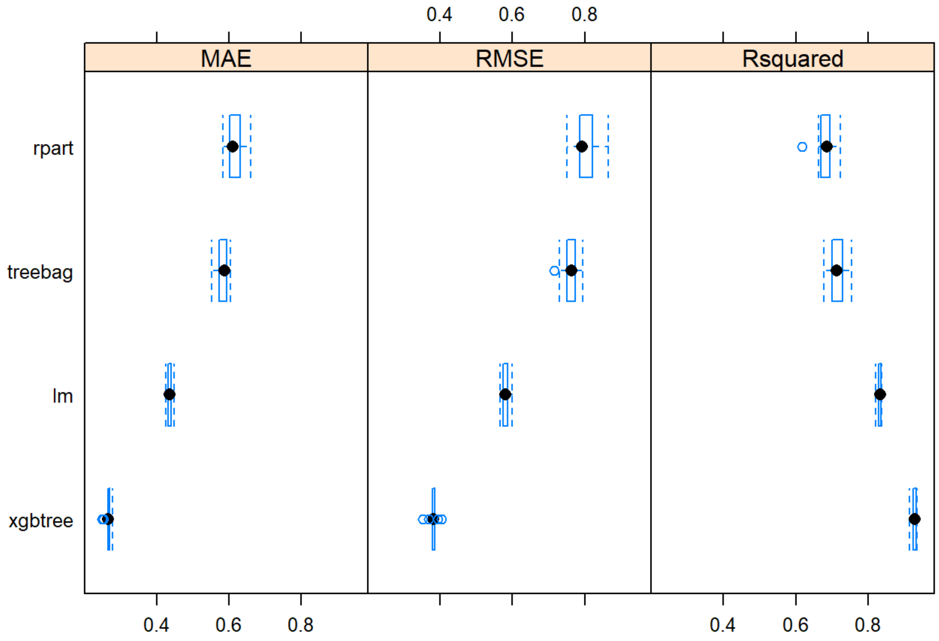
```
tuneGrid <- data.frame(nrounds = 150,
                       max_depth = 3,
                       eta = 0.4,
                       gamma = 0,
                       colsample_bytree= 0.8,
                       min_child_weight = 1,
                       subsample = 0.75)

fit$xgbtree <- train(logcount ~ . -count,
                     data = training,
                     method = 'xgbTree',
                     trControl = trControl
                     ,tuneGrid = tuneGrid
                     )

fit$xgbtree
```

```
## eXtreme Gradient Boosting
##
## 8714 samples
##   12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7843, 7843, 7843, 7843, 7843, 7841, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.3808604  0.9279323  0.2644802
##
## Tuning parameter 'nrounds' was held constant at a value of 150
## Tuning parameter 'max_depth' was held
##  value of 0.8
## Tuning parameter 'min_child_weight' was held constant at a value of 1
## Tuning
##  parameter 'subsample' was held constant at a value of 0.75
```

