HW02

Keyi Jiang, Freya Wang, Rita Xu

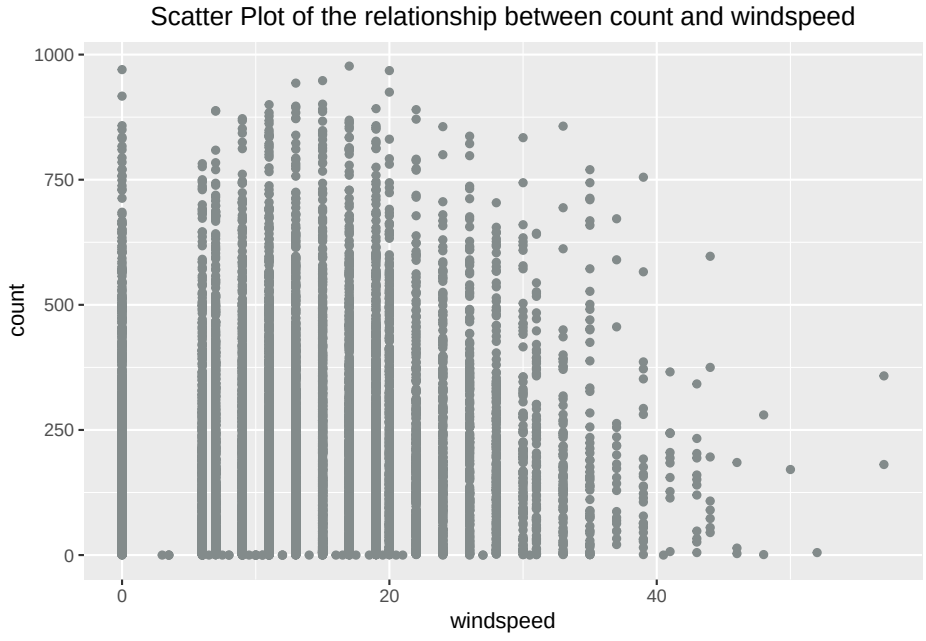
01/28/2023

Group name: Why women kill

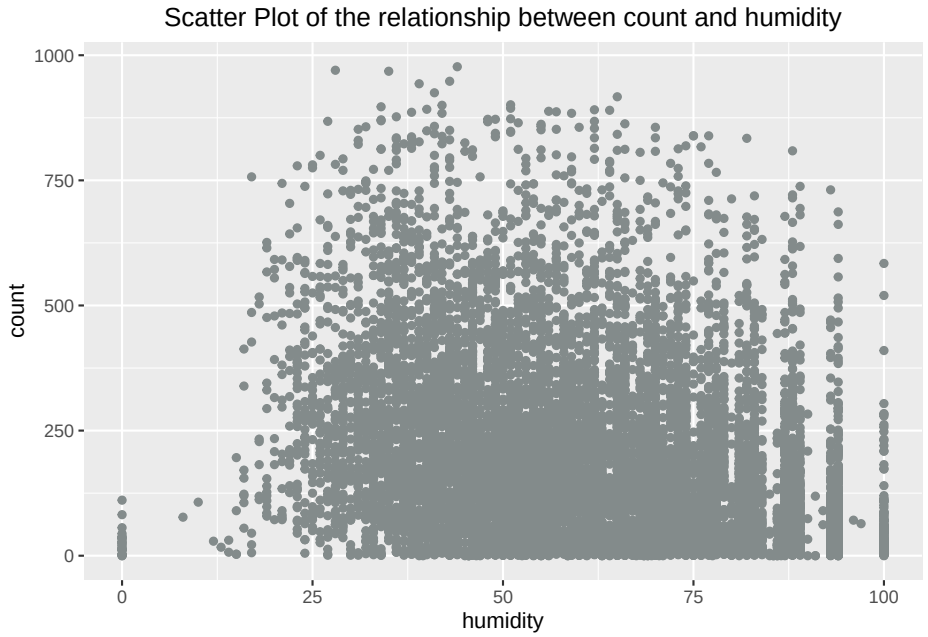
Question 1

2.a relationship between *count* and *windspeed*, *humidity*, *temp*, and *atemp*.

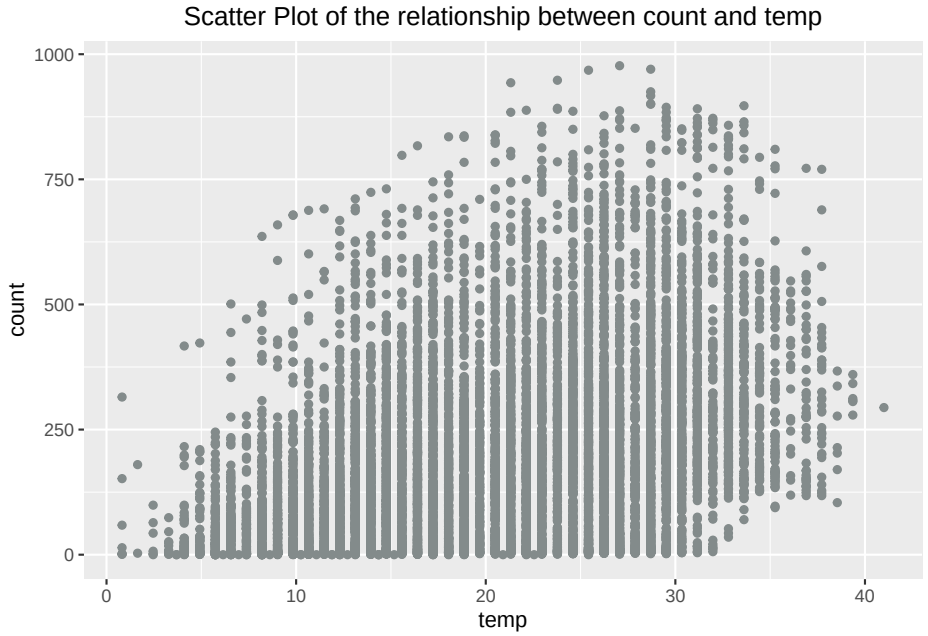
```
ggplot(bike.train)+
  geom_point(aes(x=windspeed,y=count), color = "azure4")+
  labs(title = "Scatter Plot of the relationship between count and windspeed")+
  theme(plot.title = element_text(hjust = 0.5))
```



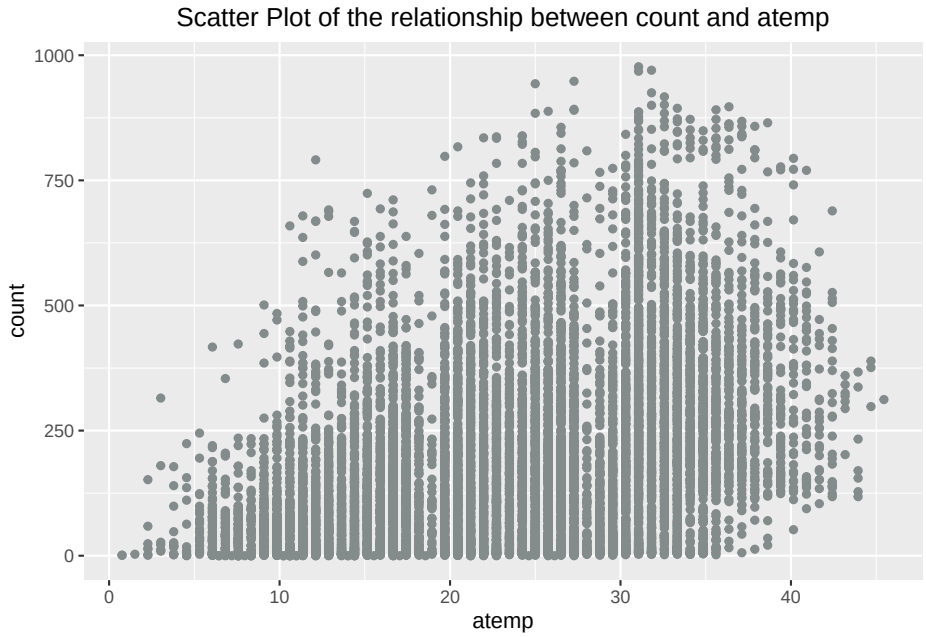

```
ggplot(bike.train)+
  geom_point(aes(x=humidity,y=count), color = "azure4")+
  labs(title = "Scatter Plot of the relationship between count and humidity")+
  theme(plot.title = element_text(hjust = 0.5))
```




```
ggplot(bike.train)+
  geom_point(aes(x=temp,y=count), color = "azure4")+
  labs(title = "Scatter Plot of the relationship between count and temp")+
  theme(plot.title = element_text(hjust = 0.5))
```

