Bike Share Demand Forecast

November 27, 2019

[ ]:
```
# ----------- Step 1a: Define and categorize the problem statement␣
 ↪--------------

# The problem statement is to "Predict the daily bike rental count based on the␣
 ↪environmental and seasonal settings"
# This is clearly a 'Supervised machine learning regression problem' to predict␣
 ↪a number based on the input features.

# ----------- Step 1a ends here -----------------
```

[1]:
```
# ----------Step 1b: Import all the required libraries ----------

#---- for data transformations----
    #install.packages("lubridate")
    library(lubridate)

#---- for EDA Visualizations ------
    #install.packages("corrplot")
    library(corrplot)
    #install.packages("ggplot2")
    library(ggplot2)
    #install.packages("GGally")
    library(GGally)
    #install.packages("ggExtra")
    library(ggExtra)

#---- for model building----
    library(caret)
    #install.packages("Metrics")
    library(Metrics)
    #install.packages("randomForest")
    library(randomForest)
    #install.packages(gbm)
    library (gbm)


# ------------------- Step 1b ends here -------------------------
```

```
Attaching package: lubridate

The following object is masked from package:base:

    date

corrplot 0.84 loaded
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
Loading required package: lattice

Attaching package: Metrics

The following objects are masked from package:caret:

    precision, recall

randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.

Attaching package: randomForest

The following object is masked from package:ggplot2:

    margin

Loaded gbm 2.1.5
```

[2]:
```
# ------------------- Step 2: Gather the data -----------------

  # Data is provided as .csv file and already split into Test and Train.
  # The training set is comprised of the first 19 days of each month, while the␣
 ↪test set is the 20th to the end of the month.
  # Let's import the data
    bike= read.csv("/Users/snehashrungarpawar/Documents/Master in Data Science/
 ↪DPA/Project/Data/train.csv", header=TRUE)
    bike_test = read.csv("/Users/snehashrungarpawar/Documents/Master in Data␣
 ↪Science/DPA/Project/Data/test.csv", header=TRUE)
# ------------------- Step 2 ends here -------------------------
```

[ ]:
```
# ------------------- Step 3: Data Preparation ---------------------
  # 3a. Analyze Attributes: Check properties of data
  # 3b. Complete Data Perform missing value analysis and Impute if needed
  # 3c. Correct Data: Check for any invalid data points
  # 3d. Create Derived Attributes - Feature Extraction
  # 3e. Convert - Converting data to proper formats
```

[3]:
```
# 3a. Analyze Attributes: Check properties of data
      dim(bike)
      str(bike)
      head(bike, 10)
  # 3a -> Inference:
        #i. The dataset has 10,886 observations (n=10886) and 12 columns of␣
 ↪type int, num and factor.
        #ii. Season, Holiday, Working day and weather are categorical variables.
        #ii. temp, atemp, humidity, windspeed, casual, registered and count are␣
 ↪continuous numerical variables.
```

1. 10886 2. 12

```
'data.frame':   10886 obs. of  12 variables:
 $ datetime  : Factor w/ 10886 levels "2011-01-01 00:00:00",..: 1 2 3 4 5 6 7 8
9 10 ...
 $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ workingday: int  0 0 0 0 0 0 0 0 0 0 ...
 $ weather   : int  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 ...
```

datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000
2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000
2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000
2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000
2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000
2011-01-01 05:00:00	1	0	0	2	9.84	12.880	75	6.0032
2011-01-01 06:00:00	1	0	0	1	9.02	13.635	80	0.0000
2011-01-01 07:00:00	1	0	0	1	8.20	12.880	86	0.0000
2011-01-01 08:00:00	1	0	0	1	9.84	14.395	75	0.0000
2011-01-01 09:00:00	1	0	0	1	13.12	17.425	76	0.0000

[4]:
```
# 3b. Complete Data Perform missing value analysis and Impute if needed
      table(is.na(bike))
  # 3b -> Inference: There are no null values in the dataset. If it had, then␣
 ↪either the rows/columns had to be
    # dropped or the null values be imputed based on the % of null values
```

```
 FALSE
130632
```

[5]:
```
# 3c. Correct Data: Check for any invalid data points
    # From above observations data doesnot seem to have any invalid datatypes␣
↪to be handled.
    # Let's check for the outliers in EDA step
```

[6]:
```
# 3d. Create Derived Attributes - Feature Extraction
      # Lets extract 'date','month','weekday' and 'year' from 'datetime' column␣
↪as we will be needing it for analysis
      bike$date=as.factor(day(bike$datetime))
      bike$year = as.factor(year(bike$datetime))
      bike$month = as.factor(month(bike$datetime))
      bike$hour = as.factor(hour(bike$datetime))
      bike$wkday = as.factor(wday(bike$datetime))

      bike_test$date=as.factor(day(bike_test$datetime))
      bike_test$year = as.factor(year(bike_test$datetime))
      bike_test$month = as.factor(month(bike_test$datetime))
      bike_test$hour = as.factor(hour(bike_test$datetime))
      bike_test$wkday = as.factor(wday(bike_test$datetime))

      # Drop datetime as we have extracted all the above needed information␣
↪from it
      bike = bike[-c(1)]
      bike_test = bike_test[-c(1)]

      head(bike, 5)
      head(bike_test, 5)

  # 3d -> Inference: There are no null values in the dataset. If it had, then␣
↪either the rows/columns had to be
                  #dropped or the null values be imputed based on the % of␣
↪null values.
```

season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered
1	0	0	1	9.84	14.395	81	0	3	13
1	0	0	1	9.02	13.635	80	0	8	32
1	0	0	1	9.02	13.635	80	0	5	27
1	0	0	1	9.84	14.395	75	0	3	10
1	0	0	1	9.84	14.395	75	0	0	1

season	holiday	workingday	weather	temp	atemp	humidity	windspeed	date	year	month
1	0	1	1	10.66	11.365	56	26.0027	20	2011	1
1	0	1	1	10.66	13.635	56	0.0000	20	2011	1
1	0	1	1	10.66	13.635	56	0.0000	20	2011	1
1	0	1	1	10.66	12.880	56	11.0014	20	2011	1
1	0	1	1	10.66	12.880	56	11.0014	20	2011	1

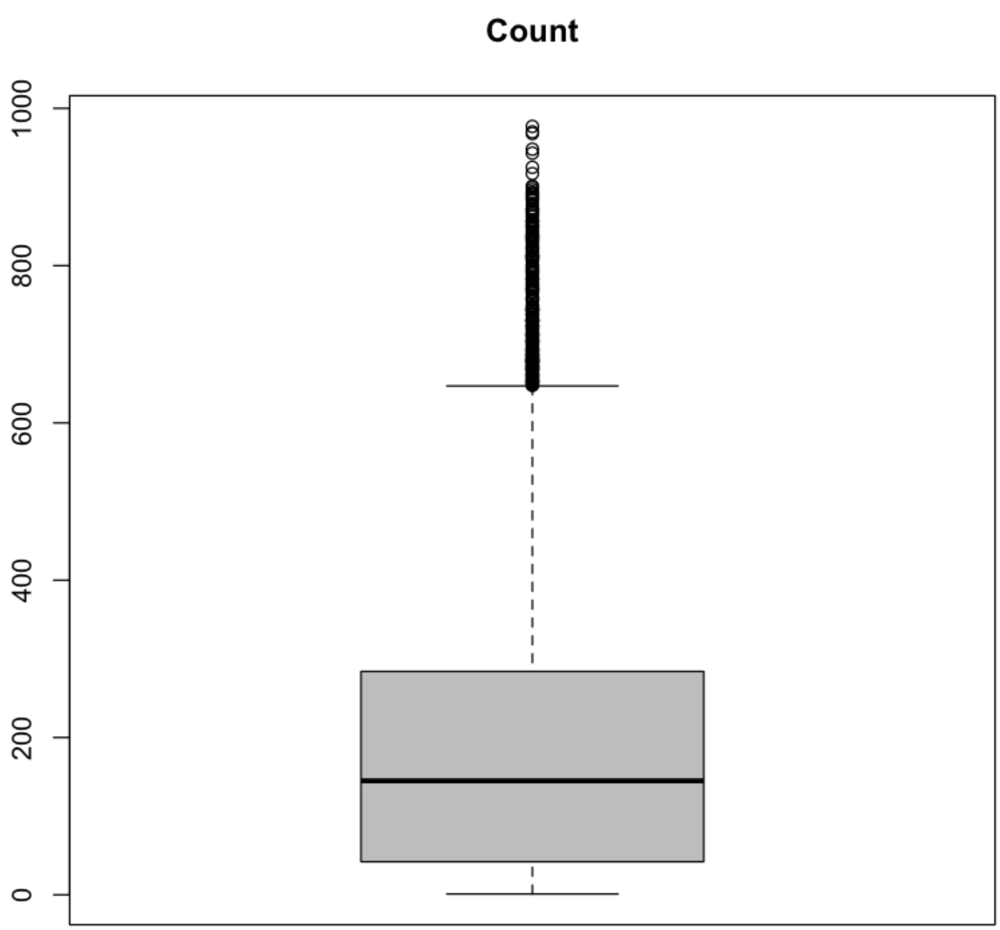
[7]:
```
# 3e. Convert - Converting data to proper formats
```

```
    # We can clearly see that "season", "holiday", "workingday" and"weather"␣
↪are categories rather than continous variable.
    # Let's convert them to categories
      names = c("season", "holiday", "workingday", "weather")
      bike[,names] = lapply(bike[,names], factor)
      bike_test[,names] = lapply(bike_test[,names], factor)

      str(bike)
      str(bike_test)

# ------------------- Step 3: Data Preparation ends here␣
↪--------------------------
```

```
'data.frame':   10886 obs. of  16 variables:
 $ 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 ...
 $ date      : Factor w/ 19 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ 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 ...
 $ wkday     : Factor w/ 7 levels "1","2","3","4",..: 7 7 7 7 7 7 7 7 7 7 ...
'data.frame':   6493 obs. of  13 variables:
 $ 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": 2 2 2 2 2 2 2 2 2 2 ...
 $ weather   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 2 ...
 $ temp      : num  10.7 10.7 10.7 10.7 10.7 ...
 $ atemp     : num  11.4 13.6 13.6 12.9 12.9 ...
 $ humidity  : int  56 56 56 56 56 60 60 55 55 52 ...
 $ windspeed : num  26 0 0 11 11 ...
 $ date      : Factor w/ 12 levels "20","21","22",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ 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 ...
 $ wkday     : Factor w/ 7 levels "1","2","3","4",..: 5 5 5 5 5 5 5 5 5 5 ...
```

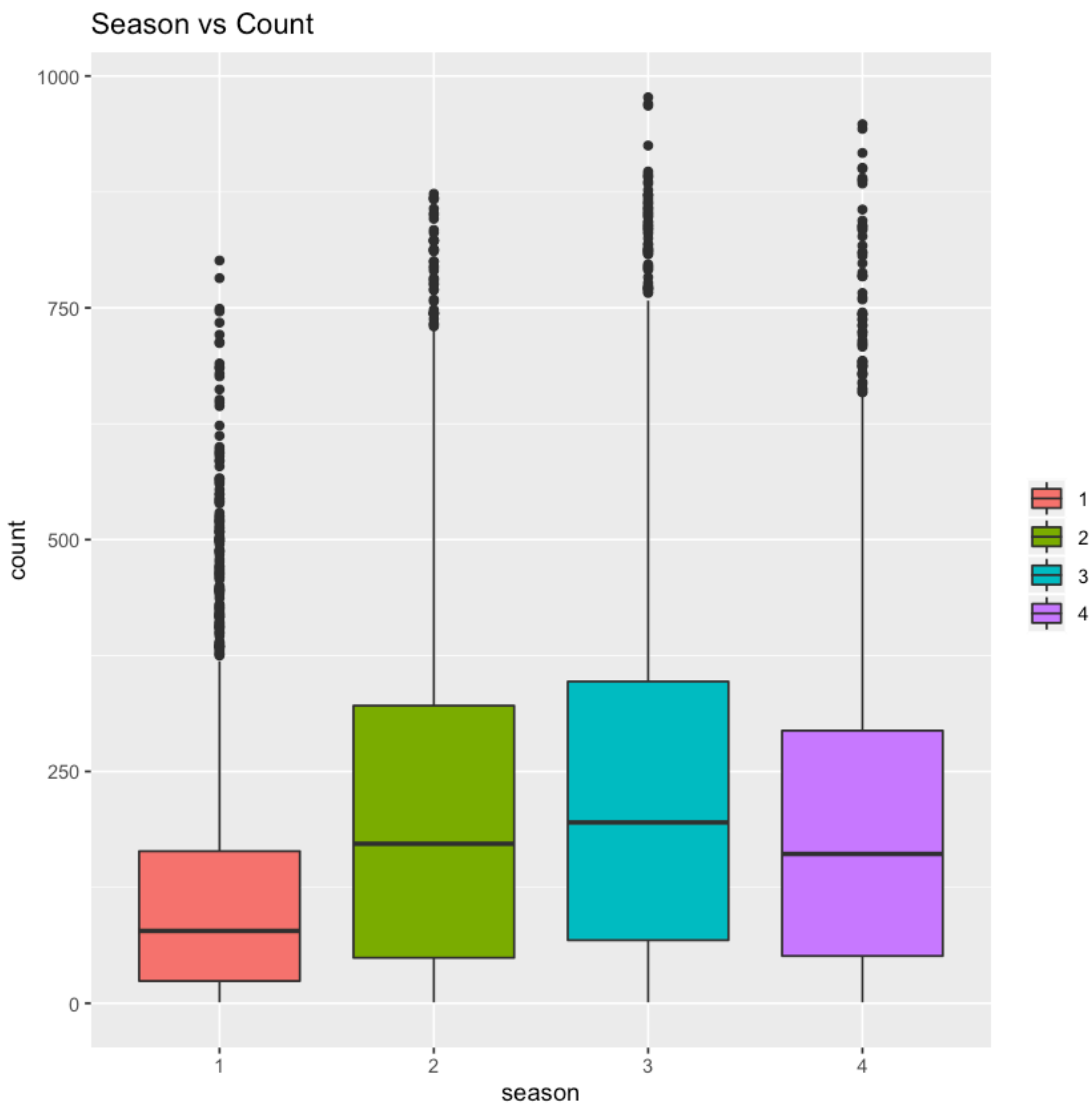
[27]:
```
# ------------- Step 4: Exploratory Data Analysis -----------
    # 4a. Outlier Analysis
```

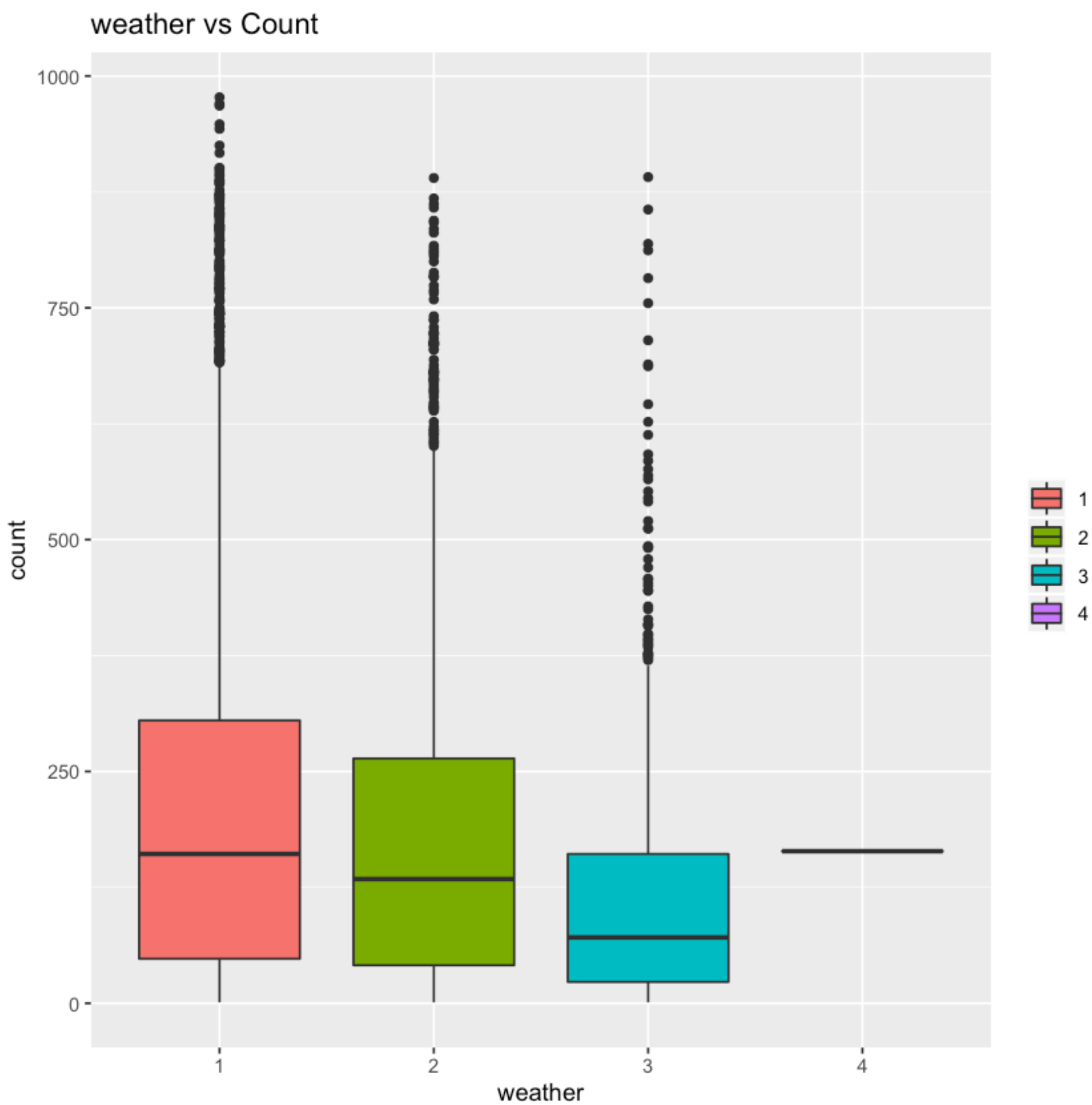
```
# 4a(1). Visualize continuos variables
        par(mfrow=c(1,5))
        boxplot(bike$count, main="Count", col="Gray", border = "black")
        boxplot(bike$temp, main="Temperature", col="blue", border = "black")
        boxplot(bike$atemp, main="Feels Like Temp", col="purple", border =␣
↪"black")
        boxplot(bike$humidity, main="Humidity", col="green", border = "black")
        boxplot(bike$windspeed, main="Windspeed", col="orange", border =␣
↪"black")
```



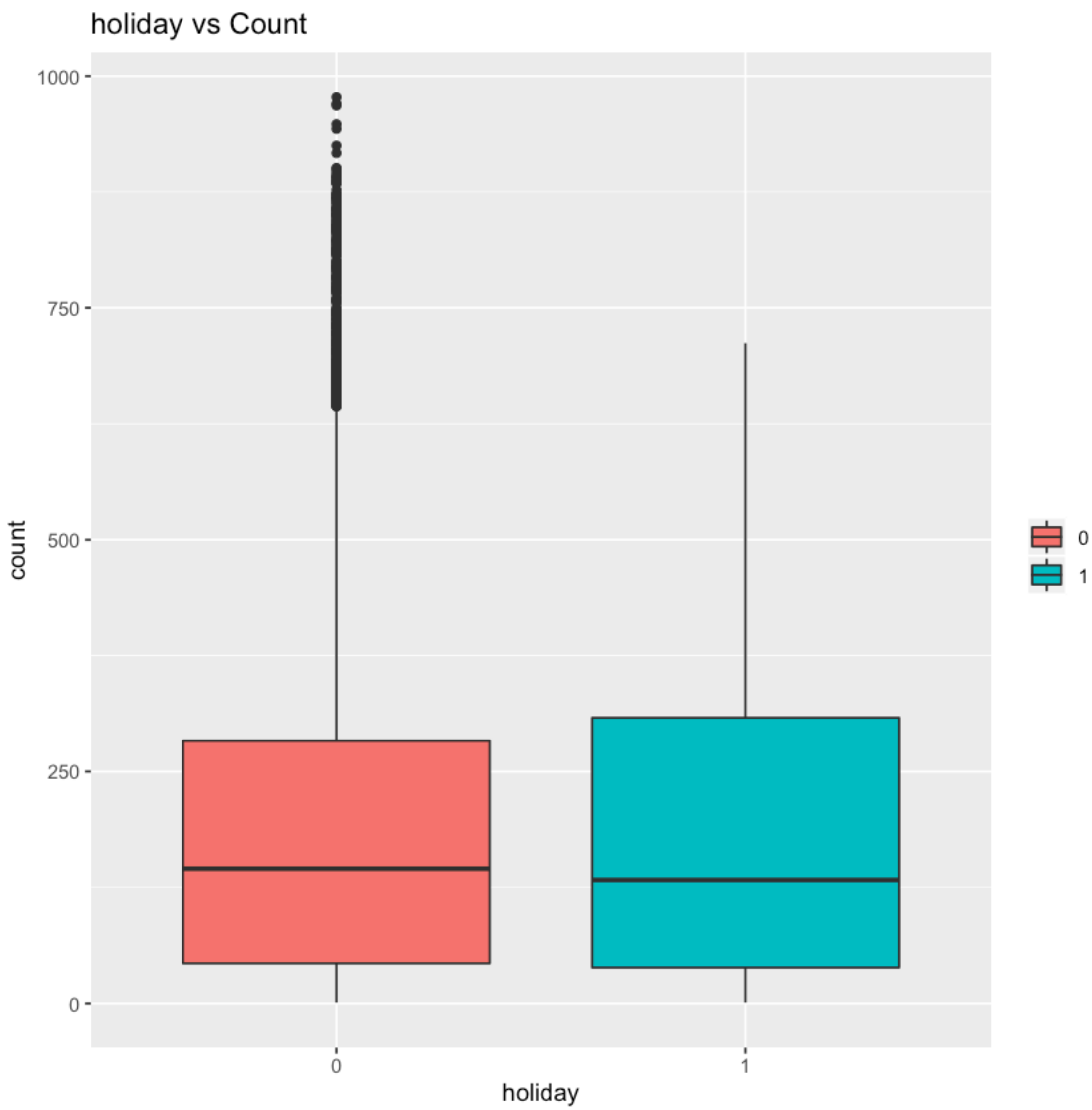
[9]:
```
# 4a(2). Visualize categorical variables wrt target variable
    par(mfrow=c(3,4))
```

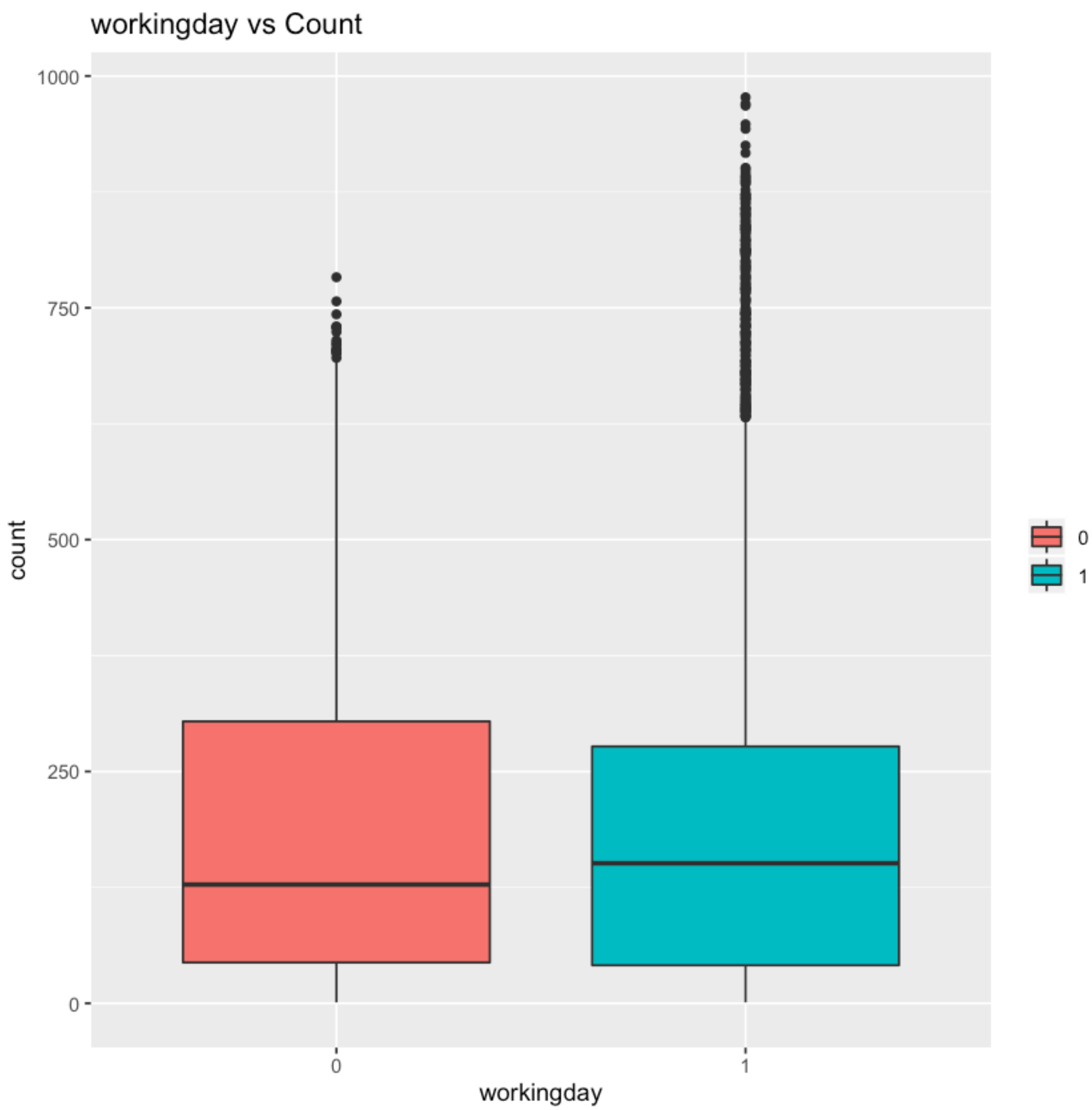
```r
ggplot(data = bike, aes(x=season, y=count, fill=as.factor(season))) + geom_boxplot() + labs(title="Season vs Count") + theme(legend.title = element_blank())
ggplot(data = bike, aes(x=weather, y=count, fill=as.factor(weather))) + geom_boxplot() + labs(title="weather vs Count") + theme(legend.title = element_blank())
ggplot(data = bike, aes(x=holiday, y=count, fill=as.factor(holiday))) + geom_boxplot() + labs(title="holiday vs Count") + theme(legend.title = element_blank())
ggplot(data = bike, aes(x=workingday, y=count, fill=as.factor(workingday))) + geom_boxplot() + labs(title="workingday vs Count") + theme(legend.title = element_blank())
ggplot(data = bike, aes(x=year, y=count, fill=as.factor(year))) + geom_boxplot() + labs(title="year vs Count") + theme(legend.title = element_blank())
ggplot(data = bike, aes(x=month, y=count, fill=as.factor(month))) + geom_boxplot() + labs(title="month vs Count") + theme(legend.title = element_blank())
ggplot(data = bike, aes(x=wkday, y=count, fill=as.factor(wkday))) + geom_boxplot() + labs(title="weekday vs Count") + theme(legend.title = element_blank())
ggplot(data = bike, aes(x=hour, y=count, fill=as.factor(hour))) + geom_boxplot() + labs(title="hour vs Count") + theme(legend.title = element_blank())
ggplot(data = bike, aes(x=date, y=count, fill=as.factor(day(date)))) + geom_boxplot() + labs(title="date vs Count") + theme(legend.title = element_blank())
```


