Assignment R Bootcamp

Tu Tran and Andy Gubser

06 March 2020

Contents

Abstract

The purpose of this project is to estimate hourly demand for shared bikes in New York City. In particular, we explore first the provided bike sharing in dependency on weather data and dates. Then we create different models for the prediction and define the best model with the least prediction error rate.

Methodology

The analysis bases on the bike sharing data sets for New York City in 2016 from Kaggle [^1 https://www.kaggle.com/samratp/bikeshare-analysis#NYC-CitiBike-2016.csv] and the corresponding weather data from the Weather.com API [^2 https://api.weather.com/v1/location/KLGA:9:US/observations/historical.json].

After data preparation, we provide an graphical analysis to approach the research question. Then, we fit a poisson model, high level polynomial linear model followed by a random forest model to deepen the analysis.

Analysis

```
# remove all objects loaded and clear memory
rm(list = ls(all.names = TRUE))
gc()
```

```
##           used (Mb) gc trigger (Mb) max used (Mb)
## Ncells  534439 28.6    1216302   65   621331 33.2
## Vcells 1009425  7.8    8388608   64  1600107 12.3
```

```
library(checkpoint)
```

```
##
## checkpoint: Part of the Reproducible R Toolkit from Microsoft
## https://mran.microsoft.com/documents/rro/reproducibility/
```

```
#checkpoint(snapshotDate = "2099-12-29")

knitr::opts_knit$set(root.dir = rprojroot::find_rstudio_root_file())
knitr::opts_chunk$set(echo=TRUE)
set.seed(19)
```

```
## user defined functions and constants ##

## holidays in New York City in 2016
public_holidays <- c("2016-01-01", "2016-01-18", "2016-02-12", "2016-02-15",
                     "2016-05-30", "2016-07-04", "2016-09-05", "2016-10-10",
                     "2016-11-11", "2016-11-24", "2016-12-26")

## plot bike rental count over 24 hours for different category groups
## @param d.data: expects properties: category, hour, rental
## @param title: the title of the plot
## @param category_name: the category group name used as the legend's name in the plot
show_24h_category_statistics_plot <- function (d.data, title, category_name) {
  ggplot(d.data, aes(x=hour, y=rental, color=category)) +
    geom_point(data=d.data, aes(group=category)) +
    geom_line(data=d.data, aes(group=category)) +
    ggtitle(title) + ylab("rentals / hour") +
    scale_color_hue(category_name, breaks=levels(d.data$category))
}
```

Process bike sharing data (found in process_bike_dataset.R)

The bike data set consists of 276'798 observation and 15 variables. These variables include the two categories user type and gender as well as the numeric birth year. User type defines if the user is a registered subscriber

of the bike sharing service or a casual user. Gender says if the user is female, male or unknown. The continous variable age is calculated from the birth year.

Further, the data set include start and stop time stamps as well as trip duration of individual trips, station names, ids and its coordinates. By checking for outliers, one observation was detected to have meaningless coordinates. These observations are excluded from further analysis. The prepared data set includes 242'746 observations and 21 variables.

Process weather data (found in process_weather_dataset.R)

The weather data set has hourly weather records for 366 days in 2016 for New York City. The weather record includes the measurements for date, time, temperature, dew point, humidity, windspeed and weather condition (e.g. Cloudy, Mostly Cloudy, Rainy, Foggy etc.).

Read bike and weather dataset and merge for the analysis

```
## read bike and weather data sets
d.bike <- readRDS("./data/d.bike.rds")
colnames(d.bike)
```

```
## [1] "date"         "month"        "weekday"      "hour"         "tripduration"
## [6] "rental_count"
```

```
dim(d.bike)
```

```
## [1] 8232    6
```

```
d.weather <- readRDS("./data/d.weather.rds")
colnames(d.weather)
```

```
## [1] "date"        "hour"        "temperature" "dewpoint"    "humidity"
## [6] "windspeed"   "type"
```

```
dim(d.weather)
```

```
## [1] 8784    7
```

```
## merge both bike and weather data for analysis
d.total <- merge(d.bike, d.weather, all.x=TRUE)
# order by data and hour
d.total <- d.total[with(d.total, order(date, hour)),]
d.total$weekday <- factor(d.total$weekday,
                          levels = c("Montag", "Dienstag", "Mittwoch",
                                     "Donnerstag", "Freitag", "Samstag", "Sonntag"))
# reset row index to normal
rownames(d.total) <- NULL
```

Public holiday data analysis

```
## public holiday analysis ##

## rental statistics in weekdays
rentweekday <- d.total %>%
  group_by(weekday) %>%
  summarise(rental=round(sum(rental_count)/52),
```

```
            tripduration=round(sum(tripduration)/52)) %>% # 52 weeks of a year
  arrange(weekday)
rentweekday
```

```
## # A tibble: 7 x 3
##   weekday    rental tripduration
##   <fct>       <dbl>        <dbl>
## 1 Montag        757        11408
## 2 Dienstag      815        11661
## 3 Mittwoch      858        12562
## 4 Donnerstag    852        12407
## 5 Freitag       796        12754
## 6 Samstag       641        12056
## 7 Sonntag       603        11329
```

```
## rentals statistics in public holidays
rentholiday <- d.total %>%
  filter(as.character(date) %in% public_holidays) %>%
  group_by(date) %>%
  summarise(rental=sum(rental_count),
            tripduration=sum(tripduration)) %>%
  arrange(date)
rentholiday
```

```
## # A tibble: 11 x 3
##    date       rental tripduration
##    <date>      <int>        <dbl>
##  1 2016-01-01    205         3780
##  2 2016-01-18    249         2730
##  3 2016-02-12    323         8665
##  4 2016-02-15    117         1129
##  5 2016-05-30    601        12632
##  6 2016-07-04    689        13326
##  7 2016-09-05    910        15916
##  8 2016-10-10   1017        16348
##  9 2016-11-11    956        13598
## 10 2016-11-24    249         3641
## 11 2016-12-26    215         4297
```

```
## As we can see, the rental in public holidays are different from the weekdays and
## among themselves. Public holiday data can be used for the model fitting if we
## would like to predict for a different year.
## However, only data from 2016 are used for both the train and test data sets.
## Droping the public holiday data would reduce the variance in the model.

## remove public holiday dataset
d.total <- filter(d.total, !as.character(date) %in% public_holidays)
```