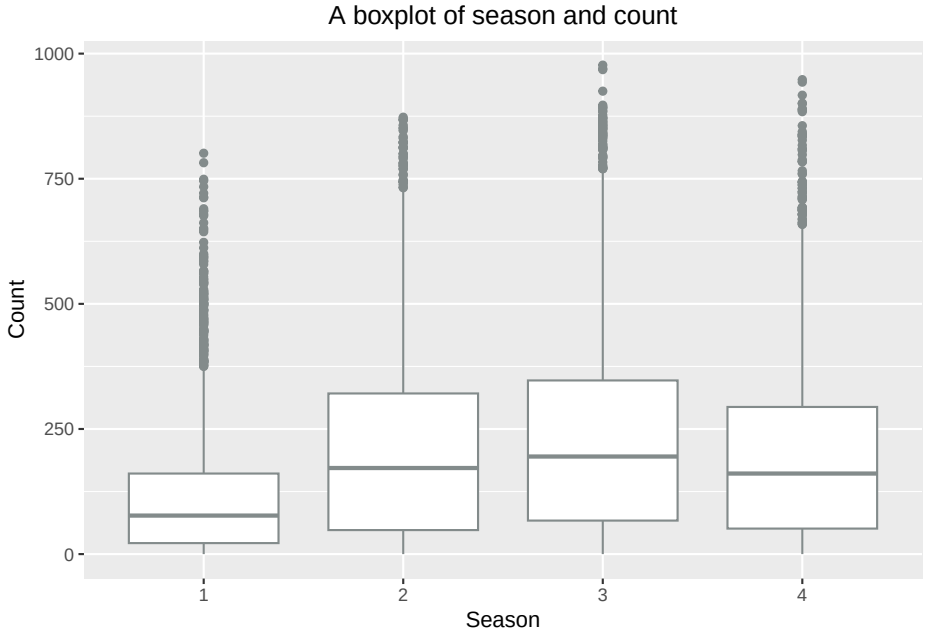



```
ggplot(bike.train)+
  geom_point(aes(x=atemp,y=count), color = "azure4")+
  labs(title = "Scatter Plot of the relationship between count and atemp")+
  theme(plot.title = element_text(hjust = 0.5))
```



2.b How does count depend on the season?

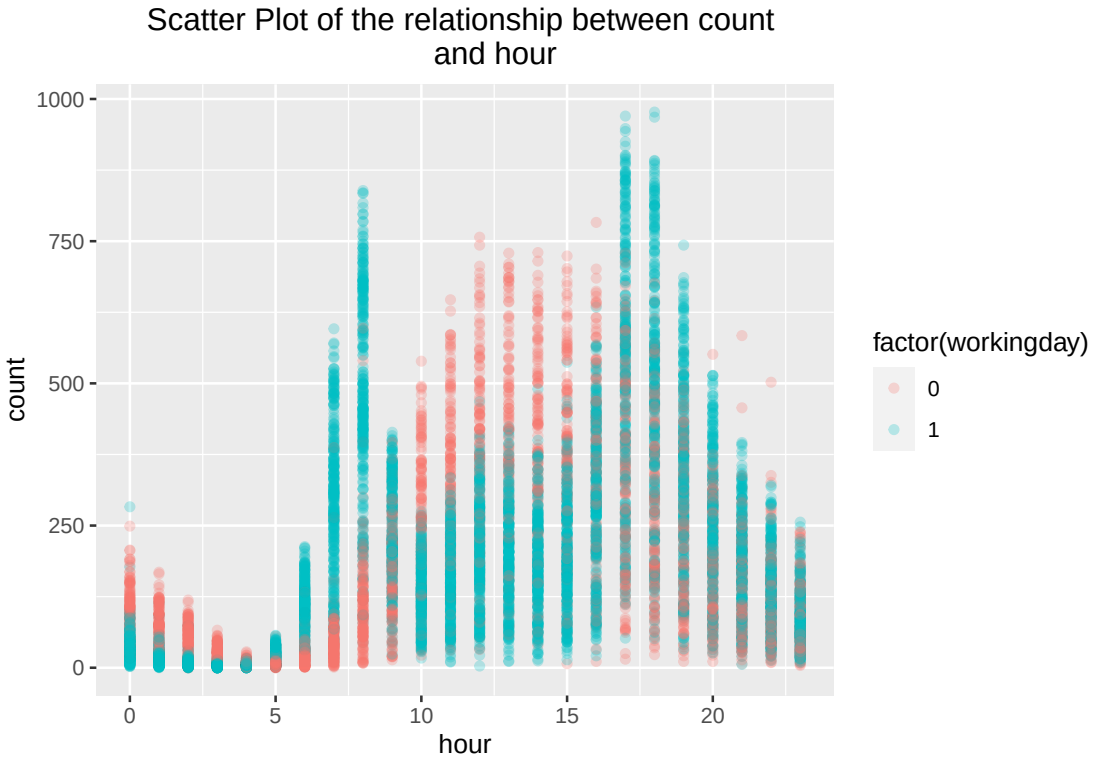
```
ggplot(bike.train,aes(x=factor(season),y=count))+
  geom_boxplot( color = "azure4")+
  labs(title = "A boxplot of season and count",x="Season", y = "Count")+
  theme(plot.title = element_text(hjust = 0.5))
```



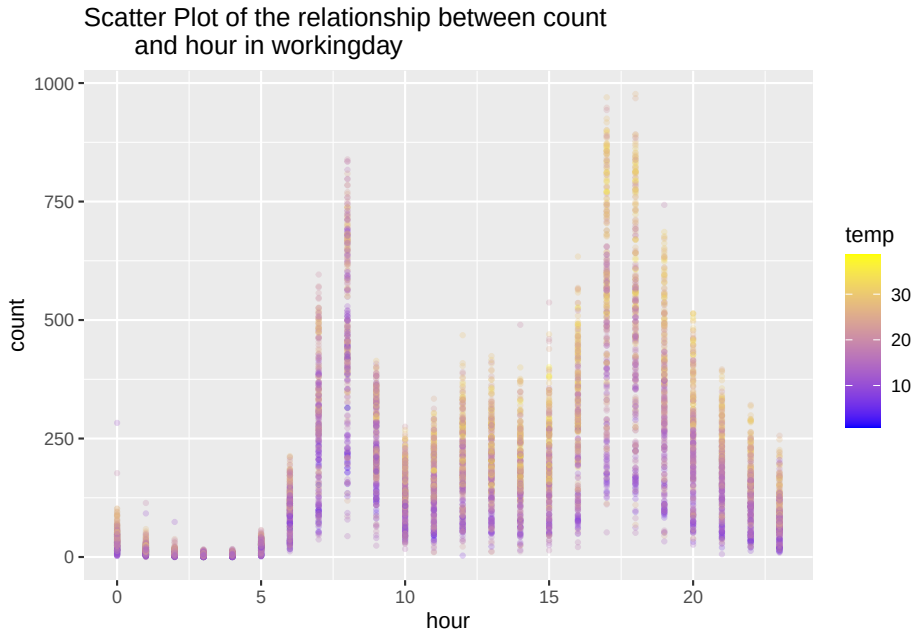
Conclusion: Summer has the highest average of bike rental counts, followed by spring and fall. The average count appears to be the smallest in winter.

