Assignment R Bootcamp

Tu Tran and Andy Gubser

/today

Contents

Abstract

Purpose / Research question

The purpose of this project is to analyse the hourly demand of using bike sharing in New York City in 2016. In particular, we would like to explore the provided bike sharing, weather data and then create different models to predict the hourly demand for sharing bikes. These models will be compared to find one best model with a 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]. The weather data is retrieved from the Weather.com API [^2 https://api.weather.com/v1/location/KLGA:9:US/observations/historical.json]. After the data preparation step, we provide an graphical analysis to approach the research question on the usage of shared bikes. Then, we fit a poisson model, high level polynomial linear model followed by a random forest model to deepen our analysis.

Analysis

R and knitr Setup

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

```
##           used (Mb) gc trigger (Mb) max used (Mb)
## Ncells  532942 28.5    1212025 64.8   621332 33.2
## Vcells 1005922  7.7    8388608 64.0  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)
```

Library Import

User Defined Functions and Constants

```
## 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

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 categorical variable age_cor is created, where
users over 70 are put into a single bin in order to reduce potential bias of these observations. 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

The weather data set includes 366 observations and the 7 variables maximum, minimum and average
temperature, precipitation, snow fall and snow depth on specific dates. In order to merge the two data sets
on the date variable, this variable is created in the bike sharing data set by extracting the date from the start
timestamps. The merged data set consists of 242'746 observations and 15 variables.

Read bike and weather dataset and merge for the analysis

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

## 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

```r
## 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         756        10773
## 2 Dienstag       815        11250
## 3 Mittwoch       858        11892
## 4 Donnerstag     852        11769
## 5 Freitag        796        11086
## 6 Samstag        641        11132
## 7 Sonntag        603        10475
```

```r
## 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     321         3671
##  4 2016-02-15     117         1129
##  5 2016-05-30     600        10715
##  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     214         2520
```

```r
## 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.
## Dropping 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, one is the working days and one is the weekend. In the working days, we can see two peaks at the rush hours at about 8:00 AM and from 5:00 PM to 6:00 PM. These are the time people go to work and go home. In the weekend, people are more relaxing and may take longer sleep. During the weekend, the number of rental bikes goes slow up hill, reaches its top at about 57 rentals around 3:00 PM and goes 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 mininum 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")
```


24h Rental by Weekday
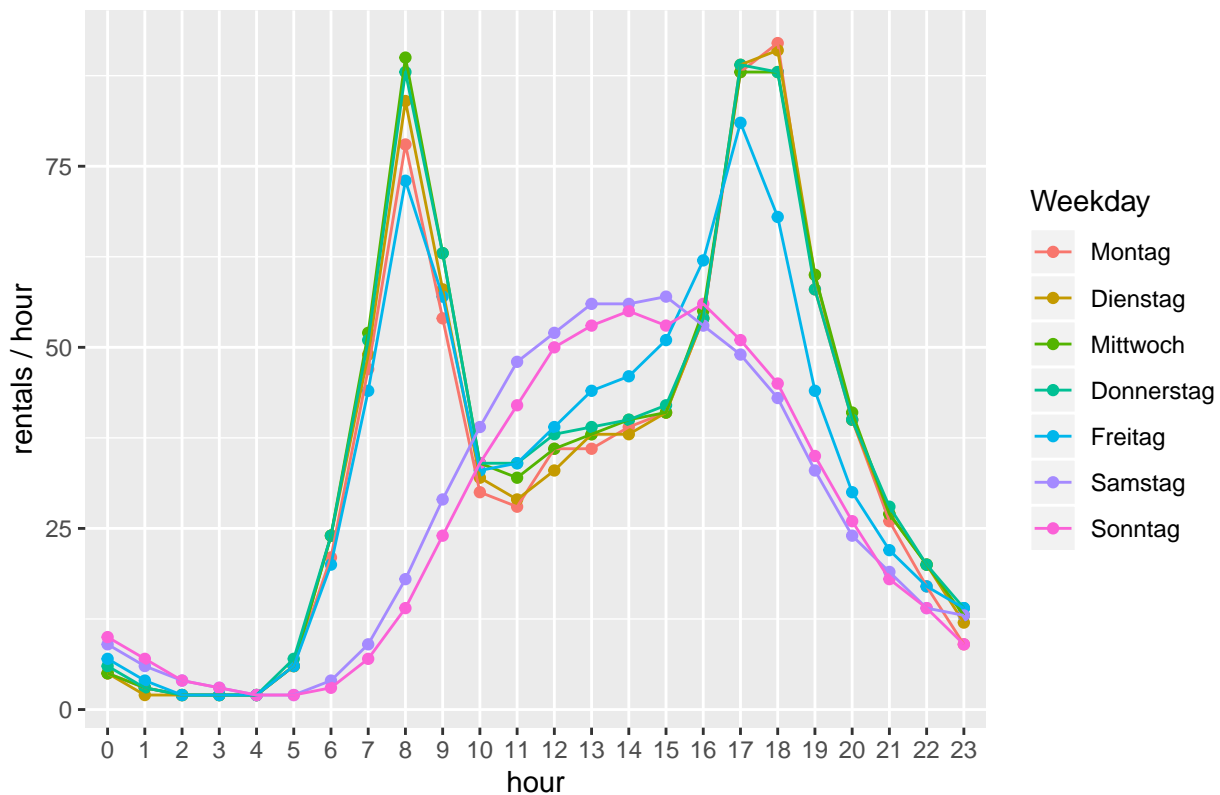
ggplot rentals over 24h for different months

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")
```
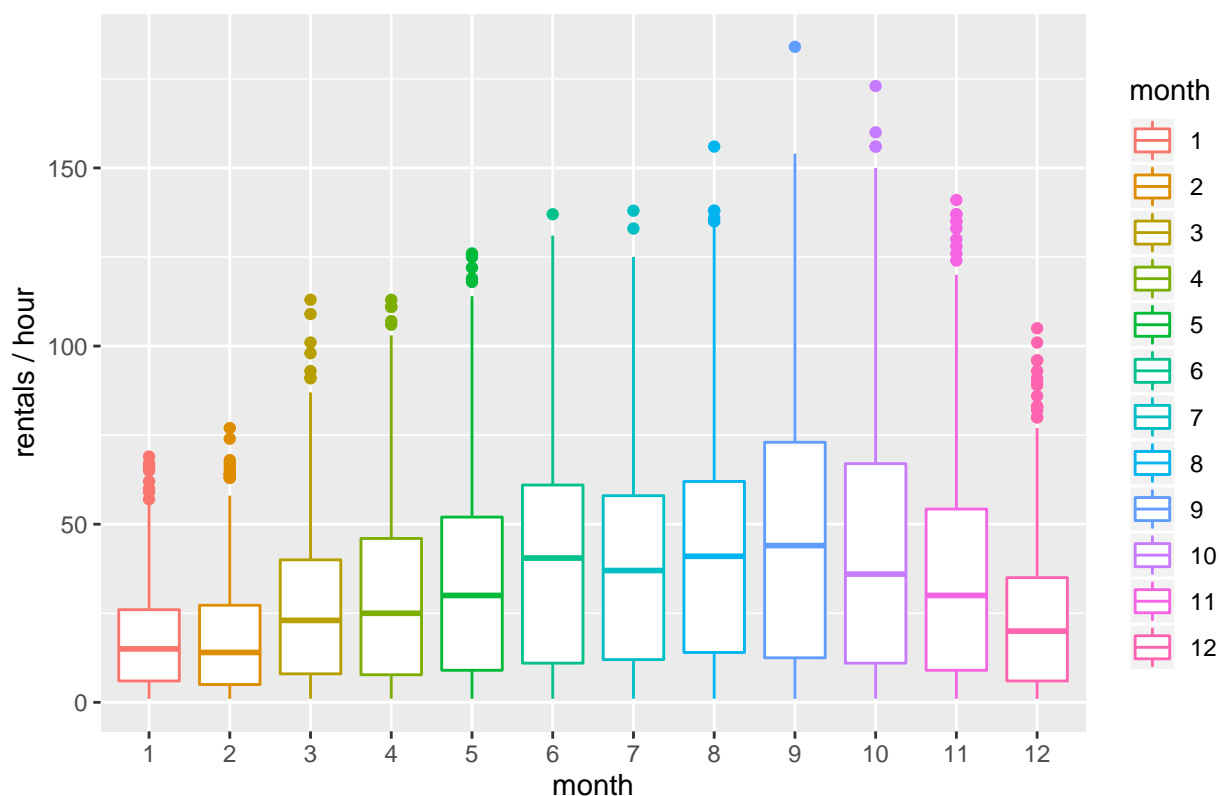

