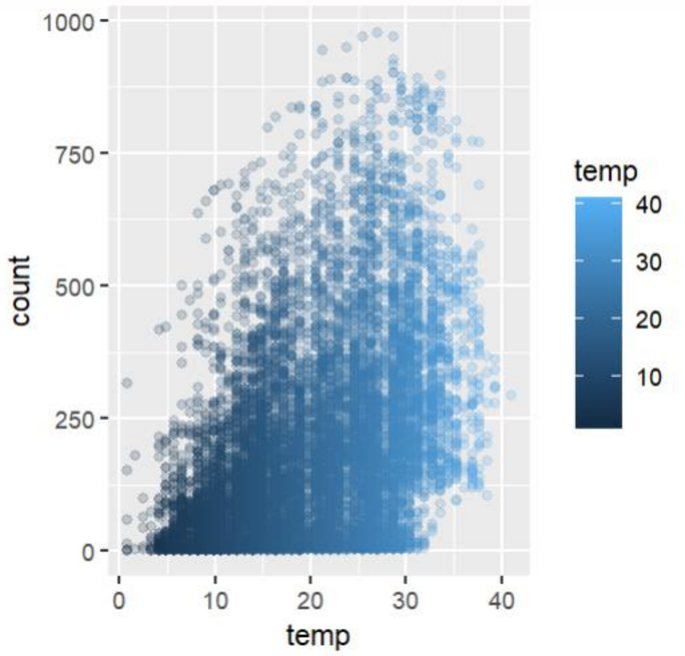
1. Read in bikeshare.csv file and set it to a dataframe called bike, and Check the head of df.

```
bike <- read.csv('C:/Users/shrey/Documents/bikeshare.csv')
bike
head(bike)
```

2. Create a scatter plot of count vs temp. Set a good alpha value.

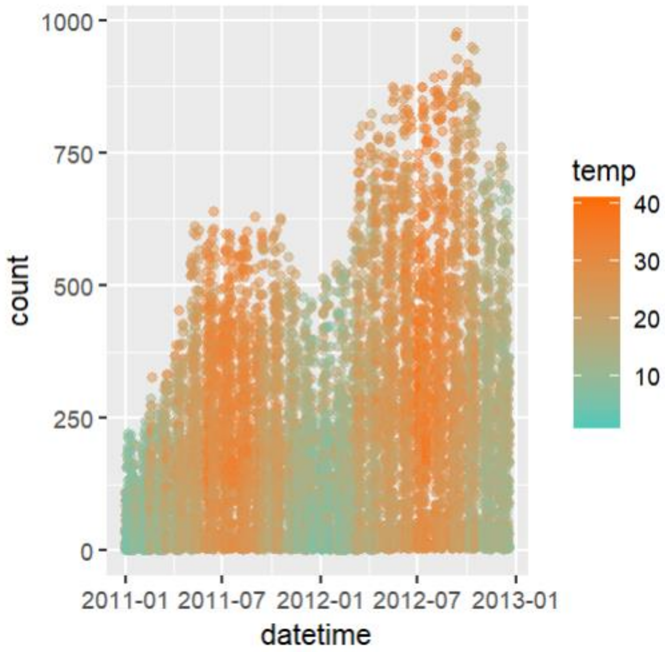
```
install.packages('plotly')
install.packages('ggplot2')
library(ggplot2)
library(plotly)
p1 <- ggplot(data = bike,aes(y=count,x=temp)) +
geom_point(aes(color=temp),alpha=0.2)
p1
```



3. Plot count versus datetime as a scatterplot with a color gradient based on temperature. You'll need to convert the datetime column into POSIXct before plotting.

```
library(ggplot2)
library(plotly)
bike$datetime <- as.POSIXct(as.character(bike$datetime),format = "%Y-%m-%d %H:%M:%S")
```

```
pl <- ggplot(bike,aes(y=count,x=datetime)) +
geom_point(aes(color=temp),alpha=0.5) + scale_color_gradient(low = '#4ACABB'
,high = '#ff6b00')
print(pl)
```