6 Model Evaluation

6.1 Performance on training set

```
rs <- resamples(fit)
bwplot(rs)
```

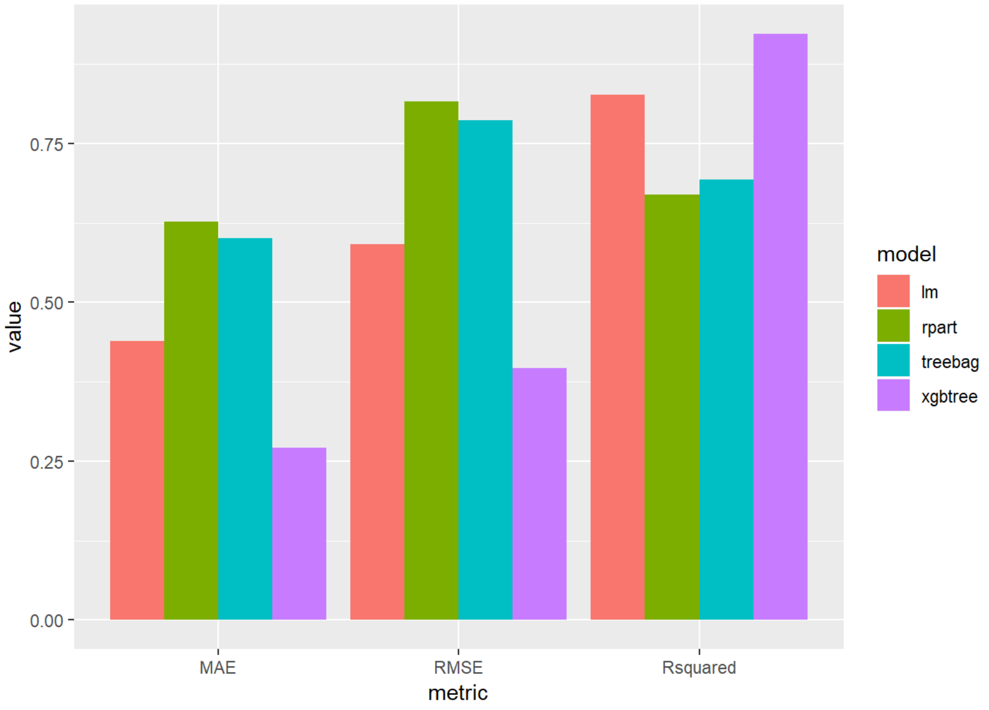


6.2 Performance on test set

We apply the prediction of all models on the test set and compare performance measures. It is important to note, that in order to avoid overfitting on the test set, a selection for the “best” model should already have been made on the basis of the cross-validated training results.

Applying all models on the test set is only shown for demonstrative purposes here and is definitely not best practive.

```
prediction <- lapply(fit, predict, newdata = testing) %>%
    bind_cols()

prediction %>%
    sapply(postResample, obs = testing$logcount) %>%
    as.data.frame() %>%
    rownames_to_column('metric') %>%
    gather(model, value, -metric) %>%
    ggplot(aes(x=metric, y = value, fill = model)) +
    geom_col(position='dodge')
```

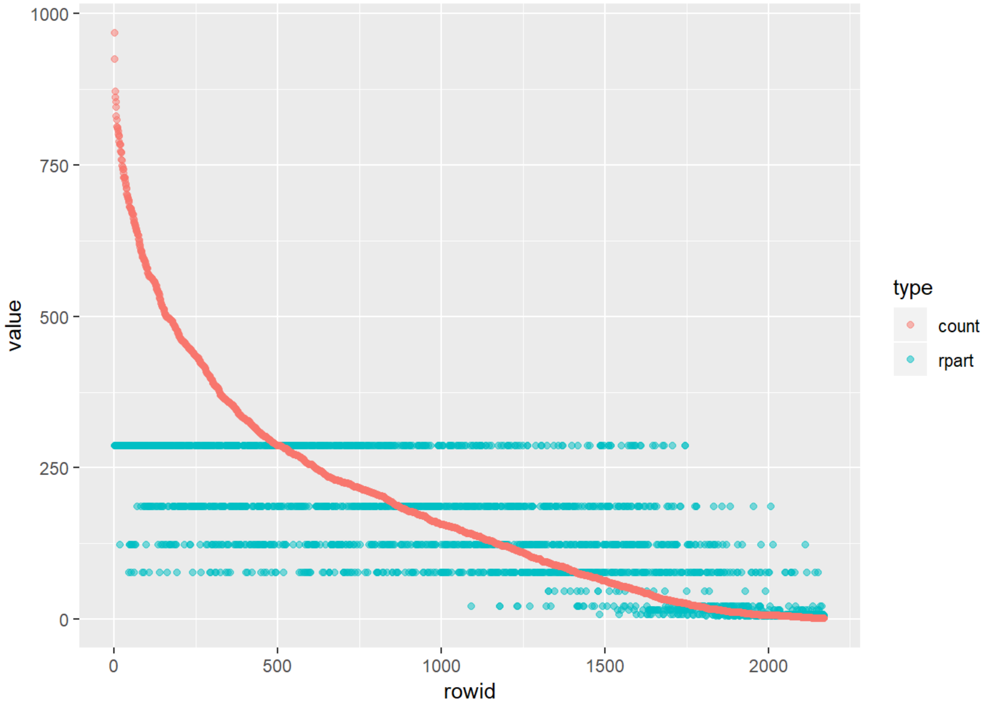


```
##                 lm     rpart   treebag   xgbtree
## RMSE     0.5914255 0.8159588 0.7870897 0.3960307
## Rsquared 0.8267350 0.6698294 0.6928026 0.9223947
## MAE      0.4396355 0.6266494 0.6014541 0.2711873
```

