CWRU DSCI351-351m-451: Lab Exercise LE2

Timeseries Analysis, ggplot2, Text Mining

R. H. French, L. S. Bruckman, P. Leu, K. Davis, S. Cirlos

21 September, 2022

Contents

2.0.1 LE2, 7 points, 3 questions.

Summary of points (use Cntrl + Shift + O for seeing sub-questions easily):-

LE2-1: 2.5 points

- LE2-1a: 1 point
- LE2-1b: 1.5 points

LE2-2: 2 points

- LE2-2a, LE2-2b: 0.5 points each

- LE2-2c, LE2-2d: 0.5 points each

LE2-3: 2.5 points

- LE2-3a, LE2-3b: 0.5 point each
- LE2-3c: 1 point
- LE2-3d: 0.5 point

2.0.1.1 Lab Exercise (LE) 2

2.1 LE2-1. Time series analysis: (2.5 points)

Time series are a common type of data,

- consisting of measurements that are continuous over a time range.

In this project we will be using classical decomposition

- to perform analysis on a time series.

First as an introduction to decomposition we will have a quick example.

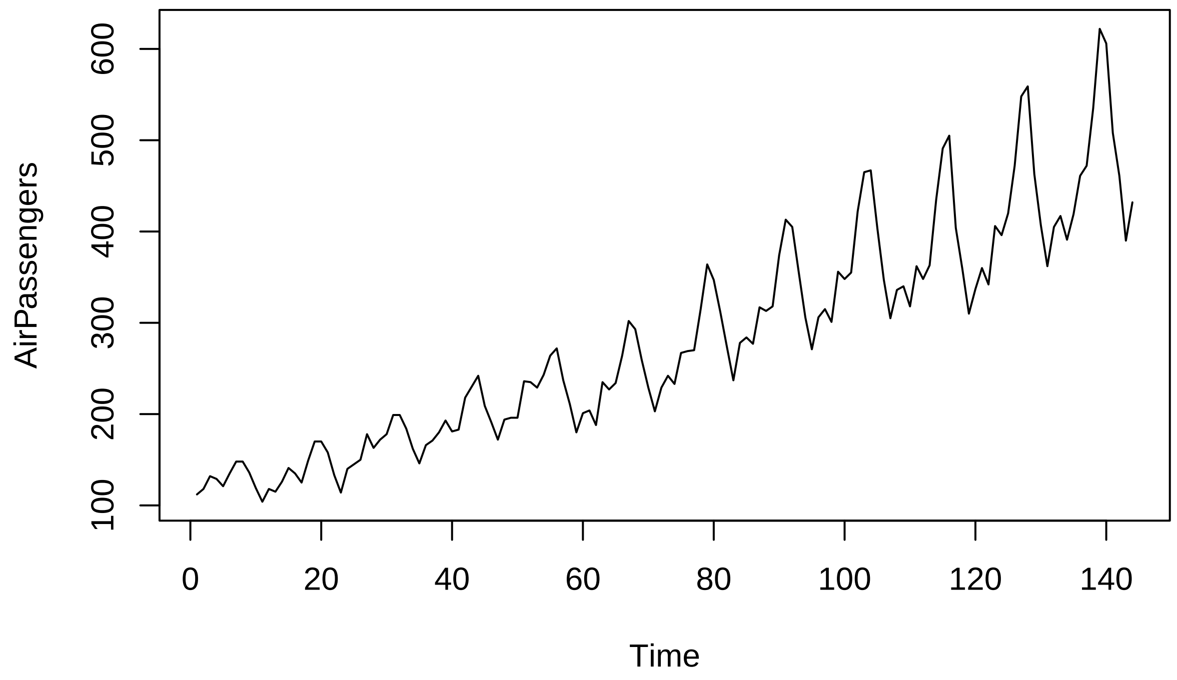
2.1.1 LE2-1a (1 point)

- What is the decomposition of a timeseries?
- The AirPassengers data set of airline passengers every month for 12 years

```
AirPassengers <- as.data.frame(AirPassengers)
```

- Plot the total time series of air passengers
- What do you notice about the plot?

```
ts.plot(AirPassengers)
```



ANSWER (what do you notice about the plot?) -> pattern stays the same but gets much larger