2.c How does count depend on the time of the day (hour)?

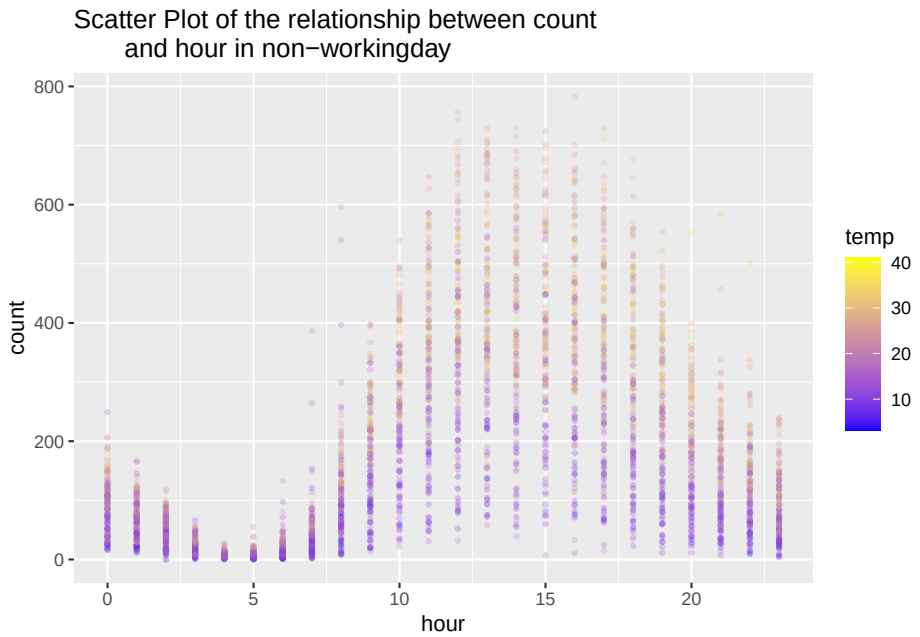
```
ggplot(bike.train,aes(x=hour,y=count))+
  geom_point(aes(color = factor(workingday)),alpha = 0.25)+
  labs(title = "Scatter Plot of the relationship between count
       and hour")+
  theme(plot.title = element_text(hjust = 0.5))
```




```
bike.train %>%
  filter(workingday == 1) %>%
  ggplot(aes(x=hour,y=count))+
  geom_point(aes(color = temp),size=0.8,alpha = 0.25)+
  labs(title = "Scatter Plot of the relationship between count
       and hour in workingday")+
   scale_color_gradient(low="blue", high="yellow")
```




```
bike.train %>%
  filter(workingday == 0) %>%
  ggplot(aes(x=hour,y=count))+
  geom_point(aes(color = temp),size=0.8,alpha = 0.25)+
  labs(title = "Scatter Plot of the relationship between count
       and hour in non-workingday")+
   scale_color_gradient(low="blue", high="yellow")
```



Conclusion: Generally speaking(figure.1.), *count* remains high during the daytime with two peaks at 8am and 17pm. It depends on whether it is a *workingday*. During working day(figure.2.) *count* presented a concave shape during daytime with the same two peaks as in figure.1. During non working day(figure.3.), *count* increases from morning to noon, keeps high during the afternoon and decreases from about 17pm till

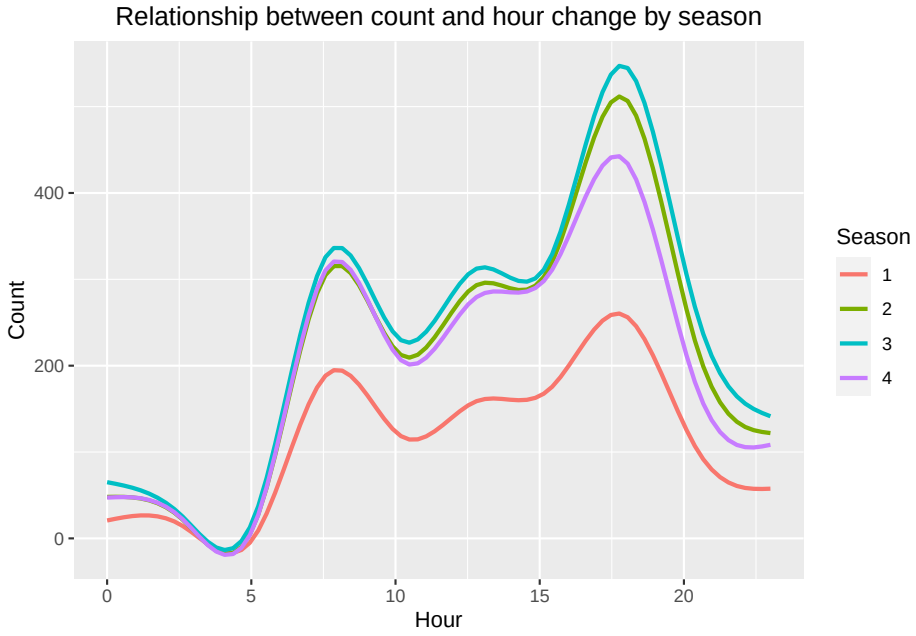
midnight and early morning.

By coloring the scatterplot using temperature, we observe that the temperature will affect hourly number of rentals: low temperatures will hinder people from renting bike. People tend to rent bike under a higher temperature.