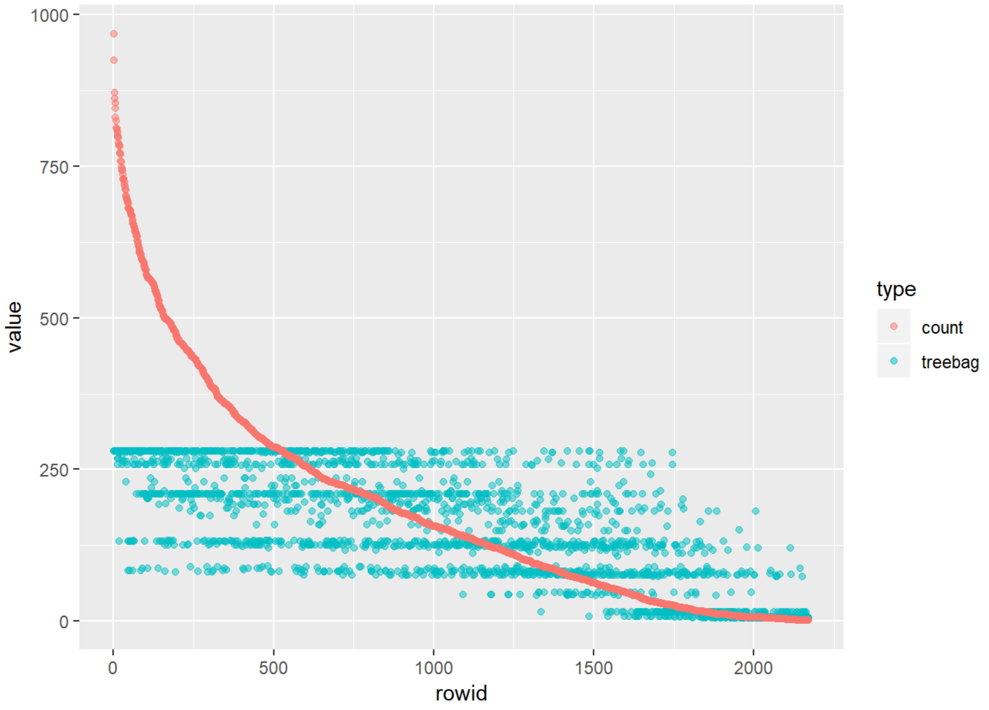
Also on the test set the boosted trees are performing best.

```
# Model fit on training
plotdat <- prediction %>%
    mutate_all(function(x) exp(x) - 1) %>%
    bind_cols(testing %>% select(count)) %>%
    arrange(desc(count)) %>%
    rowid_to_column() %>%
    gather(type, value, -rowid)
```