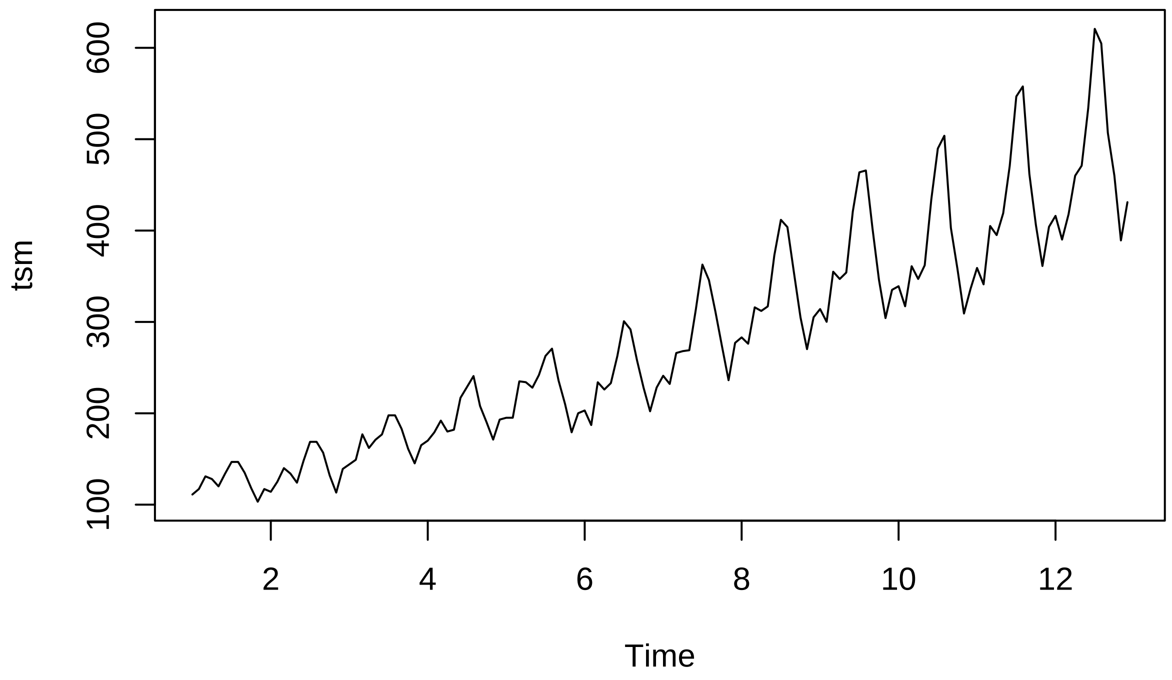
- Use the ts() function in base R
  - to define AirPassengers as a time series with a yearly trend
- If the data is taken monthly,
  - what will the frequency (points per season) of a yearly season be?

```
# Create a time series object with a yearly trend
tsy <- ts(AirPassengers, frequency = 1)
tsm <- ts(AirPassengers, frequency = 12)
```

ANSWER (what will the frequency (points per season) of a yearly season be?) -> 12 (monthly for 12 years, 144 months, 12 months each)

- Use the decompose() function -to decompose the time series and remove the seasonality
- The type for this time series is multiplicative
- Plot the decomposed time series, what do you notice about the trend?

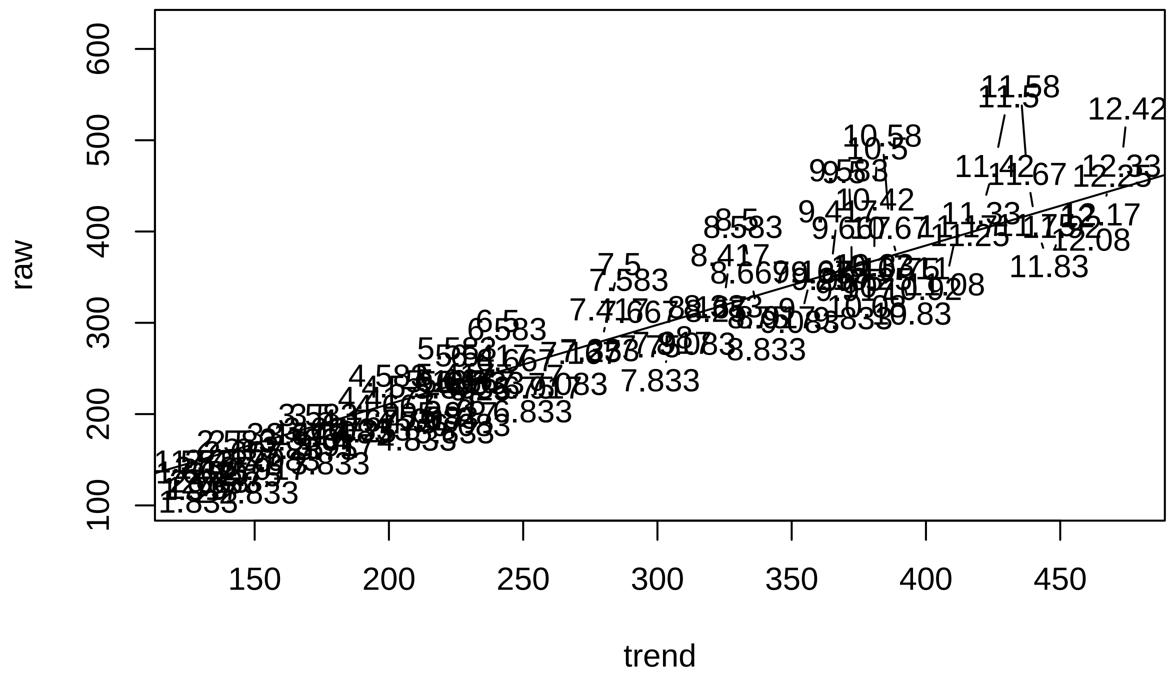
```
dcp <- decompose(tsm, type = "multiplicative")
dcs <- tsm - dcp$seasonal
plot(dcs)
```



ANSWER (what do you notice about the trend?) -> when seasonality is removed, trend stays the same, graph is the same as the standard time series.