ggplot rentals over 24h for different months

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")
```

24h Rental by Weather Type
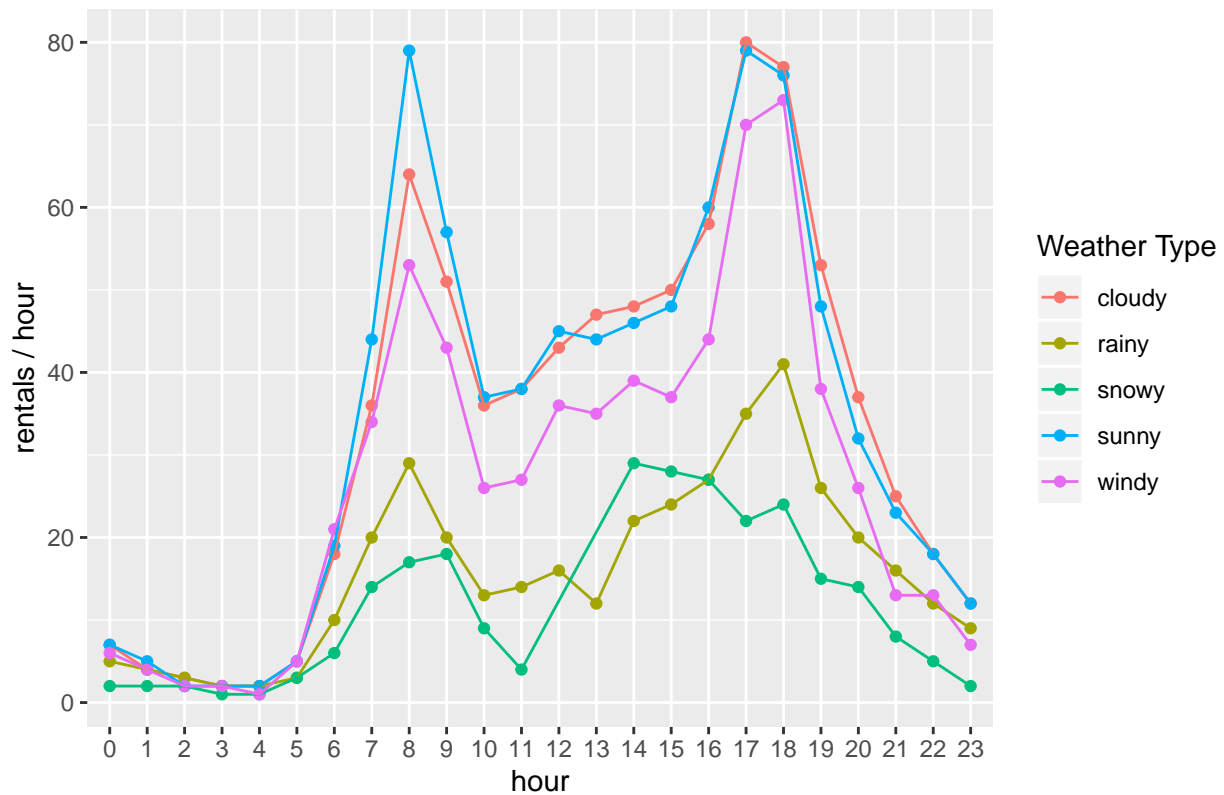
ggplot for the rental of the weekdays with an influence of the weather factor

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.