Season vs Count
1000
750
500
250
0
count
1
2
3
4
season
1
2
3
4

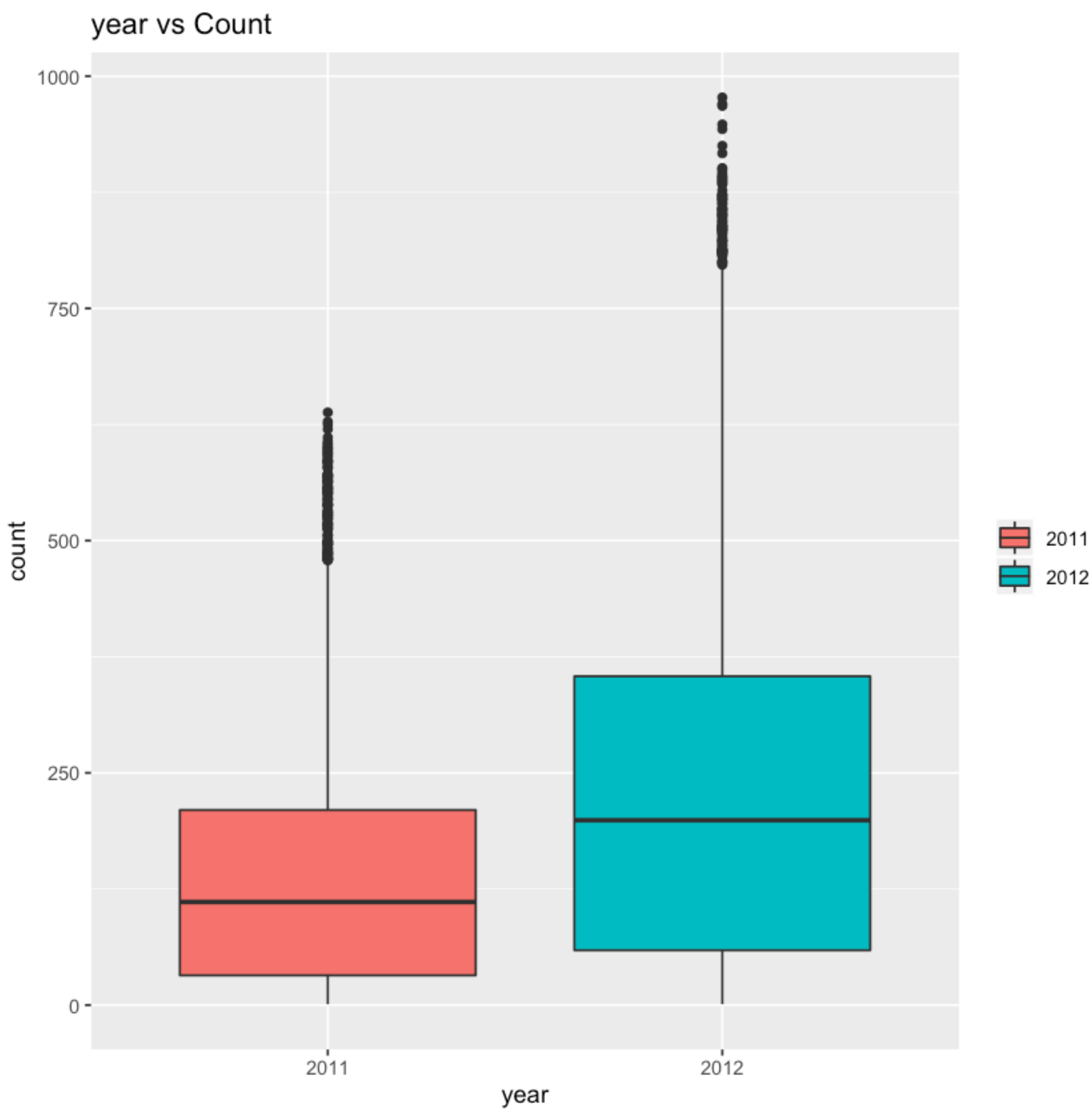


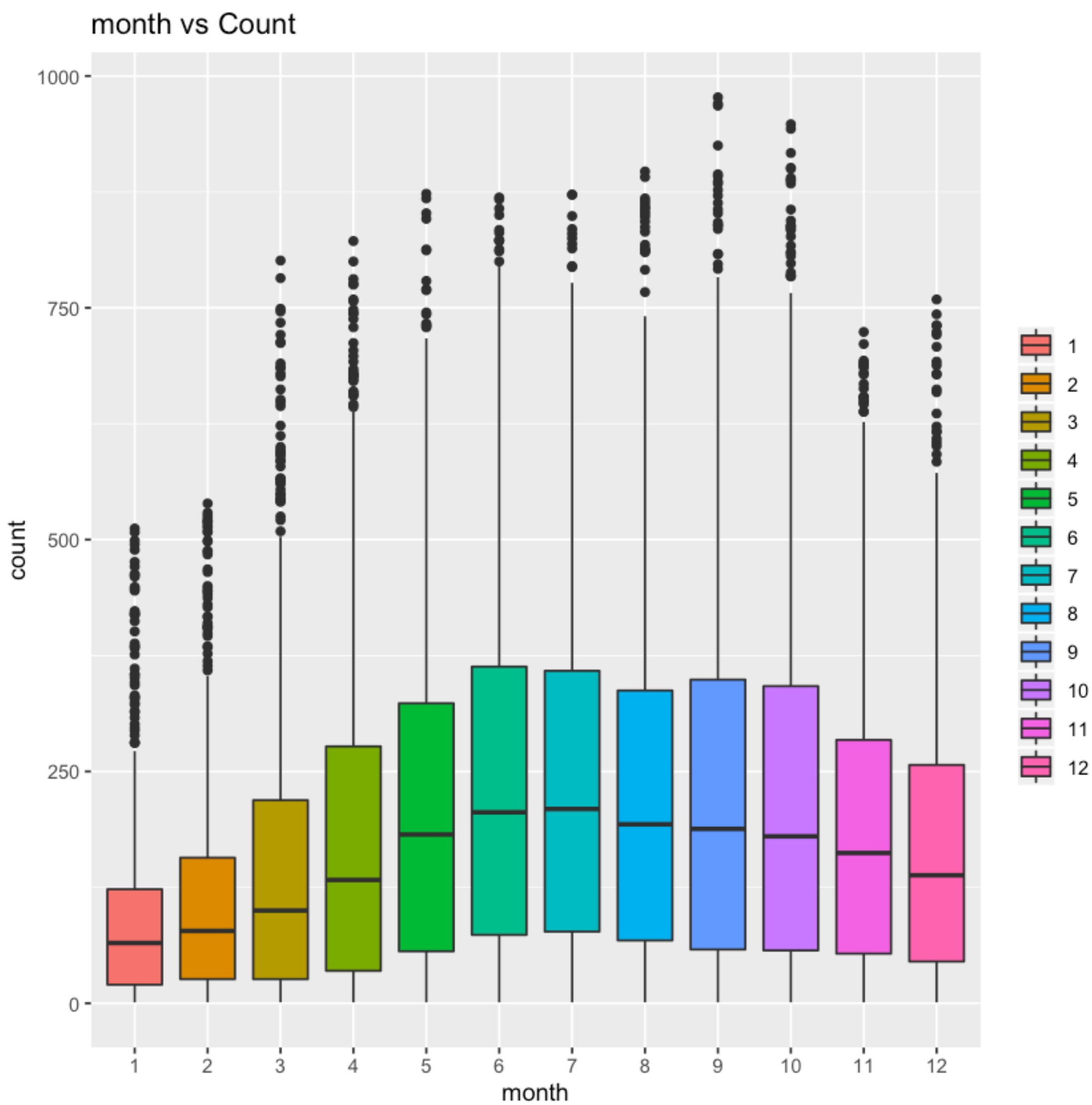
weather vs Count
1000
750
500
250
0
count
1
2
3
4
weather
1
2
3
4

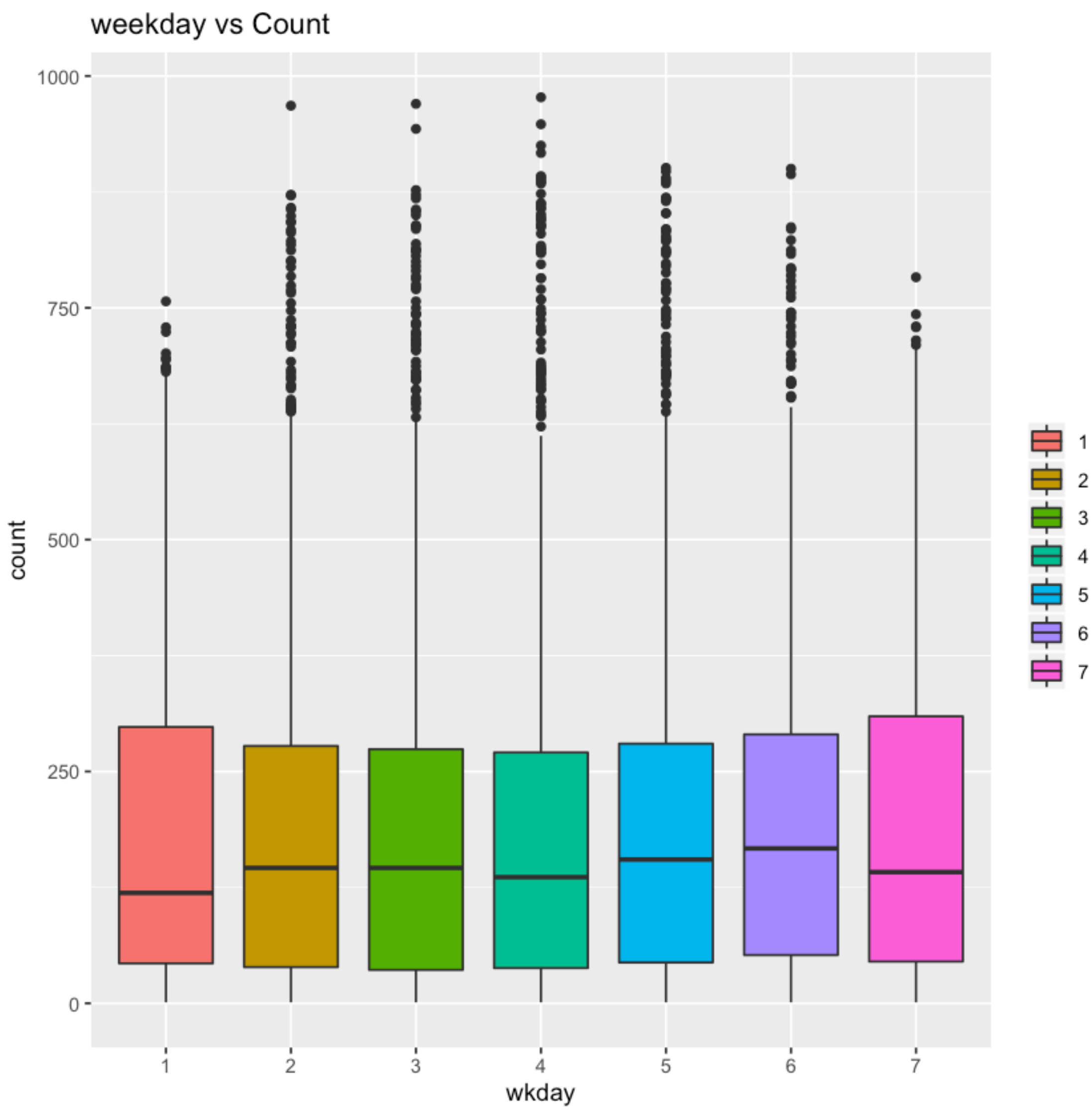

holiday vs Count
1000
750
500
250
0
count
0
1
holiday
0
1



workingday vs Count
1000
750
500
250
0
count
0
1
workingday
0
1

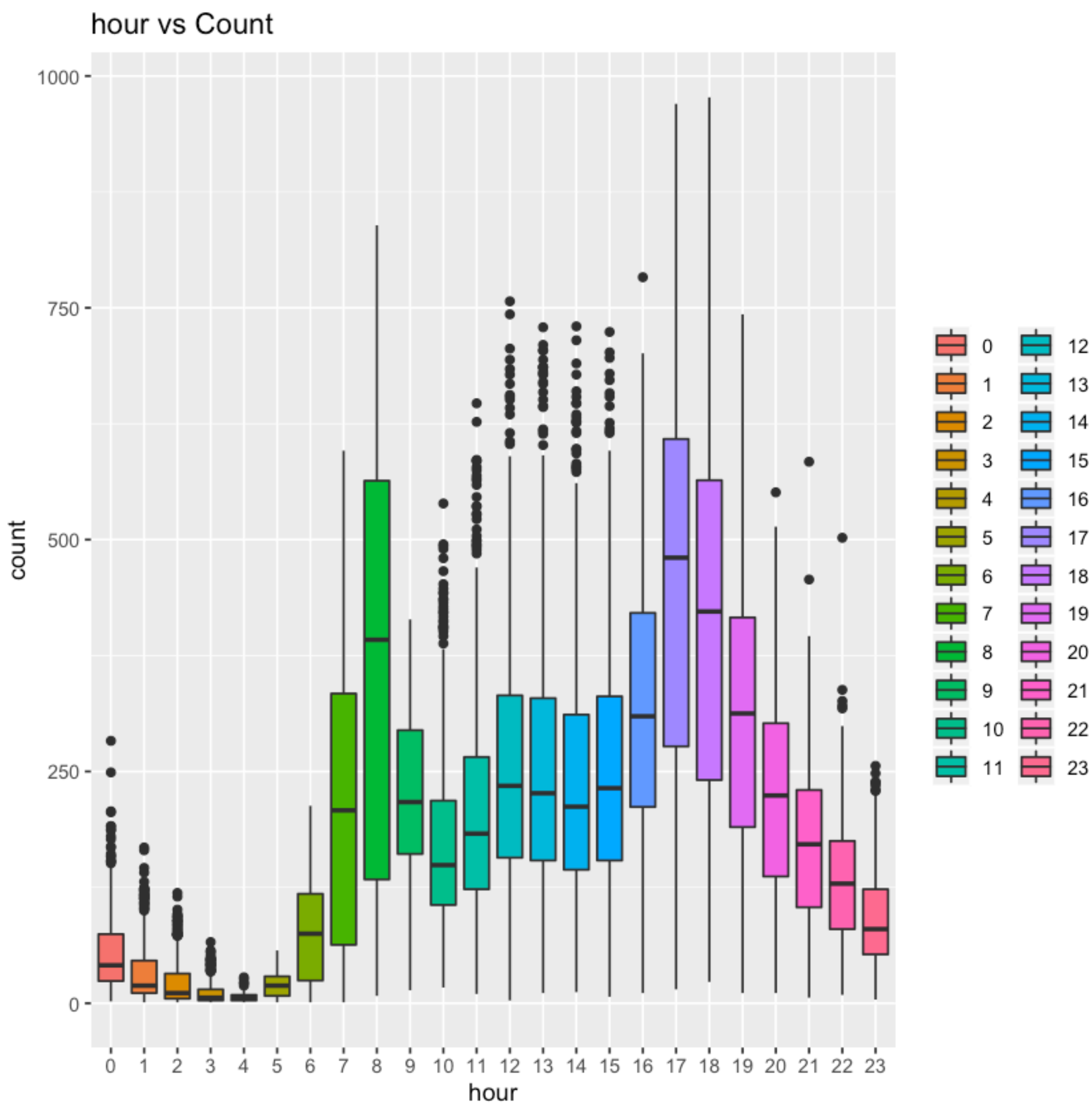

year vs Count
1000
750
500
250
0
count
2011
2012
year
2011
2012


month vs Count
1000
750
500
250
0
count
1
2
3
4
5
6
7
8
9
10
11
12
month
1
2
3
4
5
6
7
8
9
10
11
12



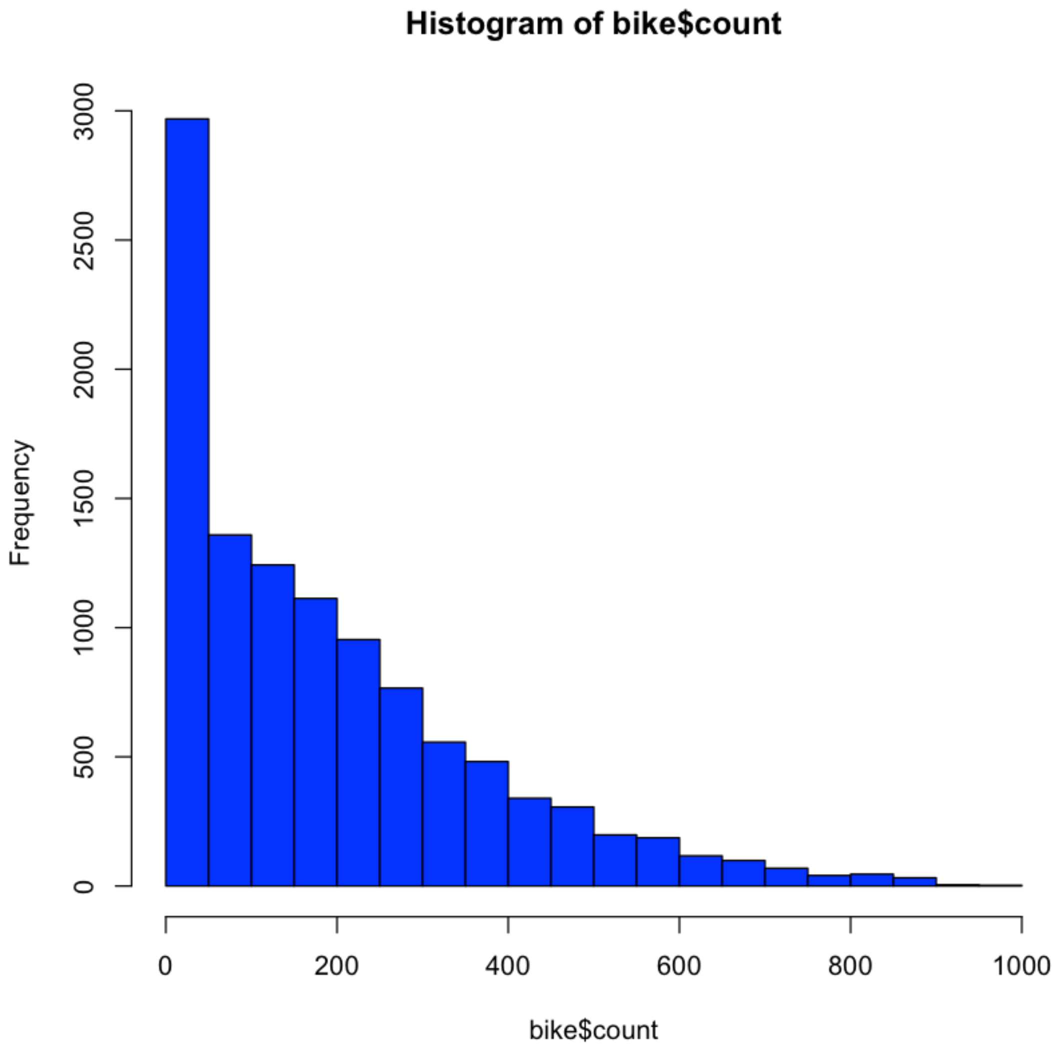

```
ERROR while rich displaying an object: Error in
as.POSIXlt.character(as.character(x), ...): character string is not in a
standard unambiguous format

Traceback:
1. FUN(X[[i]], ...)
2. tryCatch(withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
```

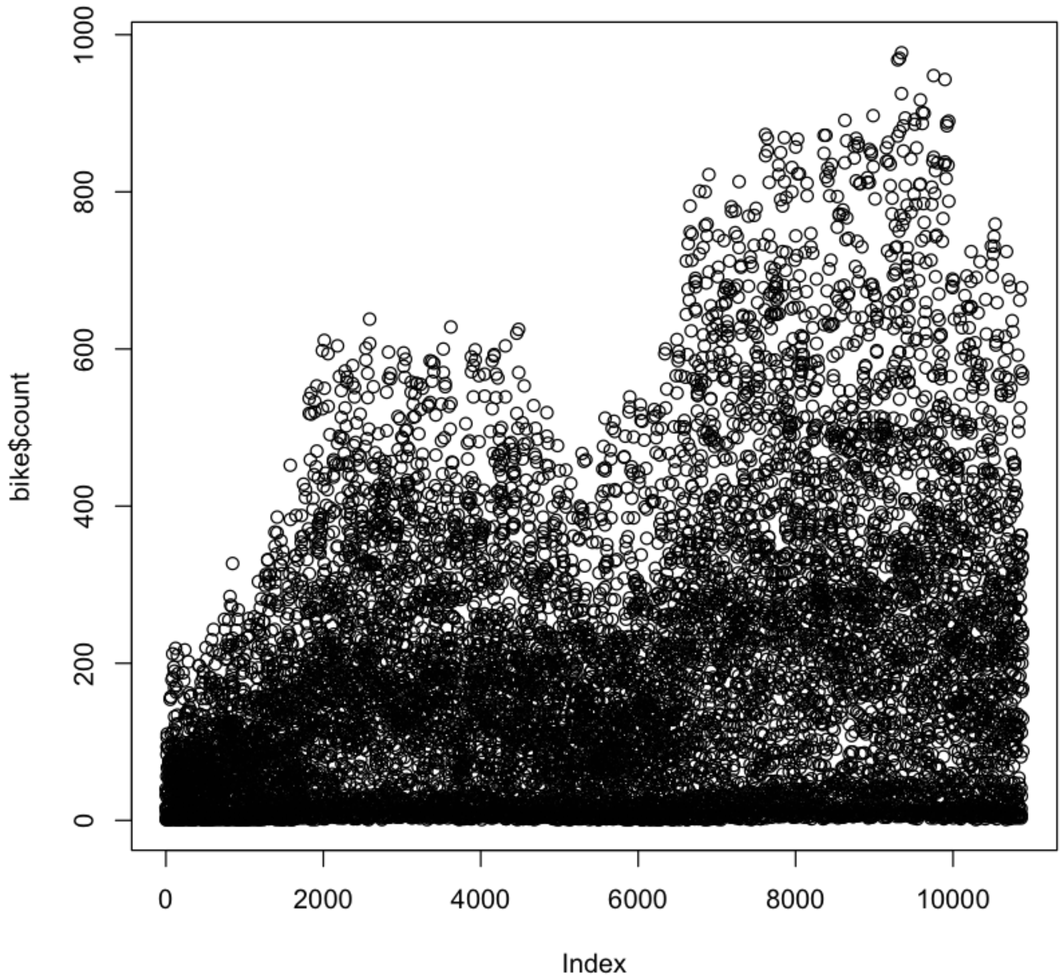
```
. }, error = error_handler), error = outer_handler)
3. tryCatchList(expr, classes, parentenv, handlers)
4. tryCatchOne(expr, names, parentenv, handlers[[1L]])
5. doTryCatch(return(expr), name, parentenv, handler)
6. withCallingHandlers({
.     if (!mime %in% names(repr::mime2repr))
.         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
.     rpr <- repr::mime2repr[[mime]](obj)
.     if (is.null(rpr))
.         return(NULL)
.     prepare_content(is.raw(rpr), rpr)
. }, error = error_handler)
7. repr::mime2repr[[mime]](obj)
8. repr_text.default(obj)
9. paste(capture.output(print(obj)), collapse = "\n")
10. capture.output(print(obj))
11. evalVis(expr)
12. withVisible(eval(expr, pf))
13. eval(expr, pf)
14. eval(expr, pf)
15. print(obj)
16. print.ggplot(obj)
17. ggplot_build(x)
18. ggplot_build.ggplot(x)
19. by_layer(function(l, d) l$compute_aesthetics(d, plot))
20. f(l = layers[[i]], d = data[[i]])
21. l$compute_aesthetics(d, plot)
22. f(..., self = self)
23. scales_add_defaults(plot$scales, data, aesthetics, plot$plot_env)
24. lapply(aesthetics[new_aesthetics], eval_tidy, data = data)
25. FUN(X[[i]], ...)
26. as.factor(day(date))
27. is.factor(x)
28. day(date)
29. mday.default(date)
30. as.POSIXlt(x, tz = tz(x))
31. as.POSIXlt.factor(x, tz = tz(x))
32. as.POSIXlt(as.character(x), ...)
33. as.POSIXlt.character(as.character(x), ...)
34. stop("character string is not in a standard unambiguous format")
```


hour vs Count
count
1000
750
500
250
0
hour
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

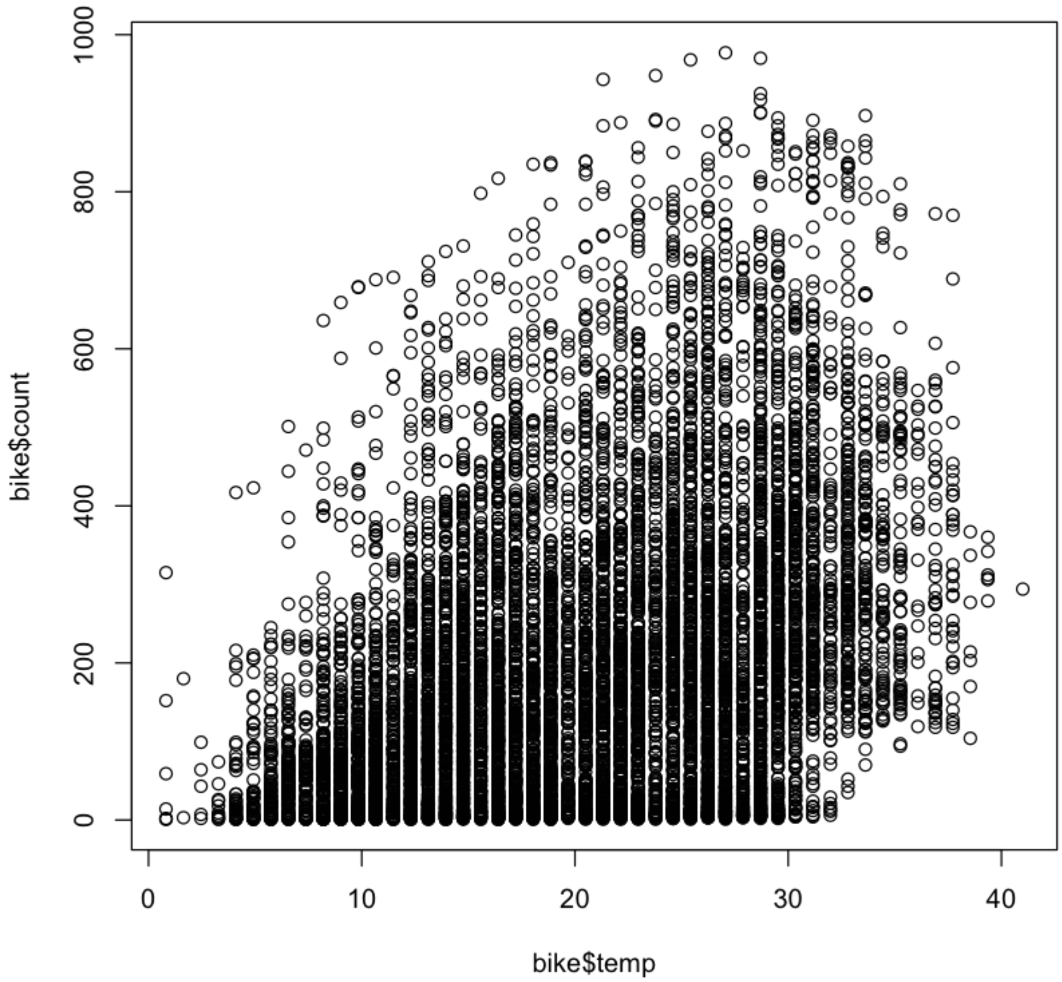
[ ]:
```
# 4b. Correlation Analysis

# --------------- Explore Continuous Variables----------------
    # 4b(1). Explore continous features
        # i. Check distribution of target variable
        # ii. Explore correlation between independent continuous variables with␣
 ↪target variable
        # iii. Plot heatmap for correlation matrix (to check for␣
 ↪multicolinearity)
        # iv. Visualize the relationship among all continuous variables using␣
 ↪pairplots
        # v. Explore relationship between independent continuous variables and␣
 ↪dependent variables using Joint Plot
```