2.d Does the relationship between count and hour change by season?

```
ggplot(data=bike.train)+
  geom_smooth(mapping=aes(hour,count,color=factor(season)),se=FALSE)+
  labs(title = "Relationship between count and hour change by season",
       x="Hour",y = "Count",col="Season")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



Conclusion: Yes, the relationship between *count* and *hour* change by *season*. From the graph, we can summarise that the relationship between *count* and *hour* remains almost the same pattern and tendency across the seasons. However, regarding the average count of bike rental, winter appear to have the least bike rental and other three seasons have a higher similar amount, with summer to be the highest cross the hours.

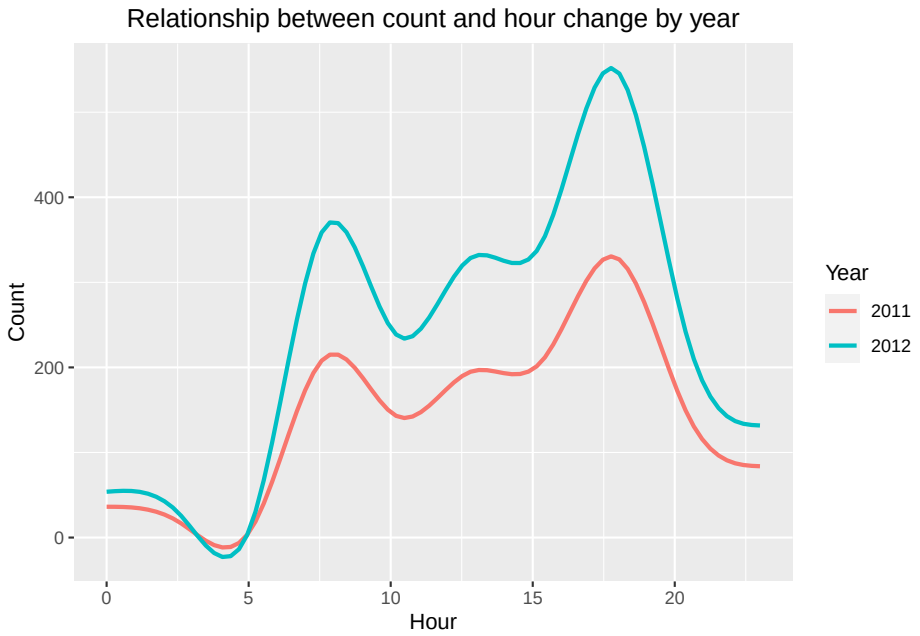
2.e Does the distribution of hourly number of rentals change between 2011 and 2012?

What does this tell you about the rental business?

```
ggplot(data=bike.train)+
  geom_smooth(mapping=aes(hour,count,color=factor(year)),se=FALSE)+
  labs(title = "Relationship between count and hour change by year",
```

```
      x="Hour", y = "Count",col="Year")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



Conclusion: Yes, the distribution of hourly number of rentals in 2012 is higher than the number in 2011 at almost all the hours, though sharing the similar pattern.

It tells us that the rental business flourished during that year because the number of rentals during the peak time almost doubles from 2011 to 2012.

Question 2

First, we explored some data and try to find any special patterns.

```
train <- data.table(bike.train)
test <- data.table(bike.test)
```

1. Deal with strange weather value

When filter *weather*, we find that there exists strange value of *weather*, such as 1.5 and 2.5. So we choose to round up and replace them with 2 and 3 respectively.

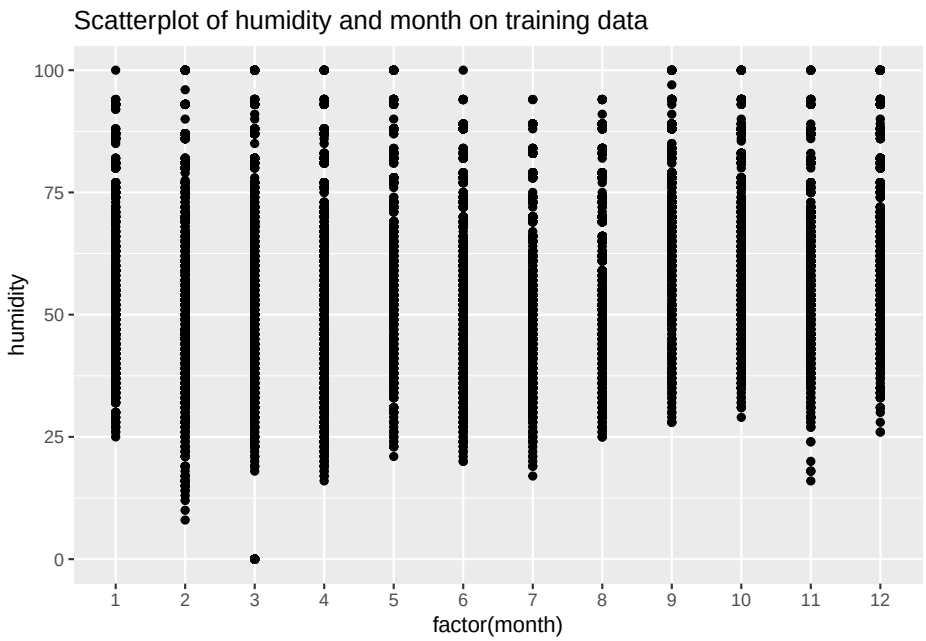
```
train<-train %>%
  mutate(weather=ifelse(weather==1.5, 2, weather)) %>%
  mutate(weather=ifelse(weather==2.5, 3, weather))
```

2. Unusual humidity & atemp & temp

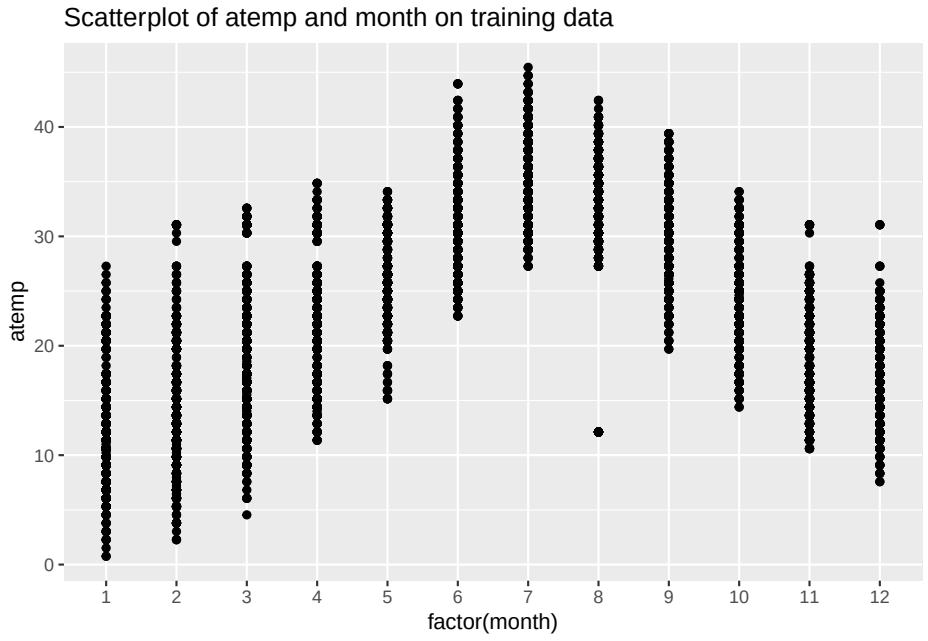
For train data:

It is obvious from the scatter plot below that there is one day in March with zero humidity record and one day in August with abnormally low temperature. A search in the training data frame reveals that their daylabels are 69 and 595, respectively. We will make average for these outliers by hours between values from the day before and the day after.