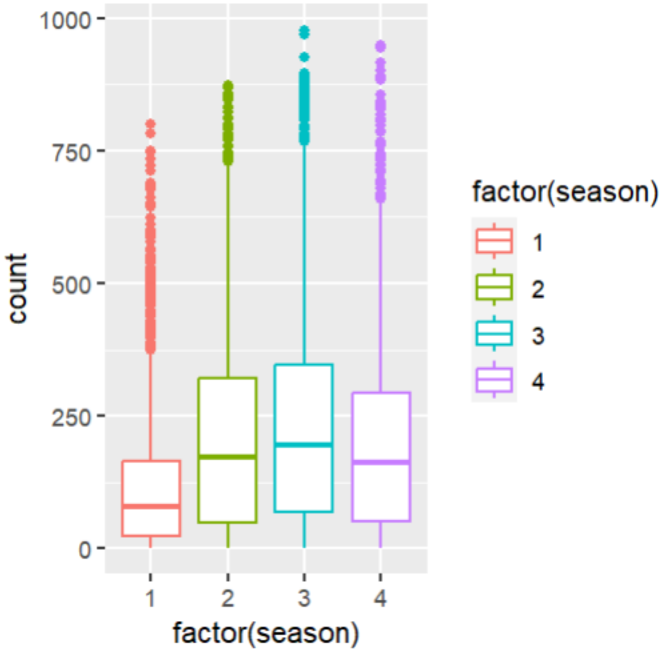


4. What is the correlation between temp and count?

```
install.packages('corrgram')
install.packages('corrplot')
library(corrgram)
library(corrplot)
cor.data <- cor(bike[,c('temp','count')])
print(cor.data)
```

5. Let's explore the season data. Create a boxplot, with the y axis indicating count and the x axis begin a box for each season.

```
library(ggplot2)
library(plotly)
pl <- ggplot(bike,aes(y = count,x = factor(season))) + geom_boxplot(aes(color =
factor(season)))
pl
```



6. Create an "hour" column that takes the hour from the datetime column. You'll probably need to apply some function to the entire datetime column and reassign it.

   Hint:
   time.stamp <- bike$datetime[4] format(time.stamp, "%H")

   ```
   bike$hour <- format(bike$datetime, '%H')
   print(head(bike))
   ```

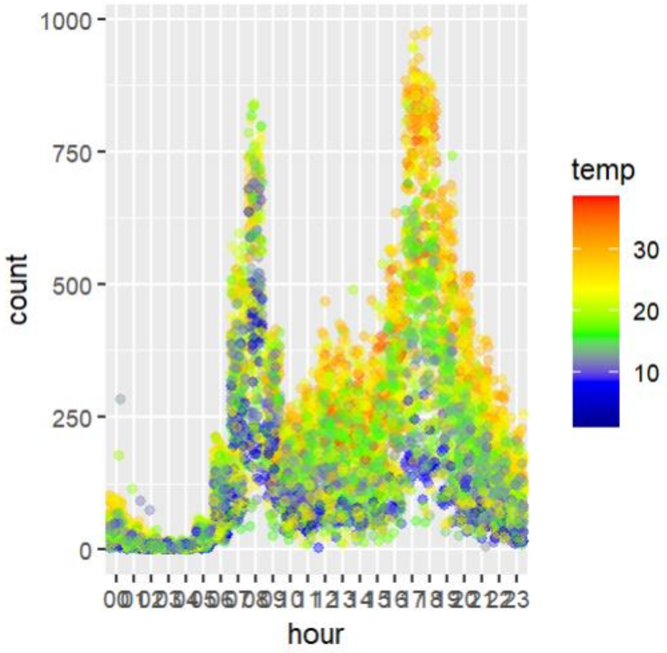
7. Now create a scatterplot of count versus hour, with color scale based on temp. Only use bike data where workingday==1.