- Isolate the trend and plot the trend on top of the raw data with the seasonality included
- How well does the trend represent the data?

```
# Extract the trend from the decomposition
trend <- dcp$trend
raw <- dcp$x
# Plot the raw data annotated with the trend
plot(raw~trend)
abline(lm(trend~raw))
```



ANSWER (how well does the trend represent the data?) -> Very well, trend is within bounds of the linear increase, in fact represents the linear increase of the data.

2.1.2 LE2-1b (1.5 points)

Now lets try this with a real world time series. We'll be using one month of power and weather data from a solar power plant.

The data set variables are as follows:

- time: The timestamp at which the data was taken
- ghir: Global Horizontal Irradiance from a sensor at the site,
  - the power from the sunlight over an area normal to the surface of the earth ($Watts/m^2$)
- iacp: The AC power from the power plant (kW)
- temp: The air temperature ($Celsius$)
- ghi_solargis: The Global Horizontal Irradiance, not from a sensor,
  - but predicted using weather modeling ($Watts/m^2$)
- clear: A logical indicating whether the sky was "clear" during measurement,
  - determined by comparing the ghi and ghi_solargis data
- ratio: the ratio of the Global Horizontal Irradiance
  - and the Plane of Array Irradiance (the irradiance normal to the surface of a tilted module)