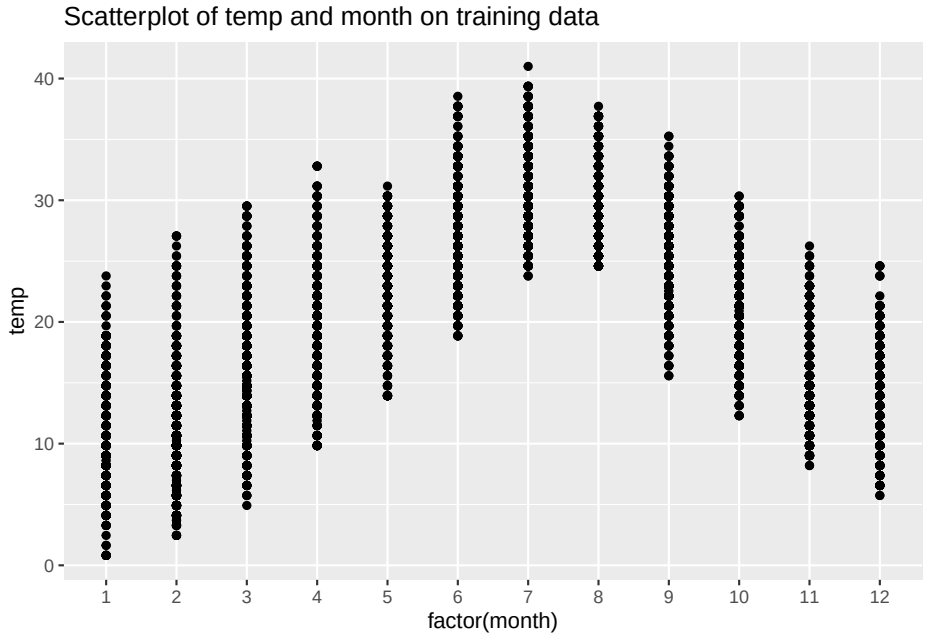
```
ggplot(bike.train)+
  geom_point(aes(y=humidity, x=factor(month)))+
  labs(title="Scatterplot of humidity and month on training data")
```



```
ggplot(bike.train)+
  geom_point(aes(y=atemp, x=factor(month)))+
  labs(title="Scatterplot of atemp and month on training data")
```




```
ggplot(bike.train)+
  geom_point(aes(y=temp, x=factor(month)))+
  labs(title="Scatterplot of temp and month on training data")
```

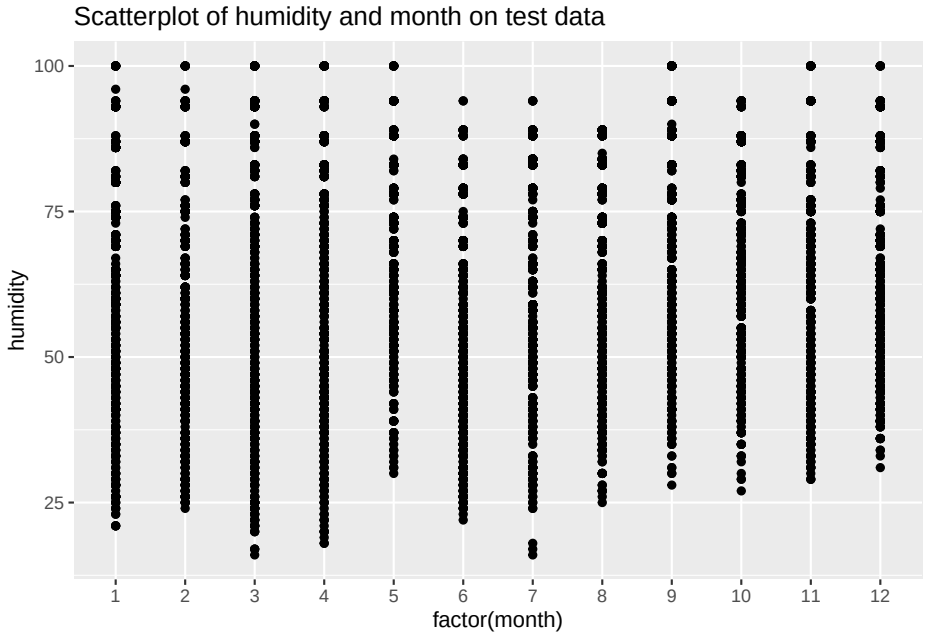


```
#For abnormal humidity & atemp: make average for abnormal humidity & atemp by hours between records fro
for (i in 0:23){
  train$humidity[which(train$daylabel == 69 & train$hour == i)] = (train$humidity[which(train$daylabel =
  train$atemp[which(train$daylabel == 595 & train$hour == i)] = (train$atemp[which(train$daylabel == 59
}
```

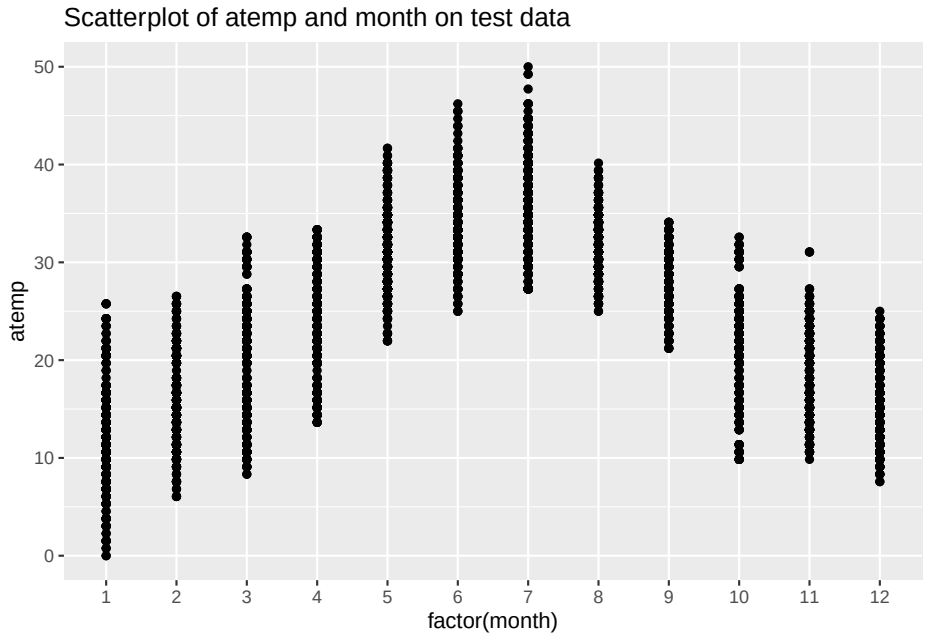
For test data:

There is no abnormal humidity, atemp and temp records.

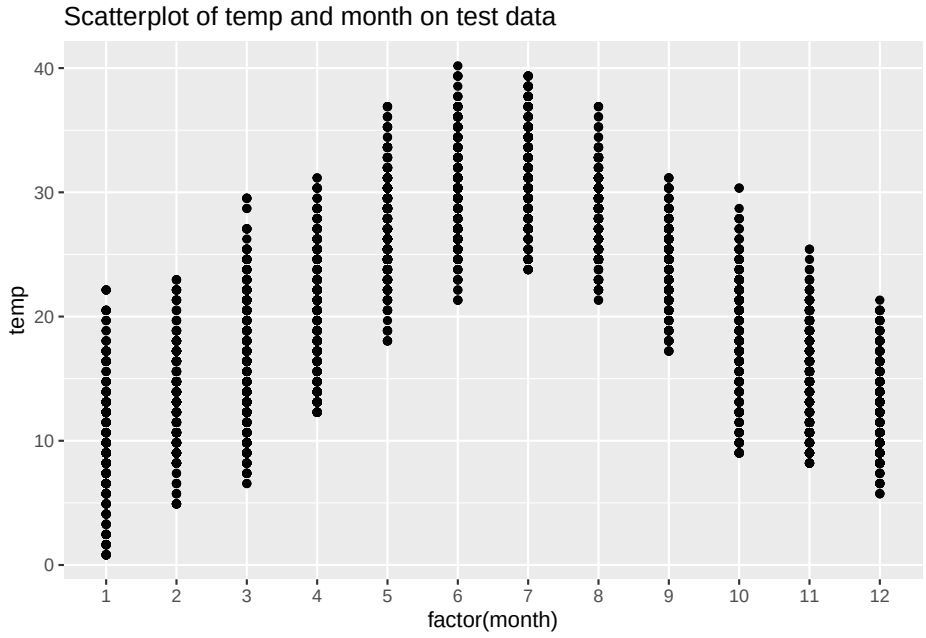
```
ggplot(bike.test)+
  geom_point(aes(y=humidity, x=factor(month)))+
  labs(title="Scatterplot of humidity and month on test data")
```




```
ggplot(bike.test)+
  geom_point(aes(y=atemp, x=factor(month)))+
  labs(title="Scatterplot of atemp and month on test data")
```




```
ggplot(bike.test)+
  geom_point(aes(y=temp, x=factor(month)))+
  labs(title="Scatterplot of temp and month on test data")
```