Make a copy of the analysis data and store in a file

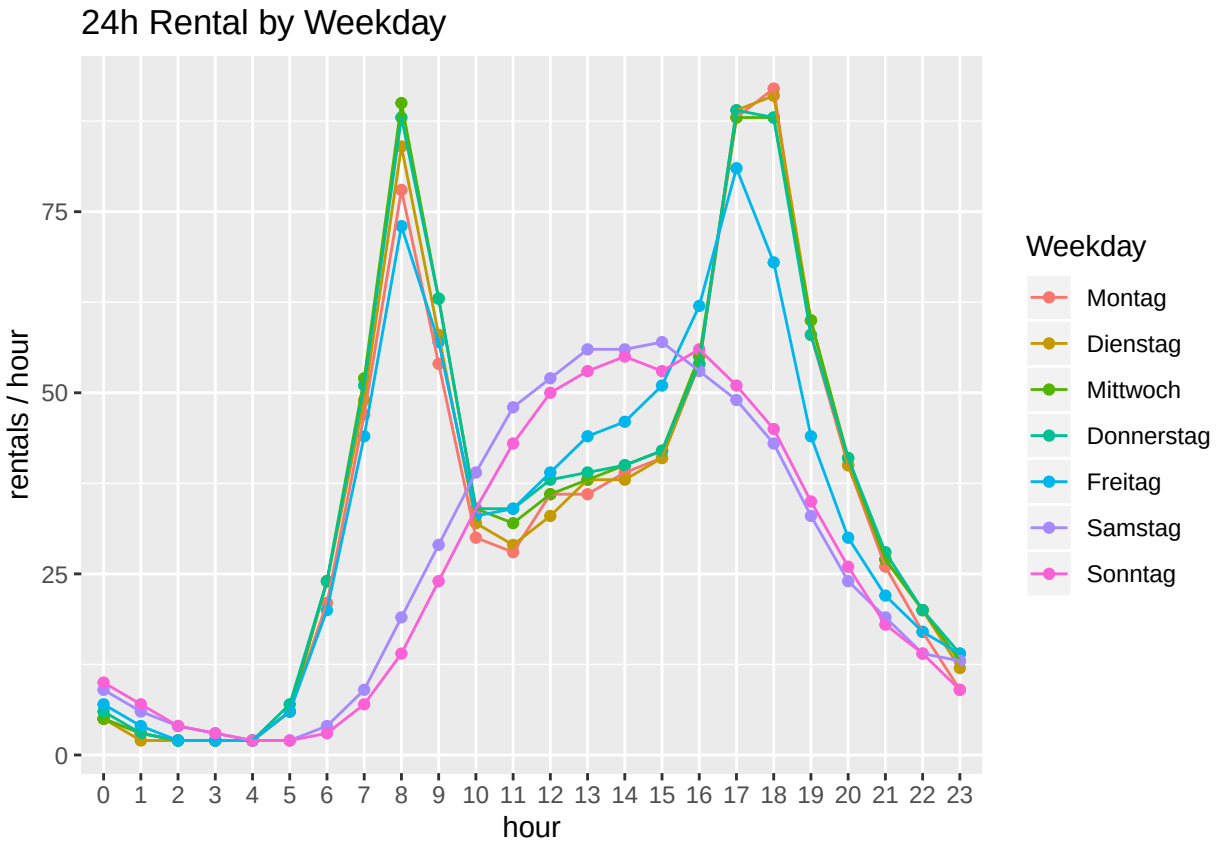
```
## save processed data to file
saveRDS(d.total, file = "./data/d.total.rds")
```

Exploratory Data Analysis, Graphical Analysis

ggplot rentals over 24h for different weekdays

The plot shows the usage pattern of shared bikes in New York City. It is clear that there are two different usage patterns for working days and weekend. On working days, we can see two peaks during the rush hours. One is at about 8:00 AM when people go to work and another one is between 5:00 PM and 6:00 PM when they finish work and go back home. During the weekends, people are more relaxing and may take longer sleep. The number of bikes goes slow up hill, reach its top at around 2:00 PM and get smoothly down hill to the end of the day. The least bike usage period are at round 1:00 AM to 5:00 AM, with its minimum at 4:00 AM.

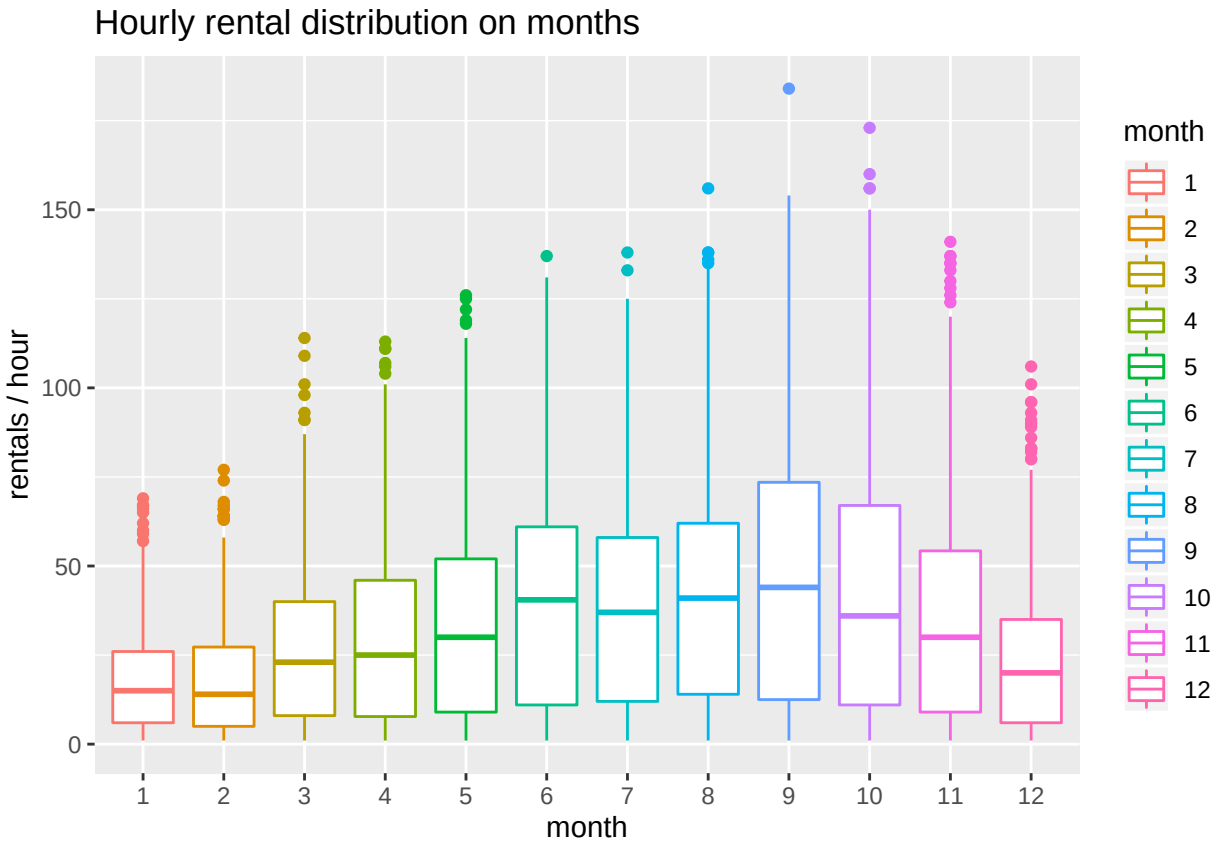
```
# ggplot 24h by weekday
d.weekday <- d.total %>% group_by(weekday, hour) %>%
  summarise(rental = round(mean(rental_count))) %>%
  rename(category = weekday)

show_24h_category_statistics_plot(d.weekday, "24h Rental by Weekday", "Weekday")
```



ggplot rentals over 24h for different months

As illustrated on the box plot, overall, the rental low season is in winter and the high season is in autumn. The lowest months are January and February with a median rental per hour for only about 15. From March to Juni, there is an increasing trend in the rental. July witnesses a small drop, but then going upward to reach its peak in September. At its peak, the trend drops about 10% to 15% over months still January. The rental trend could possibly be explained with the temperature and climate over the months. When it is colder, there are less rentals. When it is getting warmer or cooler, the trend increases. When it is much hotter, e.g. mid-sommer, there is a drop in trend. And people enjoy biking at cool and fair weather the most.