```r
## 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()
```
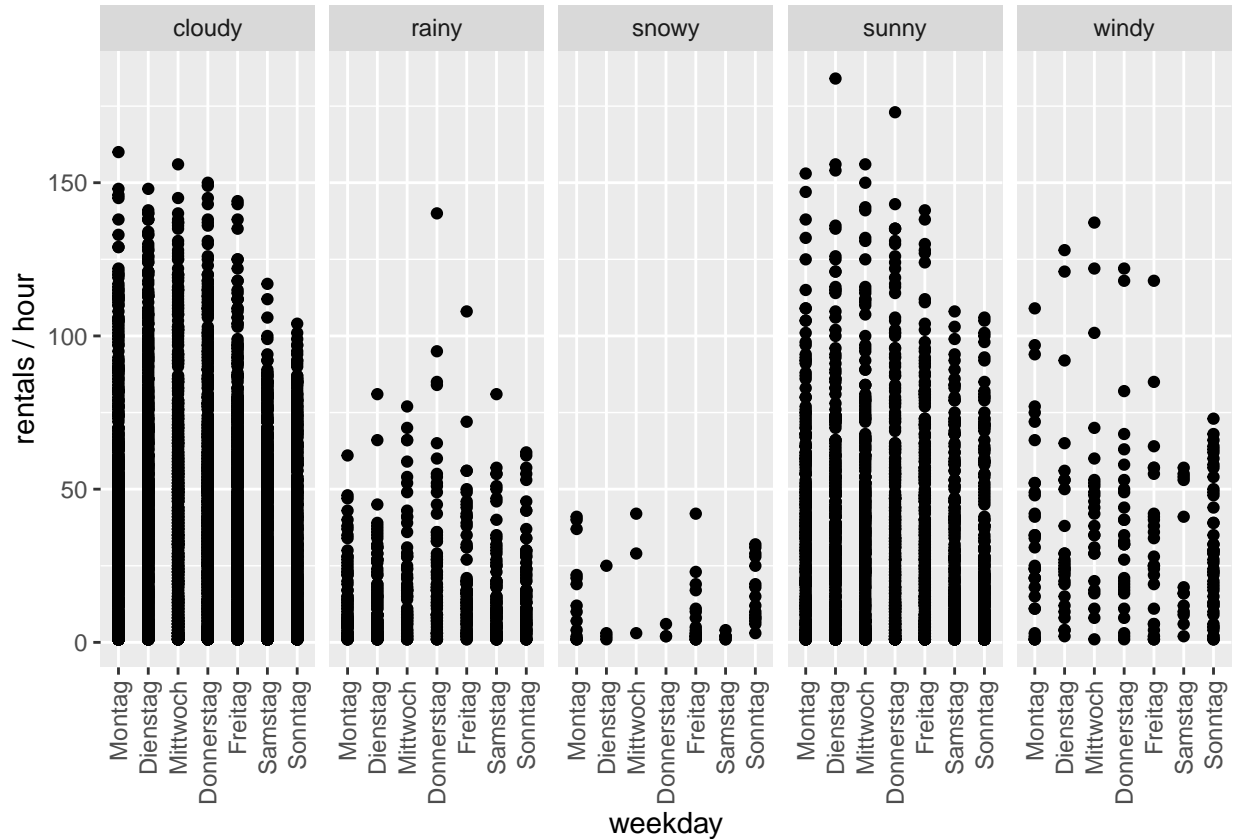
ggplot heatmap for correlation analysis between numeric predictors

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")
```
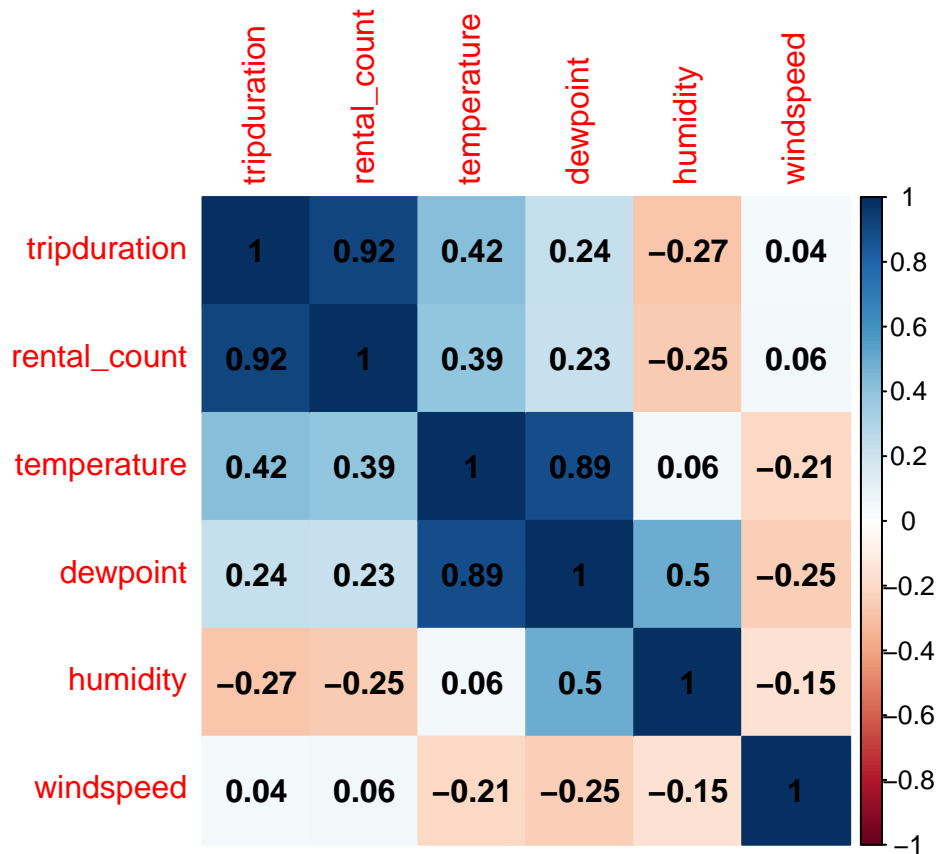
Fitting models