3. Delete abnormal windspeed

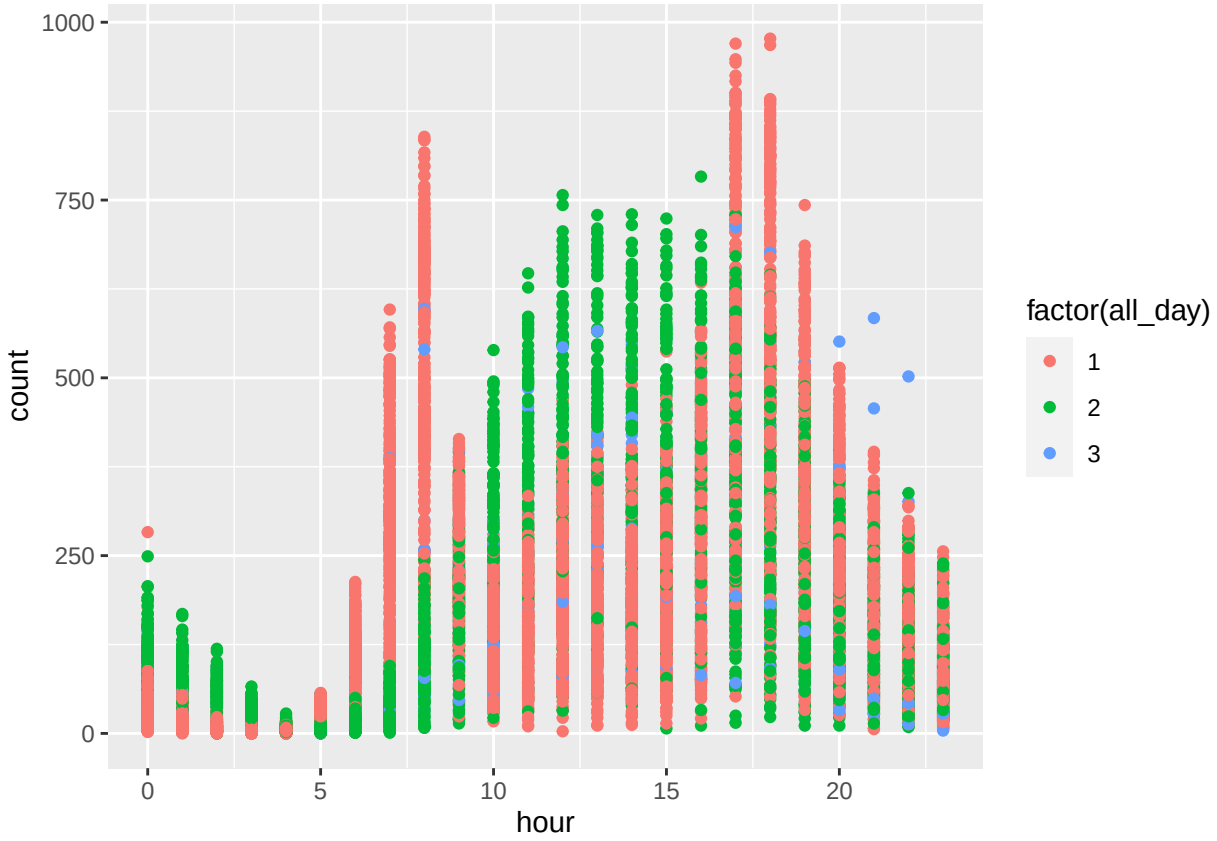
We observe two outliers from Q2.1, so we delete them.

```
train[daylabel == 184 & hour %in% c(17, 18), c("windspeed")] = NA
train <- na.omit(train)
```

4. Unusual workingday & holiday

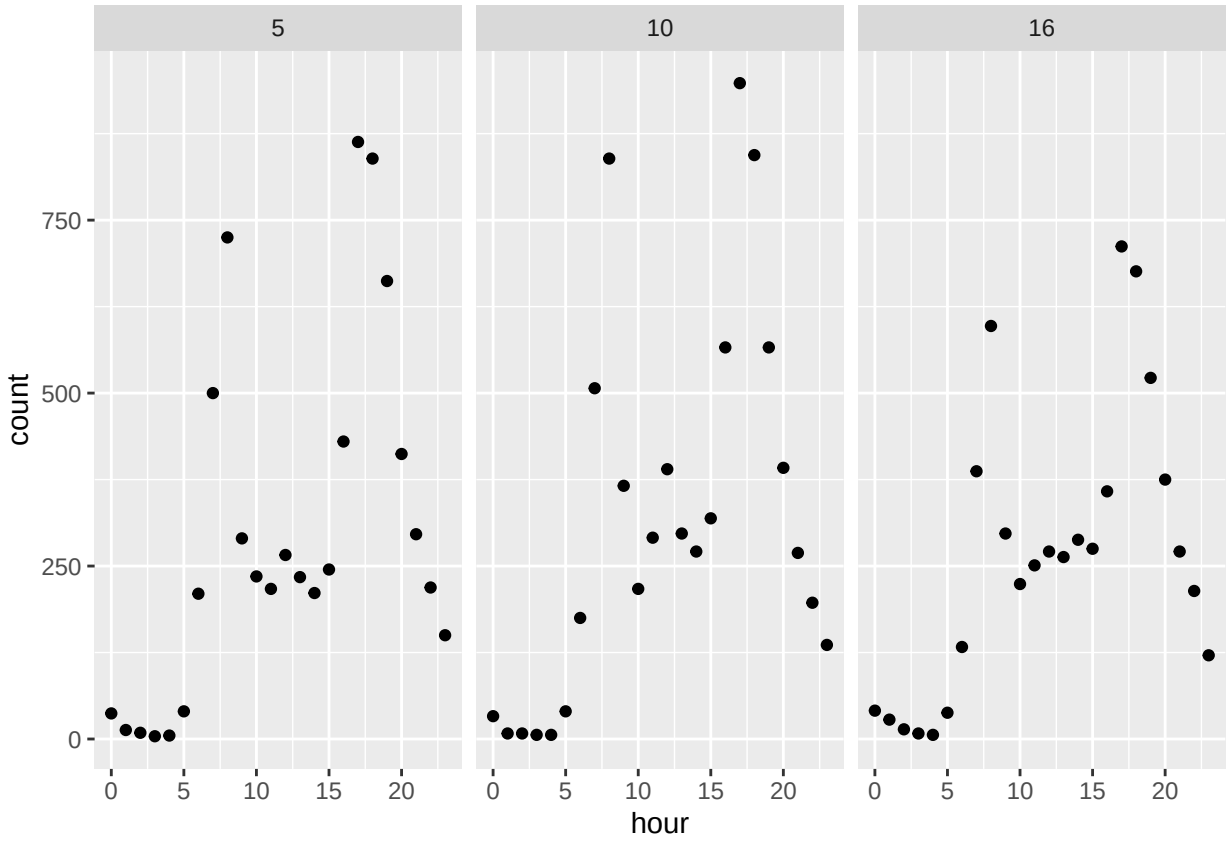
We find that there appears several outliers to have extremely high count around 8pm, which is not that normal for "holiday".

```
train %>%
  mutate(all_day=ifelse(holiday==1,3,ifelse(workingday==1,1,2))) %>%
  ggplot()+
  geom_point(aes(x=hour,y=count,col=factor(all_day)))
```