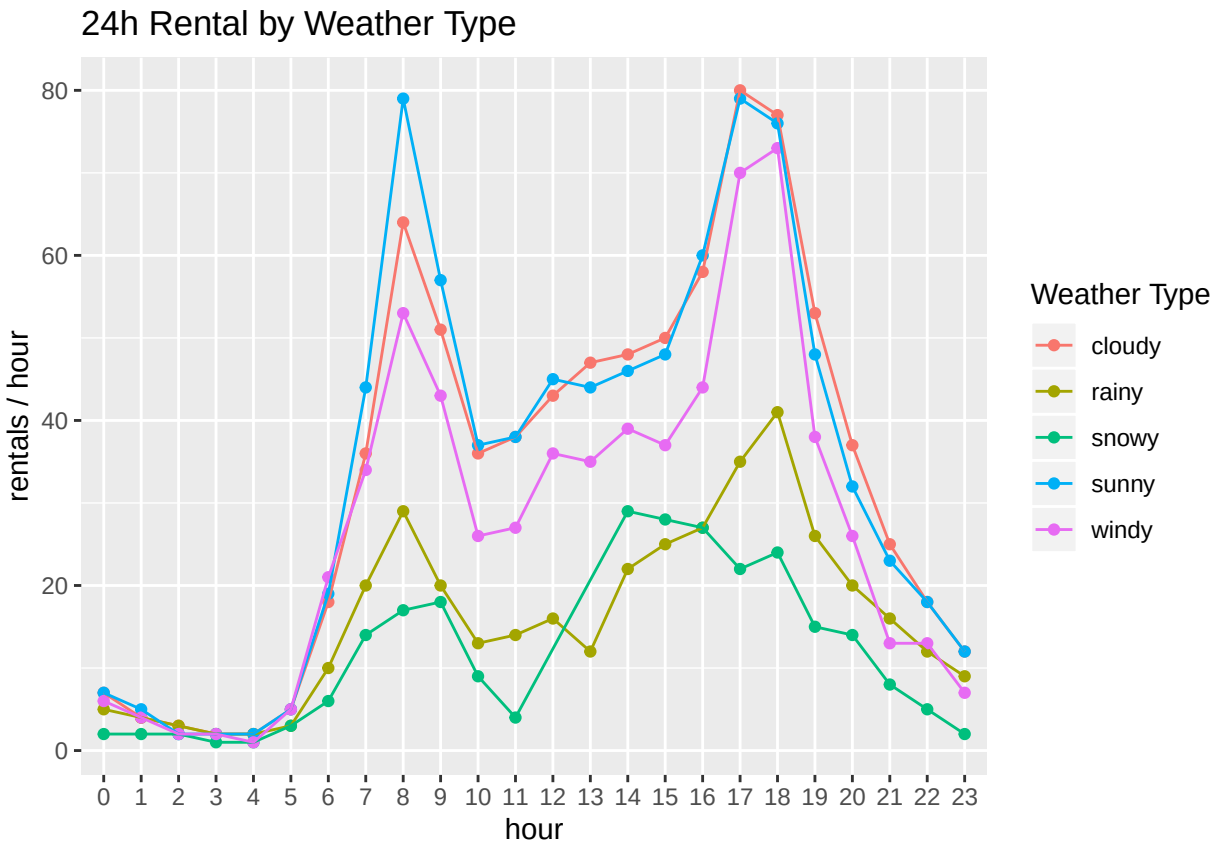
```
ggplot(d.total, aes(x=month, y=rental_count, color=month)) +
  geom_boxplot(data=d.total, aes(group=month)) +
  ggtitle("Hourly rental distribution on months") + ylab("rentals / hour")
```



ggplot rentals over 24h for different months

With this figure, we can see the highest demands on sunny and cloudy days with its peak at 80 rental at 8:00 AM or 5:00 PM. There is a drop about 10% to 20% in demand when it is windy. The need for bike is dramatically drop more than a half for rainy and snowy weather as compared to a sunny day. Here, we could again easily recognize the usage pattern during rush hours as described in the first plot.

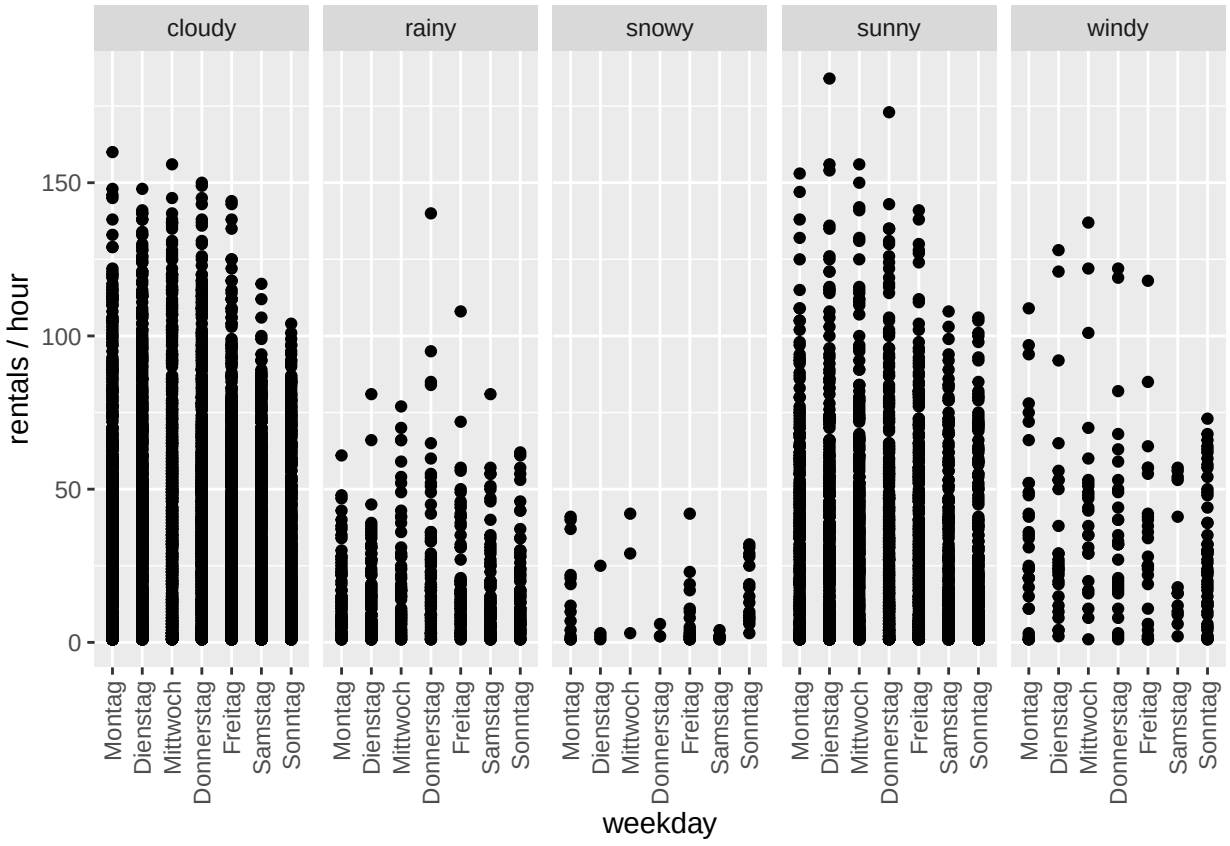
```
# ggplot 24h by weather types
d.wtype <- d.total %>% group_by(type, hour) %>%
  summarise(rental = round(mean(rental_count))) %>%
  rename(category = type)
show_24h_category_statistics_plot(d.wtype, "24h Rental by Weather Type", "Weather Type")
```



ggplot for the rental of the weekdays with an influence of the weather factor

In general, the usage pattern of the weekdays are stable, but then be strongly influenced and changed by the different weather conditions. The cloudy and sunny days has similar pattern. However, snowy, rainy and windy weather have strong impact and affect to change its usage pattern noticeably.