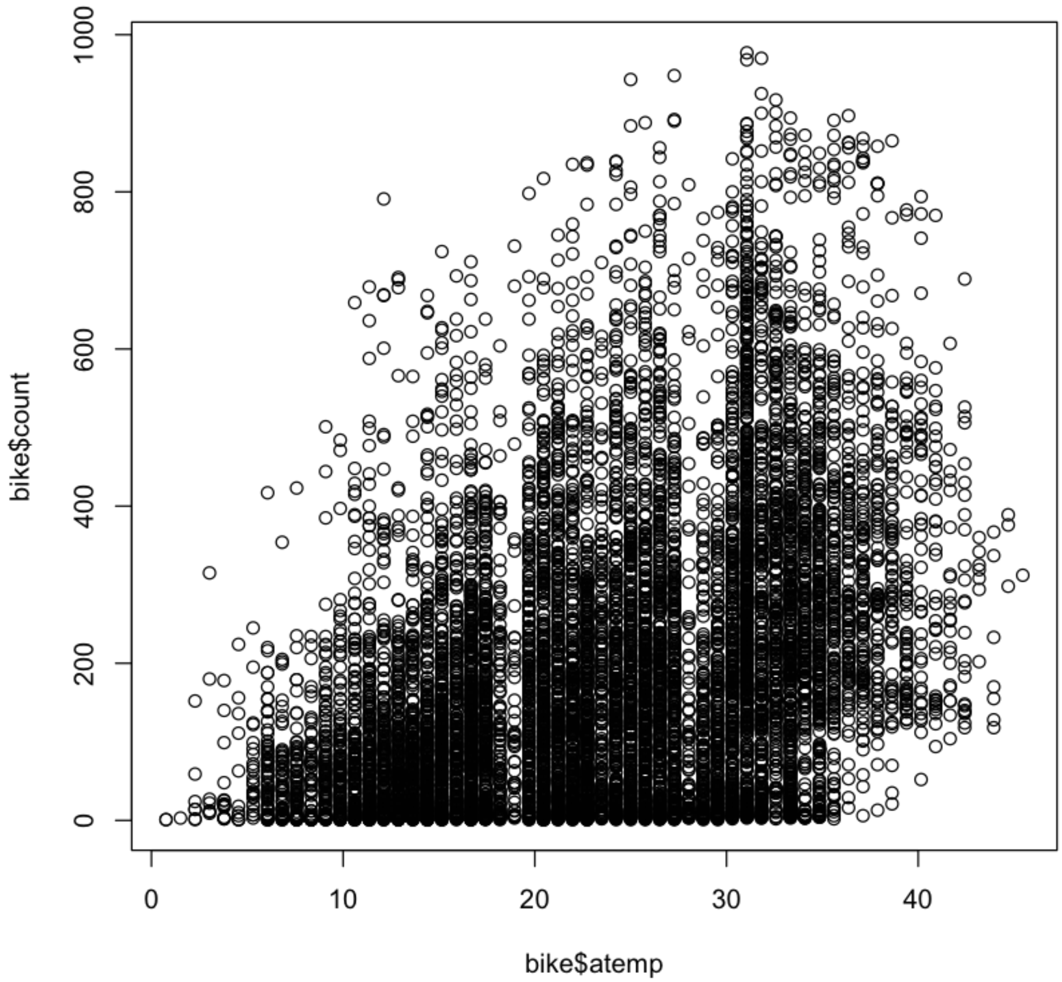
[10]:
```
# 4b(1) i. Check distribution of target variable
          hist(bike$count, col="blue")
          plot(bike$count)
   # Inference: Target variable "count" is almost normally distributed.
```

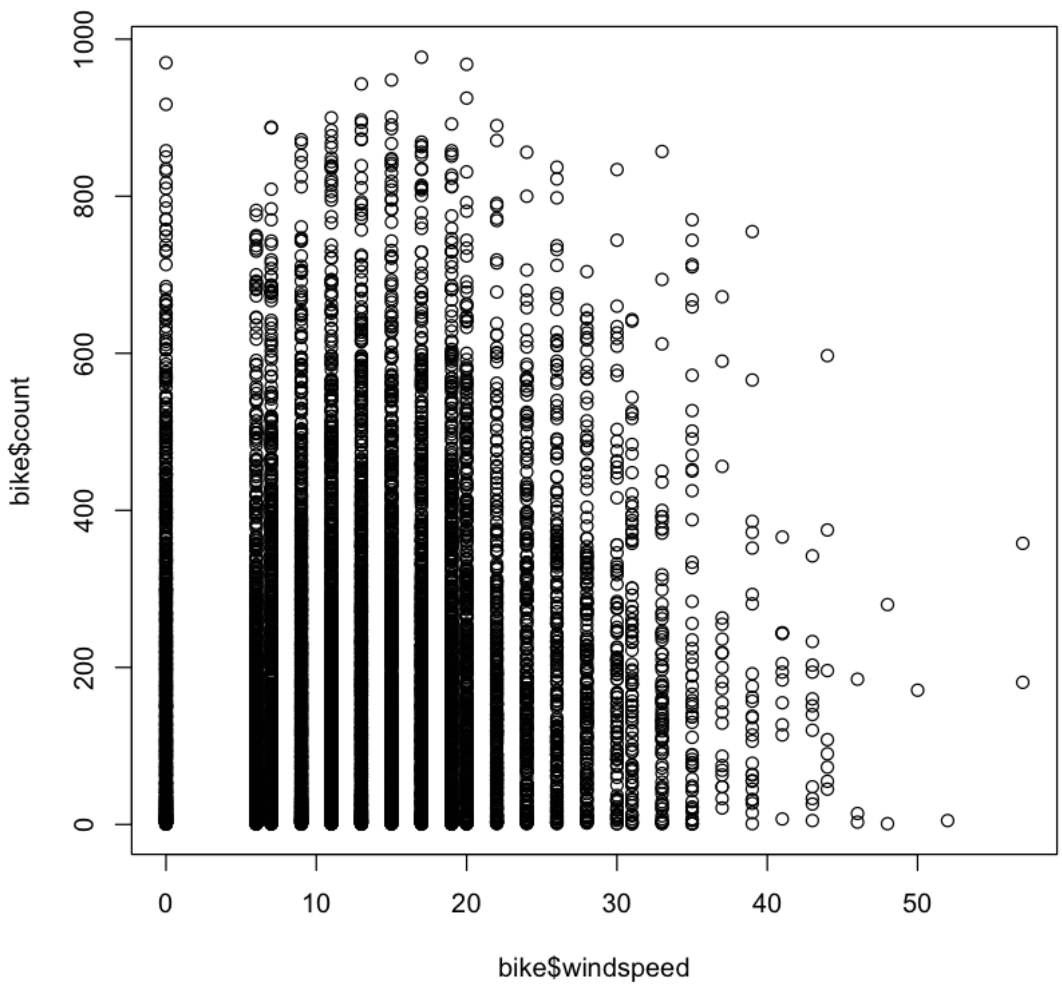


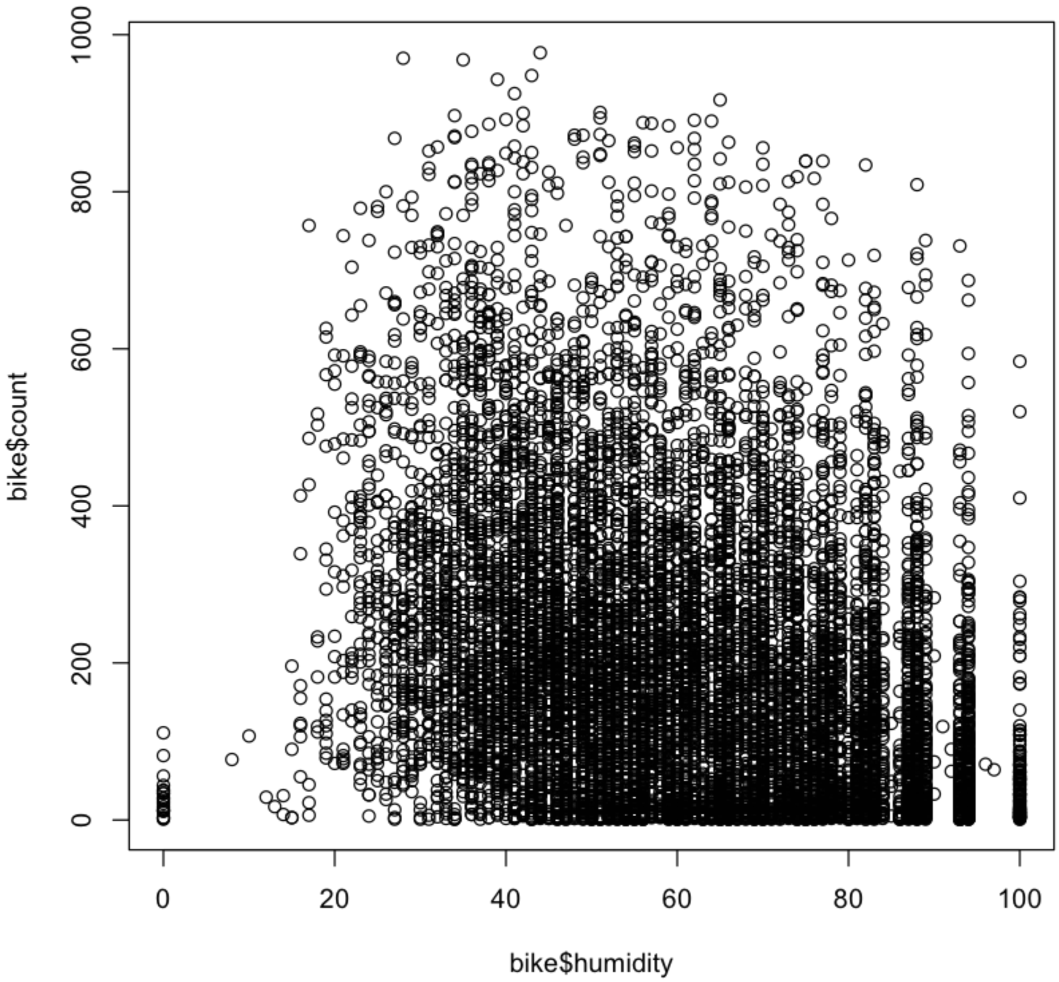


[11]:
```
# 4b(1) ii. Explore correlation between independent continuous variables with␣
 ↪target variable
        plot(bike$temp,bike$count)
        plot(bike$atemp,bike$count)
        plot(bike$windspeed,bike$count)
        plot(bike$humidity,bike$count)
```

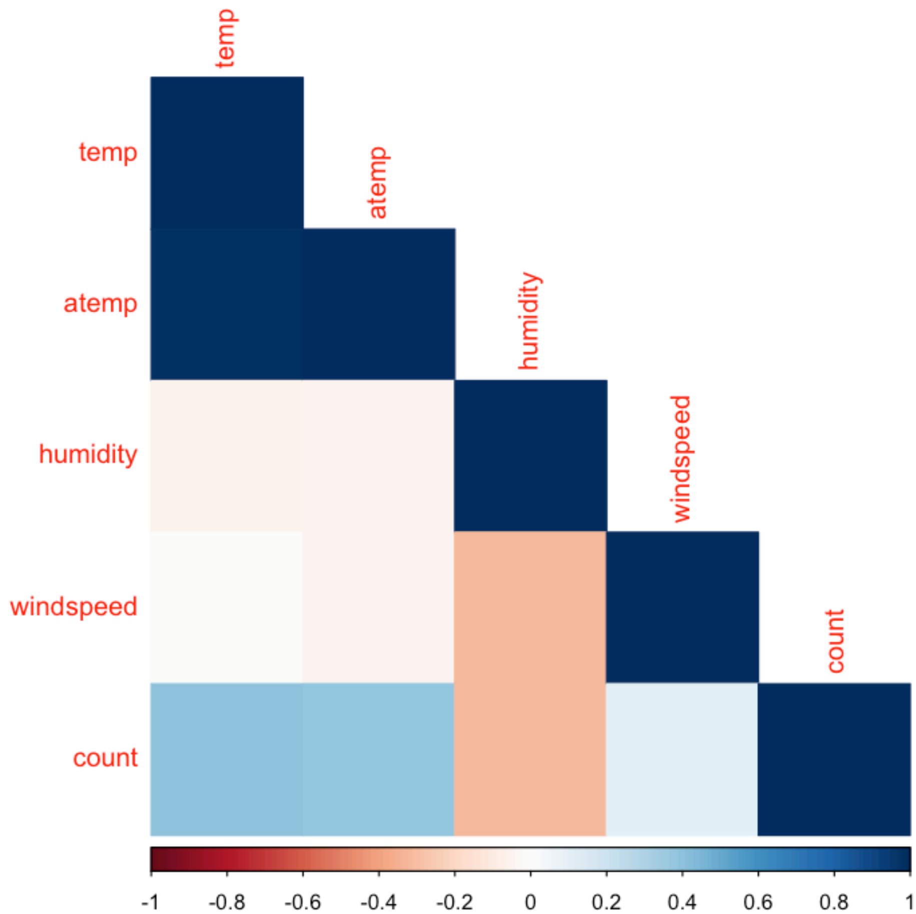

1000
800
600
400
200
0
bike$count
0
10
20
30
40
bike$temp


bike$count
bike$atemp
0
200
400
600
800
1000
0
10
20
30
40

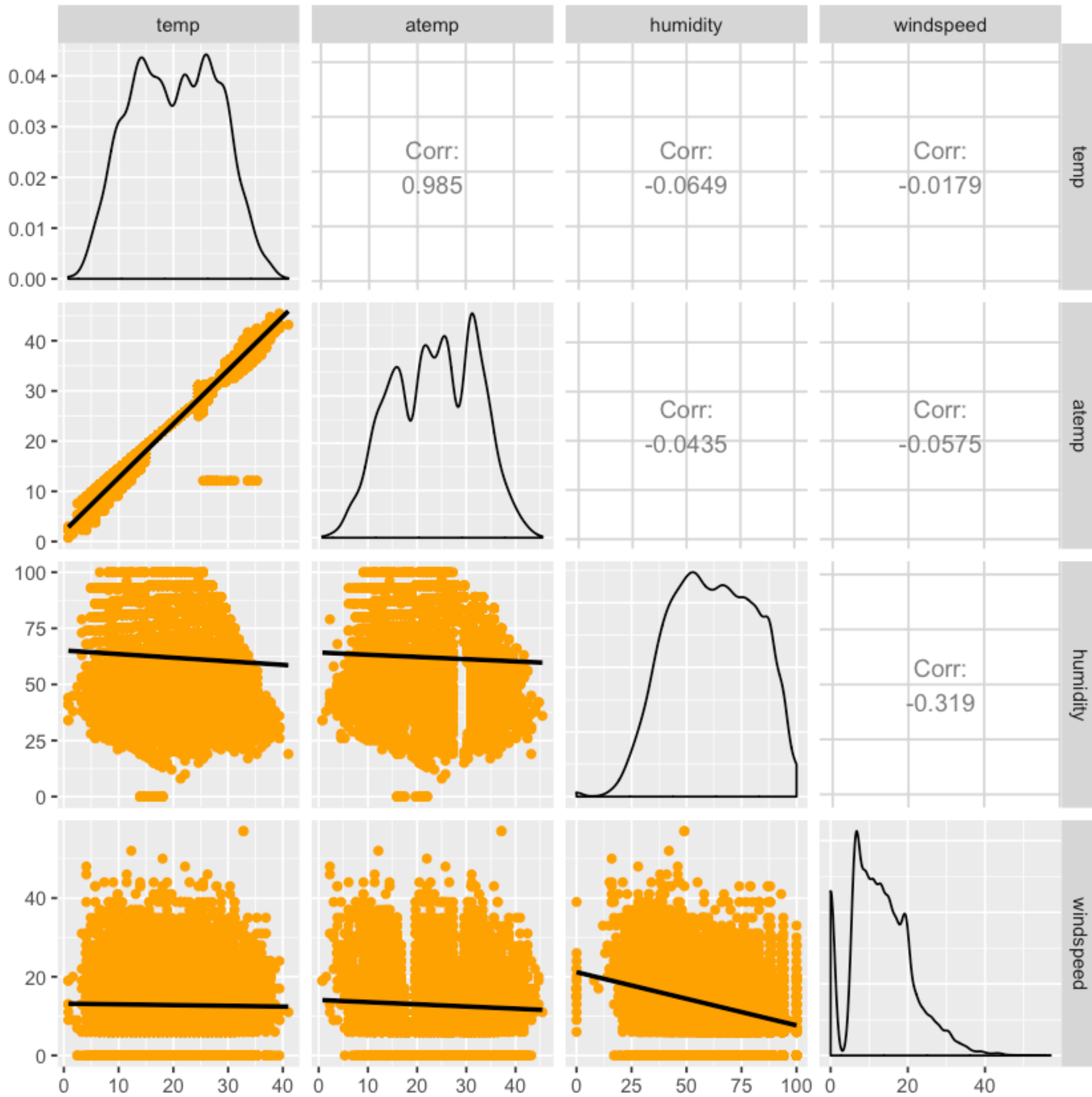

bike$count
0
200
400
600
800
1000
0
10
20
30
40
50
bike$windspeed



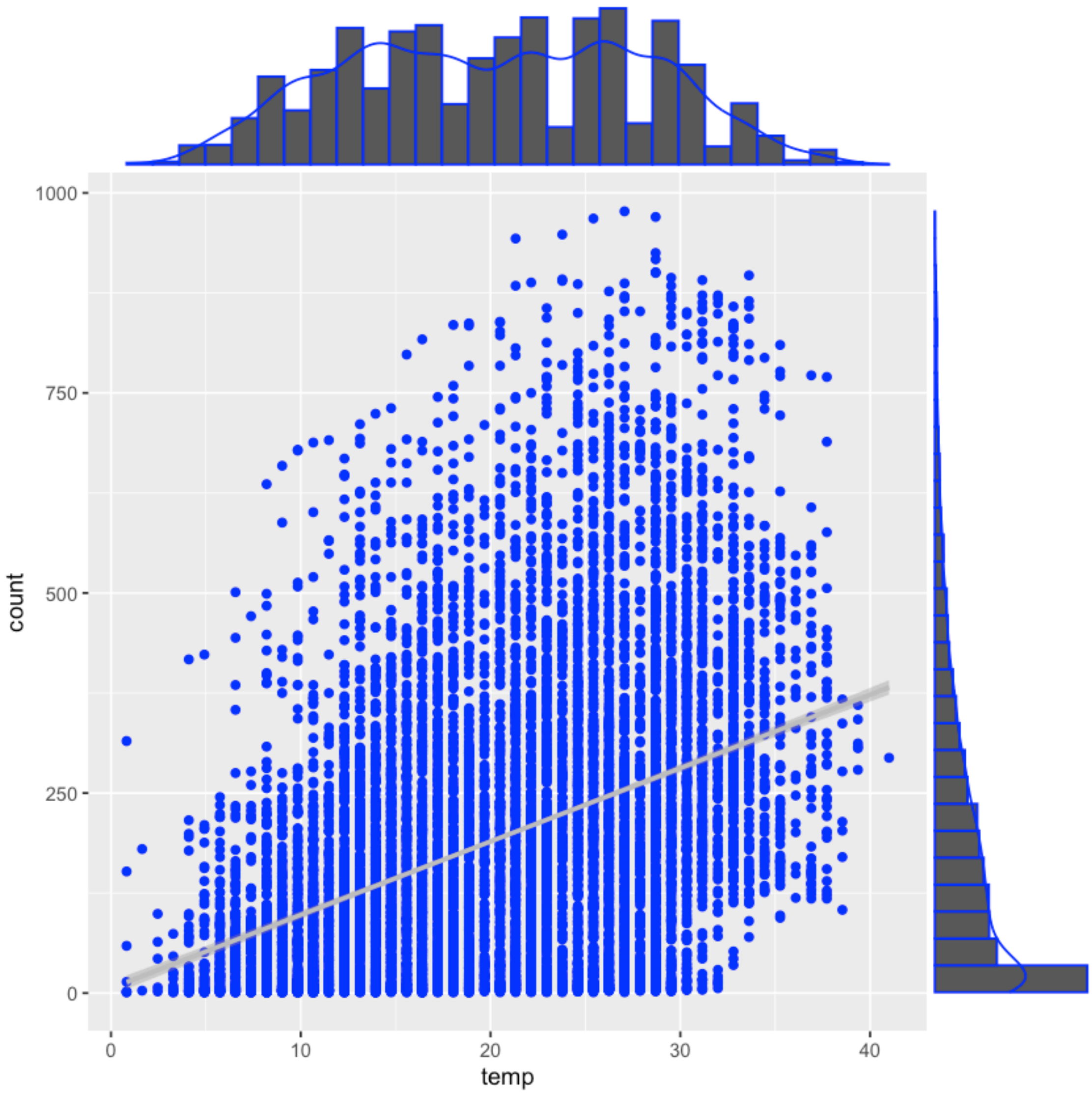
[12]:
```
# 4b(1) iii. Plot heatmap for correlation matrix (to check for␣
↪multicolinearity)
        corr <- as.data.frame(lapply(bike[c(5:8, 11)], as.numeric))
        corrplot(cor(corr), method = "color", type='lower')
    # Inference:
        # i. temp and atemp are highly correlated, we would need to drop one of␣
↪them to remove multicolinearity.
        # ii. We can also drop Registered and Casual from our analysis as␣
↪Counts are categorized as Registered and Casual
             # and we will be predicting "Count" variable only.
```



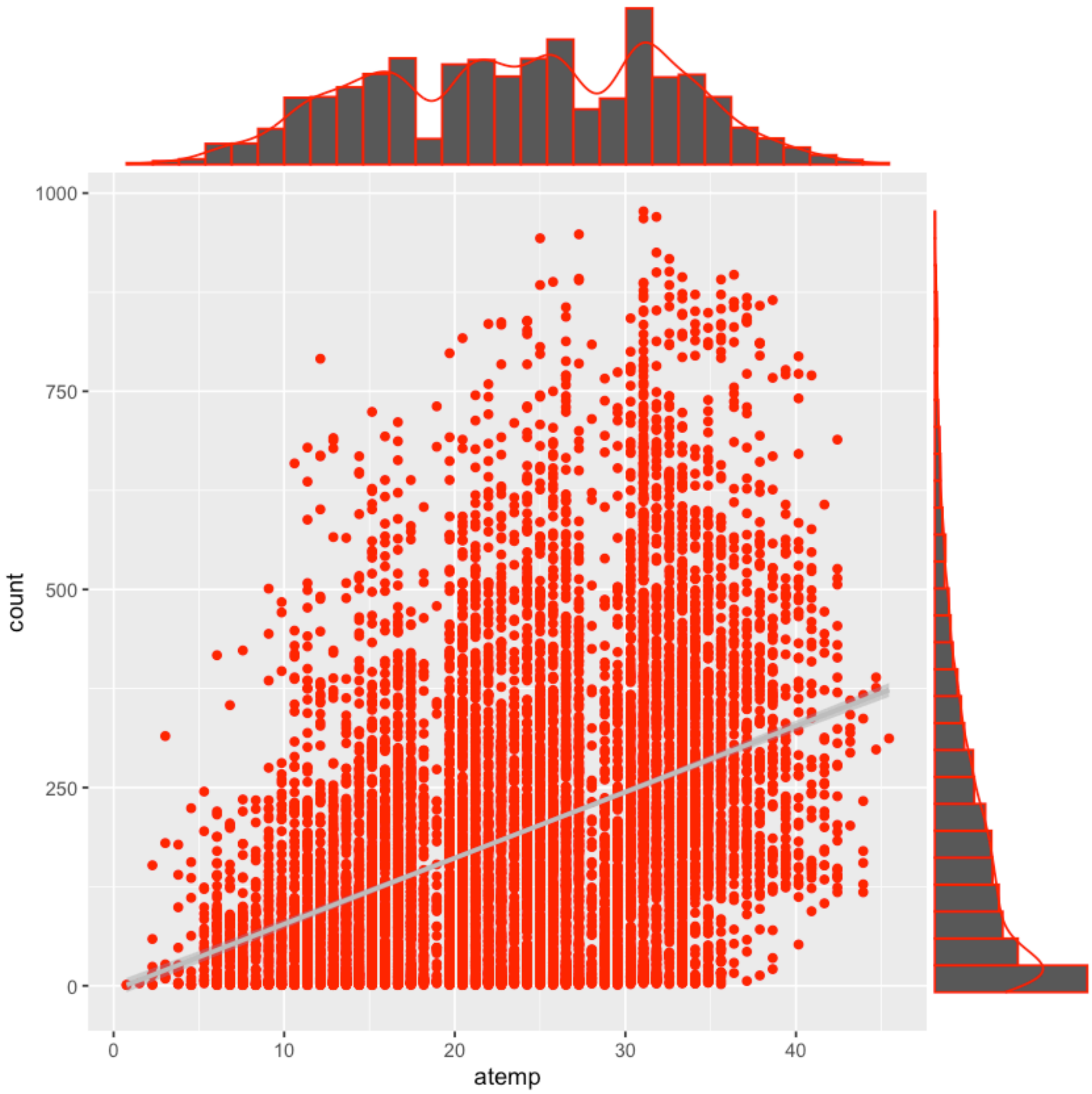
[13]:
```
# 4b(1) iv. Visualize the relationship among all continuous variables using␣
↪pairplots
        ggpairs(bike[c(5:8)], lower=list(continuous=wrap("smooth",␣
↪colour="orange")) )
```



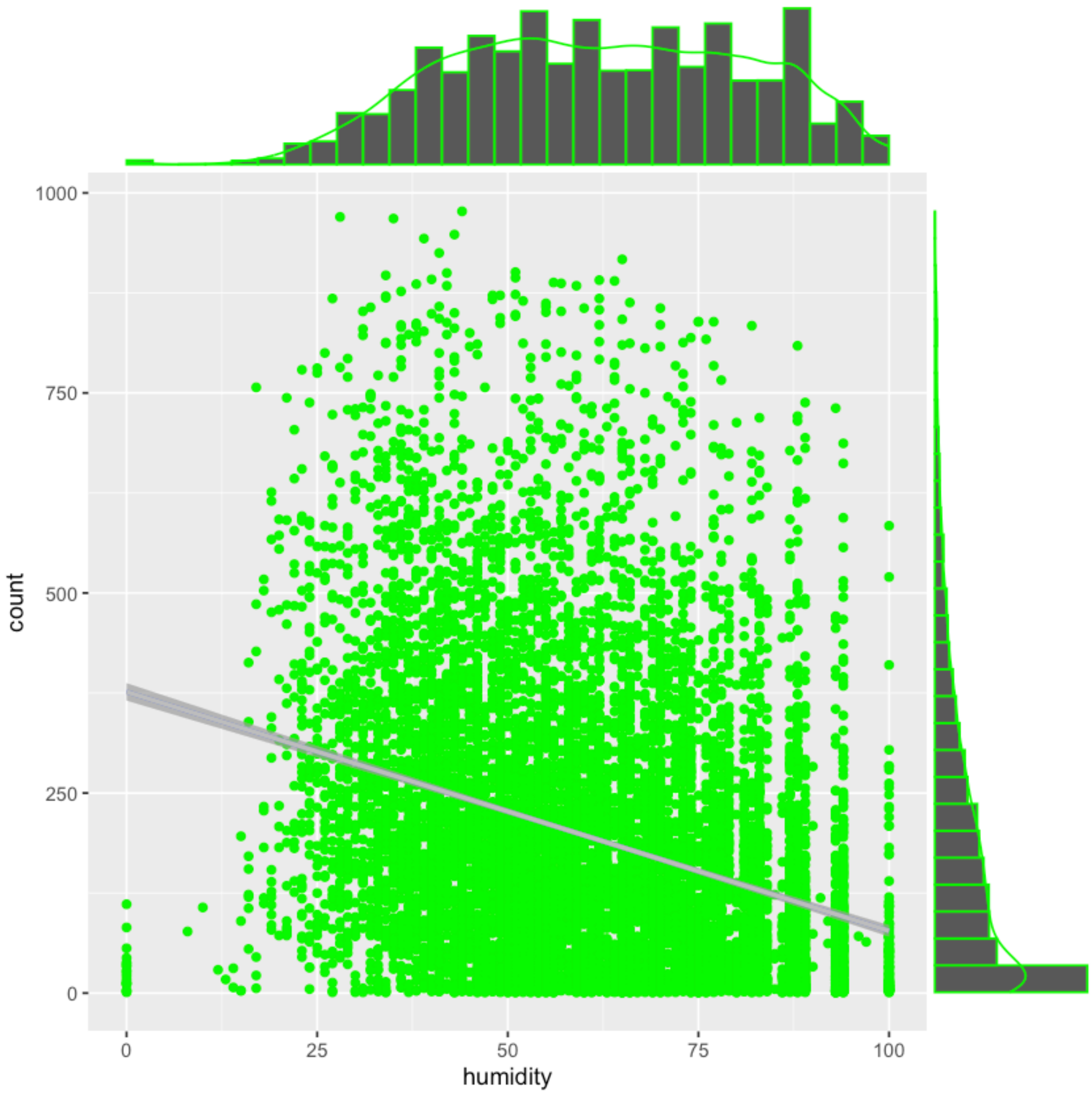
[14]:
```
# 4b(1) v. Explore relationship between independent continuous variables and␣
↪dependent variables using Joint Plot

        # 1. temp vs Count
        plot_center = ggplot(bike, aes(x=temp,y=count)) +␣
↪geom_point(colour="blue") + geom_smooth(method="lm", colour="grey")
        ggMarginal(plot_center, type="densigram", colour="blue")
        # Inference: temp has good correlation with count.
```