So we pick them up and find several crucial date: Sep 15th, Oct 10th and Apr 16th in 2012. They are tagged as holiday but they demonstrate a working day pattern in the plot.

```
train %>%
  filter((year==2012&month==9&day==5)|
           (year==2012&month==10&day==10)|
           (year==2012&month==4&day==16)) %>%
  ggplot()+
  geom_point(aes(x=hour,y=count ))+
  facet_wrap(~day)
```



So we changed the tag of these date from *holiday* to *workingday*.

```
train<-train %>%
  mutate(workingday=ifelse(year==2012&month==10&day==10,1,workingday),
         holiday=ifelse(year==2012&month==10&day==10,0,holiday))

train<-train %>%
  mutate(workingday=ifelse(year==2012&month==9&day==5,1,workingday),
         holiday=ifelse(year==2012&month==9&day==5,0,holiday))

train<-train %>%
  mutate(workingday=ifelse(year==2012&month==4&day==16,1,workingday),
         holiday=ifelse(year==2012&month==4&day==16,0,holiday))
```

5. Transform daylabel to weekday

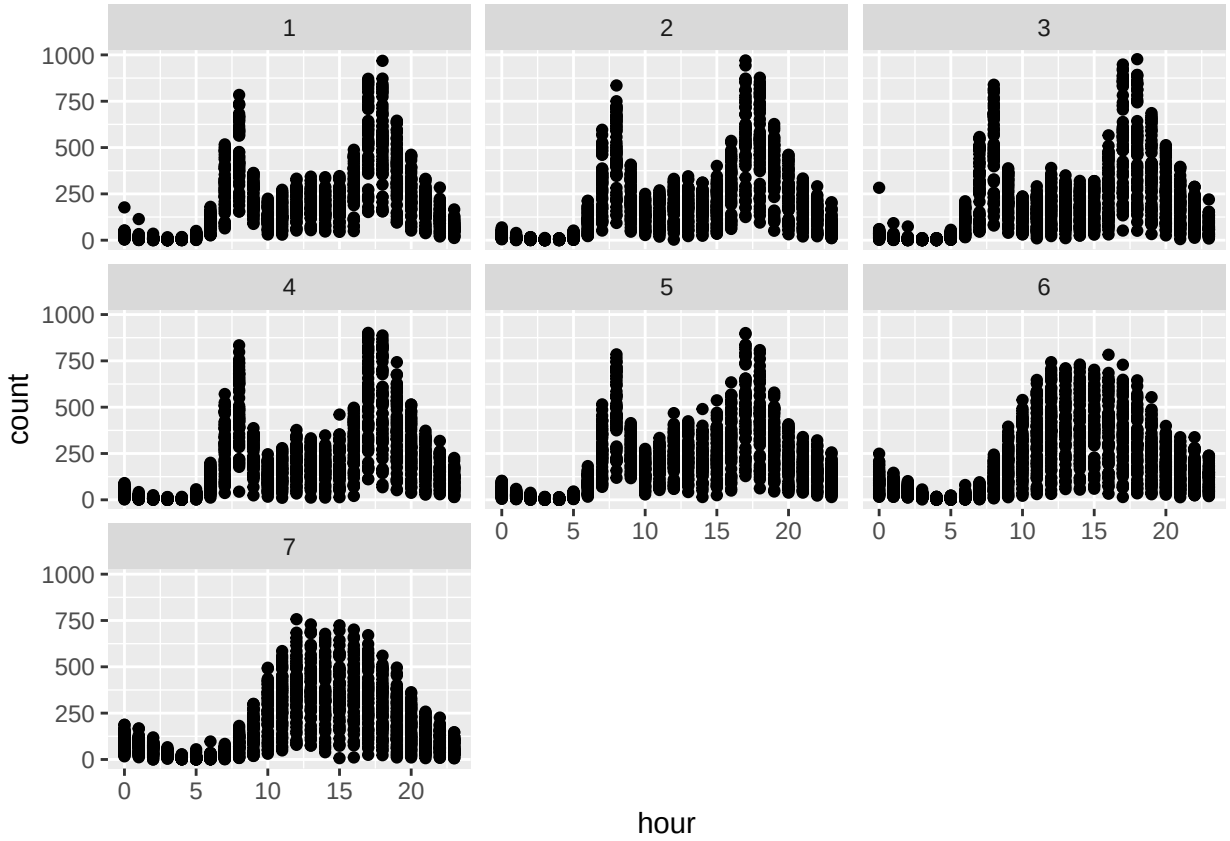
After finding certain holiday to have same pattern of working day, we then came up with idea that every working day might have different traits. For example, people may feel more casual on friday and thus rent more bike. So we add a column named *weekday* to see if they have any special.

```
train<-train %>%
  mutate(weekday=ifelse((daylabel-3)%%7==0,1,
                 ifelse((daylabel-3)%%7==1,2,
                 ifelse((daylabel-3)%%7==2,3,
                 ifelse((daylabel-3)%%7==3,4,
```

```
                ifelse((daylabel-3)%%7==4,5,
                ifelse((daylabel-3)%%7==5,6,7)))))) %>%
  select(-1)              #drop daylabel
```

Then we draw seven plots from Monday(1) to Sunday(7):

```
train %>%
  filter(holiday==0) %>%
  ggplot()+
  geom_point(aes(x=hour,y=count))+
  facet_wrap(~ weekday)
```



We can easily oberve that two weekends day follow the exact pattern just like in Q2.c. However, weekdays, though sharing quite similar shape, Monday seems to have higher peak and lower concave part while Friday is lean to the weekend pattern. So we decide to add *weekday* as a new variable and drop *daylabel* because it does not offer any information.

6. Transform year, month, season, weather into dummies

When exploring the data, we also find out that there are some categorical variables, which should be transform into dummy variables when running the models. So we choose to transform *year*, *month*, *season*, *weather* into dummies. The reason why we don't transform *hour* is because it will significantly decrease the speed of running the model. And we can also interpret *hour* as a numeric variable. For example, 0.5 hour means 30 minutes.