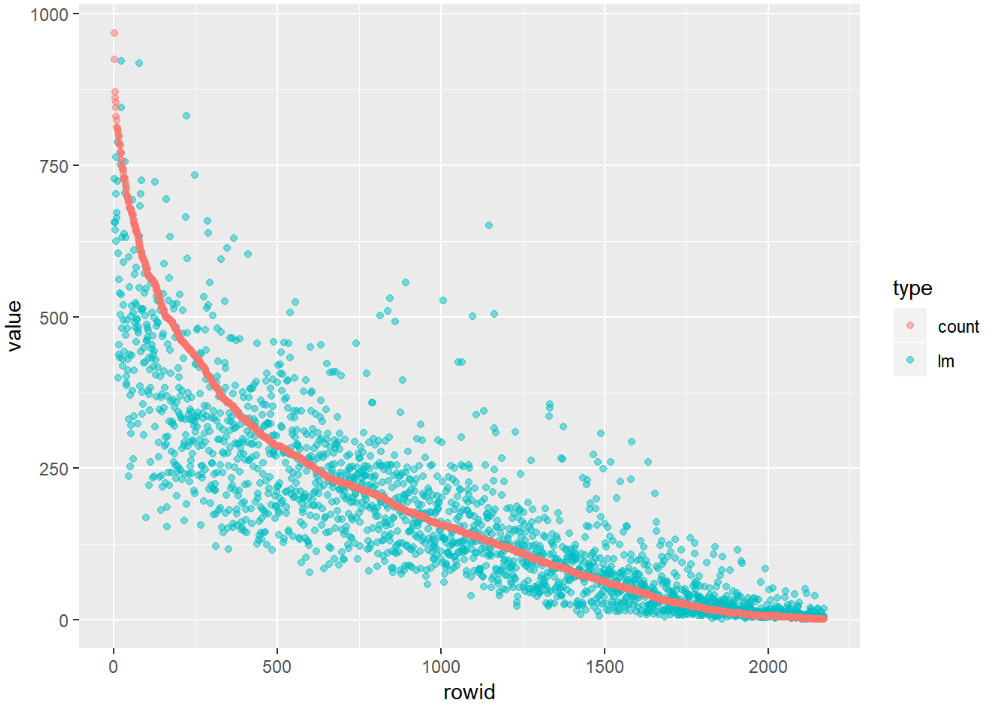
```
plotdat %>% filter(type %in% c('count', 'rpart')) %>%
    ggplot(aes(x=rowid, y=value, color=type)) + geom_point(alpha =.5)
```



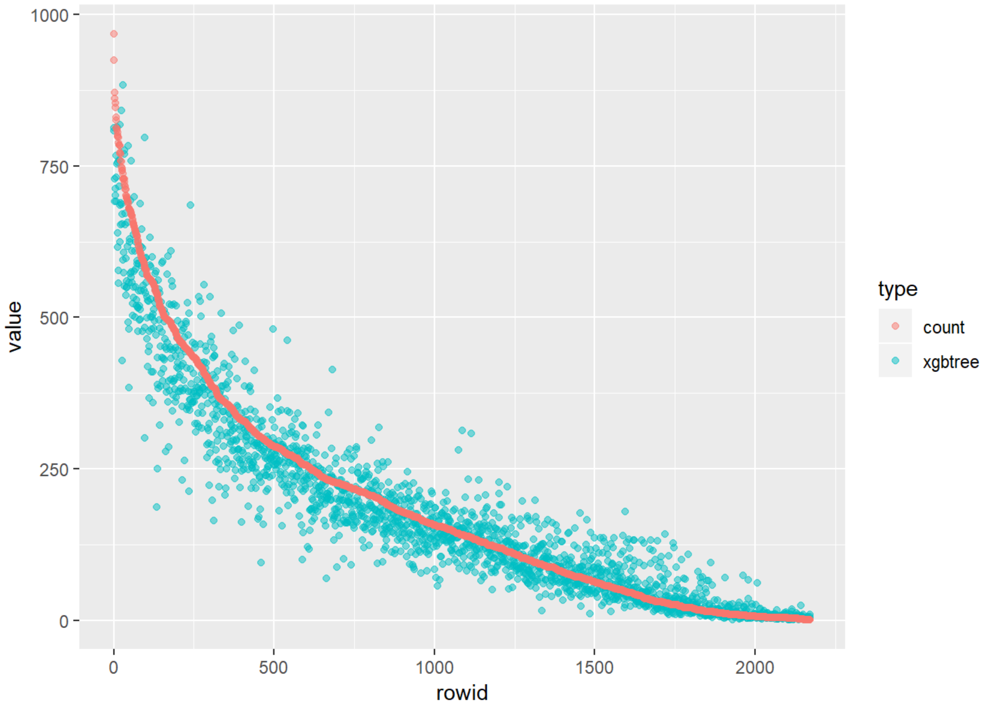
```
plotdat %>% filter(type %in% c('count', 'treebag')) %>%
    ggplot(aes(x=rowid, y=value, color=type)) + geom_point(alpha =.5)
```



```
plotdat %>% filter(type %in% c('count', 'lm')) %>%
    ggplot(aes(x=rowid, y=value, color=type)) + geom_point(alpha =.5)
```



```
plotdat %>% filter(type %in% c('count', 'xgbtree')) %>%
    ggplot(aes(x=rowid, y=value, color=type)) + geom_point(alpha =.5)
```