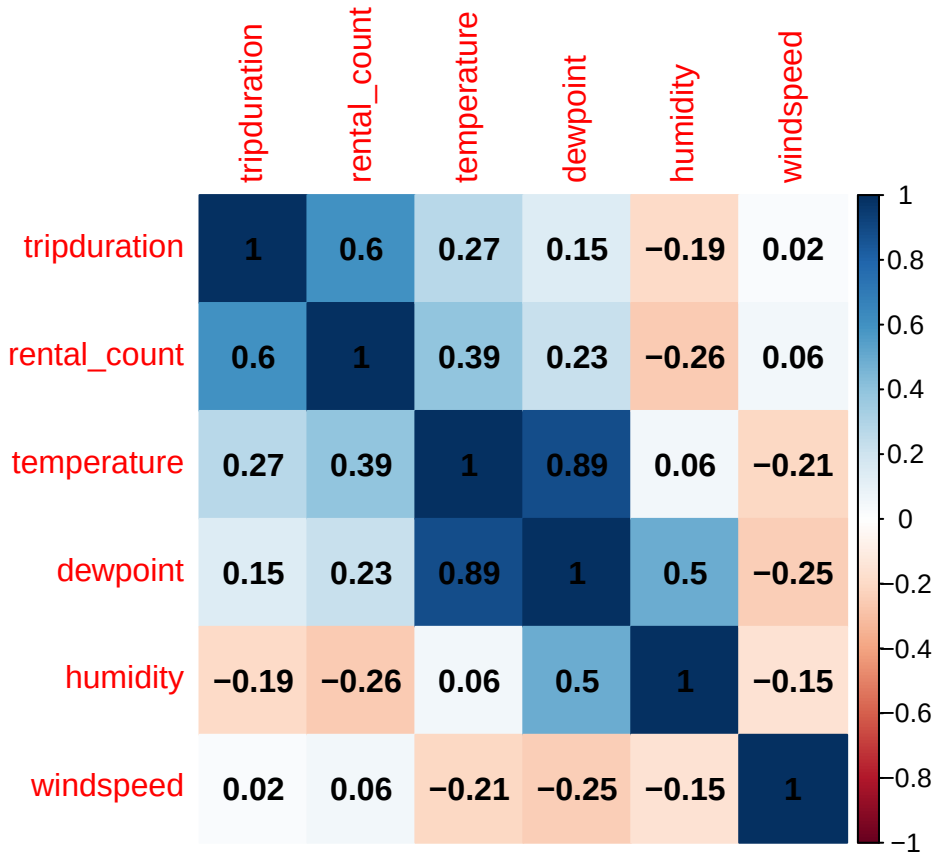
```
## the rental of the weekdays with an influence of the weather
ggplot(d.total, aes(x=weekday, y=rental_count)) +
  geom_point() +
  ylab("rentals / hour") +
  facet_grid(. ~ type) +
  ggpubr::rotate_x_text()
```



ggplot heatmap for correlation analysis between numeric predictors

The below heatmap shows some strong correlation between several predictors. High correlations can be seen for the pairs of dew point and the temperature, dew point and humidity, tripduration and rental_count. Interestingly, there is a moderate correlation between temperature and tripduration, but not for temperature and rental count, even though, tripduration and rental_count are highly correlated. These information are used to decide which predictors will be considered for the model fitting. High correlated predictors should not be used together as predictors in a model.

```
## correlation plot
d.total[,5:10] %>% cor() %>% corrplot(method = 'color', addCoef.col="black")
```



Fitting models

We would like to fit different models for predicting the hourly bike rental demand in New York City. In this section, we will apply different machine learning regression models on the data sets. Furthermore, we compare the performances of these different model based on the mean squared error and comment on the outcomes.

A simple Poisson model

As the predicted amount of bike rental count data, we will use the Poisson model for the fitting.

```
## split a train and a test set
set.seed(1)
d.split <- sample.split(d.total, SplitRatio = 0.7)
d.train <- subset(d.total, subset = d.split)
d.test <- subset(d.total, subset = !d.split)

# Poisson model with all possible predictors
pm.fit <- glm(rental_count ~ . - rental_count - tripduration - date,
              family="poisson", data=d.train)

# predict model on test data
pm.pred <- predict(pm.fit, newdata = d.test, type="response")
# compute MSE
pm.mse <- mean((d.test$rental_count - pm.pred)^2)
cat("Complex Poisson model MSE: ", pm.mse)
```

```
## Complex Poisson model MSE:  181.9917
```

Residual Analysis The residuals vs Fitted figure shows that the smoother red line is on zero. Therefore, we may conclude that the model equation is correct. The Normal Q-Q plot shows the residual standard deviance is normal distribution almost on the diagonal line. We may conclude that the Poisson model is an appropriate choice.

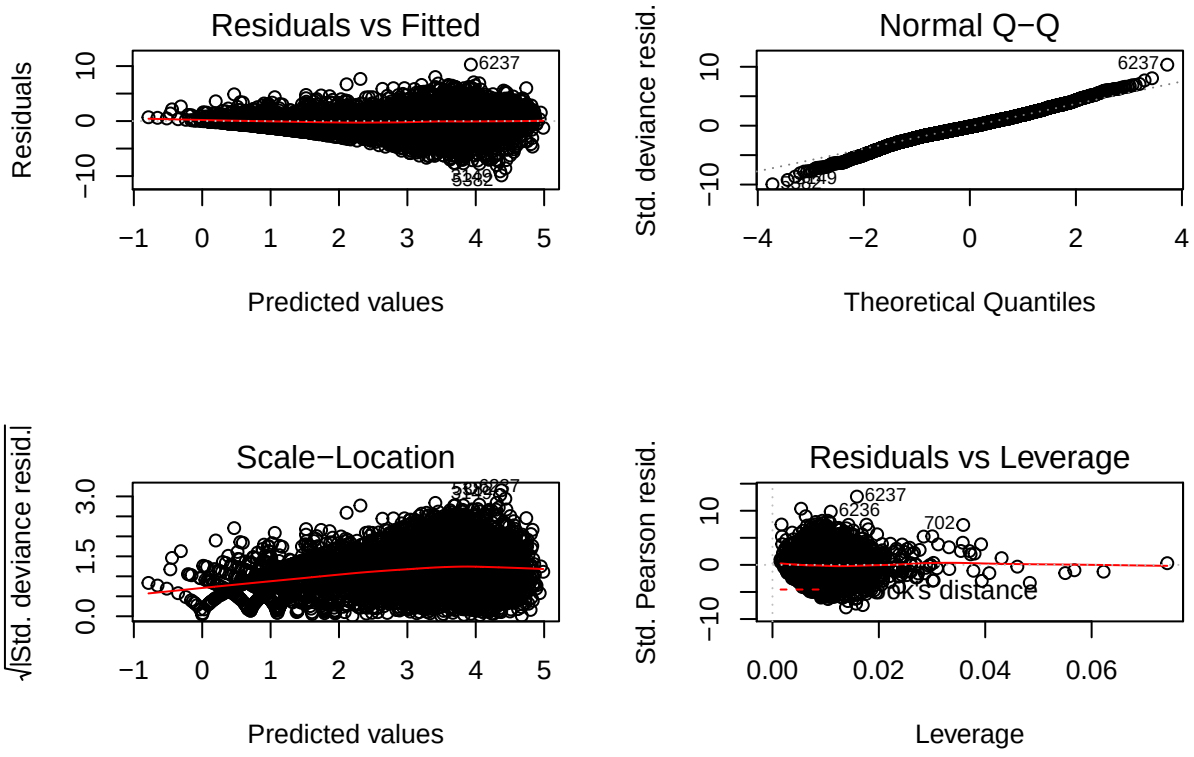
```
# residual
resid_values <- resid(pm.fit)
cat("Residual length: ", length(resid_values), "\n")
```

```
## Residual length:  5082
```

```
cat("Residual header: ", head(resid_values), "\n")
```

```
## Residual header:  -0.7884204 -0.8488808 0.6034056 -0.8801195 2.457451 1.342713
```