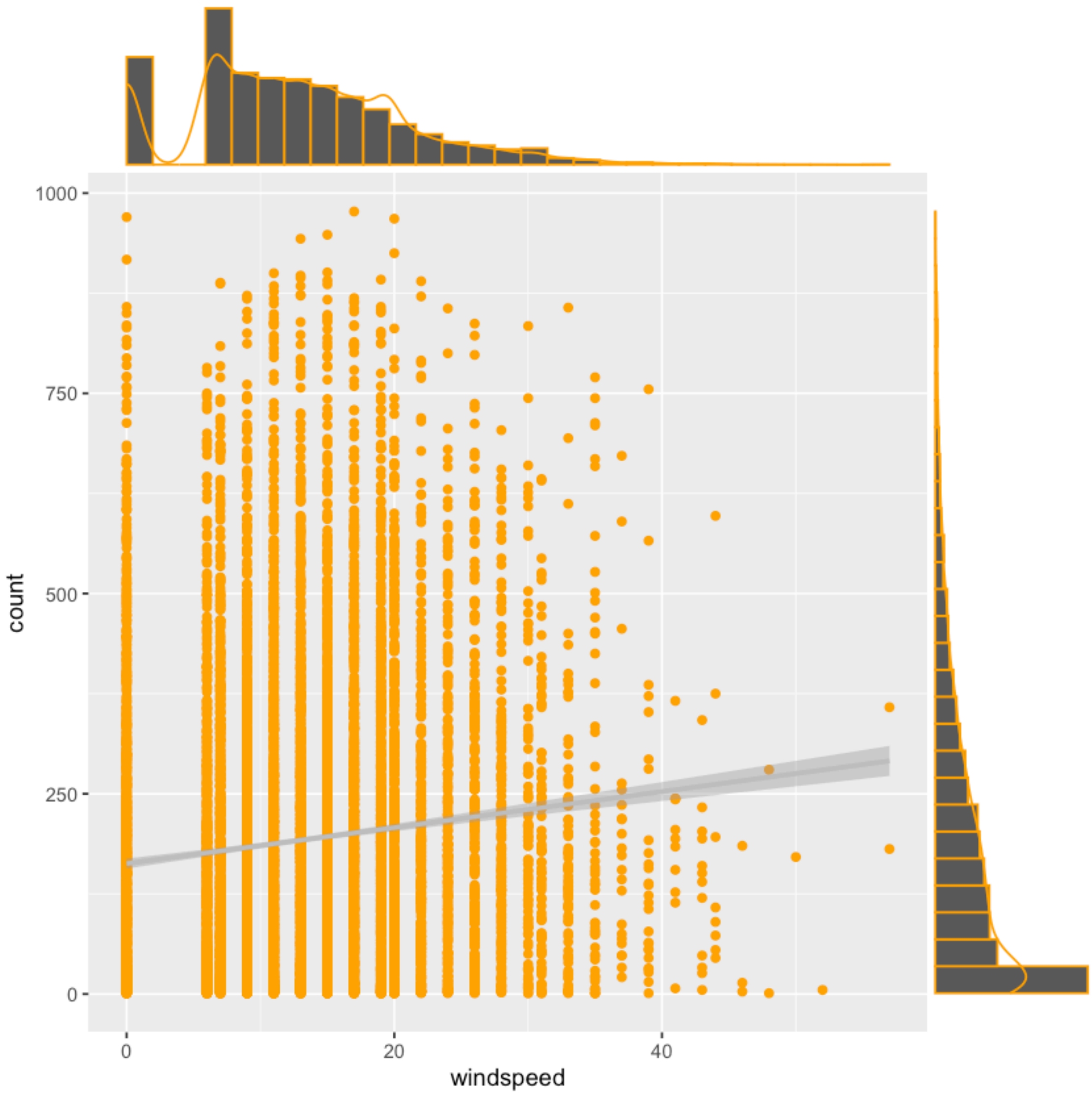
[15]:
```
# 4b(1).v.2. atemp vs Count
        plot_center = ggplot(bike, aes(x=atemp,y=count)) +␣
 ↪geom_point(colour="red") + geom_smooth(method="lm", colour="grey")
        ggMarginal(plot_center, type="densigram", colour="red")
        # Inference: atemp has good correlation with count.
```



[16]:
```
# 4b(1).v.3. humidity vs Count
        plot_center = ggplot(bike, aes(x=humidity,y=count)) +␣
 ↪geom_point(colour="green") + geom_smooth(method="lm") +␣
 ↪geom_smooth(method="lm", colour="grey")
        ggMarginal(plot_center, type="densigram", colour="green")
        # Inference: Humidity has low correlation with count.
```

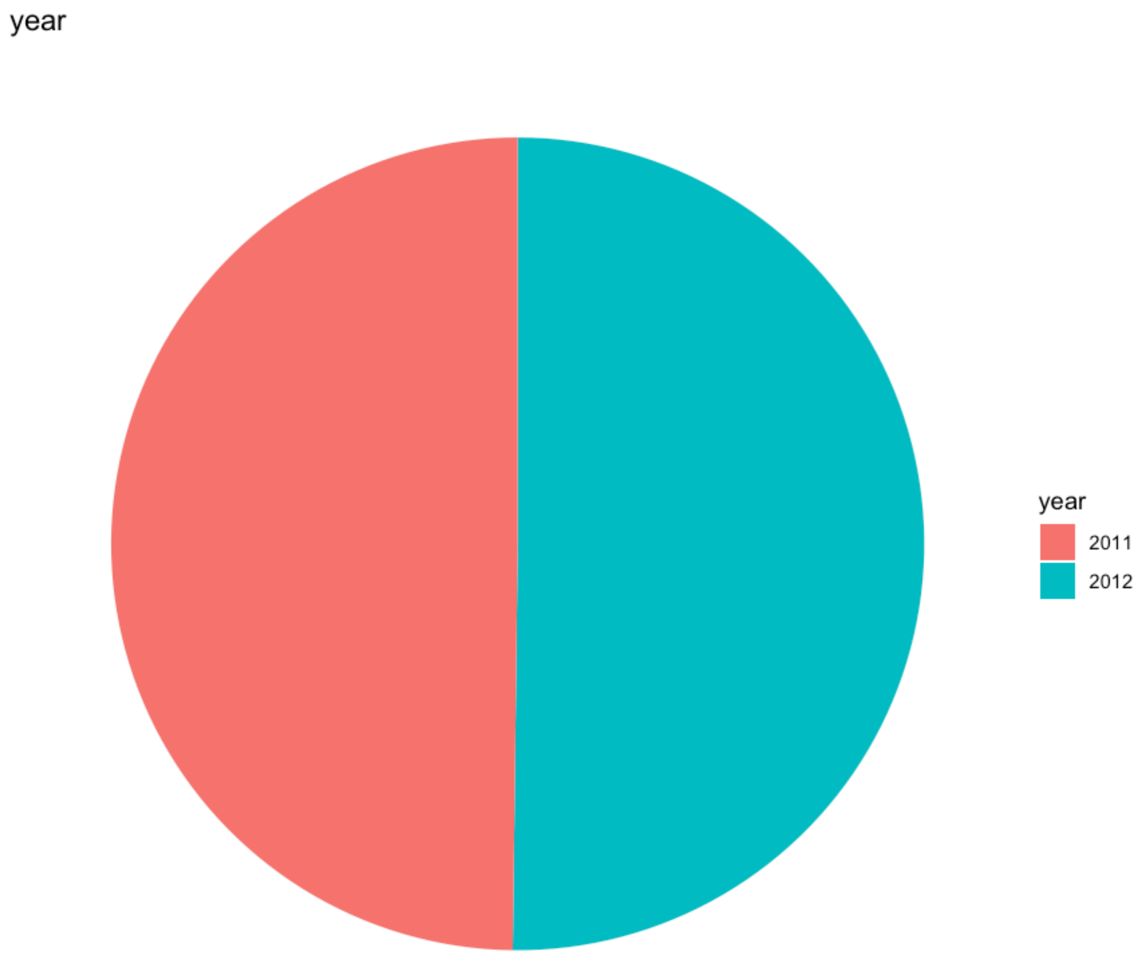


[17]:
```
# 4b(1).v.4. windspeed vs Count
        plot_center = ggplot(bike, aes(x=windspeed,y=count)) +␣
 ↪geom_point(colour="orange") + geom_smooth(method="lm", colour="grey")
        ggMarginal(plot_center, type="densigram", colour="orange")
```



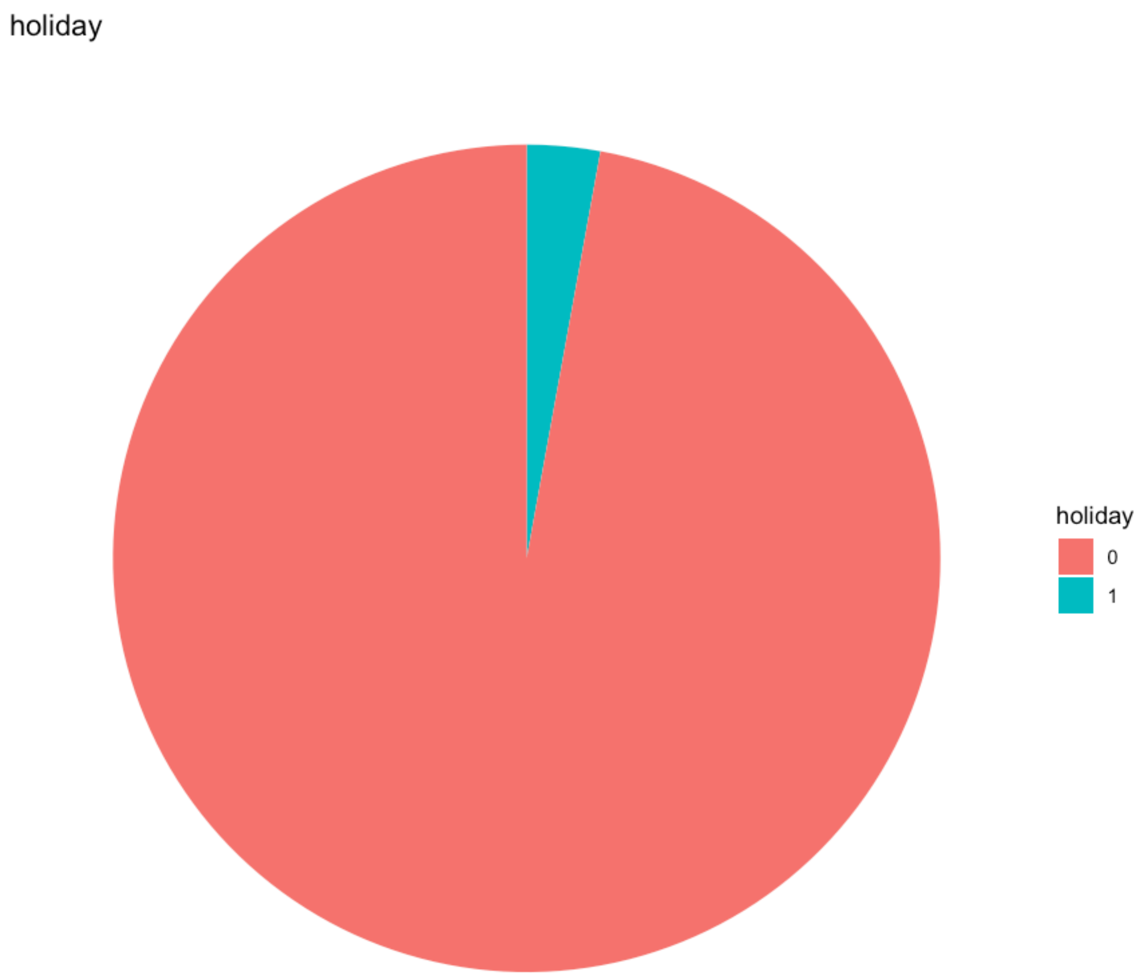
[ ]:
```
# 4b(1) Inferences Summary - Analysis of continous variables
        # 1. Target variable 'count' is almost normally distributed.
        # 2. From correlation with dependent variable "count", we can see that␣
↪'casual','registered' are very
             # highly correlated to cnt. Needs to be dropped from the dataset.
        # 3. 'humidity' has low correlation with 'count'. For now, lets keep it.
        # 4. atemp and temp has good correlation with 'count'
        # 5. From heatmap, we can see that atemp and temp are highly correlated.
↪ So we need to drop 1 to remove multicollinearity.
        # 6. Since, as seen from jointplot, p(atemp) < p(temp), we can drop␣
↪'temp' and retain 'atemp' in the dataset.
```

[18]:
```
# --------------- Explore Catogorical Variables----------------
    # 4b(2) Explore categorical features
          # i. Check distribution of categorical variables
            ggplot(bike, aes(x=" ",fill=year))+ geom_bar(width = 1)+␣
↪coord_polar("y")+labs(title = "year")+theme_void()

            ggplot(bike, aes(x=" ",fill=month))+ geom_bar(width = 1)+␣
↪coord_polar("y")+labs(title = "month")+theme_void()
            bike$season = factor(bike$season)

            ggplot(bike, aes(x=" ",fill=season))+ geom_bar(width = 1)+␣
↪coord_polar("y")+labs(title = "Season")+theme_void()
            bike$holiday = factor(bike$holiday)

            ggplot(bike, aes(x=" ",fill=holiday))+ geom_bar(width = 1)+␣
↪coord_polar("y")+labs(title = "holiday")+theme_void()

            ggplot(bike, aes(x=" ",fill=wkday))+ geom_bar(width = 1)+␣
↪coord_polar("y")+labs(title = "weekday")+theme_void()
            bike$workingday = factor(bike$workingday)

            ggplot(bike, aes(x=" ",fill=workingday))+ geom_bar(width = 1)+␣
↪coord_polar("y")+labs(title = "workingday")+theme_void()
            bike$weather = factor(bike$weather)

            ggplot(bike, aes(x=" ",fill=weather))+ geom_bar(width = 1)+␣
↪coord_polar("y")+labs(title = "weather")+theme_void()
```


year
year
2011
2012

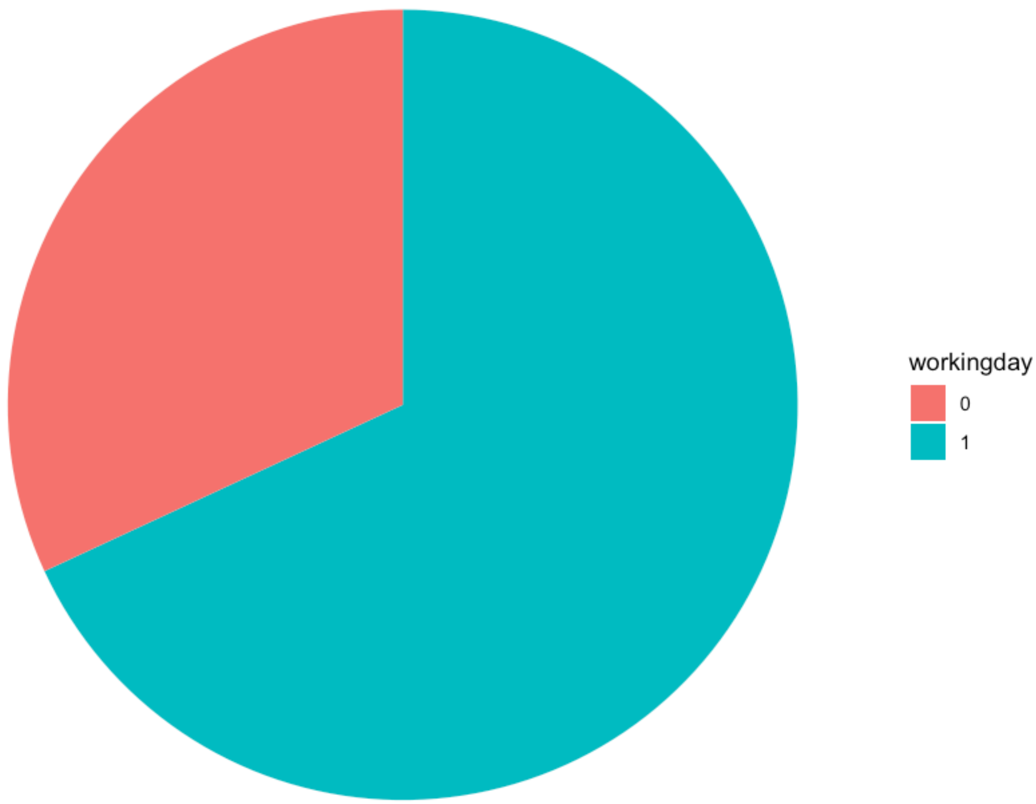

month
month
1
2
3
4
5
6
7
8
9
10
11
12

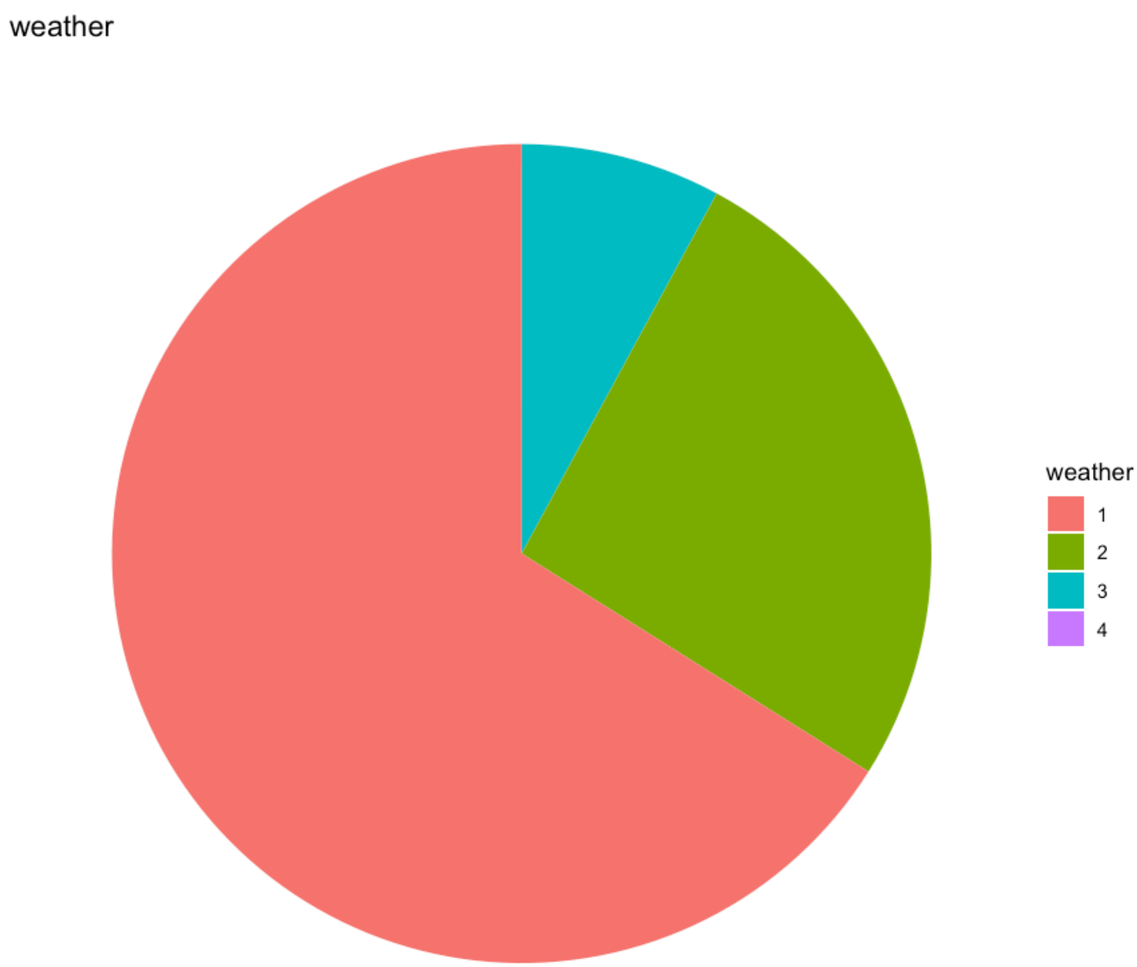

Season
season
1
2
3
4


holiday
holiday
0
1


weekday
wkday
1
2
3
4
5
6
7

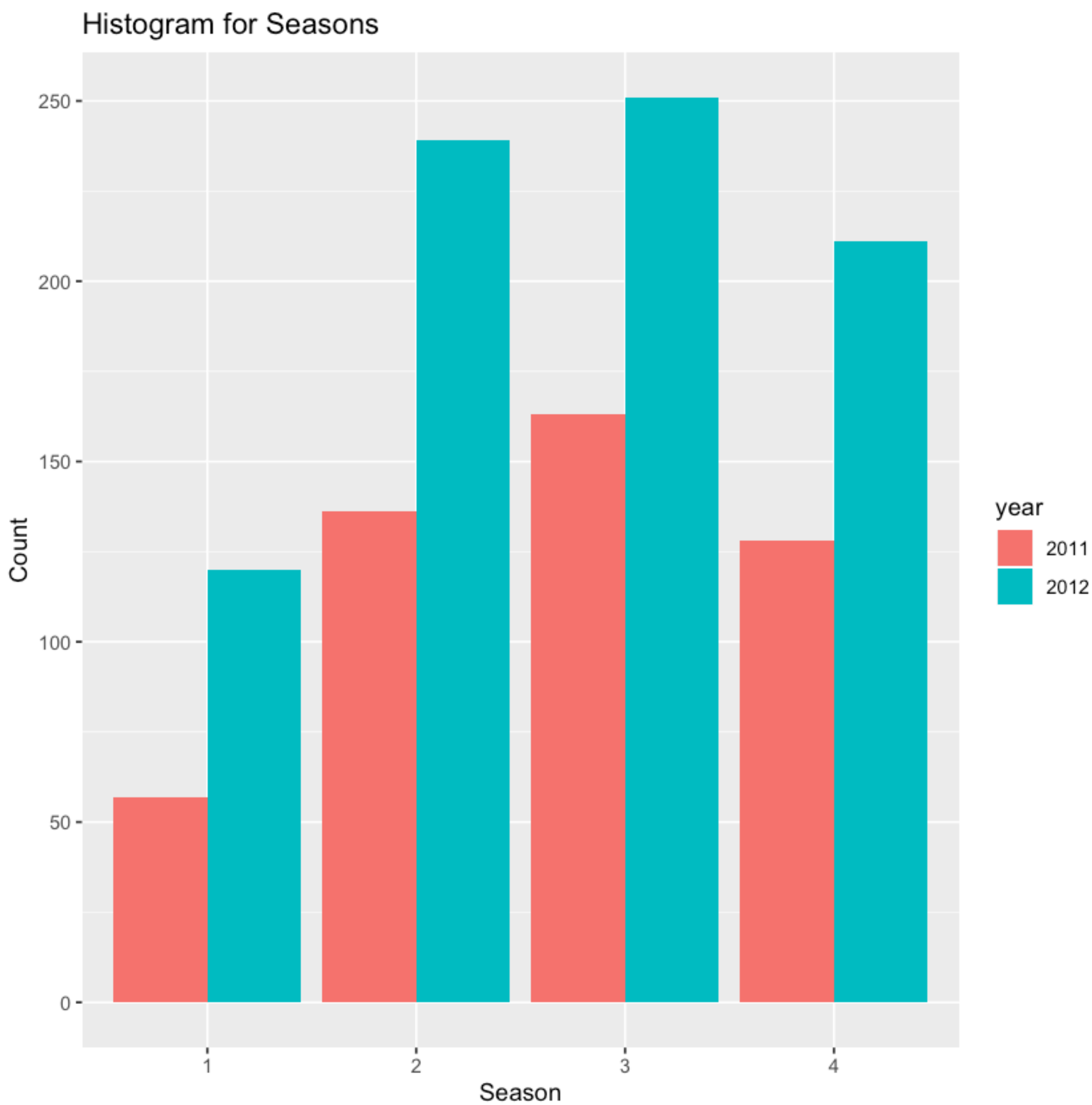
workingday

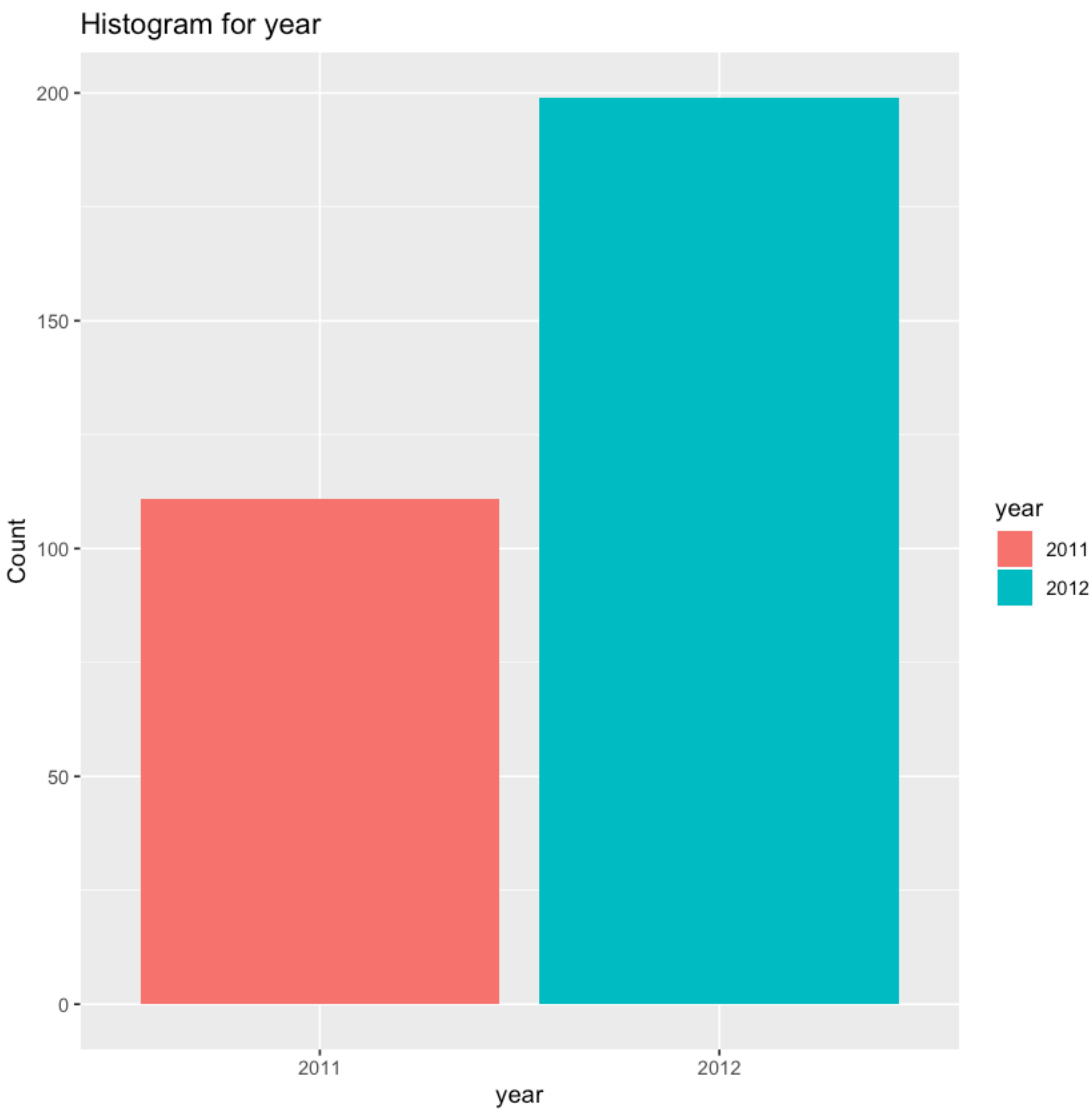




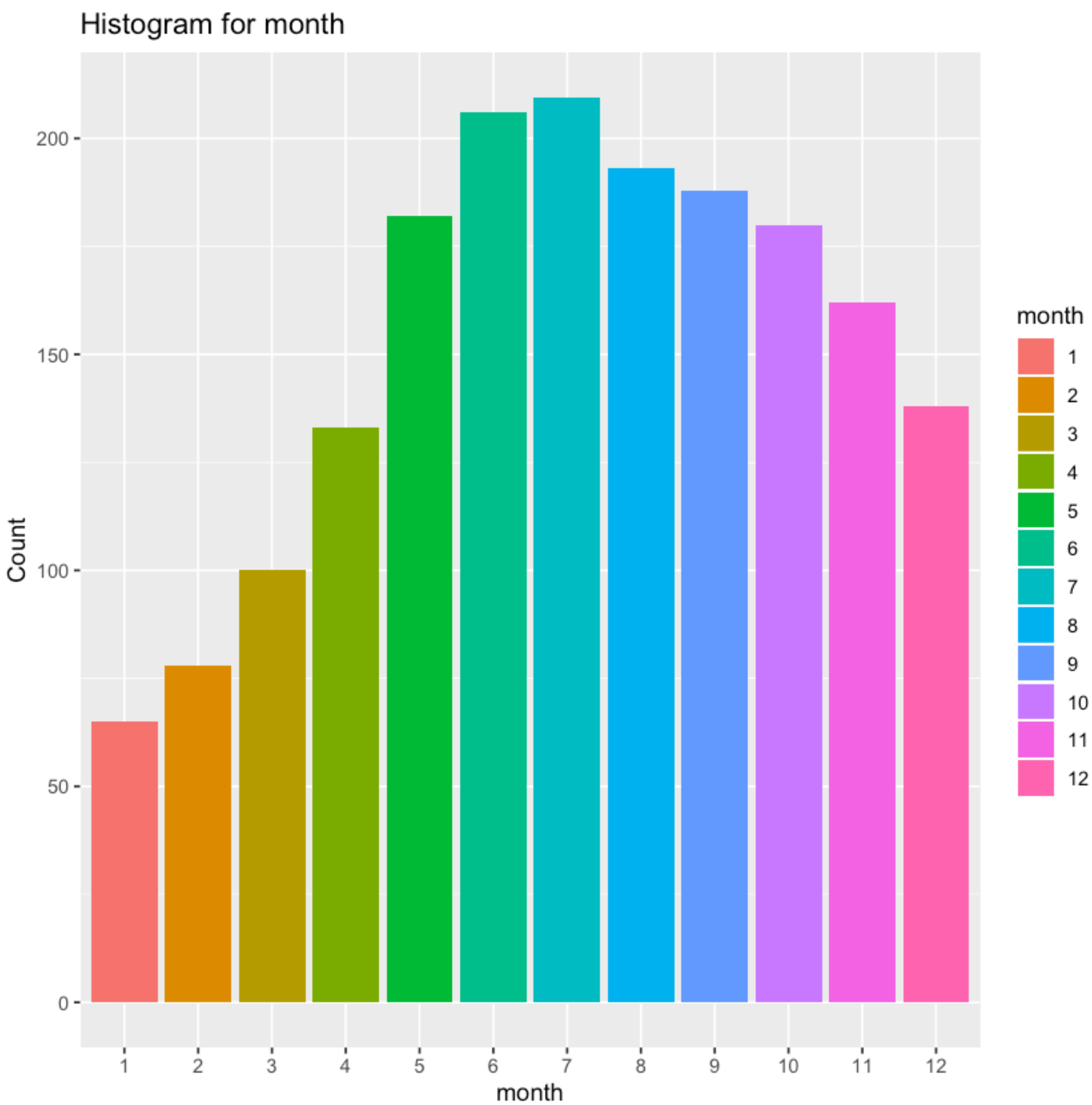
[19]:
```
# ii. Check how individual categorical features affects the target variable
        ggplot(bike, aes(x=season, y=count, fill=year)) +
          stat_summary(
            fun.y=median,
            geom='bar',
            position=position_dodge(),
            ) + labs(title="Histogram for Seasons") +  labs(x="Season",␣
 ↪y="Count")

        ggplot(bike, aes(x=year, y=count, fill=year)) +
          stat_summary(
            fun.y=median,
            geom='bar',
```