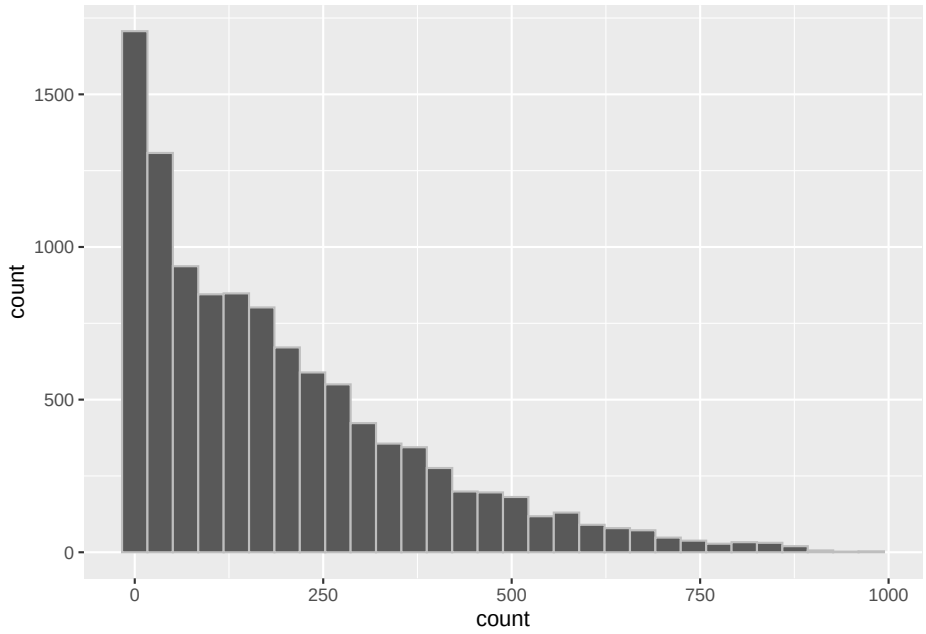
```
train = dummy_cols(train, select_columns = c("year","month","season","weather"), remove_selected_columns
test = dummy_cols(test, select_columns = c("year","month","season","weather"), remove_selected_columns =
```

7. Transform Count to log format

The histogram of *count* variable shows a notable right-skewed pattern. So we choose to use the log format of the *count*. If we directly use *log(count)*, there may be negative infinity so we use *log(count+1)*.

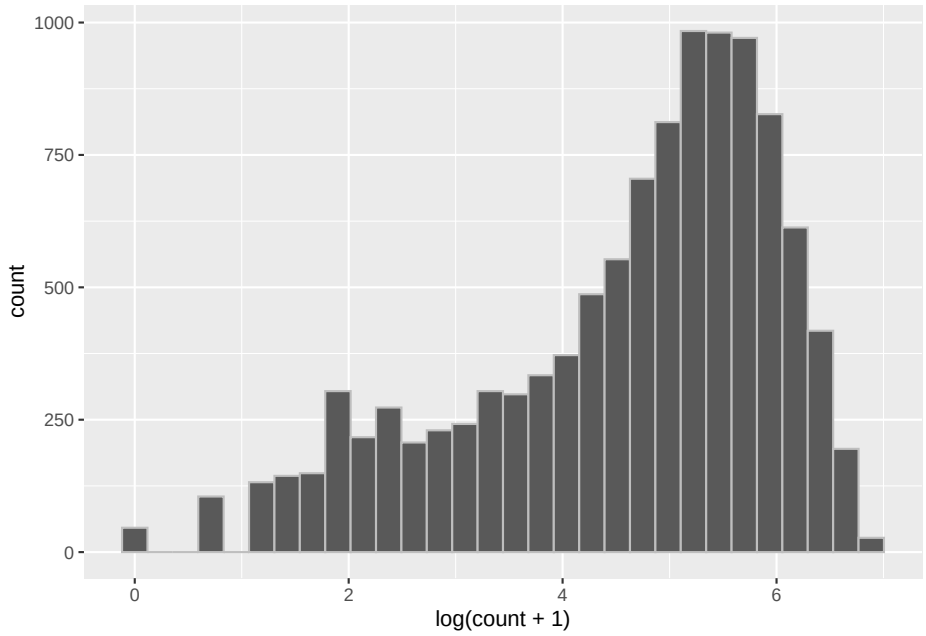
```
### count log format
ggplot(train)+
  geom_histogram(aes(x=count), color = "grey")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(train)+
  geom_histogram(aes(x=log(count+1)), color = "grey")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
#log count
train = train %>%
  mutate(log_count=log(count + 1))
train = train[,-c("count")]
```

8. Model

Random Forest

```
tic()
#Creat a grid first
hyper_grid <- expand.grid(
  mtry       = seq(9, 30, by = 2),
  node_size  = c(5,25,50),
  sample_size = c(.55, .632, .80),
  OOB_RMSE   = 0
)

for(i in 1:nrow(hyper_grid)) {

  # train model
  RF <- ranger(
    formula         = log_count ~ .,
    data            = train,
    num.trees       = 1000,
    mtry            = hyper_grid$mtry[i],
    min.node.size   = hyper_grid$node_size[i],
    sample.fraction = hyper_grid$sample_size[i],
    seed            = 1108
  )

  # add OOB error to grid
  hyper_grid$OOB_RMSE[i] <- sqrt(RF$prediction.error)
}

(oo = hyper_grid %>%
  dplyr::arrange(OOB_RMSE) %>%
  head(10))
toc()
```

```
rf.fit.final <- ranger(
    formula         = log_count ~ .,
    data            = train,
    num.trees       = 1000,
    mtry            = oo[1,]$mtry,
    min.node.size   = oo[1,]$node_size,
    sample.fraction = oo[1,]$sample_size,
    importance      = 'impurity'
    )

yhat.rf = exp(predict(rf.fit.final, data = test)$predictions) -1
```

Boosting

Transform data frame to matrices:

```
X.train = as.matrix(train[,-9])
Y.train = train$log_count

X.test = as.matrix(test)
```

```
#Create a grid
hyper_grid <- expand.grid(
  shrinkage = c(.001, .005, .01,),     ## controls the learning rate
  interaction.depth = c(5, 10), ## tree depth
  n.minobsinnode = c(5, 10, 15), ##  minimum number of observations required in each terminal node
  bag.fraction = c(.5, .632, .9),  ##  percent of training data to sample for each tree
  optimal_trees = 0,               ## a place to dump results
  min_RMSE = 0                     ## a place to dump results
  )



for(i in 1:nrow(hyper_grid)) {

  # create parameter list
  params <- list(
    eta = hyper_grid$shrinkage[i],
    max_depth = hyper_grid$interaction.depth[i],
    min_child_weight = hyper_grid$n.minobsinnode[i],
    subsample = hyper_grid$bag.fraction[i]
  )

  # reproducibility
  set.seed(1108)

  # train model using Cross Validation
  xgb.tune <- xgb.cv(
    params = params,
    data = X.train,
    label = Y.train,
    nrounds = 3000, # number of trees
    nfold = 5,
    objective = "reg:squarederror",     # for regression models
    verbose = 0,                        # silent,
    early_stopping_rounds = 10          # stop if no improvement for 10 consecutive trees
  )

  # add min training error and trees to grid
  hyper_grid$optimal_trees[i] <- which.min(xgb.tune$evaluation_log$test_rmse_mean)
  hyper_grid$min_RMSE[i] <- min(xgb.tune$evaluation_log$test_rmse_mean)
}

(oo = hyper_grid %>%
      dplyr::arrange(min_RMSE) %>%
      head(10))
```

```r
# parameter list
params <- list(
  eta = oo[1,]$shrinkage,
  max_depth = oo[1,]$interaction.depth,
  min_child_weight = oo[1,]$n.minobsinnode,
  subsample = oo[1,]$bag.fraction
)

# train final model
xgb.fit.final <- xgboost(
  params = params,
  data = X.train,
  label = Y.train,
  nrounds = oo[1,]$optimal_trees,
  objective = "reg:squarederror",
  verbose = 0
)

yhat.xgb <- exp(predict(xgb.fit.final, newdata=X.test)) - 1
```

Conclusion: We choose boosting model to be the final model because it generates a smaller rmse in either validation or test process.

9. Appendix

```r
#sampleSubmission is a dataframe with columns 'Id' and 'count'
sampleSubmission = data.frame(Id=1:length(yhat.xgb), count=yhat.xgb)
write.csv(sampleSubmission,
          file = "sampleSubmission_xgb.csv",
          row.names = FALSE,
          quote = FALSE)
```