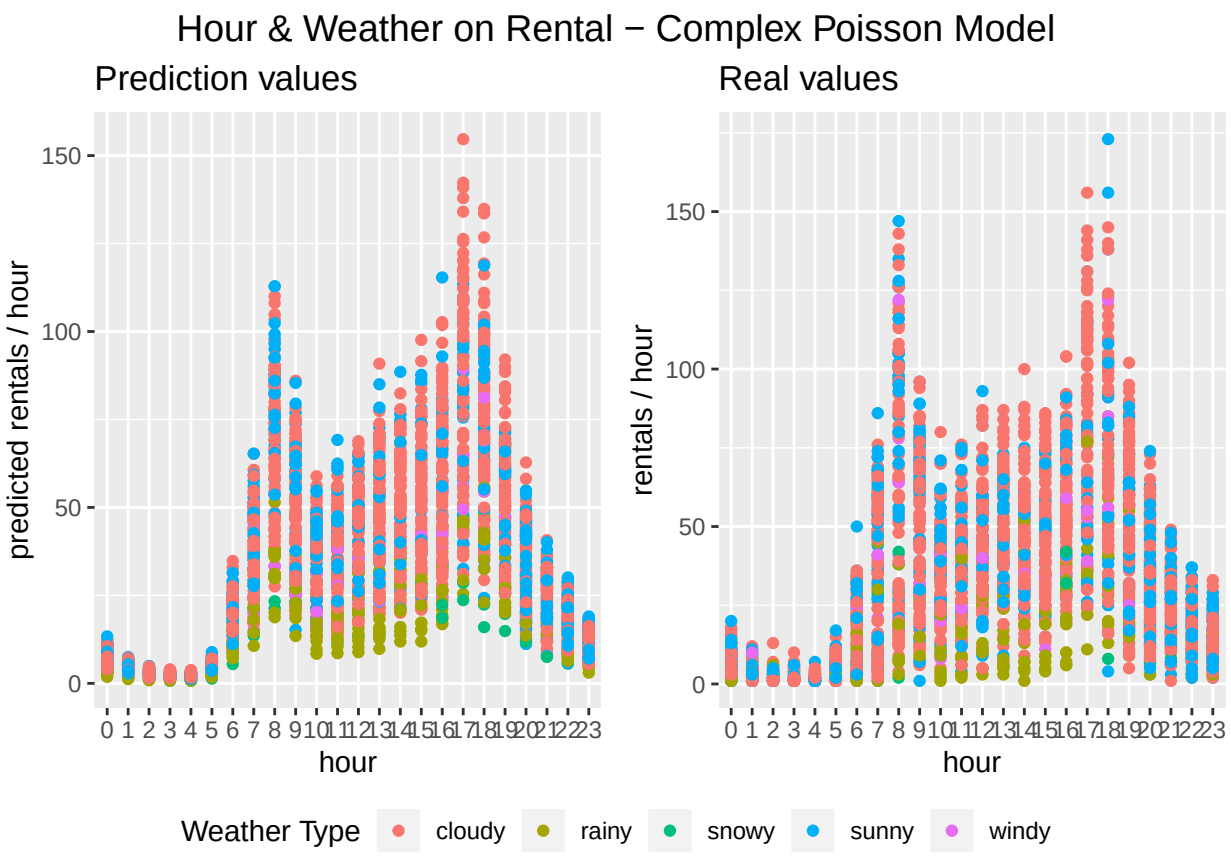
```
# residual analysis plots
par(mfrow=c(2,2))
plot(pm.fit)
```



Plotting predicted values with GGPlot The two plots below show how close the predicted values to the real values.

```
# ggplot predicted values of hourly rental for different weather types
p1 <- ggplot(d.test, aes(x=hour, y=pm.pred, color=type)) +
  geom_point(data=d.test, aes(group=type)) +
  ggtitle("Prediction values") + ylab("predicted rentals / hour") +
  scale_color_hue("Weather Type", breaks=levels(d.test$type))
```

```
# ggplot true values of hourly rental for different weather types
p2 <- ggplot(d.test, aes(x=hour, y=rental_count, color=type)) +
  geom_point(data=d.test, aes(group=type)) +
  ggtitle("Real values") + ylab("rentals / hour") +
  scale_color_hue("Weather Type", breaks=levels(d.test$type))

figure <- ggarrange(p1, p2, ncol=2, common.legend = TRUE, legend="bottom")
annotate_figure(figure, top=text_grob("Hour & Weather on Rental - Complex Poisson Model",
                                      size = 14))
```

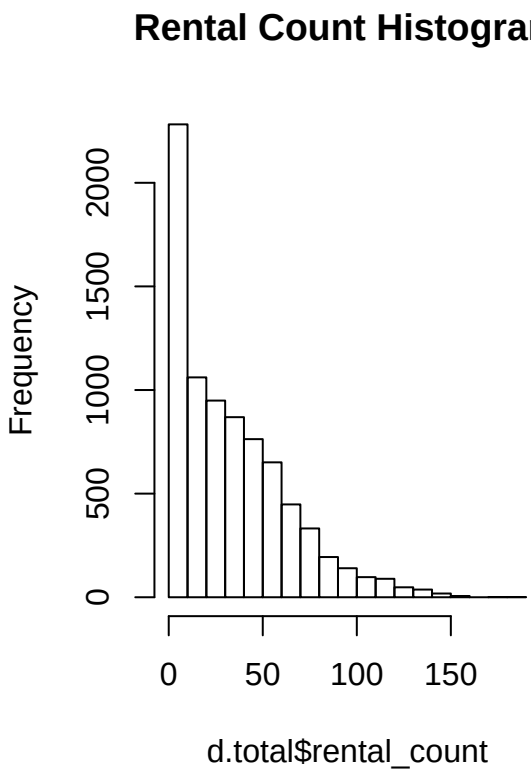


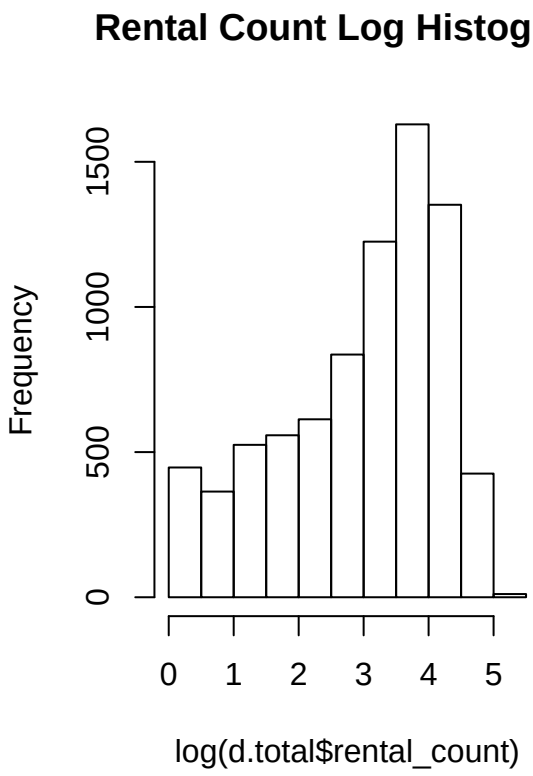
More complex model: Linear model with different polynomial levels

In this section, we apply cross validation in order to run a list of complex models with increasing polynomial levels of the temperature predictor. Besides, these models have many predictors and an interaction between humidity and dew point.

The histograms below show that the distribution of rental_count is not normal distribution. We have to use log transformation for fitting a linear model.

```
## check normal distribution for linear model fitting
par(mfrow=c(1, 2))
hist(d.total$rental_count, main = "Rental Count Histogram")
hist(log(d.total$rental_count), main = "Rental Count Log Histogram")
```





Fitting the linear model

```
set.seed(1)

cv.error.10 <- rep(0, 10)
for (i in 1:10) {
  glm.fit <- glm(log(rental_count) ~ type + month + weekday + hour + humidity * dewpoint +
                   poly(windspeed, degree=i) + poly(temperature, degree=i), data=d.train)
  cv.error.10[i] <- cv.glm(d.train, glm.fit, K=10)$delta[1]
}

best.level <- which.min(cv.error.10)
cat("Index of model with least error rate: ", best.level, "\n")
```

```
## Index of model with least error rate:  4
```

```
cat("Cross validation error of different polynomial linear models:\n", cv.error.10, "\n")
```

```
## Cross validation error of different polynomial linear models:
##  0.3014258 0.2977286 0.296371 0.2942483 0.2955557 0.2967275 0.3023312 0.3037069 0.3409857 0.3649077
```

```
glm.fit.best <- glm(log(rental_count) ~ type + month + weekday + hour +
                      humidity * dewpoint +
                      poly(windspeed, degree = best.level) +
                      poly(temperature, degree = best.level),
                    data=d.train)

# predict model on test data
```

```
glm.pred <- predict(glm.fit.best, newdata = d.test)
# convert log value to normal value via exponential
glm.pred <- exp(glm.pred)

# compute MSE
glm.mse <- mean((d.test$rental_count - glm.pred)^2)
cat("GLM polynomial model MSE: ", glm.mse)
```

```
## GLM polynomial model MSE:  199.359
```

Residual Analysis The residuals vs Fitted figure shows that the smoother red line is on zero. Therefore, we may conclude that the model equation is correct. With log transformation, the Normal Q-Q plot shows the residual standard deviance is close to normal distribution on the diagonal line.