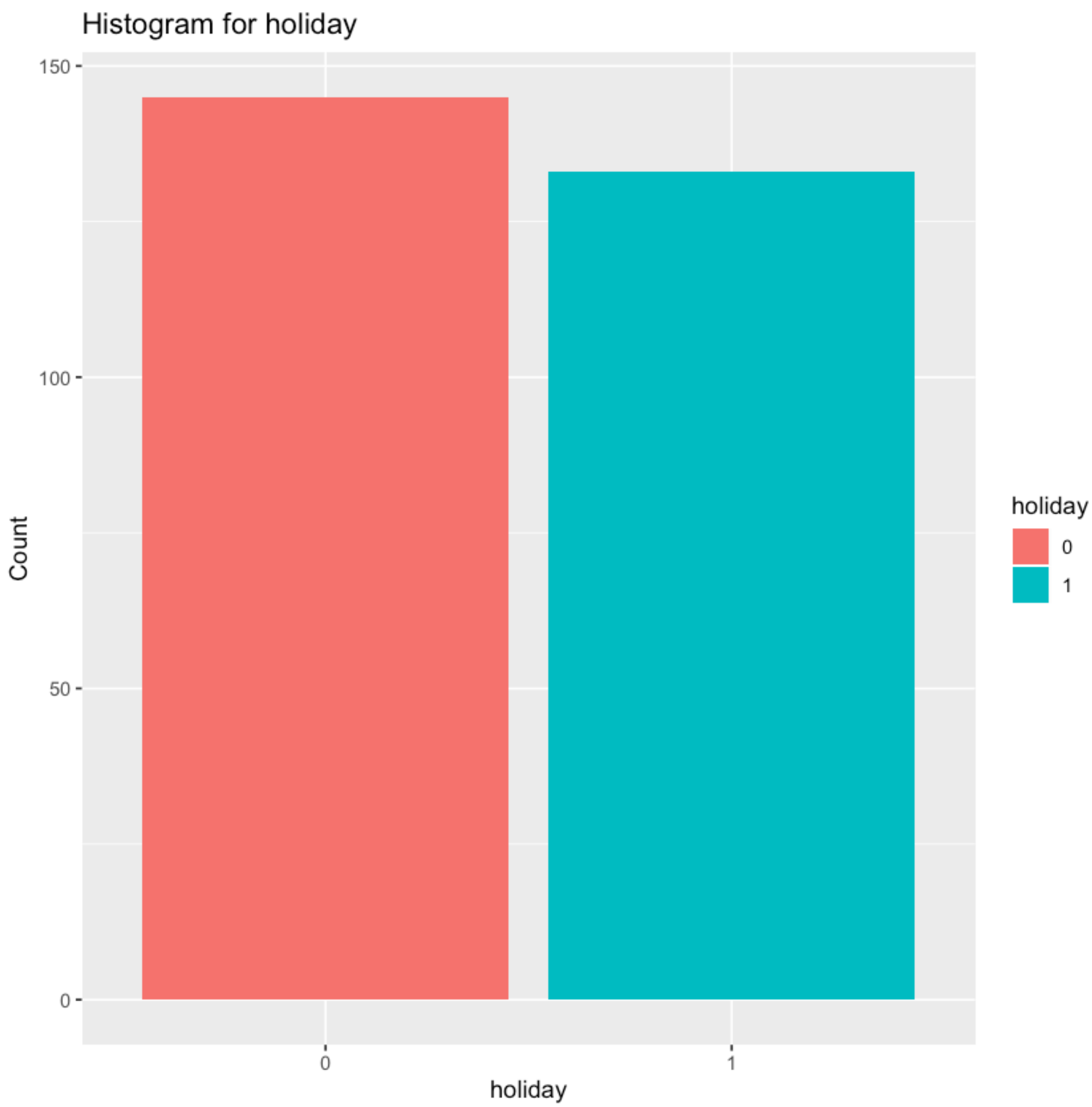
```
      position=position_dodge(),
    ) +
    labs(title="Histogram for year") +  labs(x="year", y="Count")

  ggplot(bike, aes(x=month, y=count, fill=month)) +
    stat_summary(
      fun.y=median,
      geom='bar',
      position=position_dodge(),
    ) +
    labs(title="Histogram for month") +  labs(x="month", y="Count")

  ggplot(bike, aes(x=holiday, y=count, fill=holiday)) +
    stat_summary(
      fun.y=median,
      geom='bar',
      position=position_dodge(),
    ) +   labs(title="Histogram for holiday") +labs(x="holiday",␣
↪y="Count")

  ggplot(bike, aes(x=wkday, y=count, fill=wkday)) +
    stat_summary(
      fun.y=median,
      geom='bar',
      position=position_dodge(),
    ) +    labs(title="Histogram for weekday") +labs(x="weekday",␣
↪y="Count")

  ggplot(bike, aes(x=workingday, y=count, fill=workingday)) +
    stat_summary(
      fun.y=median,
      geom='bar',
      position=position_dodge(),
    ) +    labs(title="Histogram for working day") +labs(x="working day",␣
↪y="Count")

  ggplot(bike, aes(x=weather, y=count, fill=weather)) +
    stat_summary(
      fun.y=median,
      geom='bar',
      position=position_dodge(),
    ) +  labs(title="Histogram for weather") +labs(x="weather", y="Count")
```

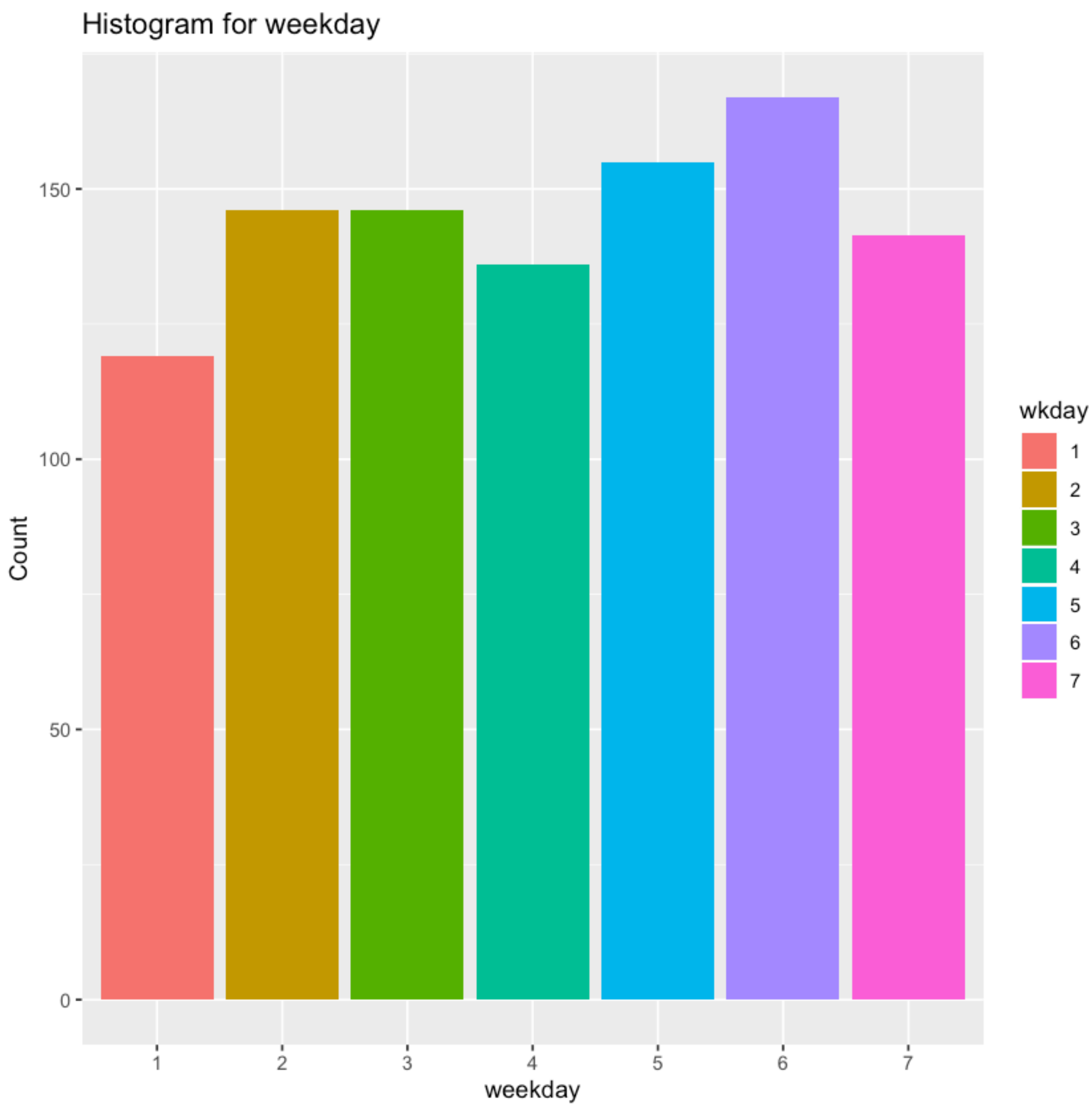

Histogram for Seasons
250
200
150
100
50
0
Count
1
2
3
4
Season
year
2011
2012



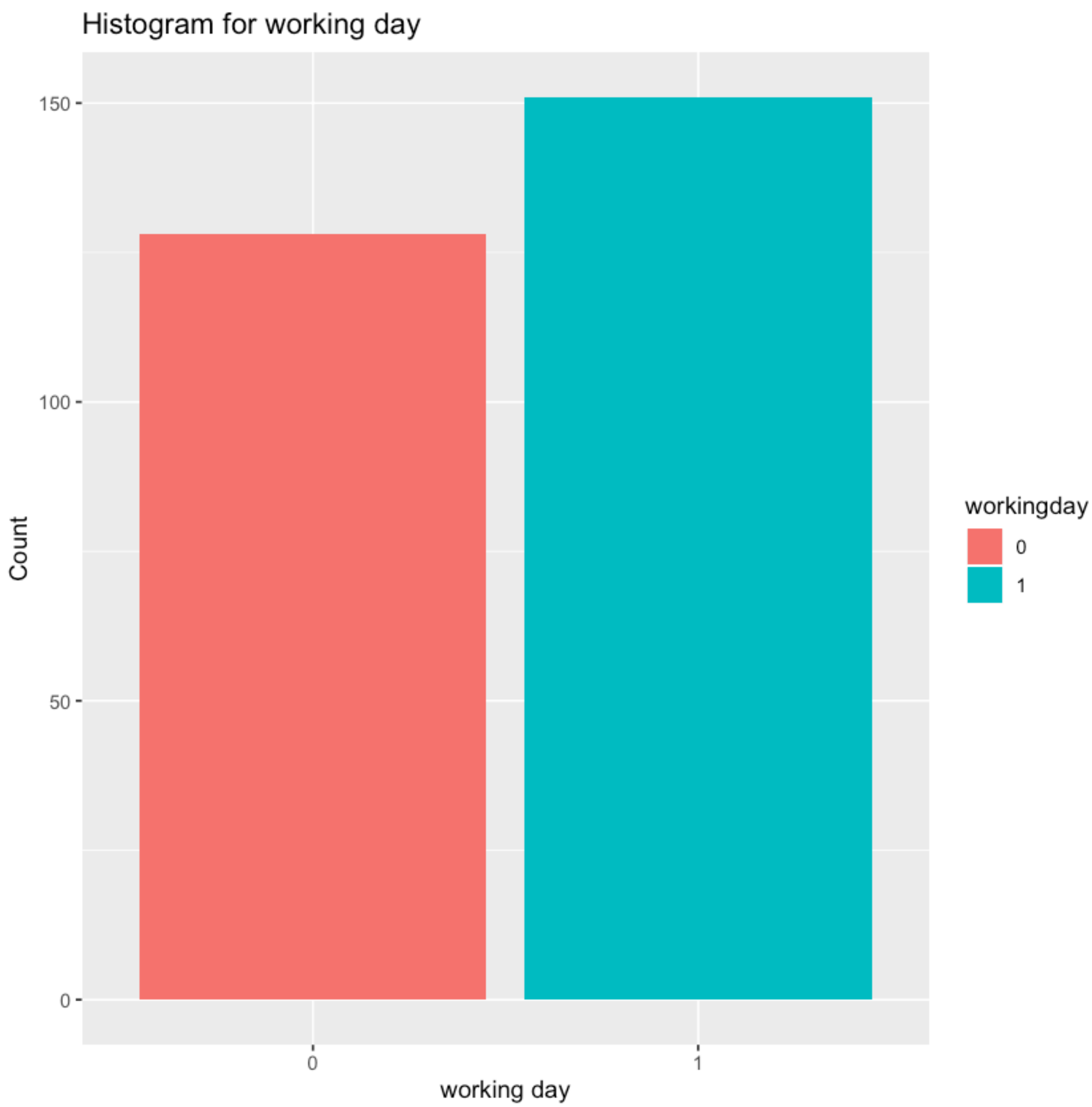
Histogram for year
200
150
100
50
0
Count
2011
2012
year
year
2011
2012


Histogram for month
Count
200
150
100
50
0
1
2
3
4
5
6
7
8
9
10
11
12
month
month
1
2
3
4
5
6
7
8
9
10
11
12

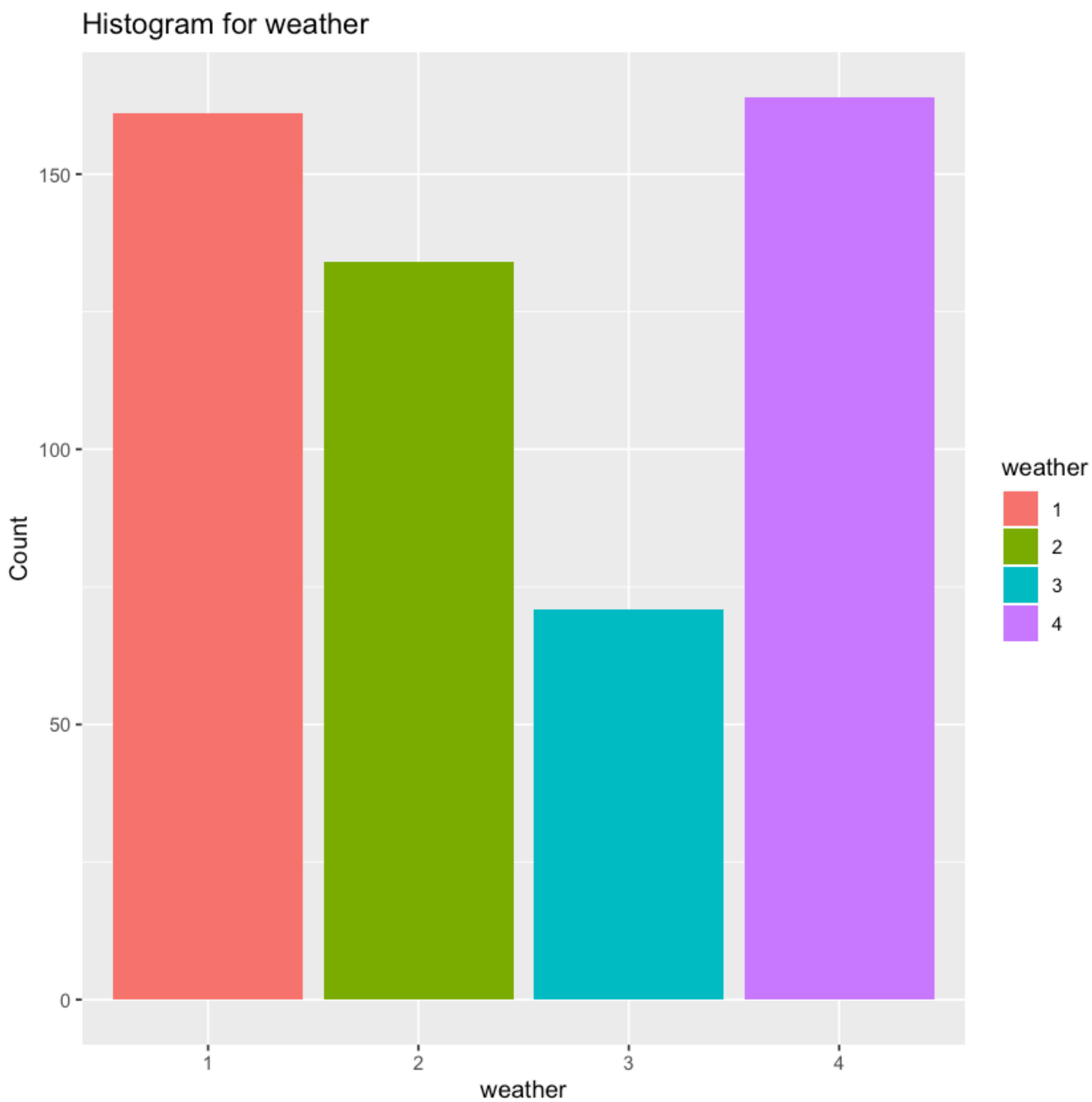

Histogram for holiday
150
100
50
0
Count
0
1
holiday
holiday
0
1



Histogram for weekday
Count
150
100
50
0
1
2
3
4
5
6
7
weekday
wkday
1
2
3
4
5
6
7

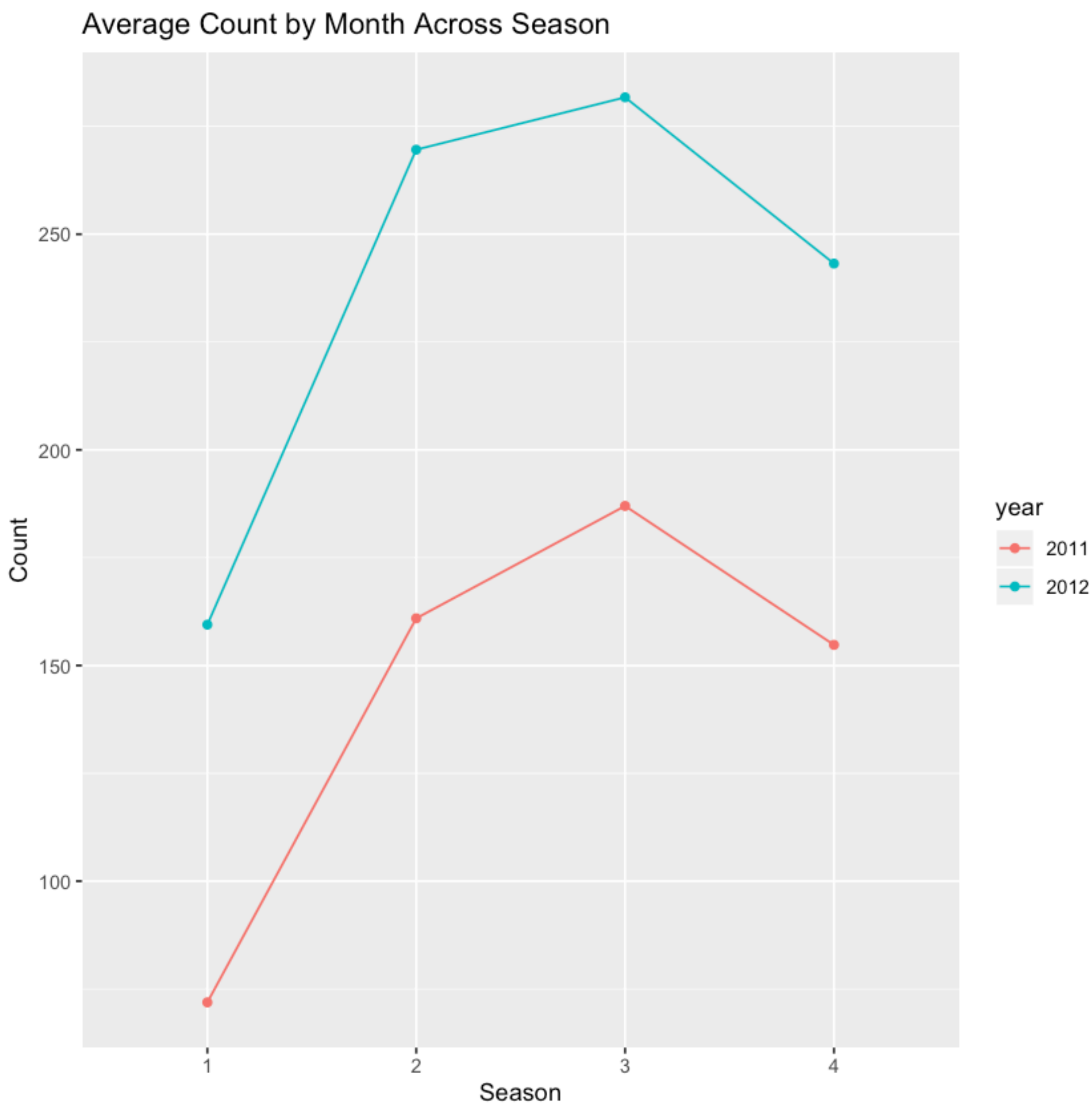


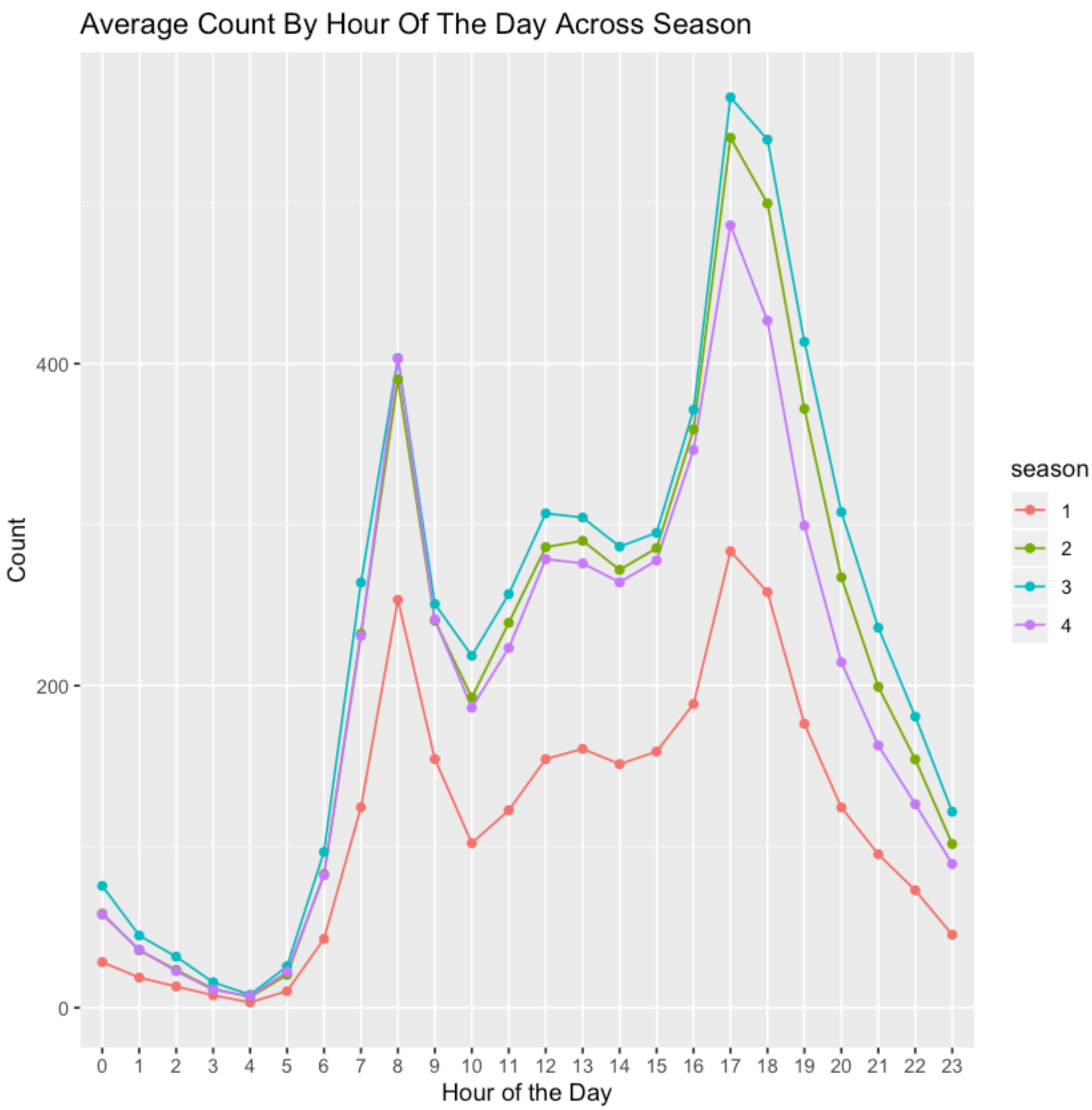
Histogram for working day
150
100
50
0
Count
0
1
working day
workingday
0
1