The power from solar panels is dependent on the irradiance hitting it,

- so a power reading is often meaningless without a corresponding irradiance measurement.

It is useful to have multiple sources of irradiance measurements.

Sensors on the ground are useful because

- they strongly represent the irradiance that is hitting the module;
- however, sensors can begin to drift if not cleaned or calibrated properly.
- An unstable sensor can render an entire data set useless.

To combat this, we also have irradiance data from SolarGIS,

- a company that uses satellite images to model and predict
- the irradiance at the surface of the earth.
- Plot the irradiance and power for the first week of data,
    - how does the irradiance look compared to
        * what you would expect from the trend of sunlight?
    - How well does the power and irradiance match up?

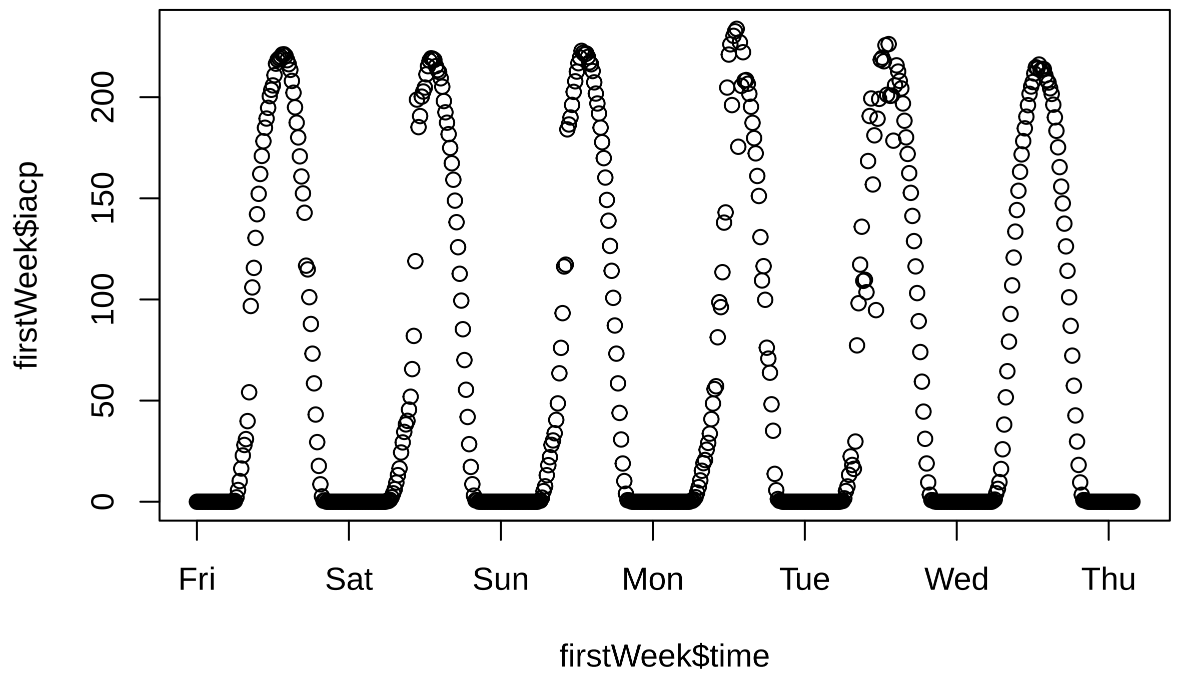
```r
# Read in the data and get a look at the structure
library(readr)
solargis <- read_csv("data/solargis.csv")
```

```
## Rows: 2880 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (5): ghir, iacp, temp, ghi_solargis, ratio
## lgl  (1): clear
## dttm (1): time
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

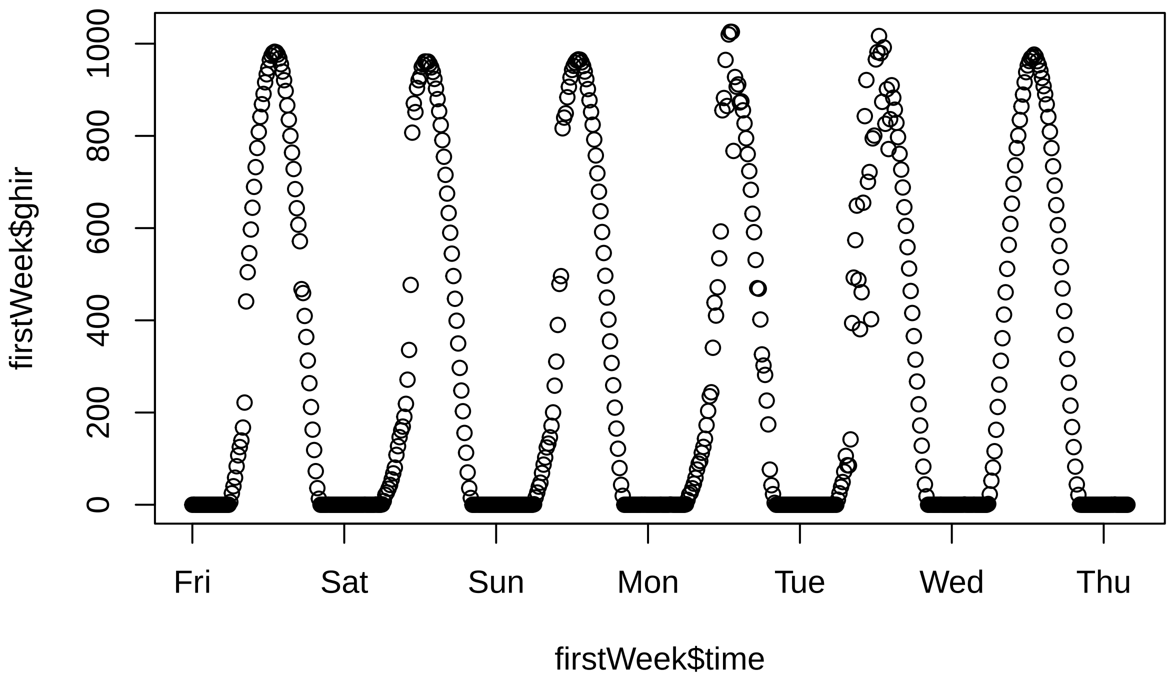
```r
# Find total length of dataset
size <- nrow(solargis)
# Find the time interval
times <- solargis$time
difftime(times[2], times[1], unit = "mins")
```