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.

```r
## 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:  181.9057
```

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.

```r
# residual
resid_values <- resid(pm.fit)
cat("Residual legnth: ", length(resid_values), "\n")
```
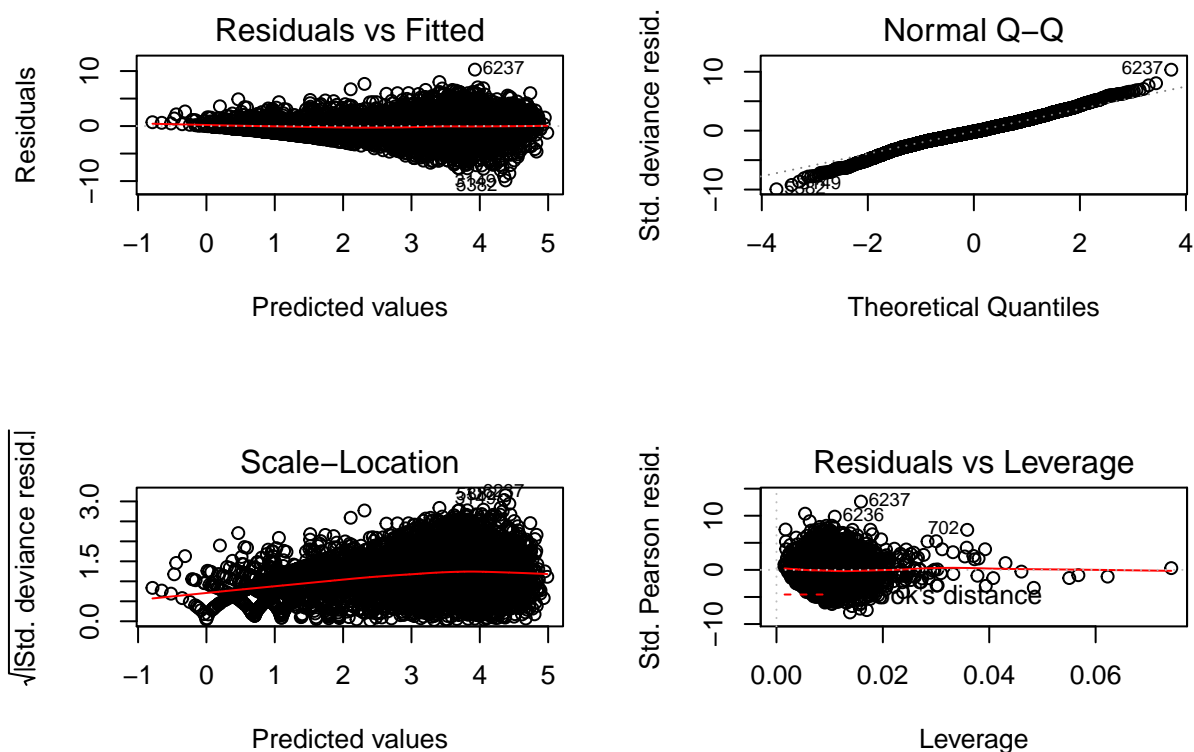
```
## Residual legnth:  5082
```

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

```
## Residual header:  -0.7866357 -0.8484774 0.603783 -0.8796609 2.460592 1.344774
```

```r
# 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.