[33]:
```
# iii. Explore trends over time ---- exploring some more pairplots

            ggplot(bike, aes(x=season, y=count, group=year, color=year)) +
              stat_summary(
                fun.y=mean,
                geom='line'
              ) +
              stat_summary(
                fun.y=mean,
                geom='point'
              ) +
              labs(title="Average Count by Month Across Season") +
              labs(x="Season", y="Count")
```

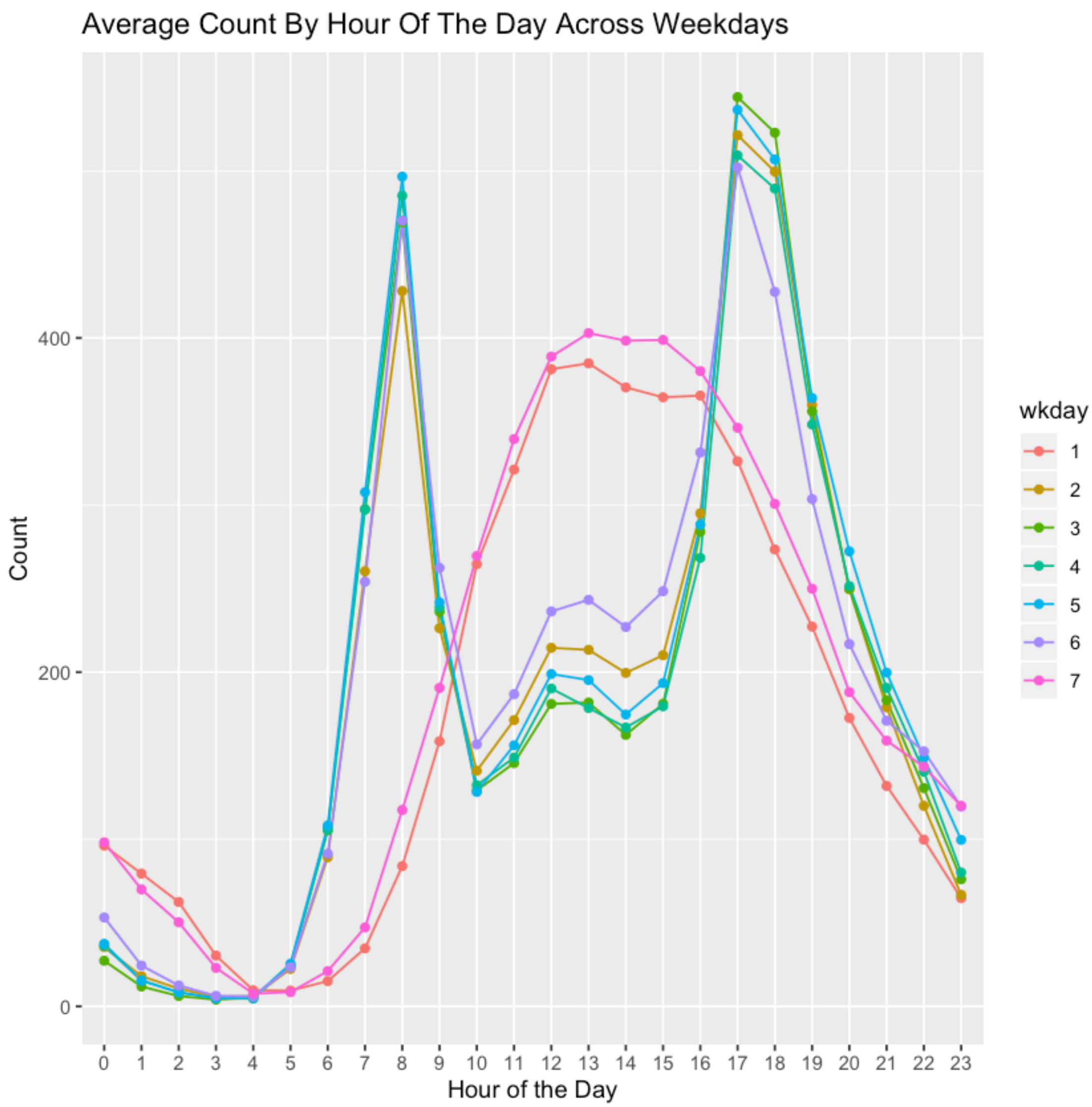
```
    ggplot(bike, aes(x=bike$hour, y=count, group=season, color=season))␣
↪+
      stat_summary(
        fun.y=mean,
        geom='line'
      ) +
      stat_summary(
        fun.y=mean,
        geom='point'
      )+
      labs(title="Average Count By Hour Of The Day Across Season") +
      labs(x="Hour of the Day", y="Count")

    ggplot(bike, aes(x=bike$hour, y=count, group=wkday, color=wkday)) +
      stat_summary(
        fun.y=mean,
        geom='line'
      ) +
      stat_summary(
        fun.y=mean,
        geom='point'
      )+
      labs(title="Average Count By Hour Of The Day Across Weekdays") +
      labs(x="Hour of the Day", y="Count")


ggplot(bike, aes(x=bike$day, y=count, group=day, color=day)) +
      stat_summary(
        fun.y=mean,
        geom='line'
      ) +
      stat_summary(
        fun.y=mean,
        geom='point'
      )+
      labs(title="Average Count By Day") +
      labs(x="HDay", y="Count")
```


Average Count by Month Across Season
250
200
150
100
Count
1
2
3
4
Season
year
2011
2012




```
Don't know how to automatically pick scale for object of type function.
Defaulting to continuous.
Don't know how to automatically pick scale for object of type function.
Defaulting to continuous.
ERROR while rich displaying an object: Error: Aesthetics must be valid data
columns. Problematic aesthetic(s): group = day, colour = day.
Did you mistype the name of a data column or forget to add stat()?

Traceback:
1. FUN(X[[i]], ...)
2. tryCatch(withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
```

```
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler), error = outer_handler)
3. tryCatchList(expr, classes, parentenv, handlers)
4. tryCatchOne(expr, names, parentenv, handlers[[1L]])
5. doTryCatch(return(expr), name, parentenv, handler)
6. withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler)
7. repr::mime2repr[[mime]](obj)
8. repr_text.default(obj)
9. paste(capture.output(print(obj)), collapse = "\n")
10. capture.output(print(obj))
11. evalVis(expr)
12. withVisible(eval(expr, pf))
13. eval(expr, pf)
14. eval(expr, pf)
15. print(obj)
16. print.ggplot(obj)
17. ggplot_build(x)
18. ggplot_build.ggplot(x)
19. by_layer(function(l, d) l$compute_aesthetics(d, plot))
20. f(l = layers[[i]], d = data[[i]])
21. l$compute_aesthetics(d, plot)
22. f(..., self = self)
23. stop(msg, call. = FALSE)
```



Average Count By Hour Of The Day Across Weekdays
Count
400
200
0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
Hour of the Day
wkday
1
2
3
4
5
6
7

[21]:

```
# 4c. Drop some variables from the dataset based on the analysis so far
        # drop temp, casual, registered and date
        bike_subset = bike[-c(5,9:10, 12)]
        head(bike_subset,5)
```

season	holiday	workingday	weather	atemp	humidity	windspeed	count	year	month	hou
1	0	0	1	14.395	81	0	16	2011	1	0
1	0	0	1	13.635	80	0	40	2011	1	1
1	0	0	1	13.635	80	0	32	2011	1	2
1	0	0	1	14.395	75	0	13	2011	1	3
1	0	0	1	14.395	75	0	1	2011	1	4

[ ]:

```
#------ Step 4: Exploratory Data Analysis ENDS Here------------------
# Final observations:
```

```
#1.) 'atemp' and 'temp' are very strongly correlated . Drop 'atemp' from the␣
↪dataset (since it has higher p-value
       #than 'temp')
#2.) 'date' does not seem to have any affect on count of bikes, it can be␣
↪dropped from the dataset
#-------------------------------------------------------------
```

[ ]:
```
#----------Part 5 : Model Builing starts here ----------------------
    # 5a. Split data into test and train set
    # 5b. Linear Regression
    # 5c. Random Forest
    # 5d. Gradient Boosting
```

[22]:
```
# 5a. Split data into test and train set
        sample_size = floor(0.8 * nrow(bike))
        set.seed(1)
        train_index = sample(nrow(bike), size = sample_size)
        train <- bike[train_index, ]
        test <- bike[-train_index, ]
```

[23]:
```
# 5b. Linear Regression
        # Fit Linear Model
        # drop atemp, registered, casual and date
        train_subset = train[-c(6,9:10, 12)]
        test_subset = test[-c(6,9:10, 12)]

        lm_fit = lm(count ~ ., data = train_subset)
        summary(lm_fit)

        # Choosing the best model by AIC in a Stepwise Algorithm
        # The step() function iteratively removes insignificant features from␣
↪the model.
        step(lm_fit)
        summary(lm_fit)

        # Calculate Train RMSLE
        y_act_train <- abs(train_subset$count)
        y_pred_train <- abs(predict(lm_fit, train_subset))
        lm_train_RMSLE = rmsle(y_act_train, y_pred_train)

        # Calculate Test RMSLE
        y_act_test <- abs(test_subset$count)
        y_pred_test <- abs(predict(lm_fit, test_subset))
        lm_test_RMSLE = rmsle(y_act_test, y_pred_test)

        # Save the results
        lm_results = predict(lm_fit, bike_test)
        hist(lm_results)
```

```
Call:
lm(formula = count ~ ., data = train_subset)

Residuals:
    Min      1Q  Median      3Q     Max
-351.68  -61.47   -7.12   50.93  438.13

Coefficients: (4 not defined because of singularities)
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -84.59331    9.33850  -9.059  < 2e-16 ***
season2       67.39603    7.87769   8.555  < 2e-16 ***
season3       76.82918    7.64092  10.055  < 2e-16 ***
season4       75.49214    5.62907  13.411  < 2e-16 ***
holiday1      11.78025    7.81568   1.507 0.131781
workingday1   14.71109    4.06430   3.620 0.000297 ***
weather2     -12.58747    2.67528  -4.705 2.58e-06 ***
weather3     -71.11945    4.47545 -15.891  < 2e-16 ***
weather4    -174.84435  100.78681  -1.735 0.082813 .
temp           5.03023    0.33352  15.082  < 2e-16 ***
humidity      -0.76332    0.07849  -9.724  < 2e-16 ***
windspeed     -0.54238    0.14408  -3.764 0.000168 ***
year2012      86.95570    2.19273  39.656  < 2e-16 ***
month2        11.13212    5.39034   2.065 0.038934 *
month3        29.41766    5.77185   5.097 3.53e-07 ***
month4       -16.76422    5.99977  -2.794 0.005215 **
month5        12.64662    5.45402   2.319 0.020431 *
month6              NA         NA      NA       NA
month7       -36.80569    5.57965  -6.596 4.46e-11 ***
month8       -27.30585    5.45357  -5.007 5.64e-07 ***
month9              NA         NA      NA       NA
month10       21.16010    5.77289   3.665 0.000248 ***
month11        1.14471    5.35424   0.214 0.830712
month12             NA         NA      NA       NA
hour1        -11.39400    7.48756  -1.522 0.128115
hour2        -24.02145    7.45155  -3.224 0.001270 **
hour3        -37.44770    7.55542  -4.956 7.32e-07 ***
hour4        -38.01239    7.44329  -5.107 3.34e-07 ***
hour5        -23.47057    7.47555  -3.140 0.001697 **
hour6         36.59158    7.41431   4.935 8.15e-07 ***
hour7        170.52864    7.39633  23.056  < 2e-16 ***
hour8        311.38508    7.44159  41.844  < 2e-16 ***
hour9        164.73930    7.38202  22.316  < 2e-16 ***
hour10       113.53297    7.46630  15.206  < 2e-16 ***
hour11       140.80547    7.50670  18.757  < 2e-16 ***
hour12       177.90103    7.56740  23.509  < 2e-16 ***
hour13       177.19756    7.66452  23.119  < 2e-16 ***
hour14       162.02489    7.65093  21.177  < 2e-16 ***
hour15       168.32943    7.59391  22.166  < 2e-16 ***
```

```
hour16       231.47403    7.62564  30.355  < 2e-16 ***
hour17       387.32373    7.66555  50.528  < 2e-16 ***
hour18       360.43568    7.58264  47.534  < 2e-16 ***
hour19       245.60360    7.43058  33.053  < 2e-16 ***
hour20       164.11798    7.51229  21.847  < 2e-16 ***
hour21       114.00143    7.44391  15.315  < 2e-16 ***
hour22        75.29220    7.45039  10.106  < 2e-16 ***
hour23        37.22724    7.37198   5.050 4.51e-07 ***
wkday2       -11.01087    4.16184  -2.646 0.008168 **
wkday3        -7.87850    4.10615  -1.919 0.055054 .
wkday4        -4.32022    4.10701  -1.052 0.292868
wkday5        -2.93373    4.07082  -0.721 0.471130
wkday6              NA         NA      NA       NA
wkday7        16.38078    3.99532   4.100 4.17e-05 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 100.5 on 8659 degrees of freedom
Multiple R-squared:  0.6936,Adjusted R-squared:  0.6919
F-statistic: 408.3 on 48 and 8659 DF,  p-value: < 2.2e-16



Start:  AIC=80333.49
count ~ season + holiday + workingday + weather + temp + humidity +
    windspeed + year + month + hour + wkday


Step:  AIC=80333.49
count ~ season + holiday + weather + temp + humidity + windspeed +
    year + month + hour + wkday


Step:  AIC=80333.49
count ~ holiday + weather + temp + humidity + windspeed + year +
    month + hour + wkday

            Df Sum of Sq       RSS   AIC
- holiday    1      1816  87400849 80332
<none>                    87399033 80333
- windspeed  1    143031  87542064 80346
- wkday      6    258475  87657508 80347
- humidity   1    954493  88353526 80426
- temp       1   2295989  89695022 80557
- weather    3   2571101  89970134 80580
- month     11   5734773  93133806 80865
- year       1  15873198 103272230 81785
- hour      23 100736154 188135187 86964
```

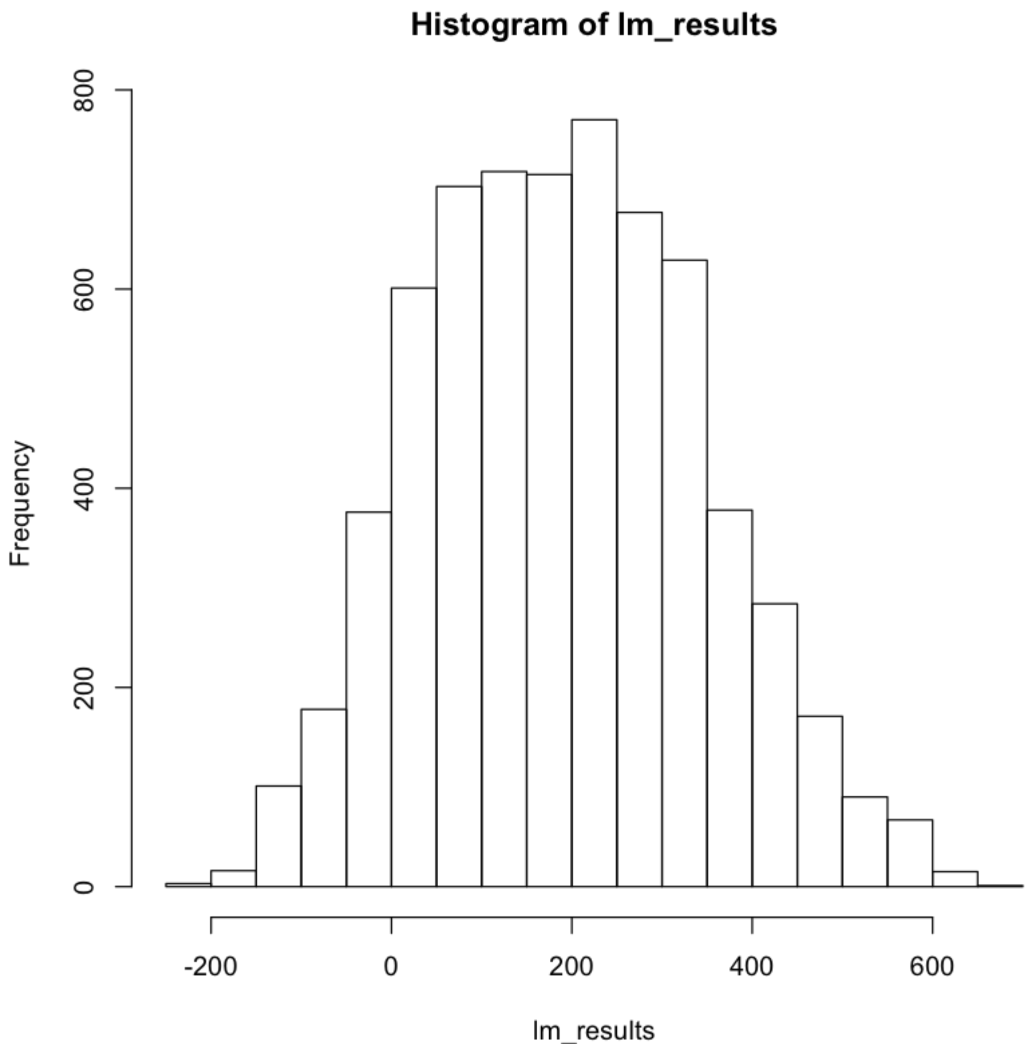
```
Step:  AIC=80331.67
count ~ weather + temp + humidity + windspeed + year + month +
    hour + wkday

            Df Sum of Sq       RSS   AIC
<none>                    87400849 80332
- windspeed  1    142973  87543822 80344
- wkday      6    265216  87666065 80346
- humidity   1    955401  88356250 80424
- temp       1   2294225  89695074 80555
- weather    3   2569967  89970816 80578
- month     11   5741240  93142089 80864
- year       1  15873066 103273914 81783
- hour      23 100753472 188154320 86963

Call:
lm(formula = count ~ weather + temp + humidity + windspeed +
    year + month + hour + wkday, data = train_subset)

Coefficients:
(Intercept)     weather2     weather3     weather4         temp     humidity
   -84.6740     -12.5972     -71.1021    -174.2638       5.0256      -0.7636
  windspeed     year2012       month2       month3       month4       month5
    -0.5423      86.9553      11.3517      29.6681      50.7423      80.3289
     month6       month7       month8       month9      month10      month11
    67.7000      40.1953      49.8426      76.9729      96.7900      76.7454
    month12        hour1        hour2        hour3        hour4        hour5
    75.7494     -11.4051     -24.0201     -37.4611     -38.0175     -23.4766
      hour6        hour7        hour8        hour9       hour10       hour11
    36.5740     170.5120     311.3926     164.7376     113.5519     140.8194
     hour12       hour13       hour14       hour15       hour16       hour17
   177.9067     177.2196     162.0334     168.3269     231.4745     387.3397
     hour18       hour19       hour20       hour21       hour22       hour23
   360.4454     245.6220     164.0973     114.0212      75.2779      37.2276
     wkday2       wkday3       wkday4       wkday5       wkday6       wkday7
     3.2566       6.8380      10.3442      11.7749      14.6134      16.3761
```

```
Call:
lm(formula = count ~ ., data = train_subset)

Residuals:
    Min      1Q  Median      3Q     Max
-351.68  -61.47   -7.12   50.93  438.13

Coefficients: (4 not defined because of singularities)
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -84.59331    9.33850  -9.059  < 2e-16 ***
season2       67.39603    7.87769   8.555  < 2e-16 ***
season3       76.82918    7.64092  10.055  < 2e-16 ***
season4       75.49214    5.62907  13.411  < 2e-16 ***
holiday1      11.78025    7.81568   1.507 0.131781
workingday1   14.71109    4.06430   3.620 0.000297 ***
weather2     -12.58747    2.67528  -4.705 2.58e-06 ***
weather3     -71.11945    4.47545 -15.891  < 2e-16 ***
weather4    -174.84435  100.78681  -1.735 0.082813 .
temp           5.03023    0.33352  15.082  < 2e-16 ***
humidity      -0.76332    0.07849  -9.724  < 2e-16 ***
windspeed     -0.54238    0.14408  -3.764 0.000168 ***
year2012      86.95570    2.19273  39.656  < 2e-16 ***
month2        11.13212    5.39034   2.065 0.038934 *
month3        29.41766    5.77185   5.097 3.53e-07 ***
month4       -16.76422    5.99977  -2.794 0.005215 **
month5        12.64662    5.45402   2.319 0.020431 *
month6              NA         NA      NA       NA
month7       -36.80569    5.57965  -6.596 4.46e-11 ***
month8       -27.30585    5.45357  -5.007 5.64e-07 ***
month9              NA         NA      NA       NA
month10       21.16010    5.77289   3.665 0.000248 ***
month11        1.14471    5.35424   0.214 0.830712
month12             NA         NA      NA       NA
hour1        -11.39400    7.48756  -1.522 0.128115
hour2        -24.02145    7.45155  -3.224 0.001270 **
hour3        -37.44770    7.55542  -4.956 7.32e-07 ***
hour4        -38.01239    7.44329  -5.107 3.34e-07 ***
hour5        -23.47057    7.47555  -3.140 0.001697 **
hour6         36.59158    7.41431   4.935 8.15e-07 ***
hour7        170.52864    7.39633  23.056  < 2e-16 ***
hour8        311.38508    7.44159  41.844  < 2e-16 ***
hour9        164.73930    7.38202  22.316  < 2e-16 ***
hour10       113.53297    7.46630  15.206  < 2e-16 ***
hour11       140.80547    7.50670  18.757  < 2e-16 ***
hour12       177.90103    7.56740  23.509  < 2e-16 ***
hour13       177.19756    7.66452  23.119  < 2e-16 ***
hour14       162.02489    7.65093  21.177  < 2e-16 ***
hour15       168.32943    7.59391  22.166  < 2e-16 ***
hour16       231.47403    7.62564  30.355  < 2e-16 ***
hour17       387.32373    7.66555  50.528  < 2e-16 ***
hour18       360.43568    7.58264  47.534  < 2e-16 ***
hour19       245.60360    7.43058  33.053  < 2e-16 ***
hour20       164.11798    7.51229  21.847  < 2e-16 ***
hour21       114.00143    7.44391  15.315  < 2e-16 ***
hour22        75.29220    7.45039  10.106  < 2e-16 ***
hour23        37.22724    7.37198   5.050 4.51e-07 ***
```

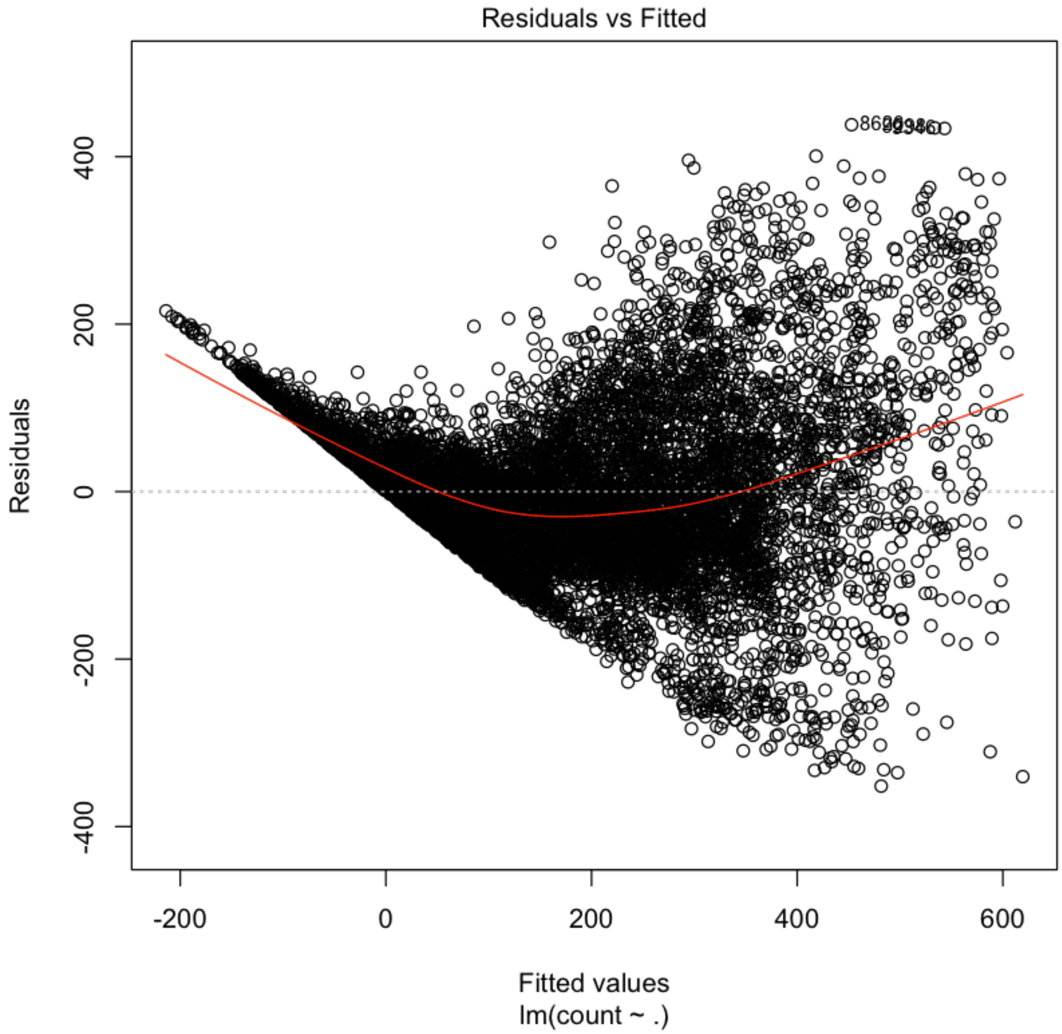
```
wkday2       -11.01087    4.16184  -2.646 0.008168 **
wkday3        -7.87850    4.10615  -1.919 0.055054 .
wkday4        -4.32022    4.10701  -1.052 0.292868
wkday5        -2.93373    4.07082  -0.721 0.471130
wkday6              NA         NA      NA       NA
wkday7        16.38078    3.99532   4.100 4.17e-05 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 100.5 on 8659 degrees of freedom
Multiple R-squared:  0.6936,Adjusted R-squared:  0.6919
F-statistic: 408.3 on 48 and 8659 DF,  p-value: < 2.2e-16
```

```
Warning message in predict.lm(lm_fit, train_subset):
prediction from a rank-deficient fit may be misleadingWarning message in
predict.lm(lm_fit, test_subset):
prediction from a rank-deficient fit may be misleadingWarning message in
predict.lm(lm_fit, bike_test):
```