```
## Time difference of 15 mins
```

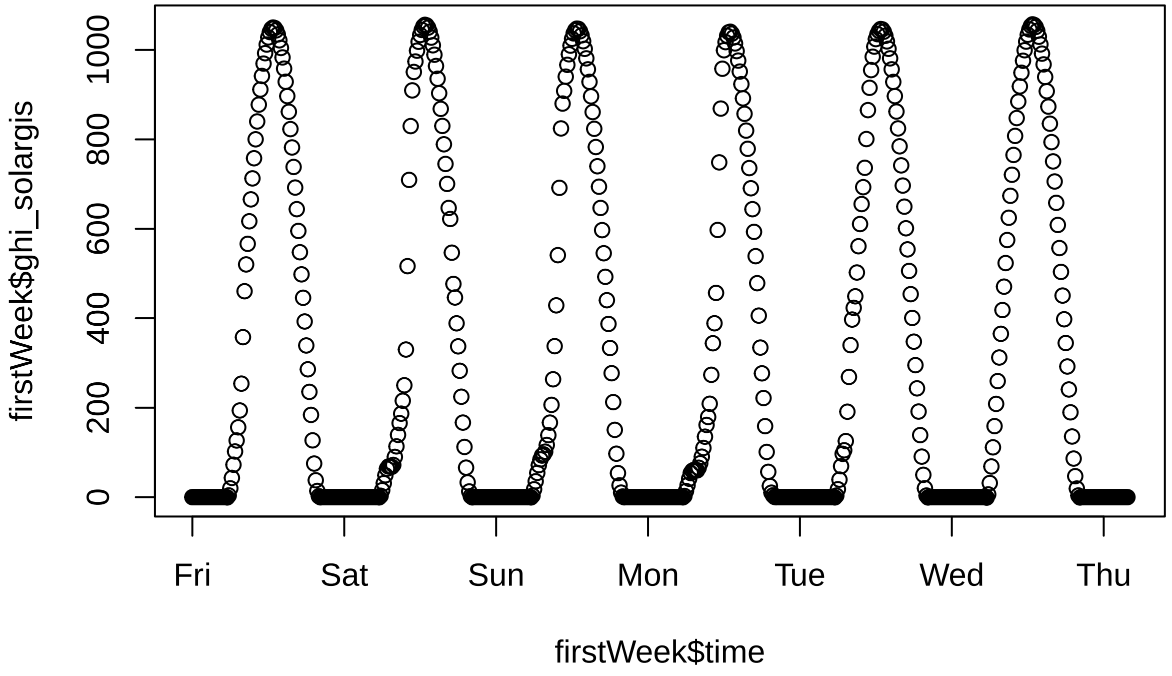
```r
# Plot the irradiance and power for the first week
# Entire dataset is in 15-minute increments
firstWeek <- solargis[solargis$time < "2012-06-07", ]
# Power plot over time
plot(firstWeek$iacp~firstWeek$time)
```