```r
# 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))
```
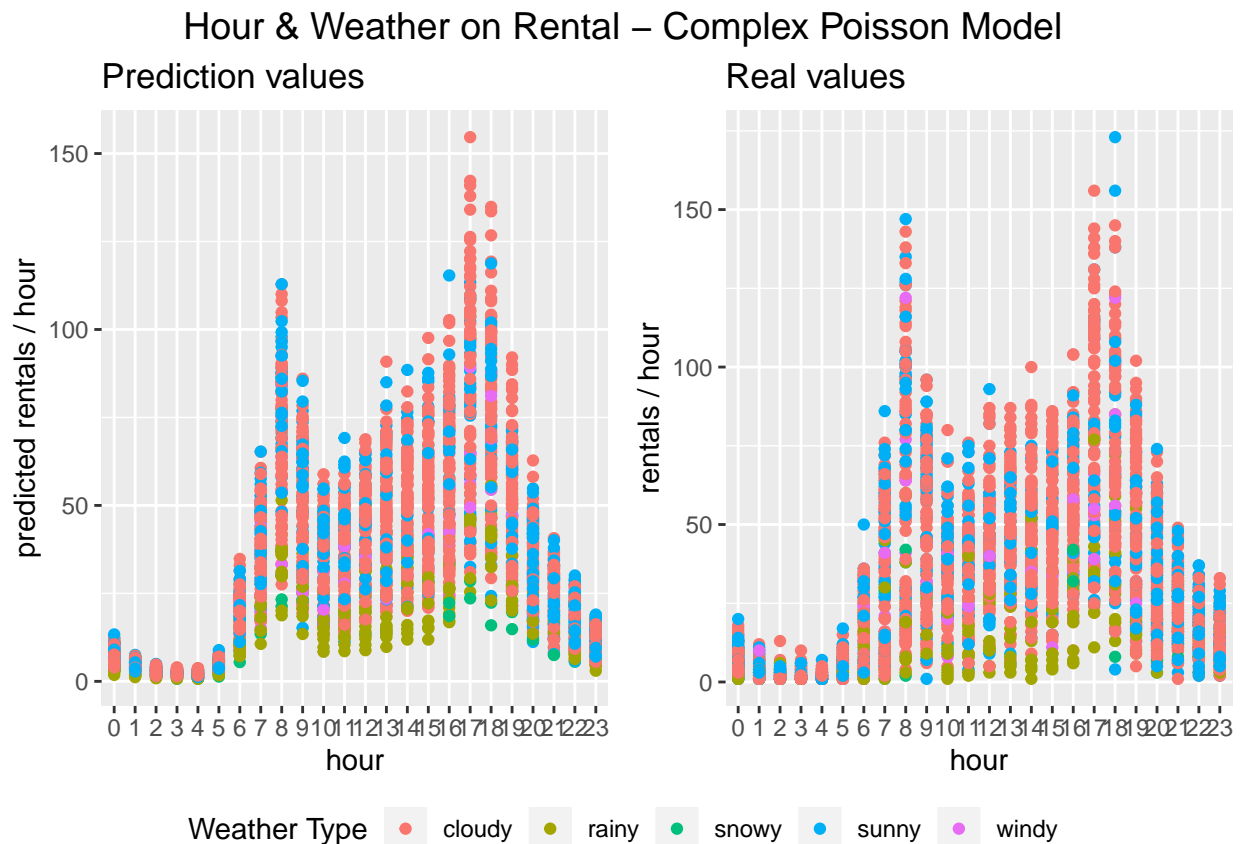

More complex model: Linear model with different polynomial levels

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 dewpoint.

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")
```