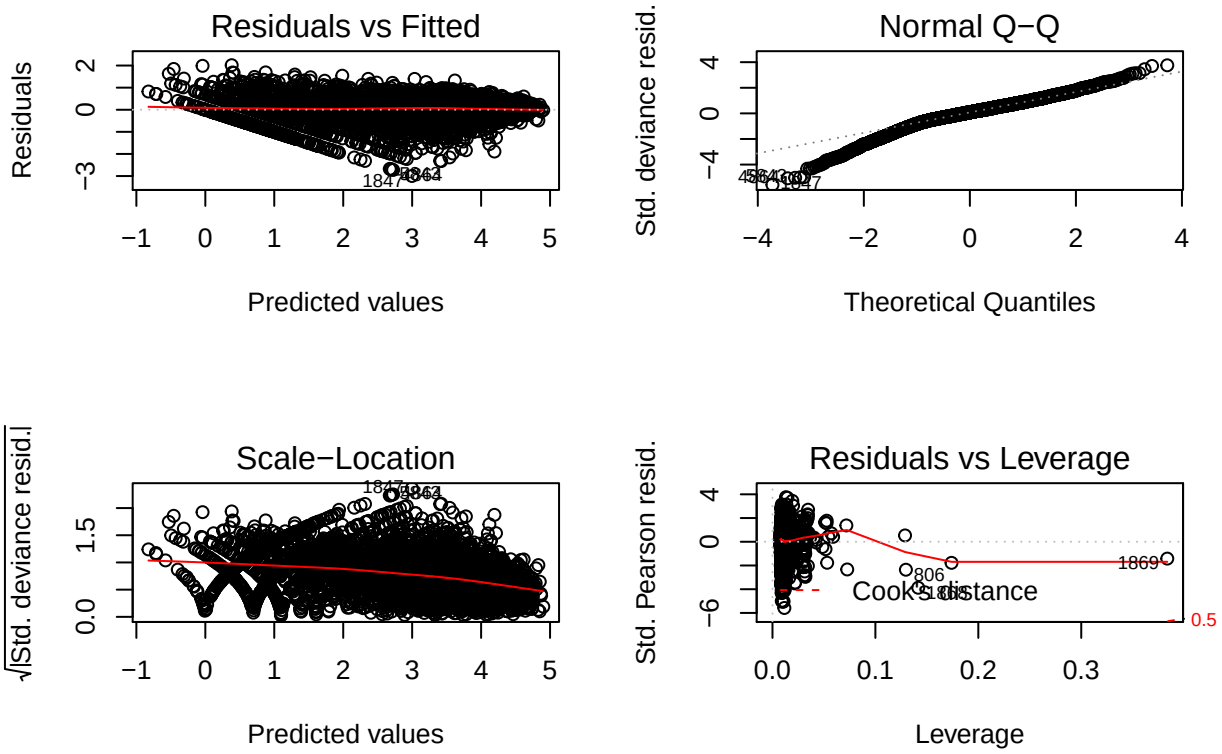
```
# residual
resid_values <- resid(glm.fit.best)
cat("Residual length: ", length(resid_values))
```

```
## Residual length:  5082
```

```
cat("Residual header: ", head(resid_values))
```

```
## Residual header:  -0.3666195 -0.5092576 0.5736475 -0.584702 0.5248364 0.3273697
```

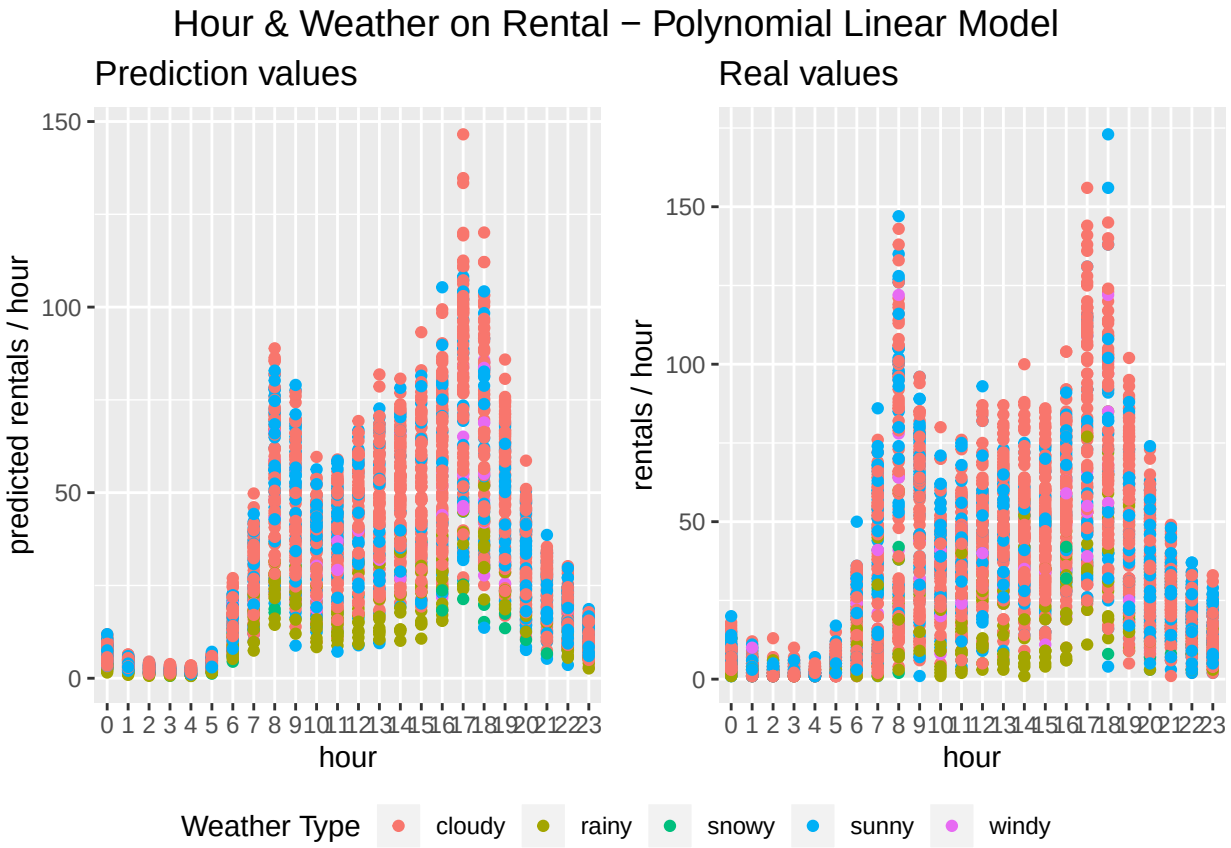
```
# plot
par(mfrow=c(2,2))
plot(glm.fit.best)
```



Plotting predicted values with GGPlot The two plots below show how close the predicted values to the real

values.

```
# ggplot predicted values of hourly rental for different weather types
p1 <- ggplot(d.test, aes(x=hour, y=glm.pred, color=type)) +
  geom_point(data=d.test, aes(group=type)) +
  ggtitle("Prediction values") + ylab("predicted rentals / hour") +
  scale_color_hue("Weather Type", breaks=levels(d.test$type))

# ggplot true values of hourly rental for different weather types
p2 <- ggplot(d.test, aes(x=hour, y=rental_count, color=type)) +
  geom_point(data=d.test, aes(group=type)) +
  ggtitle("Real values") + ylab("rentals / hour") +
  scale_color_hue("Weather Type", breaks=levels(d.test$type))

figure <- ggarrange(p1, p2, ncol=2, common.legend = TRUE, legend="bottom")
annotate_figure(figure, top = text_grob("Hour & Weather on Rental - Polynomial Linear Model",
                                        size = 14))
```