```
# Both sources of irradiance
plot(firstWeek$ghir~firstWeek$time)
```



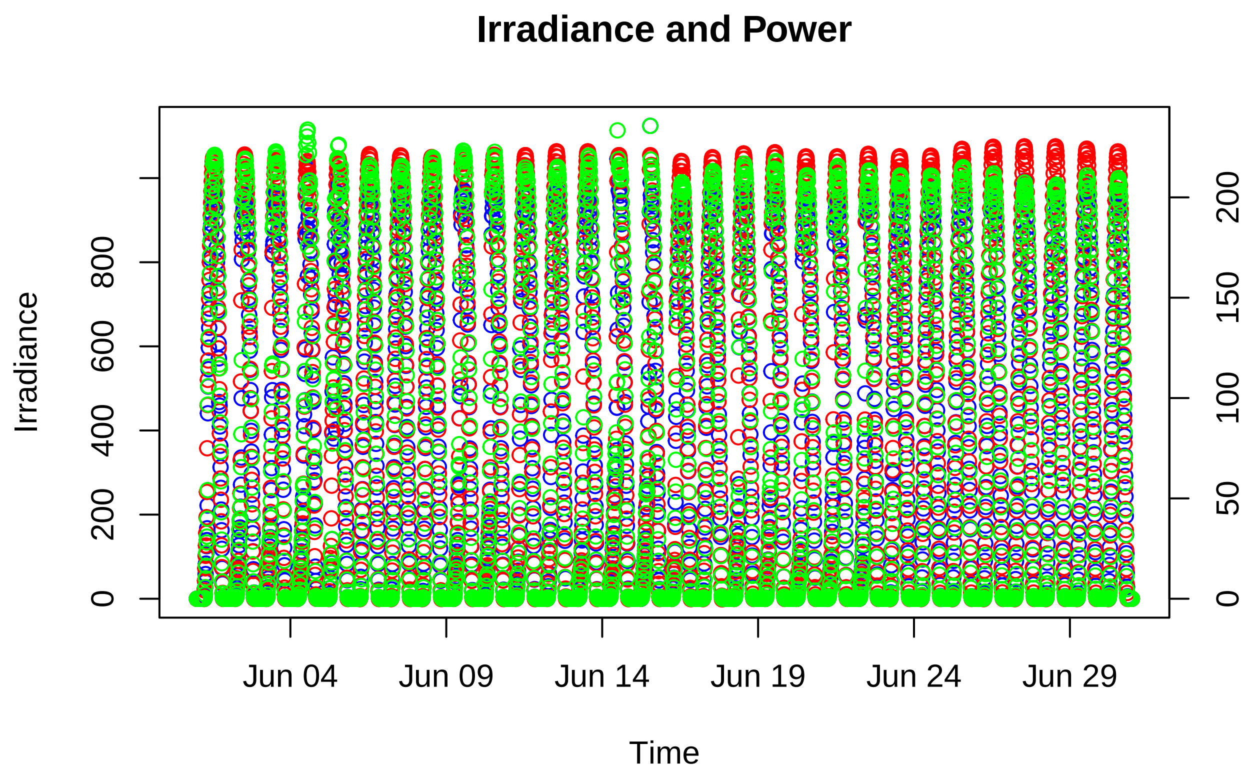
```
plot(firstWeek$ghi_solargis~firstWeek$time)
```



ANSWER (how does the irradiance look compared to what you would expect from the trend of sunlight?) -> Looks like an accurate representation of the trend of sunlight. The irradiance depends on what position the sun is in over a 24 hour period, and is usually the same unless there are weather conditions blocking the sunlight. The actual irradiance value obtained by the sensor is slightly lower than expected from the calculations obtained via weather modeling for the peak values. Trough values seem to mostly match.

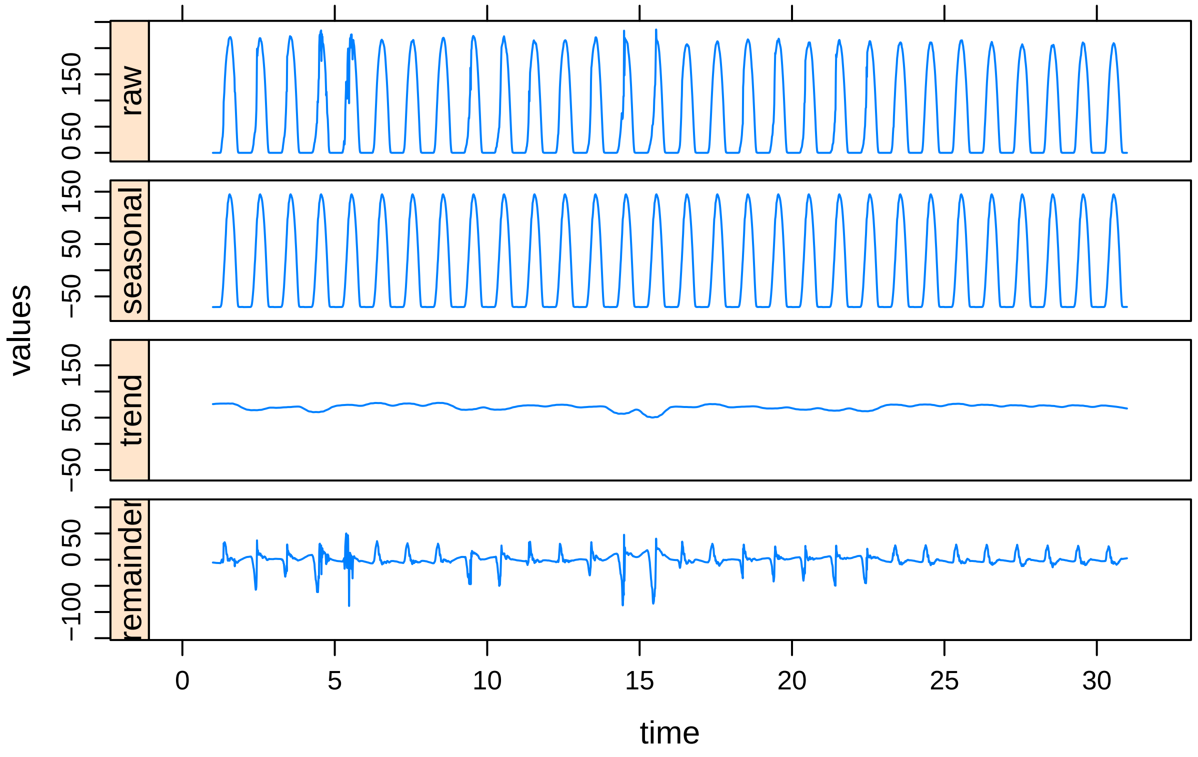
ANSWER (how well does the power and irradiance match up?) -> the trend pattern of power mostly certainly correlates to the trend pattern of irradiance. Broadly speaking, the power generated from the plant is around 1/5 of the irradiance recieved, and nullifies during the period when the sun isn't out.