Rental Count Histogram
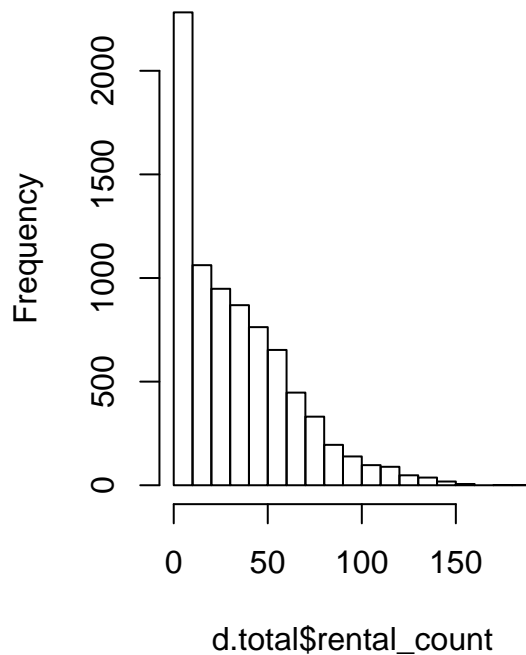
Rental Count Log Histogram
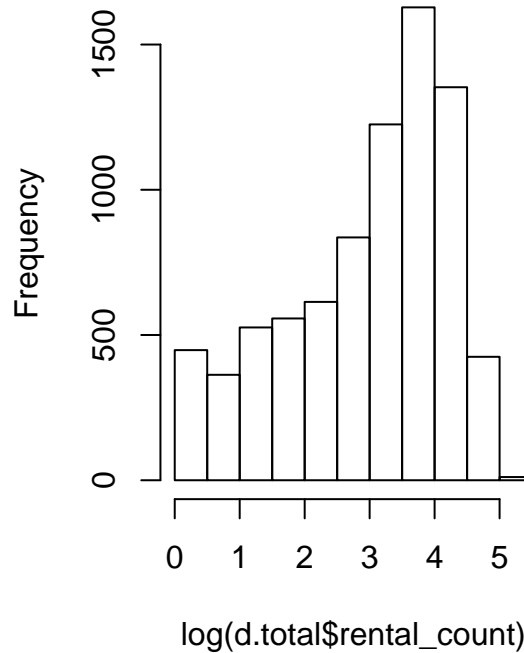
Fitting the linear model

```
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 polynominal linear models:\n", cv.error.10, "\n")
```

```
## Cross validation error of different polynominal linear models:
##  0.3013236 0.2976461 0.2962701 0.294151 0.2954633 0.2966329 0.3022395 0.3036237 0.3408784 0.3650716
```

```
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: 199.2743

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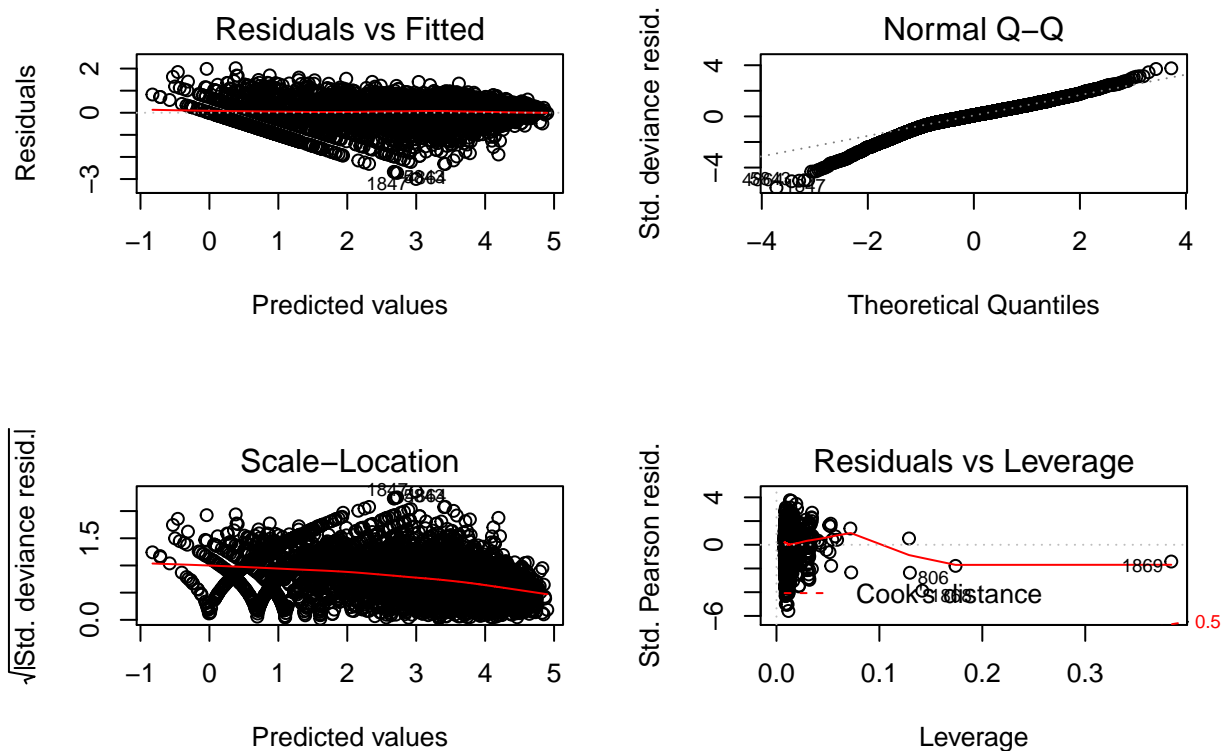
```
# residual
resid_values <- resid(glm.fit.best)
cat("Residual length: ", length(resid_values))
```

Residual length: 5082

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

Residual header: -0.3650641 -0.5091433 0.5738417 -0.5843946 0.5253908 0.3276906

```
# 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

13

values.

```r
# 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))
```
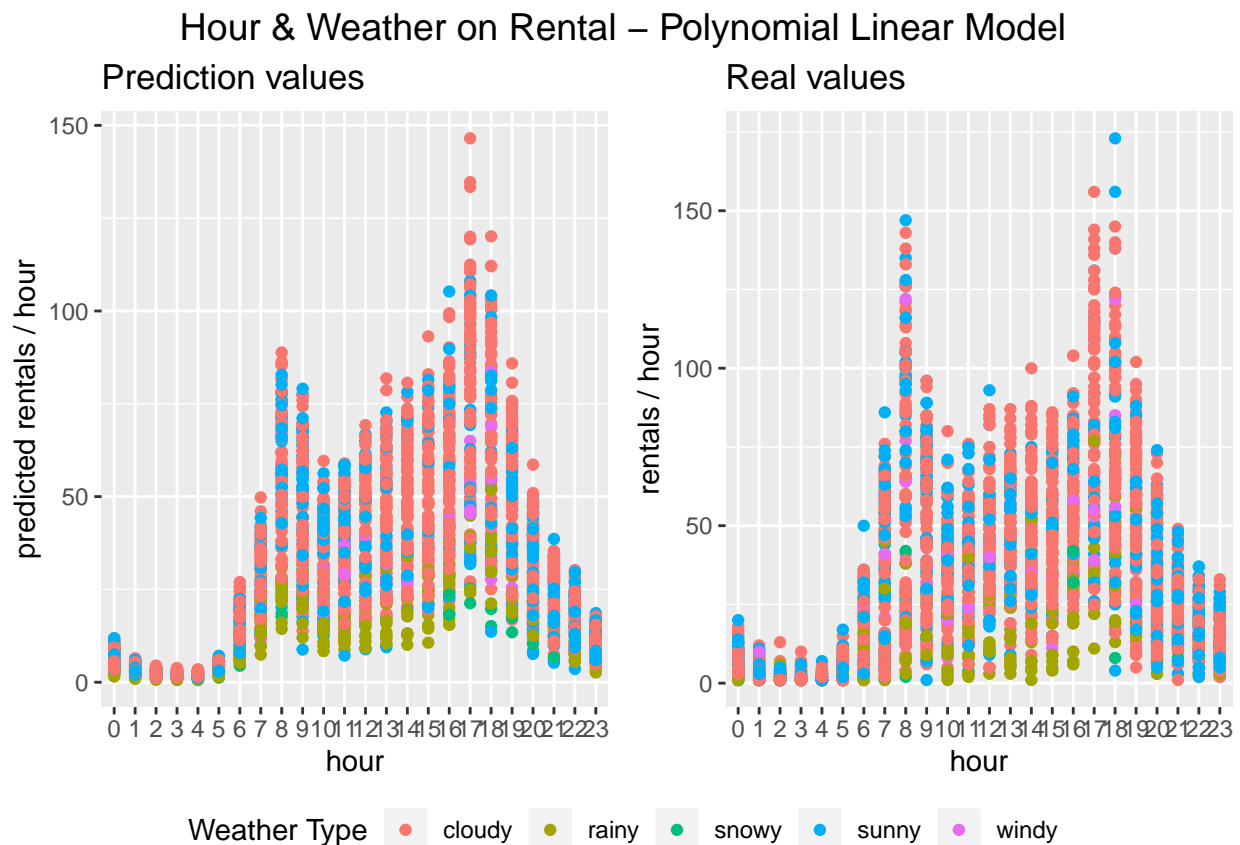