   Optional Additions:

   Use the additional layer:
   scale_color_gradientn(colors=c('color1',color2,etc..)) where the colors argument is a vector gradient of colors you choose, not just high and low.
   Use position=position_jitter(w=1, h=0) inside of geom_point() and check out what it does.

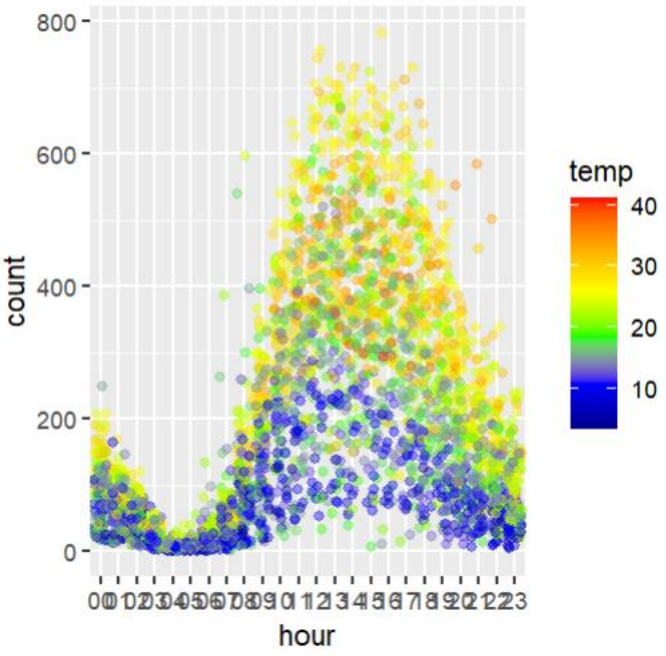
   ```
           library(ggplot2)
   library(plotly)
   library(dplyr)
   ```

```
pl <- ggplot(filter(bike,workingday == 1),aes(y = count,x = hour)) +
geom_jitter(aes(color = temp),alpha = 0.4) + scale_color_gradientn(colours = c('dark
blue','blue','green','yellow','orange','red'),guide = 'colourbar')
pl
```



8. Now create the same plot for non working days:

```
library(ggplot2)
library(plotly)
library(dplyr)
pl <- ggplot(filter(bike,workingday == 0),aes(y = count,x = hour)) +
geom_jitter(aes(color = temp),alpha = 0.4) + scale_color_gradientn(colours = c('dark
blue','blue','green','yellow','orange','red'),guide = 'colourbar')
pl
```



9. Use lm() to build a model that predicts count based solely on the temp feature, name it temp.model

```
temp.model <- lm(count ~ temp,data = bike)
```

10. Get the summary of the temp.mode.

```
print(summary(temp.model))
```

11. How many bike rentals would we predict if the temperature was 25 degrees Celsius? Calculate this two ways:
    Using the values we just got above
    Using the predict() function
    You should get around 235.3 bikes.
    Output: 1: 235.309724995272

```
coef <- as.data.frame(coefficients(temp.model))
check_temp <- 25
count1 <- coef[1,1] + coef[2,1]*check_temp
print(paste('Method 1: Count of Bikes at temp 25 Degree Celcius =',round(count1,4)))

check.tempdf <- data.frame(temp = 25)
count2 <- predict(temp.model,check.tempdf)
print(paste('Method 2: Count of Bikes at temp 25 Degree Celcius =',round(count2,4)))
```

12. Use sapply() and as.numeric to change the hour column to a column of numeric values.

```
bike$hour <- sapply(bike$hour,as.numeric)

print(str(bike))
```

13. Finally build a model that attempts to predict count based off of the following features. Figure out if theres a way to not have to pass/write all these variables into the lm() function. Hint: StackOverflow or Google may be quicker than the documentation.

    - season
    - holiday
    - workingday
    - weather
    - temp
    - humidity

- windspeed
- hour (factor)

```
final.model <- lm(formula = count ~ . - casual - registered - datetime - atemp, data =
bike)
```

14. Get the summary of the model.

```
print(summary(final.model))
```