```
# All 3 lines on the same plot with a secondary axis
plot(solargis$ghir~solargis$time, xlab = "Time", ylab = "Irradiance", col = "blue", main = "Irradiance &
points(solargis$ghi_solargis~solargis$time, col = "red")
par(new = TRUE)
plot(solargis$iacp~solargis$time, col = "green", axes = FALSE, xlab = "", ylab = "")
axis(side = 4, at = pretty(range(solargis$iacp)))
mtext("Power", side = 4, line = 3)
```

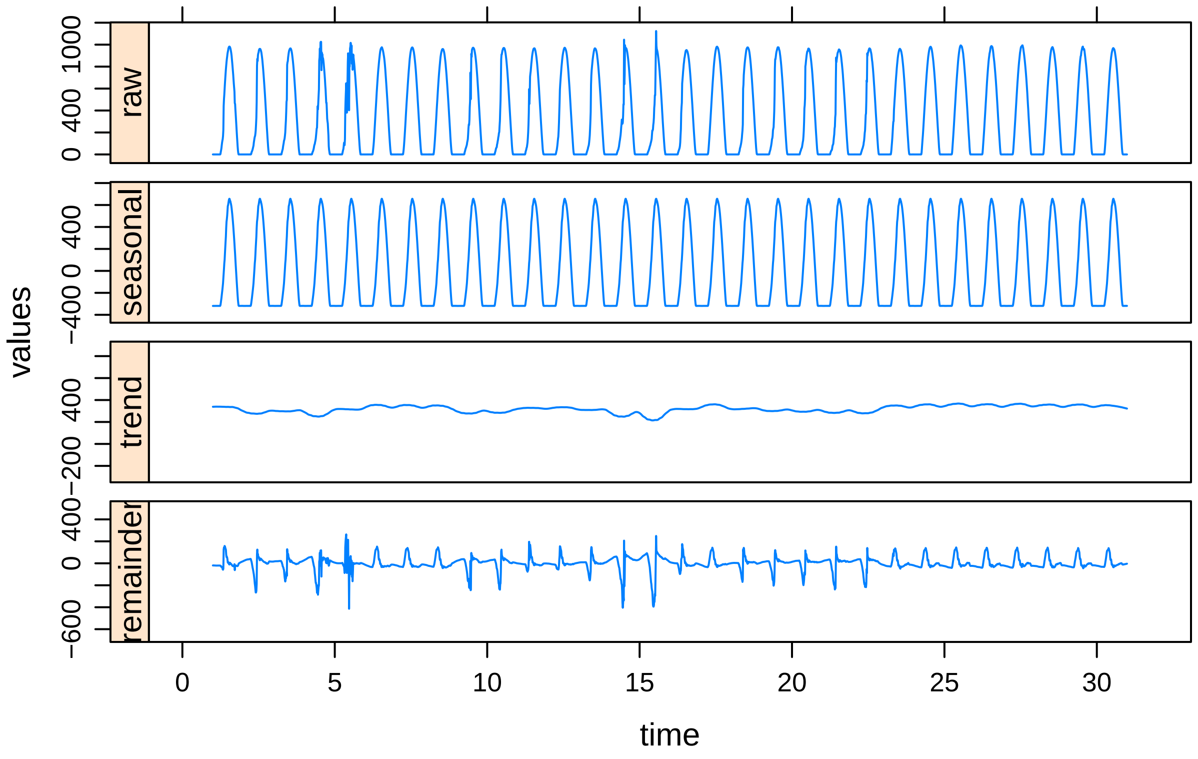


- Use the ts() functions and the stlplus() function from the stlplus package
  - to decompose the sensor and SolarGIS irradiance and the power
    - for the whole month.
  - Plot each of the decompositions, what do you notice?

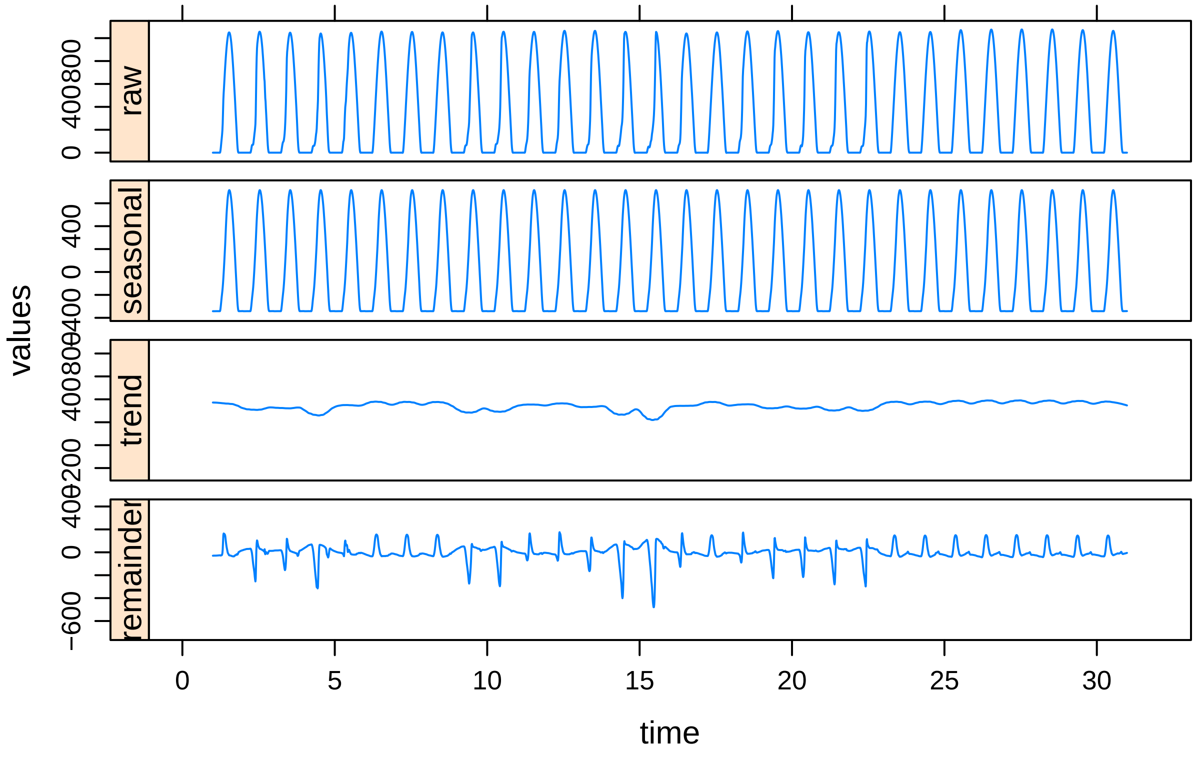
```
# think carefully about the frequency you'll need to define for this data
# what is the seasonal component to this data and how many data points make up a season? 4 (1/4 hour in
# use s.window = "periodic" for the stlplus function
library(stlplus)
power <- ts(solargis$iacp, frequency = 96)
powerstl <- stlplus(power, s.window = "periodic")
plot(powerstl)
```