Fitting with Regression Tree - Randomforest

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

```r
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("Randomforest mode MSE: ", rf.mse)
```

Randomforest mode MSE: 101.0408

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

```
##                 %IncMSE IncNodePurity
## hour          66.583721     2590888.8
## month         15.076734      395430.4
## weekday       33.926246      346111.6
## temperature   10.635035      512802.7
## dewpoint       7.960879      192456.9
## humidity      15.888705      318434.0
## windspeed      6.579759      135717.1
## type          11.851720      121317.7
```

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

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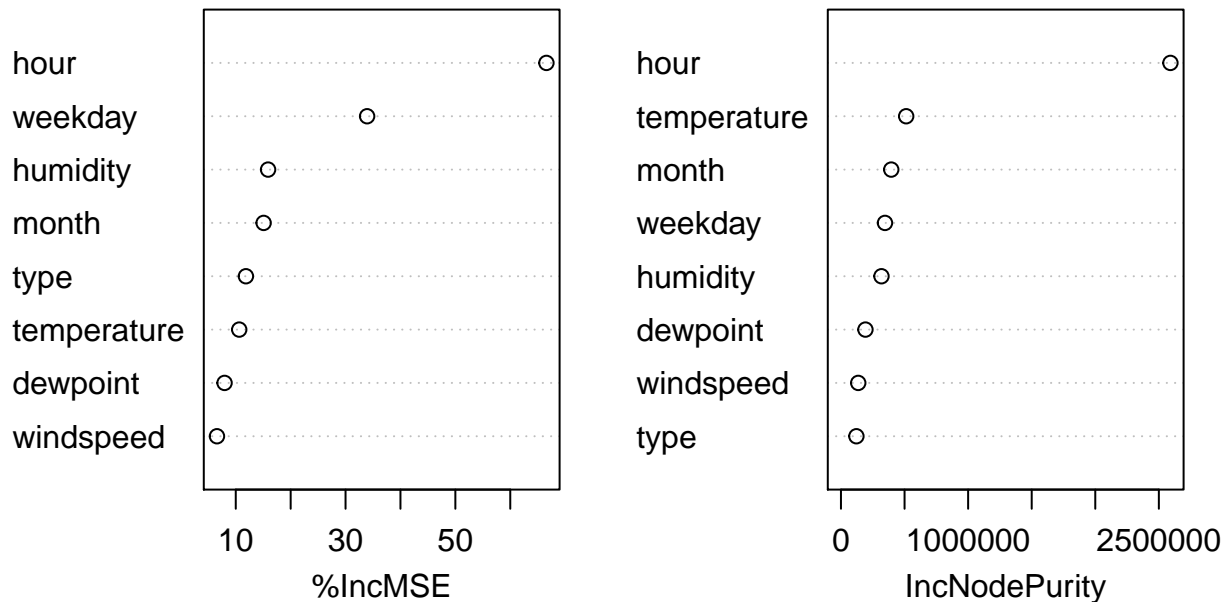
Plotting predicted values with GGPlot The two plots below show how close the predicted values to the real values.

```
# 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)) +
```