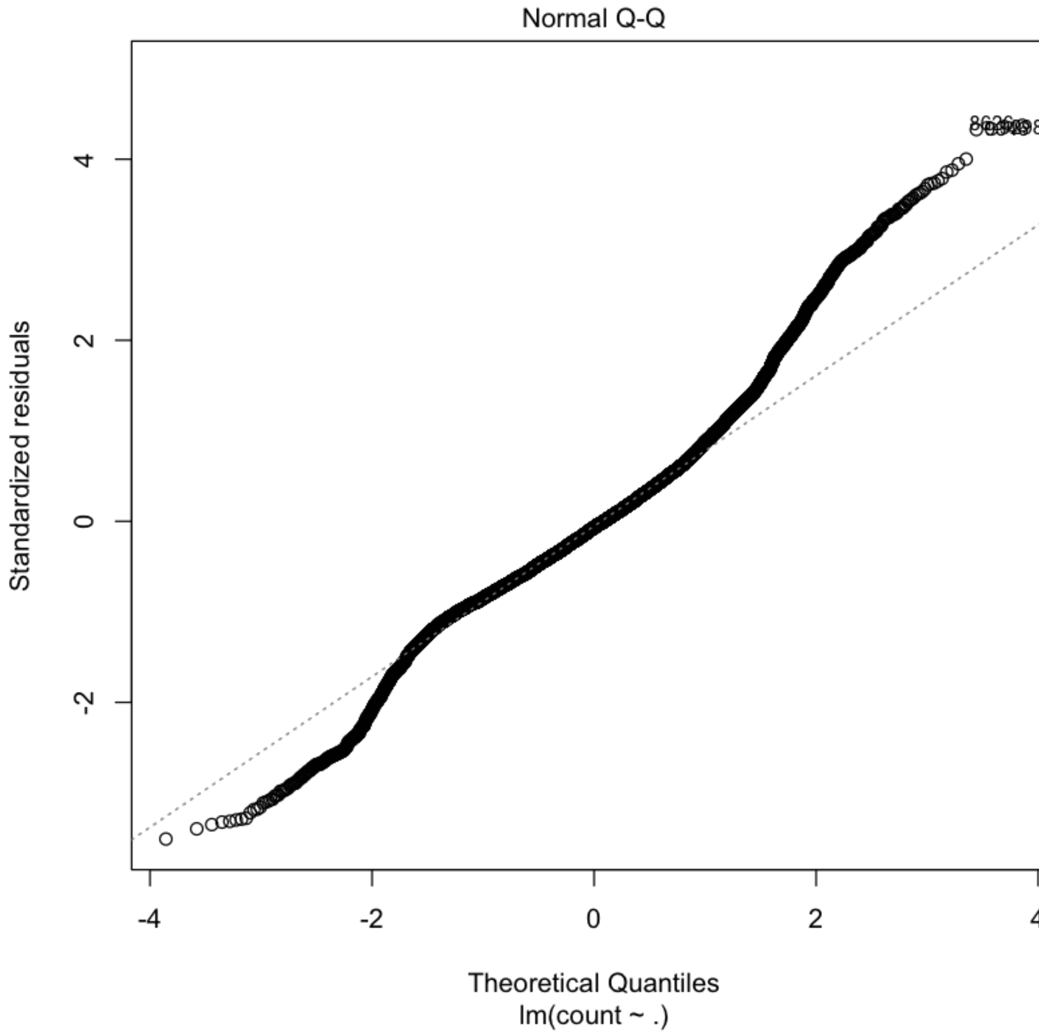


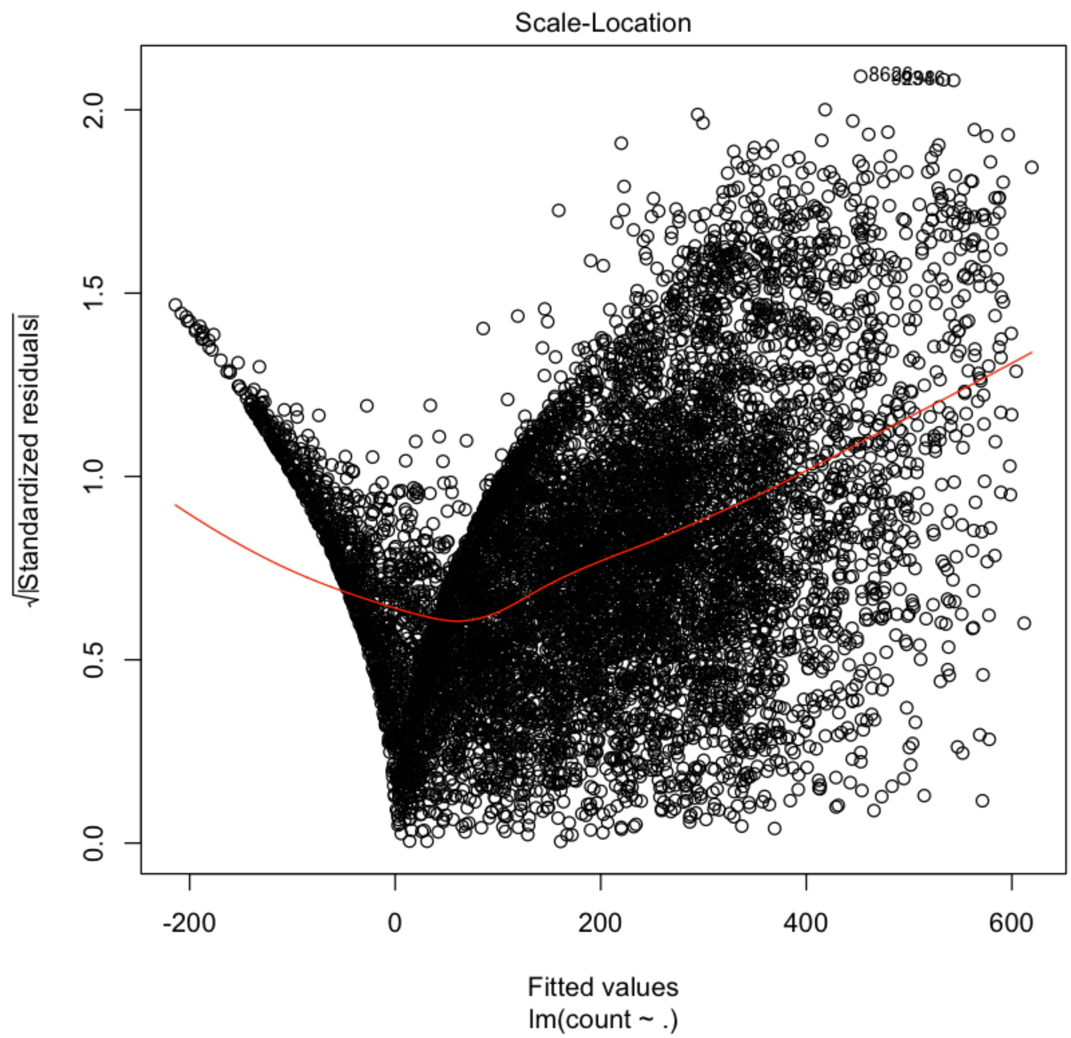
[24]:
```
plot(lm_fit)
```

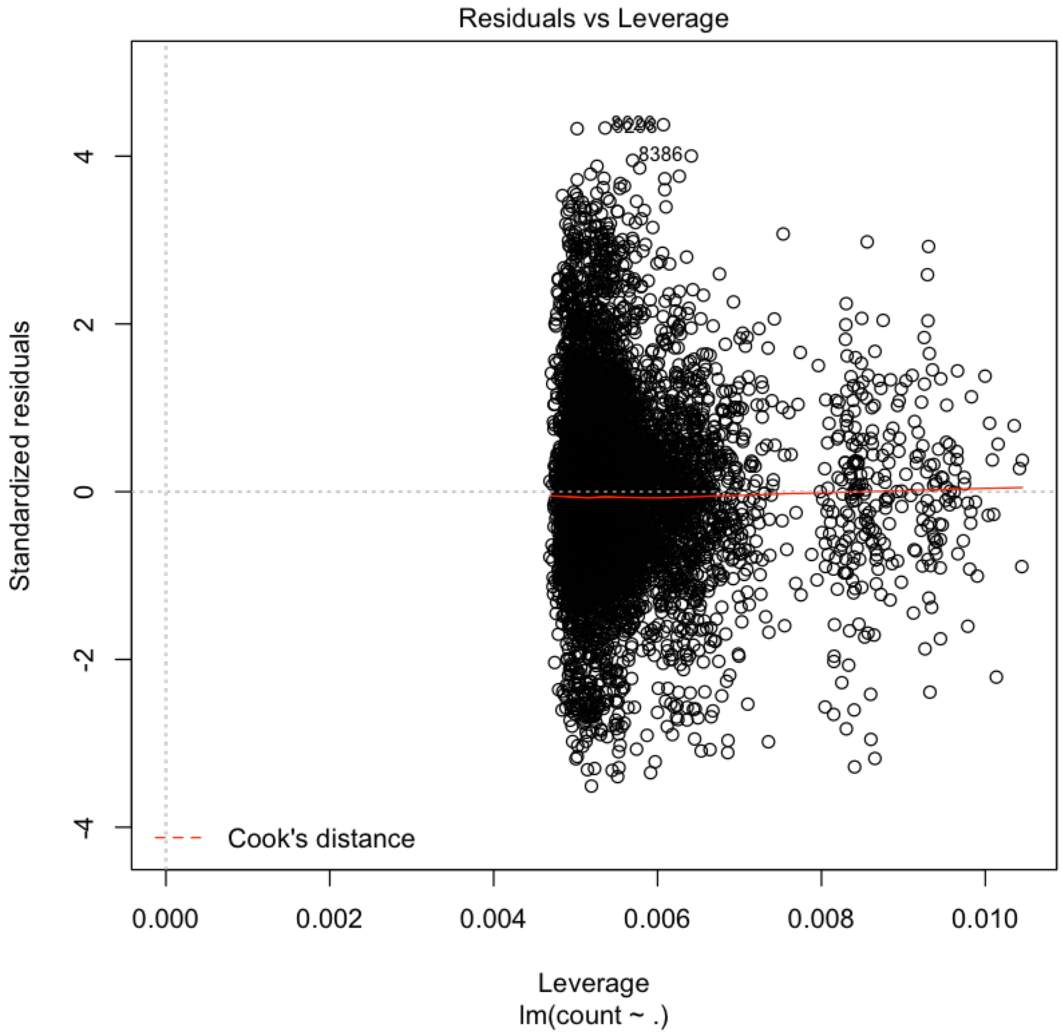
```
Warning message:
not plotting observations with leverage one:
```



Warning message:
not plotting observations with leverage one:


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(count ~ .)


Scale-Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
-200
0
200
400
600
Fitted values
lm(count ~ .)



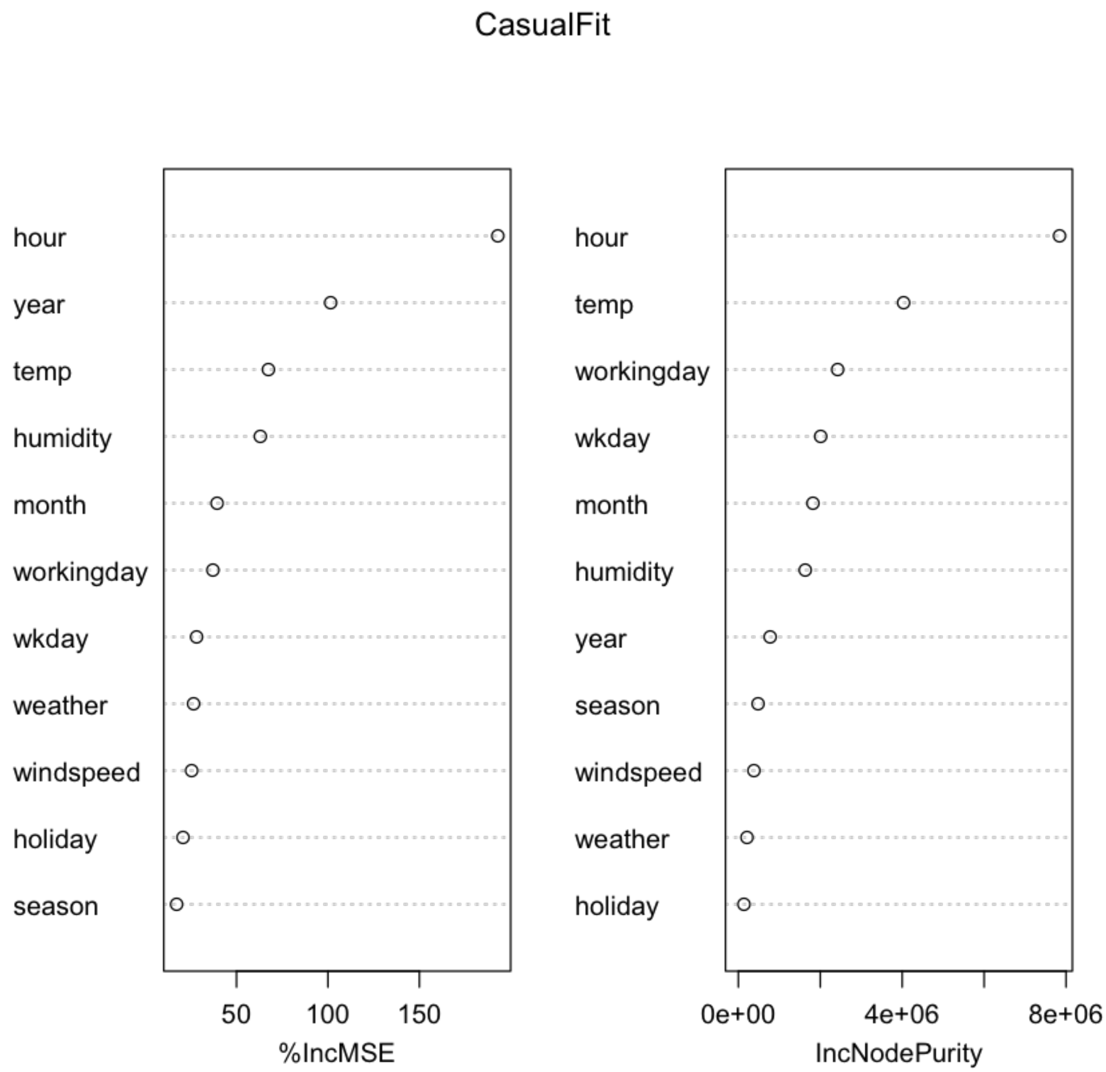
[25]:
```
mse(y_act_train,y_pred_train)
rmse(y_act_train,y_pred_train)
rmsle(y_act_train,y_pred_train)

mse(y_act_test,y_pred_test)
rmse(y_act_test,y_pred_test)
rmsle(y_act_test,y_pred_test)
```

9818.80878274557
99.0899025266731
1.02779704287991
9763.82893890566
98.8120890321911
1.00488949037201

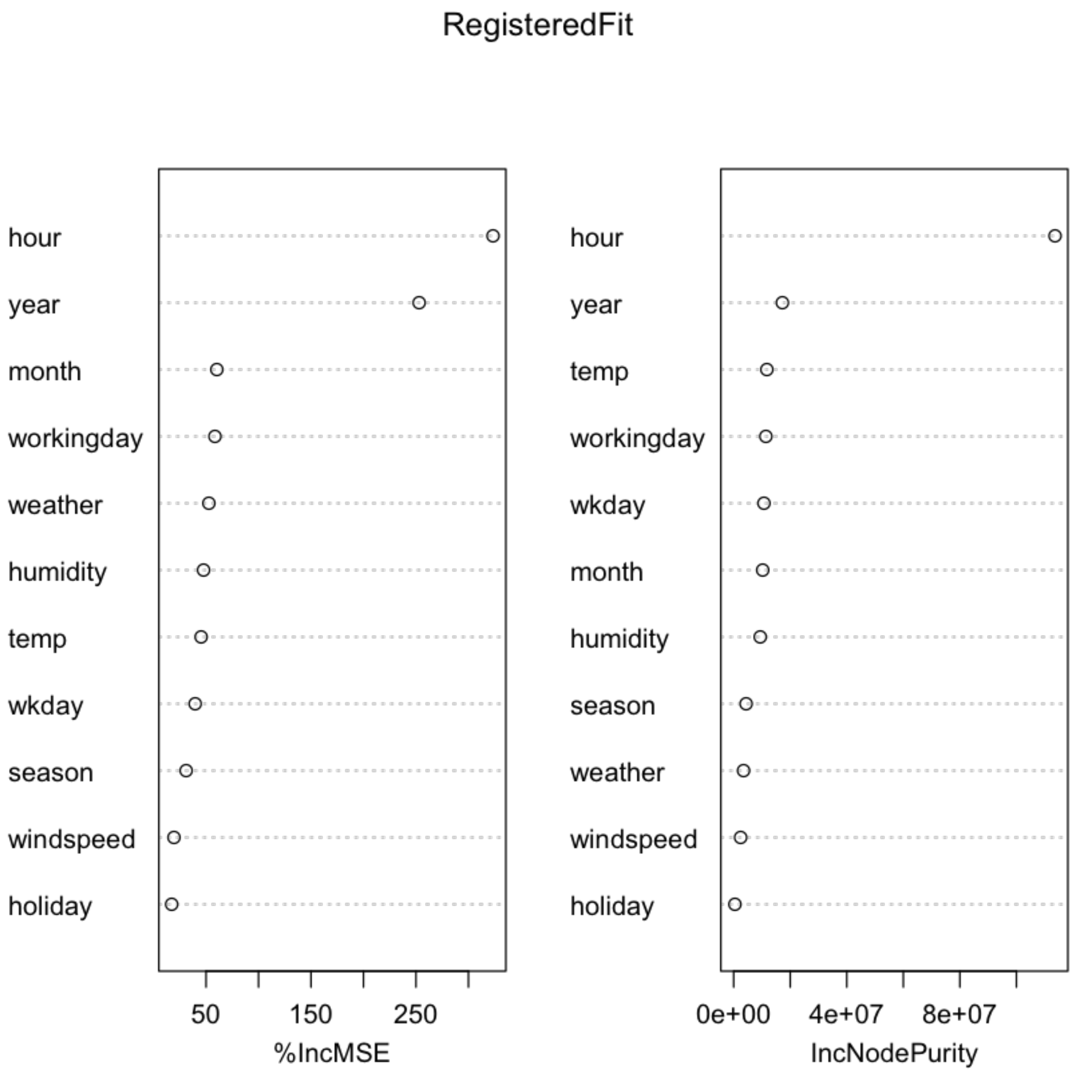
[26]:
```
# 5b. Random Forest
        Ntree=500
        Mtry = 5
        myImportance = TRUE

        # Predict Casual Counts
        set.seed(1)
        CasualData <- subset(train, select = -c(count, registered, date, atemp))
        CasualFit <- randomForest(casual ~ ., data=CasualData, ntree=Ntree,␣
↪mtry=Mtry,
                                importance=myImportance)


        # Predict Registered Counts
        RegisteredData <- subset(train, select = -c(count, casual, date, atemp))
        RegisteredFit <- randomForest(registered ~ ., data=RegisteredData,␣
↪ntree=Ntree, mtry=Mtry,
                                importance=myImportance)
```

[34]:
```
varImpPlot(CasualFit)
        varImp(CasualFit)

        varImpPlot(RegisteredFit)
        varImp(RegisteredFit)

    #Inference - Casual Fit: season, holiday, windspeed and weather are not␣
↪much significant here.
    #Inference - Registered Fit: season, holiday, windspeed and weekday are not␣
↪much significant here.
```

	Overall
season	17.24307
holiday	20.74985
workingday	37.09054
weather	26.53064
temp	67.45059
humidity	63.02898
windspeed	25.46503
year	101.47007
month	39.38764
hour	192.87762
wkday	28.16724



	Overall
season	31.02606
holiday	17.26589
workingday	58.48382
weather	52.71161
temp	45.31425
humidity	47.63264
windspeed	19.45475
year	253.04092
month	60.28591
hour	323.42262
wkday	39.65462



[35]:
```
casualFitFinal <- randomForest(casual ~ hour + year + humidity + month + temp +
↪workingday + wkday,
                             data=CasualData, ntree=Ntree,
↪mtry=Mtry,importance=myImportance)
        RegisteredFitFinal <- randomForest(registered ~ hour + year + month +
↪weather + workingday + humidity + temp,
                                      data=RegisteredData, ntree=Ntree,
↪mtry=Mtry,importance=myImportance)
```

[36]:
```
# Prediction on train data

            # Prediction on train data - casual users
            PredTrainCasual = round(predict(CasualFit, train),0)
            PredTrainCasualFinal = round(predict(casualFitFinal, train),0)
```

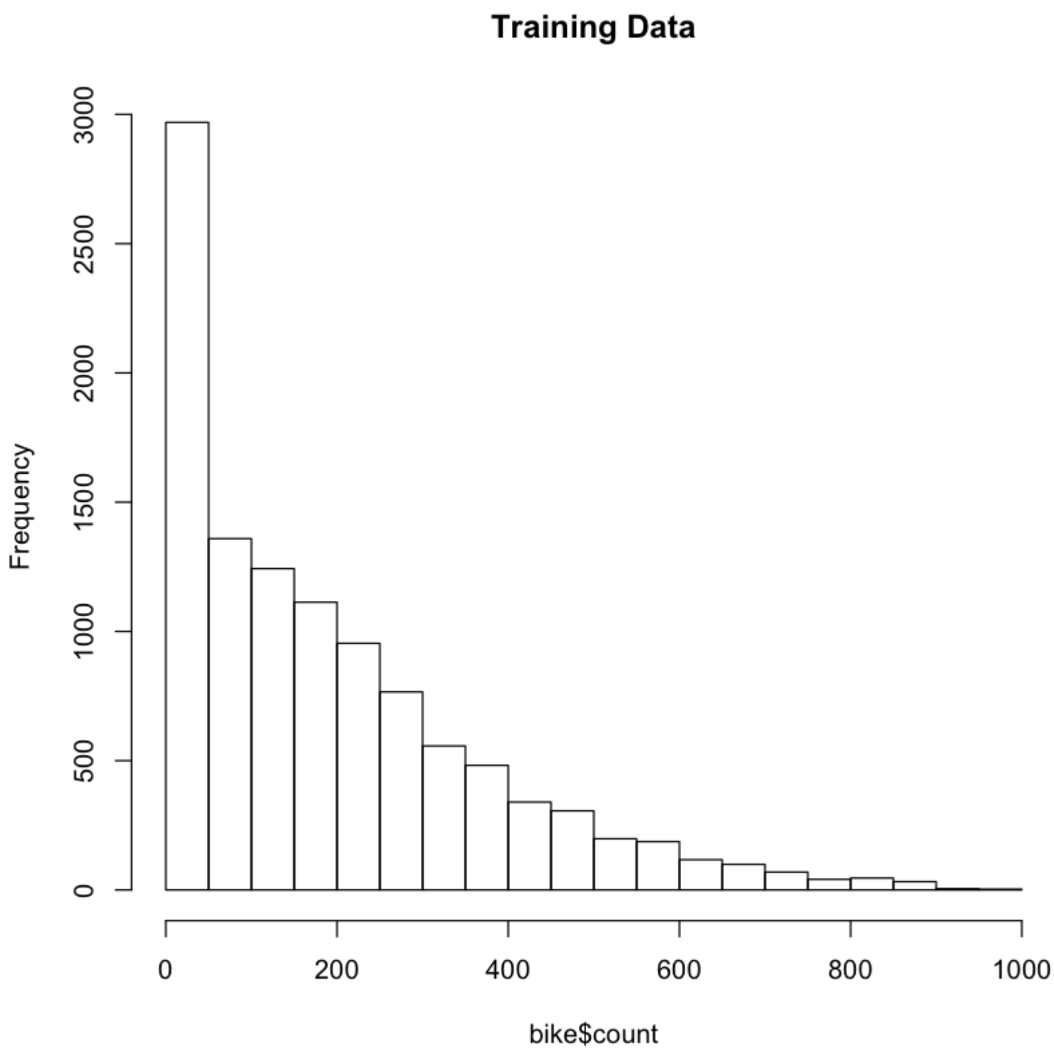
```
            # Prediction on train data - Registered users
            PredTrainRegistered = round(predict(RegisteredFit, train),0)
            PredTrainRegisteredFinal = round(predict(RegisteredFitFinal,␣
↪train),0)

            # Sum up Casual and Registered to get Total Count
            PredTrainCount = PredTrainCasual+PredTrainRegistered
            PredTrainCountFinal = PredTrainCasualFinal+PredTrainRegisteredFinal

            # Calculate Train RMSLE
            rf_train_rmsle_full = rmsle(train$count, PredTrainCount)
            rf_train_rmsle2_reduced = rmsle(train$count, PredTrainCountFinal)


        # Prediction on test data
            # Prediction on test data - casual users
            PredTestCasual = round(predict(CasualFit, test),0)
            PredTestCasualFinal = round(predict(casualFitFinal, test),0)

            # Prediction on test data - registered users
            PredTestRegistered = round(predict(RegisteredFit, test),0)
            PredTestRegisteredFinal = round(predict(RegisteredFitFinal, test),0)

            # Sum up Casual and Registered to get Total Count
            PredTestCount = PredTestCasual+PredTestRegistered
            PredTestCountFinal = PredTestCasualFinal+PredTestRegisteredFinal

            # Calculate Train RMSLE
            rf_test_rmsle_full = rmsle(test$count, PredTestCount)
            rf_test_rmsle2_reduced = rmsle(test$count, PredTestCountFinal)
```

[37]:

```
cat("Training RMSLE - Linear Regression: ", lm_train_RMSLE)
cat("\nTraining RMSLE - Random Forest (Full Model): ", rf_train_rmsle_full)
cat("\nTraining RMSLE - Random Forest (Reduced Model): : ",␣
↪rf_train_rmsle2_reduced)

cat("\n\nTest RMSLE - Linear Regression: ", lm_test_RMSLE)
cat("\nTest RMSLE - Random Forest (Full Model): ", rf_test_rmsle_full)
cat("\nTest RMSLE - Random Forest (Reduced Model): ", rf_test_rmsle2_reduced)
```

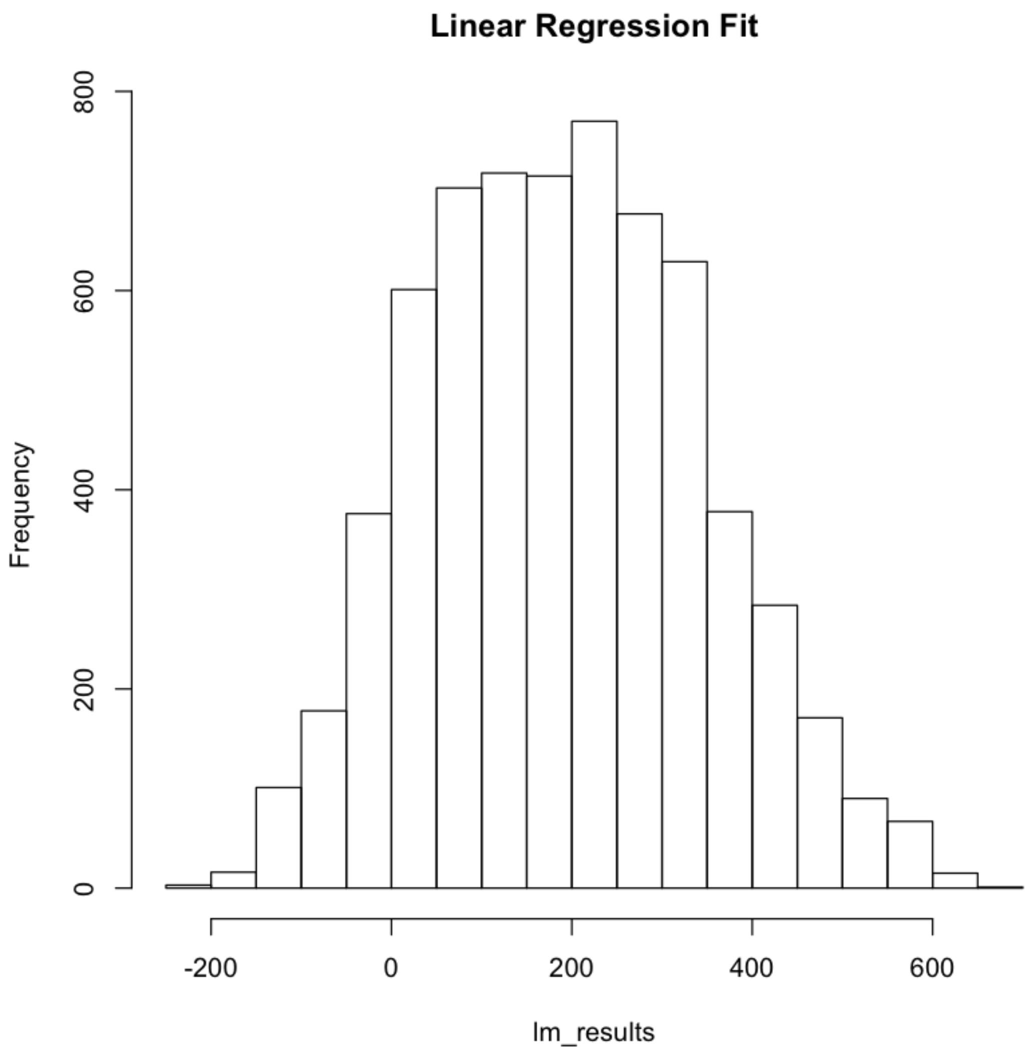
```
Training RMSLE - Linear Regression:  1.027797
Training RMSLE - Random Forest (Full Model):  0.2561525
Training RMSLE - Random Forest (Reduced Model): :  0.2147757

Test RMSLE - Linear Regression:  1.004889
Test RMSLE - Random Forest (Full Model):  0.4212346
```

[38]:
```
hist(bike$count, main="Training Data")
        hist(lm_results, main="Linear Regression Fit")
        hist(rf_results, main="Random Forest Fit")

        # Inference: The distribution of predicted count looks similar to that␣
 ↪of train data.
```




```
Error in hist(rf_results, main = "Random Forest Fit"): object␣
↪'rf_results' not found
   Traceback:
```

1. hist(rf_results, main = "Random Forest Fit")