Fitting with Regression Tree - Random forest

According to the Feature Importance diagram, while the hour and weekday predictors are the most important for the mean square error of the model, hour is the significant feature for the node purity.

```
set.seed(1)
rf.fit <- randomForest(rental_count ~ . - rental_count - tripduration - date,
                       data = d.train, mtry=3, ntree=50, importance=TRUE)
rf.pred <- predict(rf.fit, d.test)
```

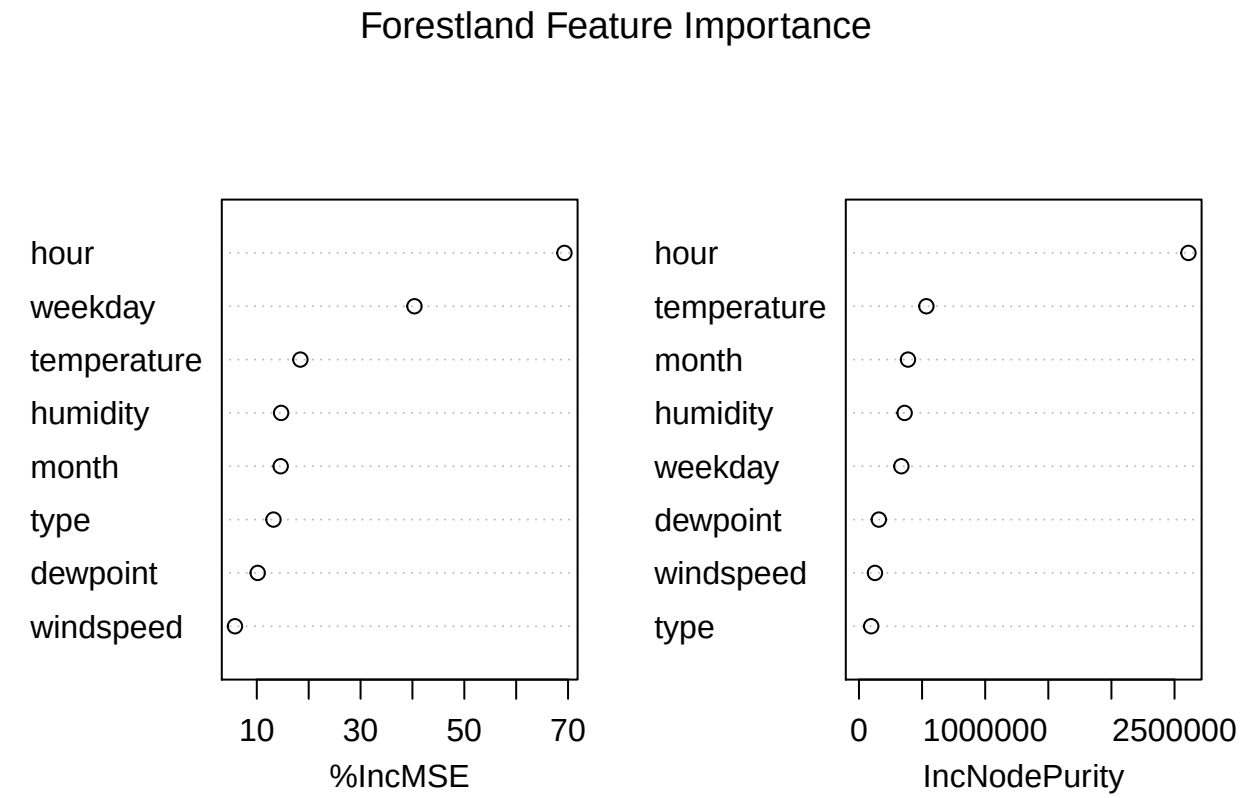
```
rf.mse <- mean((rf.pred - d.test$rental_count)^2)
cat("Random forest mode MSE: ", rf.mse)
```

```
## Random forest mode MSE:  101.2652
```

```
importance(rf.fit)
```

```
##                %IncMSE IncNodePurity
## hour         69.298351    2610039.89
## month        14.598942     388158.83
## weekday      40.390482     335645.98
## temperature 18.396173     533802.63
## dewpoint     10.173240     157345.51
## humidity     14.692209     361499.71
## windspeed     5.790877     126652.61
## type         13.213668      96851.97
```

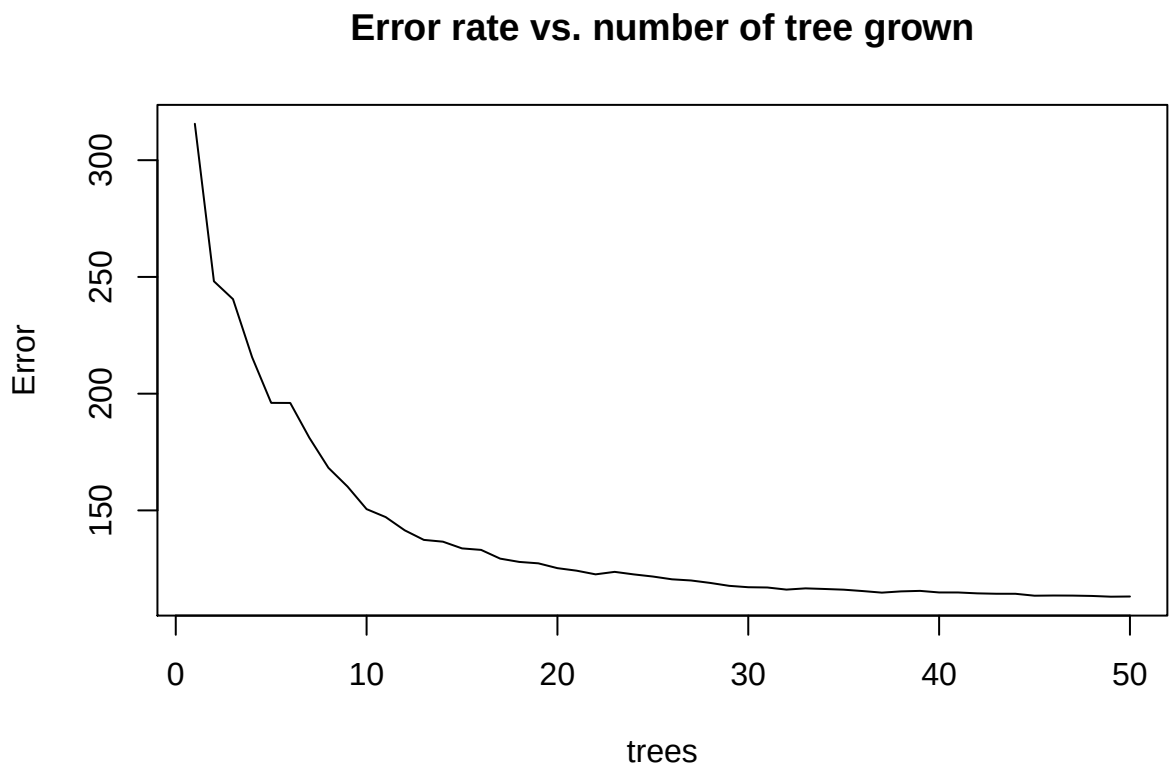
```
varImpPlot(rf.fit, main = "Forestland Feature Importance")
```