15

```
    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")
```

Error rate vs. number of tree grown
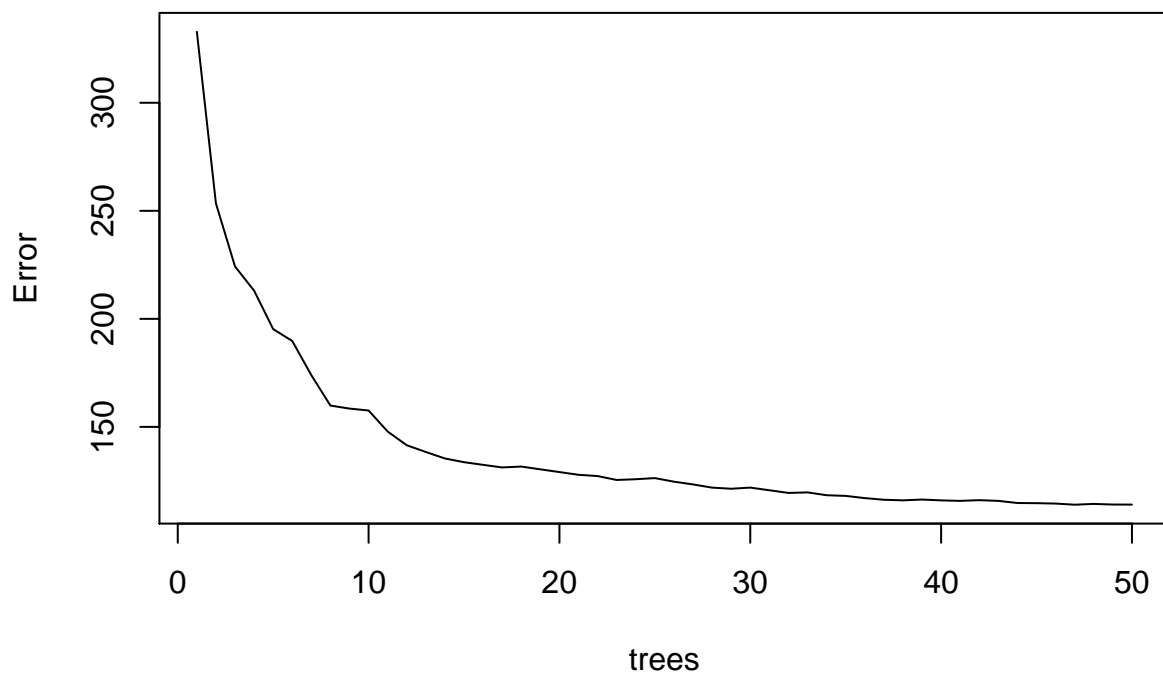

```
annotate_figure(figure, top = text_grob("Hour & Weather on Rental - Randomforest Model",
                                          size = 14))
```
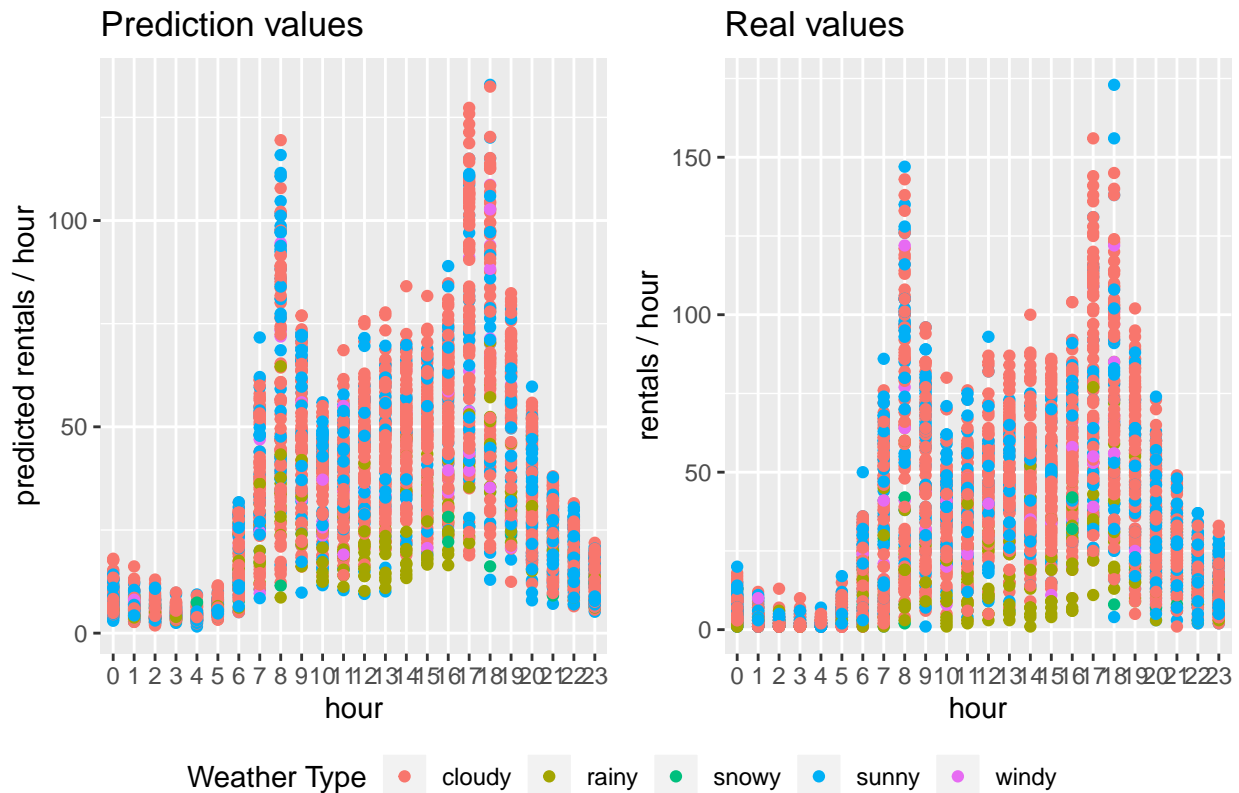
Hour & Weather on Rental – Randomforest Model

A chapter of your choice: Regression Tree Randonforest

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:  181.9057
```

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

```
## Linear Polynomial model MSE:  199.2743
```

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

```
## Randomforest model MSE:  101.0408
```