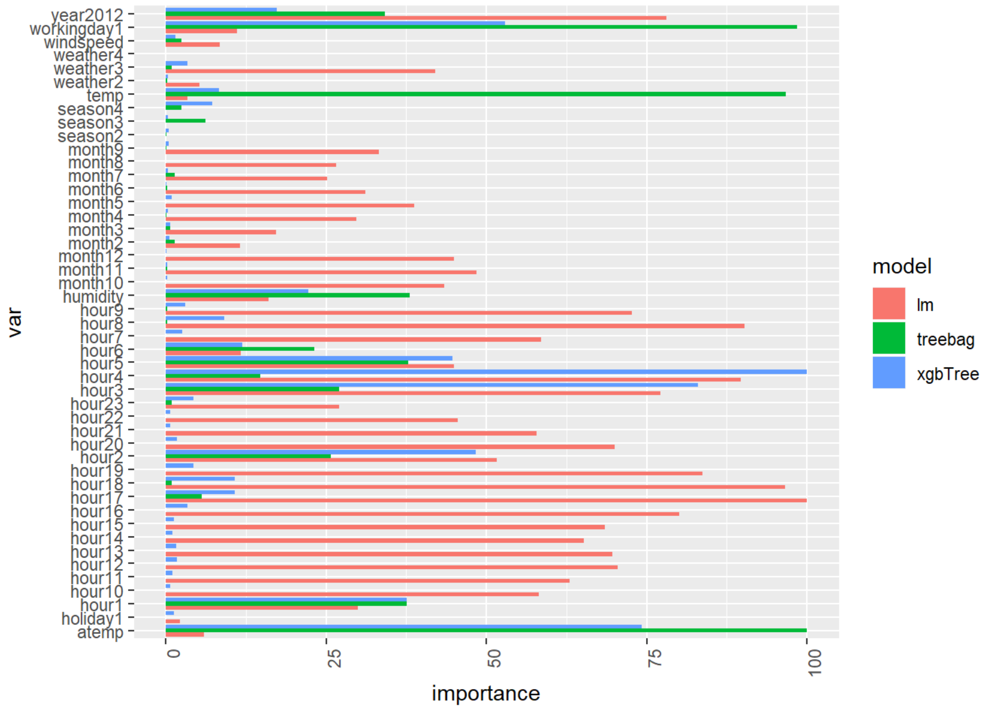
6.3 Variable importance

We can extract and plot the relative importance each model attributes to all the available variables.

The differences are quite striking.

```
vi <- lapply(fit, function(x)
    varImp(x)$importance %>%
        rownames_to_column('var') %>%
        set_names(c('var', x$method))
    )

vi$lm %>% full_join(vi$xgbtree) %>% full_join(vi$treebag) %>%
    gather(model, importance, -var) %>% arrange(var) %>%
    ggplot(aes(x=var, y = importance, fill=model)) + geom_col(position='dodge') +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    coord_flip()
```

```
## Warning: Removed 3 rows containing missing values (geom_col).
```



```r
varImp(fit$lm)
varImp(fit$rpart)
varImp(fit$treebag)
varImp(fit$xgbtree)
```