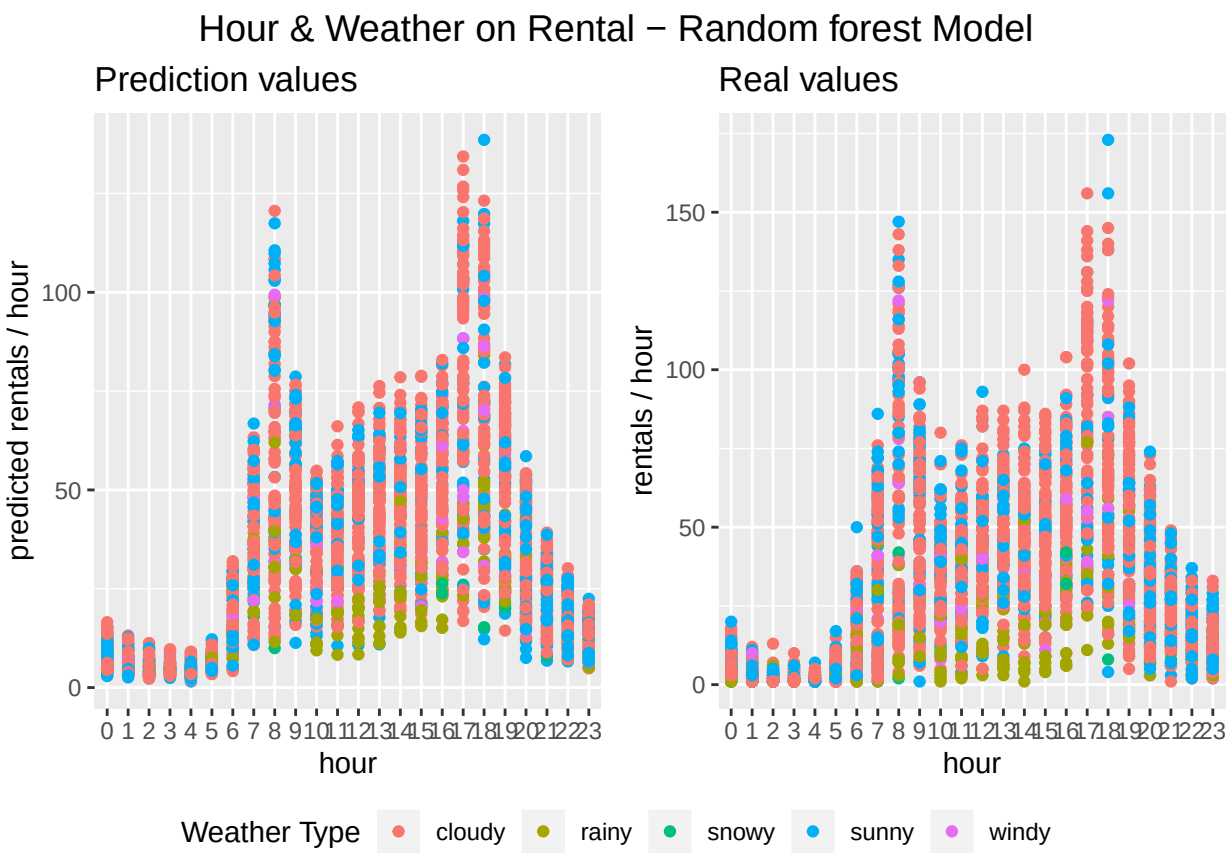
Plotting predicted values with GGPlot The two plots below show how close the predicted values to the real values.

```
# plot randomforest model fit
plot(rf.fit, main="Error rate vs. number of tree grown")

# ggplot predicted values of hourly rental for different weather types
p1 <- ggplot(d.test, aes(x=hour, y=rf.pred, color=type)) +
  geom_point(data=d.test, aes(group=type)) +
```

```
  ggtitle("Prediction values") + ylab("predicted rentals / hour") +
  scale_color_hue("Weather Type", breaks=levels(d.test$type))

# ggplot true values of hourly rental for different weather types
p2 <- ggplot(d.test, aes(x=hour, y=rental_count, color=type)) +
  geom_point(data=d.test, aes(group=type)) +
  ggtitle("Real values") + ylab("rentals / hour") +
  scale_color_hue("Weather Type", breaks=levels(d.test$type))

figure <- ggarrange(p1, p2, ncol=2, common.legend = TRUE, legend="bottom")
```




```
annotate_figure(figure, top = text_grob("Hour & Weather on Rental - Random forest Model",
                                        size = 14))
```

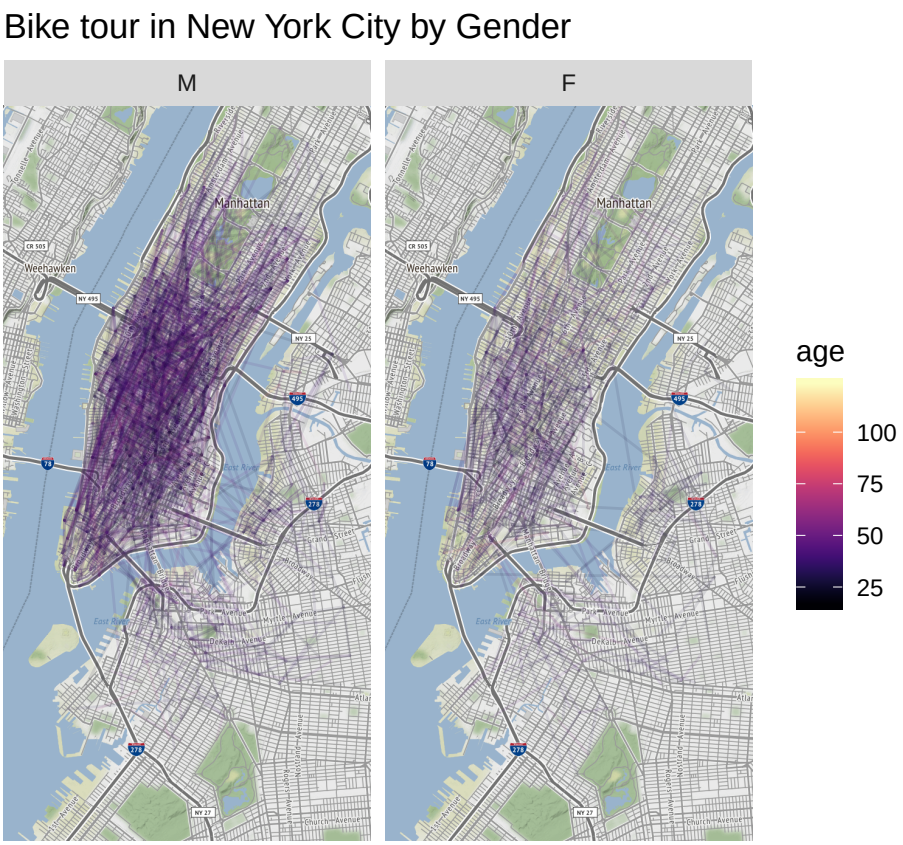


A chapter of your choice: GGMAP

The following picture bases on the bike sharing dataset only and shows the map of New York City. The lines indicate the most common routes driven by females and males, transparency indicates how often are the routes driven and the color indicates which routes are driven by younger and which one by older people.

```
ggmap(mad_map) +
 geom_leg(
   data=d.bike.map.sample,
     # color = factor(gender),
     alpha=0.1,
     aes(
       color = age,
       x = start_station_longitude,
       y = start_station_latitude,
       xend = end_station_longitude,
       yend = end_station_latitude
       )
     )+
 labs(x="",y="") +
 facet_grid(. ~ gender) +
 theme(
   axis.title.x=element_blank(),
   axis.text.x=element_blank(),
   axis.ticks.x=element_blank(),
   axis.title.y=element_blank(),
```

```
  axis.text.y=element_blank(),
  axis.ticks.y=element_blank()
  ) +
scale_color_viridis(option = "magma") + ggtitle("Bike tour in New York City by Gender")
```




```
ggsave("map_age_gender_month.pdf", device = "pdf")
```

```
## Saving 6.5 x 4.5 in image
```

Conclusion

As a result, comparing the mean squared error (MSE) of the three above mentioned models, Poisson model is simple and has a moderate MSE of 181.9. Linear polynomial model is complexer but results in a bit higher MSE of 199.3. This means, the linear polynomial model does not fit very well. Lastly, randomforest model shows the most effective fitting with its smallest MSE of 101.0. Randomforest model is definitely the choice for the fitting of the data sources.

```
cat("Poisson model MSE: ", pm.mse, "\n")
```

```
## Poisson model MSE:  181.9917
```

```
cat("Linear Polynomial model MSE: ", glm.mse, "\n")
```

```
## Linear Polynomial model MSE:  199.359
```

```
cat("Randomforest model MSE: ", rf.mse, "\n")
```

```
## Randomforest model MSE:  101.2652
```