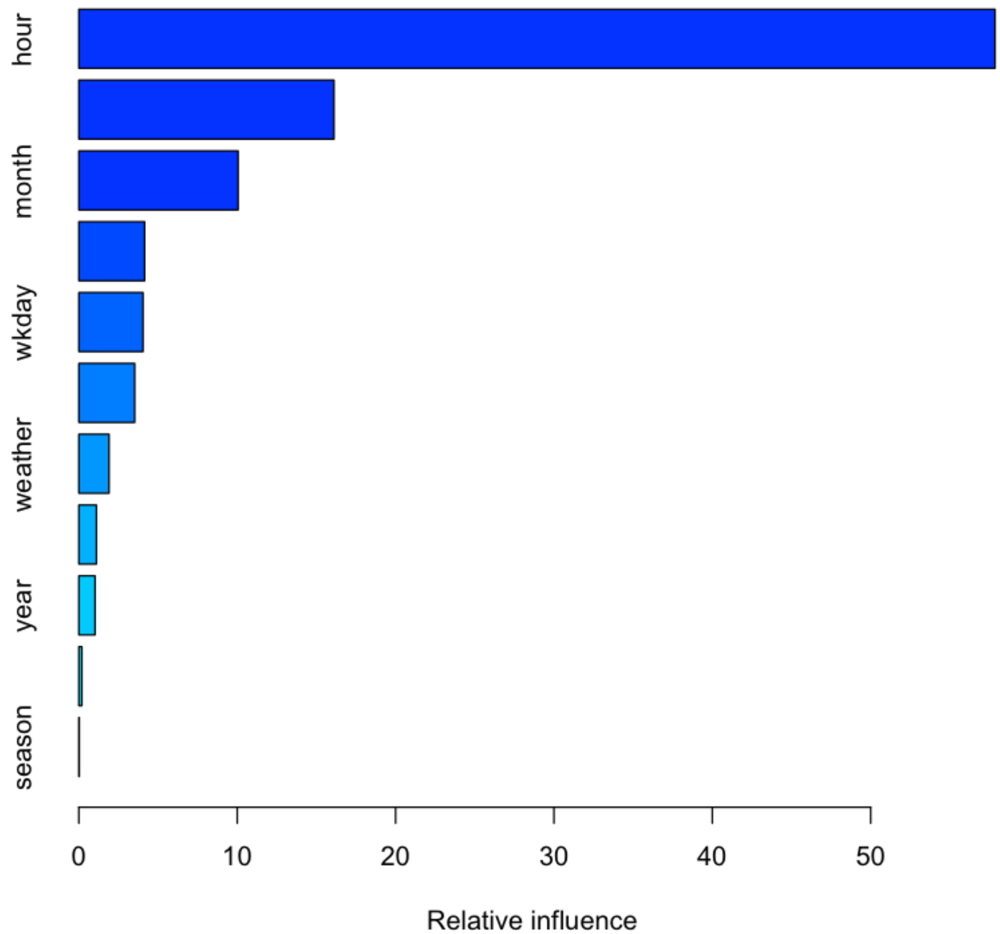
[39]:
```
# Save the RF results
        rf_test_casual = round(predict(casualFitFinal, bike_test),0)
        rf_test_registered = round(predict(RegisteredFitFinal, bike_test),)
        rf_results = rf_test_casual + rf_test_registered
```

[40]:
```
gbmtree=4000
iDepth = 3
set.seed(1)

# Predict Casual Counts
CasualData <- subset(train, select = -c(count, registered, atemp, date))
```

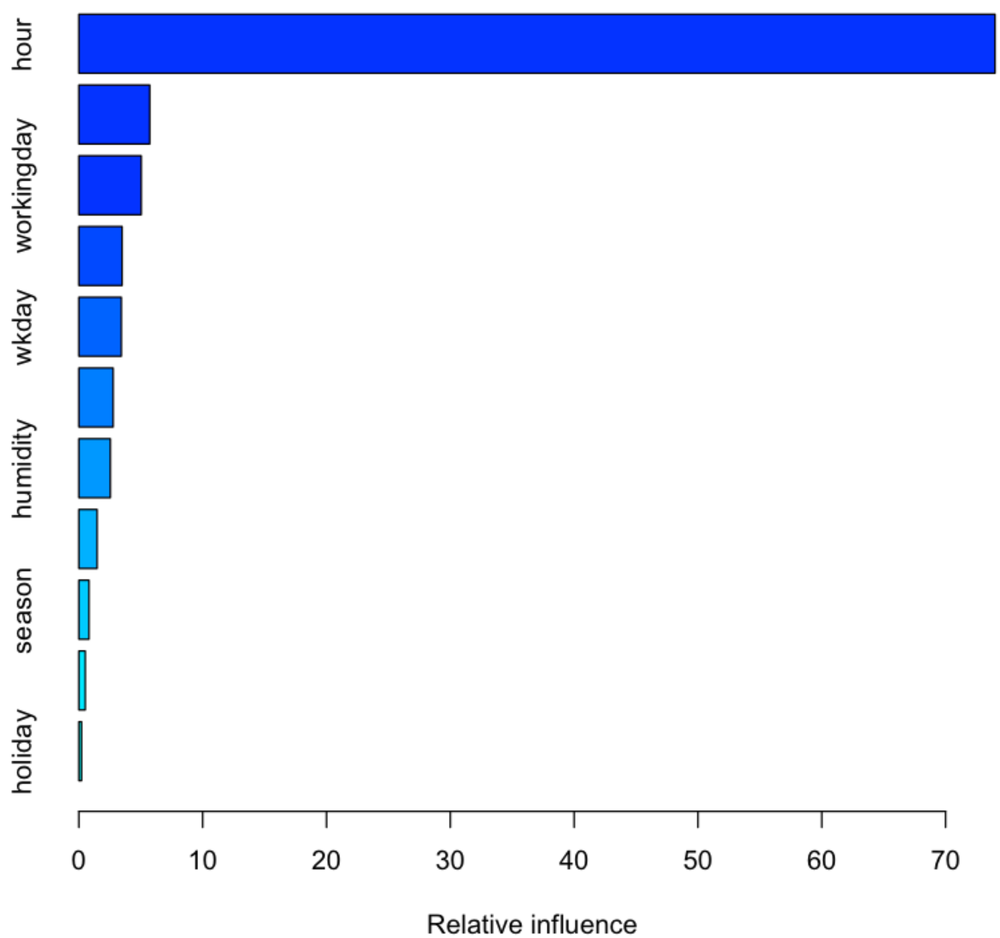
```
gbm.Casual <- gbm(log1p(casual)~.,data=CasualData,distribution= "gaussian",n.
↪trees=gbmtree,interaction.depth=iDepth)


# Predict Registered Counts
RegisteredData <- subset(train, select = -c(count, casual, atemp, date))
gbm.Registered <- gbm(log1p(registered)~.,data=RegisteredData,distribution=␣
↪"gaussian",n.trees=gbmtree,interaction.depth=iDepth)
```

[41]:
```
summary(gbm.Casual)
summary(gbm.Registered)

##Inference - gbm Casual: season, holiday, year, windspeed are not much␣
↪significant here.
##Inference - gbm Registered: holiday, windspeed, season, weather are not much␣
↪significant here.
```

	var	rel.inf
hour	hour	57.85237775
temp	temp	16.11053055
month	month	10.05511534
humidity	humidity	4.15202731
wkday	wkday	4.05126767
workingday	workingday	3.52722769
weather	weather	1.90563745
windspeed	windspeed	1.10793870
year	year	1.02249663
holiday	holiday	0.19081563
season	season	0.02456529



	var	rel.inf
hour	hour	73.9936471
month	month	5.7223401
workingday	workingday	5.0367884
year	year	3.4921877
wkday	wkday	3.4287371
temp	temp	2.7665790
humidity	humidity	2.5423855
weather	weather	1.4728776
season	season	0.8166783
windspeed	windspeed	0.5150276
holiday	holiday	0.2127515



[42]:
```
gbm.CasualFinal <- gbm(log1p(casual) ~ hour + workingday + temp + month  + ␣
↪wkday + humidity + weather,
                              data=CasualData, distribution= "gaussian",n.
↪trees=gbmtree,interaction.depth=iDepth)
gbm.RegisteredFinal <- gbm(log1p(registered) ~ hour + year + workingday + month␣
↪+ wkday + humidity + temp,
                          data=RegisteredData, distribution= "gaussian",n.
↪trees=gbmtree,interaction.depth=iDepth)
```

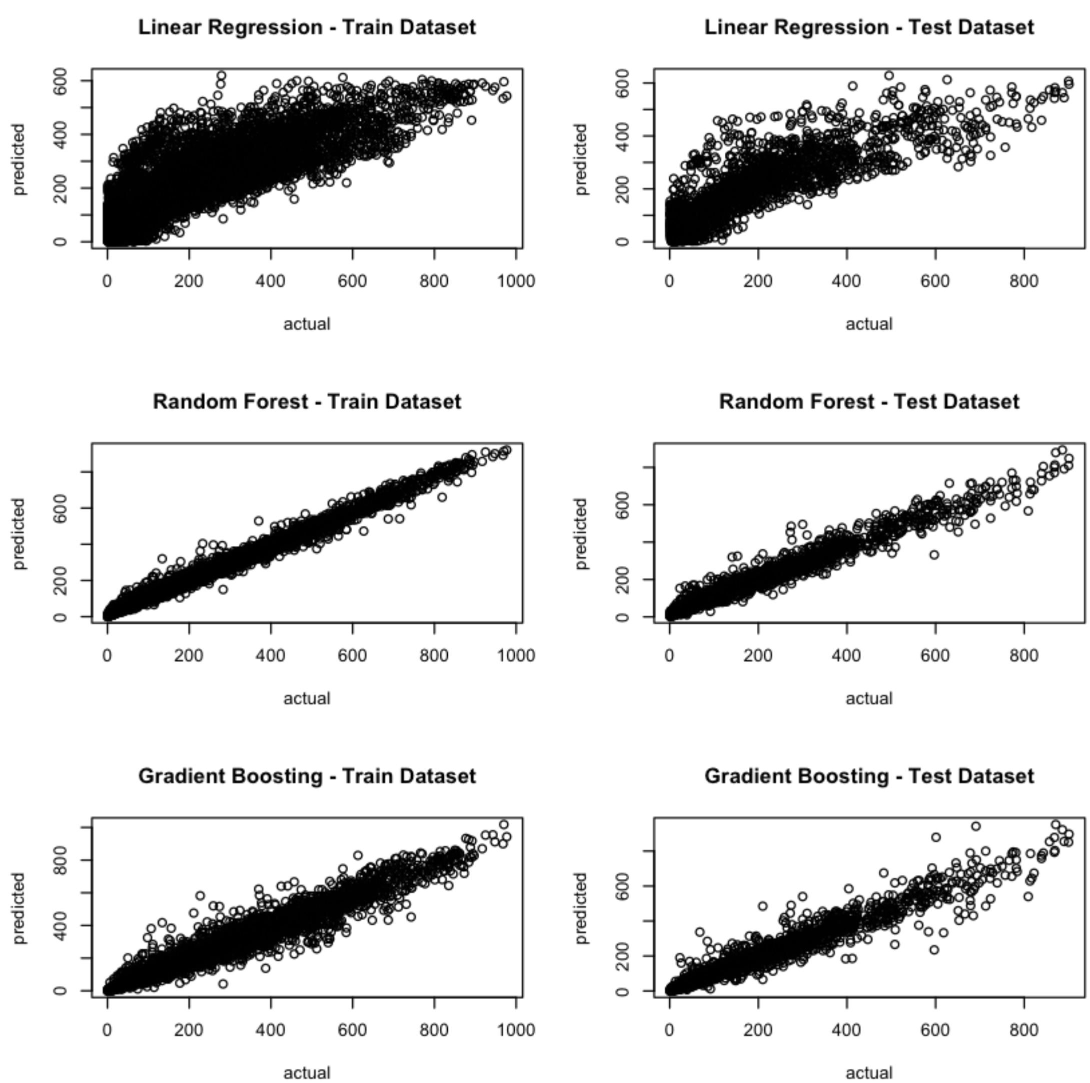
[43]:
```
# Prediction on train data
  # Prediction on train data - casual users

gbm.PredTrainCasual <- predict(gbm.Casual, train, n.trees=gbmtree)
gbm.PredTrainCasualFinal <- predict(gbm.CasualFinal, train, n.trees=gbmtree)
```

```
# Prediction on train data - Registered users
gbm.PredTrainRegistered <- predict(gbm.Registered, train, n.trees=gbmtree)
gbm.PredTrainRegisteredFinal <- predict(gbm.RegisteredFinal, train, n.
↪trees=gbmtree)

# Sum up Casual and Registered to get Total Count
gbm.PredTrainCount <- round(exp(gbm.PredTrainCasual) - 1, 0) + round(exp(gbm.
↪PredTrainRegistered) - 1, 0)
gbm.PredTrainCountFinal <- round(exp(gbm.PredTrainCasualFinal) - 1, 0) +␣
↪round(exp(gbm.PredTrainRegisteredFinal) - 1, 0)

# Calculate Train RMSLE
gbm.rf_train_rmsle_full <- rmsle(train$count, gbm.PredTrainCount)
gbm.rf_train_rmsle2_reduced <- rmsle(train$count, gbm.PredTrainCountFinal)

# Prediction on test data
# Prediction on test data - casual users
gbm.PredTestCasual = predict(gbm.Casual, test, n.trees=gbmtree)
gbm.PredTestCasualFinal = predict(gbm.CasualFinal, test, n.trees=gbmtree)

# Prediction on test data - registered users
gbm.PredTestRegistered = predict(gbm.Registered, test, n.trees=gbmtree)
gbm.PredTestRegisteredFinal = predict(gbm.RegisteredFinal, test, n.
↪trees=gbmtree)

# Sum up Casual and Registered to get Total Count
gbm.PredTestCount = round(exp(gbm.PredTestCasual) - 1, 0) + round(exp(gbm.
↪PredTestRegistered) - 1, 0)
gbm.PredTestCountFinal = round(exp(gbm.PredTestCasualFinal) - 1, 0) +␣
↪round(exp(gbm.PredTestRegisteredFinal) - 1, 0)

# Calculate Test RMSLE
gbm.rf_test_rmsle_full = rmsle(test$count, gbm.PredTestCount)
gbm.rf_test_rmsle2_reduced = rmsle(test$count, gbm.PredTestCountFinal)
```

[44]:
```
gbm.rf_train_rmsle_full
gbm.rf_train_rmsle2_reduced

gbm.rf_test_rmsle_full
gbm.rf_test_rmsle2_reduced
```

0.215991634854015
0.241986515661299
0.277804015565458
0.298095056429441

[46]:
```
par(mfrow=c(3,2))
plot(y_act_train, y_pred_train, main="Linear Regression - Train Dataset",␣
 ↪xlab="actual", ylab="predicted")
plot(y_act_test, y_pred_test, main="Linear Regression - Test Dataset",␣
 ↪xlab="actual", ylab="predicted")

plot(train$count, PredTrainCount, main="Random Forest - Train Dataset",␣
 ↪xlab="actual", ylab="predicted")
plot(test$count, PredTestCount, main="Random Forest - Test Dataset",␣
 ↪xlab="actual", ylab="predicted")

plot(train$count, gbm.PredTrainCountFinal, main="Gradient Boosting - Train␣
 ↪Dataset", xlab="actual", ylab="predicted")
plot(test$count, gbm.PredTestCountFinal, main="Gradient Boosting - Test␣
 ↪Dataset", xlab="actual", ylab="predicted")
```

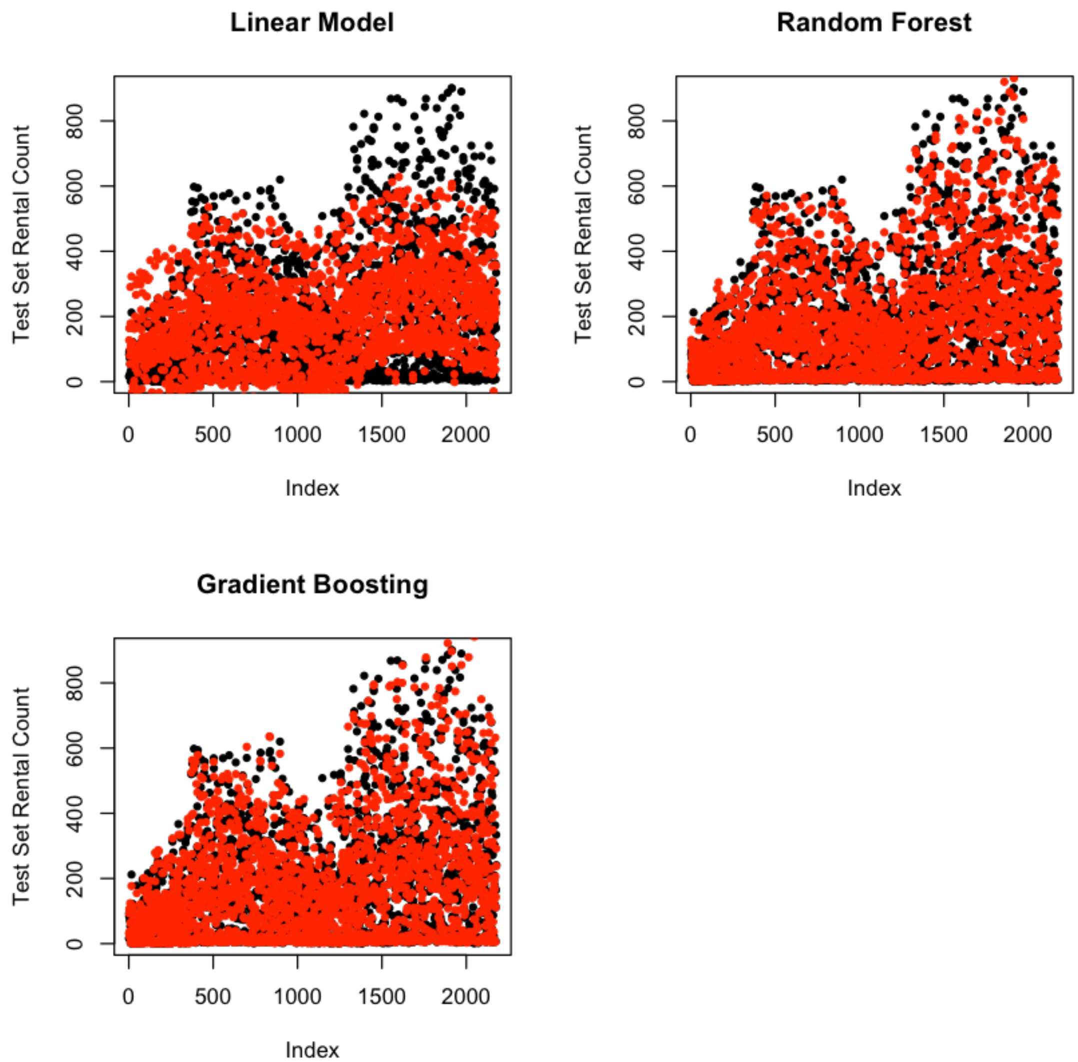


[48]:
```
cor(y_act_train, y_pred_train)
cor(y_act_test, y_pred_test)

cor(train$count, PredTrainCountFinal)
cor(test$count, PredTestCountFinal)

cor(train$count, gbm.PredTrainCountFinal)
cor(test$count, gbm.PredTestCountFinal)
```

0.843954967302359
0.846726206103257
0.993247383190163
0.975317861770545
0.978741240418922
0.970021541913715

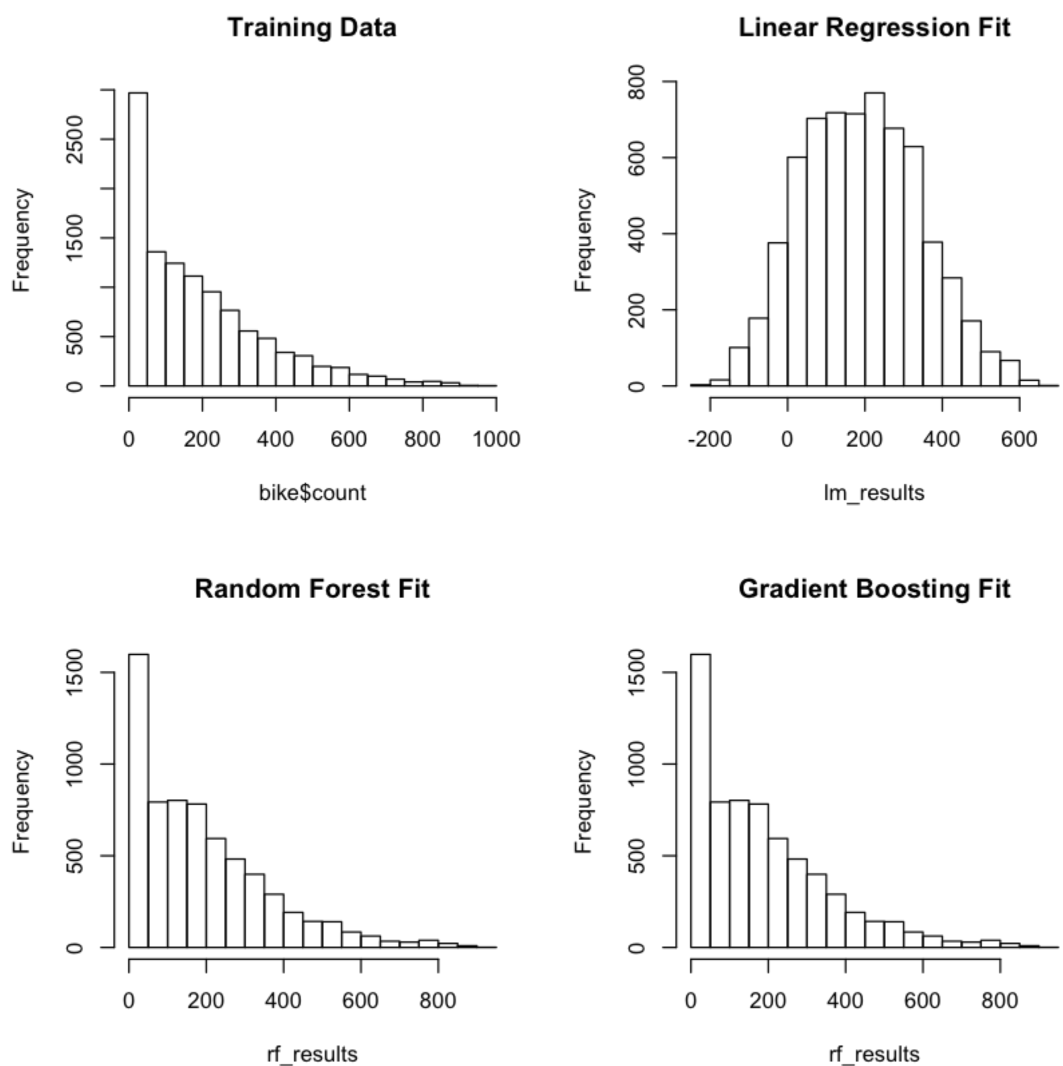
[49]:
```
par(mfrow=c(2,2))
plot(test_subset$count, main = "Linear Model", ylab = "Test Set Rental Count",␣
 ↪pch = 20)
points(predict(lm_fit, newdata = test), col = "red", pch = 20)

plot(test_subset$count, main = "Random Forest", ylab = "Test Set Rental Count",␣
 ↪pch = 20)
points(PredTestCountFinal, col = "red", pch = 20)

plot(test_subset$count, main = "Gradient Boosting", ylab = "Test Set Rental␣
 ↪Count", pch = 20)
points(gbm.PredTestCountFinal, col = "red", pch = 20)
```

```
Warning message in predict.lm(lm_fit, newdata = test):
```

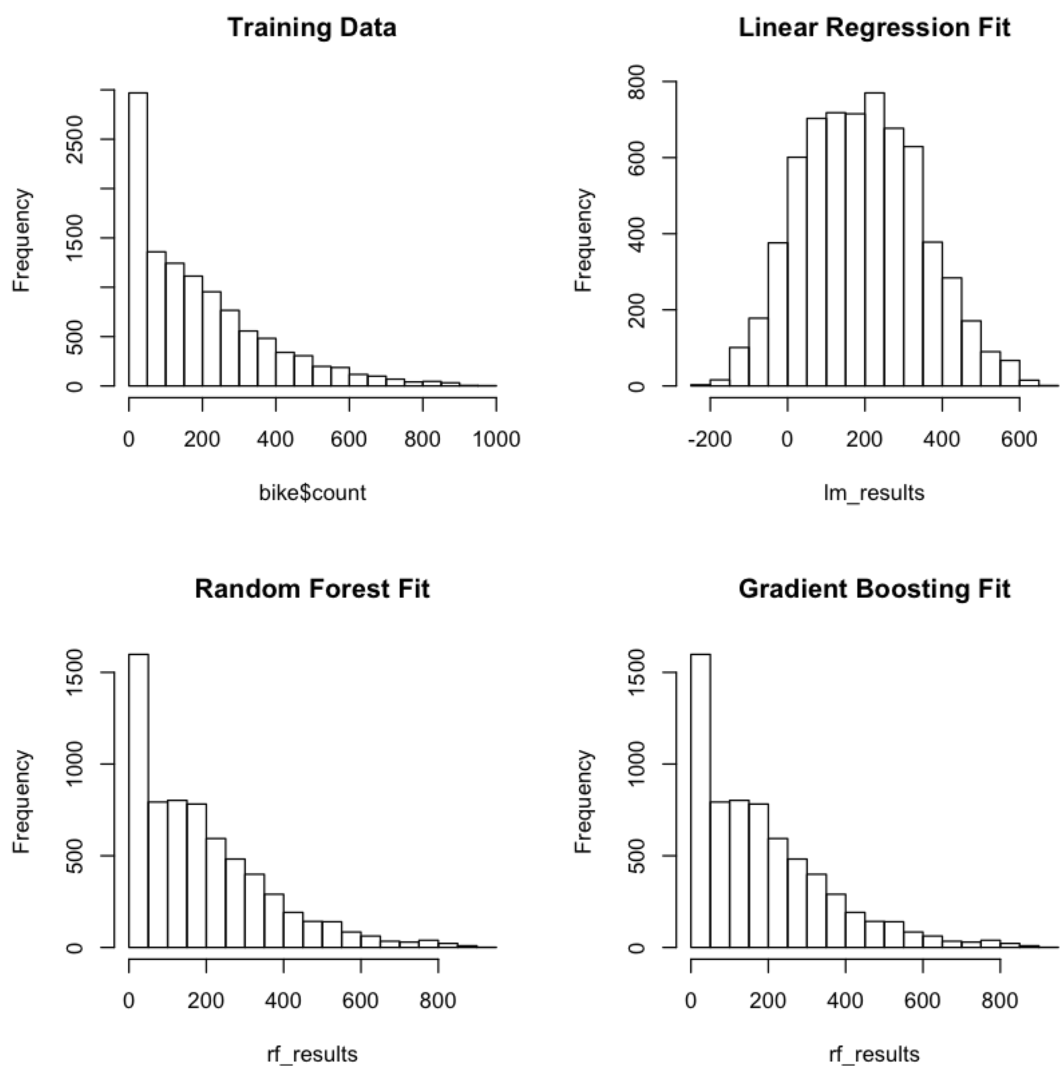


[50]:
```
# Save the RF results
        gbm_test_casual = round(predict(gbm.CasualFinal, bike_test, n.
↪trees=gbmtree),0)
        gbm_test_registered = round(predict(gbm.RegisteredFinal, bike_test, n.
↪trees=gbmtree),0)
        gbm_results = gbm_test_casual + gbm_test_registered
```

[51]:
```
par(mfrow=c(2,2))
hist(bike$count, main="Training Data")
hist(lm_results, main="Linear Regression Fit")
hist(rf_results, main="Random Forest Fit")
hist(rf_results, main="Gradient Boosting Fit")
```



[52]:
```
par(mfrow=c(2,2))
hist(bike$count, main="Training Data")
hist(lm_results, main="Linear Regression Fit")
hist(rf_results, main="Random Forest Fit")
hist(rf_results, main="Gradient Boosting Fit")
```


Training Data
Frequency
0
500
1500
2500
0
200
400
600
800
1000
bike$count
Linear Regression Fit
Frequency
0
200
400
600
800
-200
0
200
400
600
lm_results
Random Forest Fit
Frequency
0
500
1000
1500
0
200
400
600
800
rf_results
Gradient Boosting Fit
Frequency
0
500
1000
1500
0
200
400
600
800
rf_results

```
[ ]:
```