```
SensorIR <- ts(solargis$ghir, frequency = 96)
CalculatedIR <- ts(solargis$ghi_solargis, frequency = 96)
SensorIRstl <- stlplus(SensorIR, s.window = "periodic")
CalculatedIRstl <- stlplus(CalculatedIR, s.window = "periodic")
plot(SensorIRstl)
```



```
plot(CalculatedIRstl)
```



ANSWER (what do you notice about the decompositions?) -> Seasonal and raw have the same patterns, trend stays mostly constant, remainder is pretty noisy and is pulsating. Seasonality stays the same in most cases.

- Isolate the trends for the 3 time series you just decomposed
  - and build a linear model for each one.
- Compare the models to each other, how are they different?

```
powerlm <- lm(powerstl$data$trend~powerstl$time)
sensorlm <- lm(SensorIRstl$data$trend~SensorIRstl$time)
calculatedlm <- lm(CalculatedIRstl$data$trend~CalculatedIRstl$time)
powerlm
```

```
##
## Call:
## lm(formula = powerstl$data$trend ~ powerstl$time)
##
## Coefficients:
##   (Intercept)  powerstl$time
##      69.70711        0.03276
```

```
sensorlm
```

```
##
```

```
## Call:
## lm(formula = SensorIRstl$data$trend ~ SensorIRstl$time)
##
## Coefficients:
##      (Intercept)  SensorIRstl$time
##          295.882             1.434
```

```
calculatedlm
```

```
##
## Call:
## lm(formula = CalculatedIRstl$data$trend ~ CalculatedIRstl$time)
##
## Coefficients:
##          (Intercept)  CalculatedIRstl$time
##               317.52                  1.52
```

```
summary(powerlm$model$`powerstl$data$trend`)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   50.29   67.51   71.28   70.23   73.65   78.23
```

```
summary(sensorlm$model$`SensorIRstl$data$trend`)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   214.5   298.0   321.2   318.8   345.4   368.0
```

```
summary(calculatedlm$model$`CalculatedIRstl$data$trend`)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   219.4   321.4   349.4   341.8   370.2   390.8
```

ANSWER (compare the models to each other and how are they different?) -> mean < median, although by a minor factor. Data can be interpreted to be skewed to the right.

- Solar panel degradation leads to less power output over time
  - at the same irradiance conditions.
- Based on the linear models you found for the trends of power and irradiance,
  - is this system degrading over time?
- How do the sensor GHI and the SolarGIS GHI compare to power?

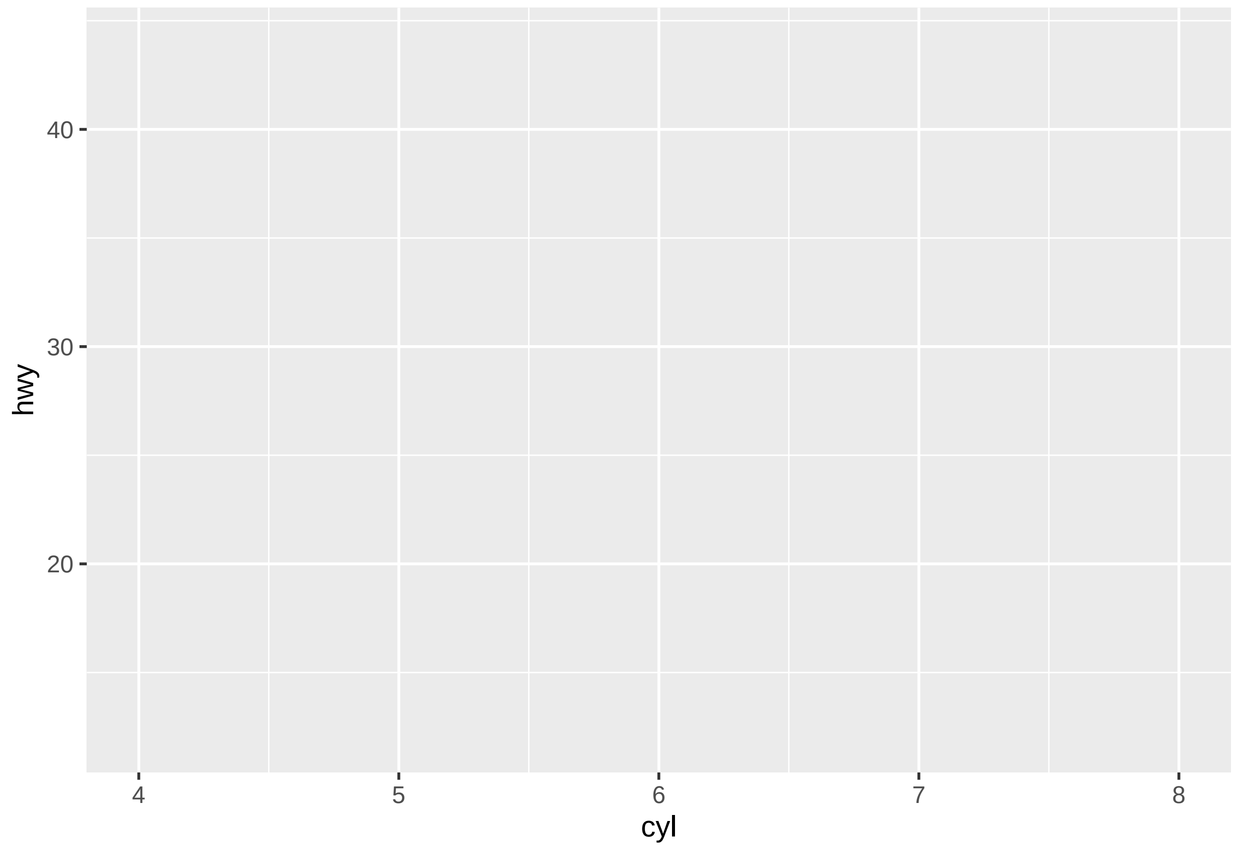
ANSWER (is this system degrading over time based on linear models?) -> Based on the linear models, there is no evidence to be found on the system degrading over time.

ANSWER (how do the sensor GHI and the SolarGIS GHI compare to power?) -> sensor/SOLARGIS GHI is approx. 5 * greater than power.

2.2 LE2-2. ggplot2: (1.5 points)

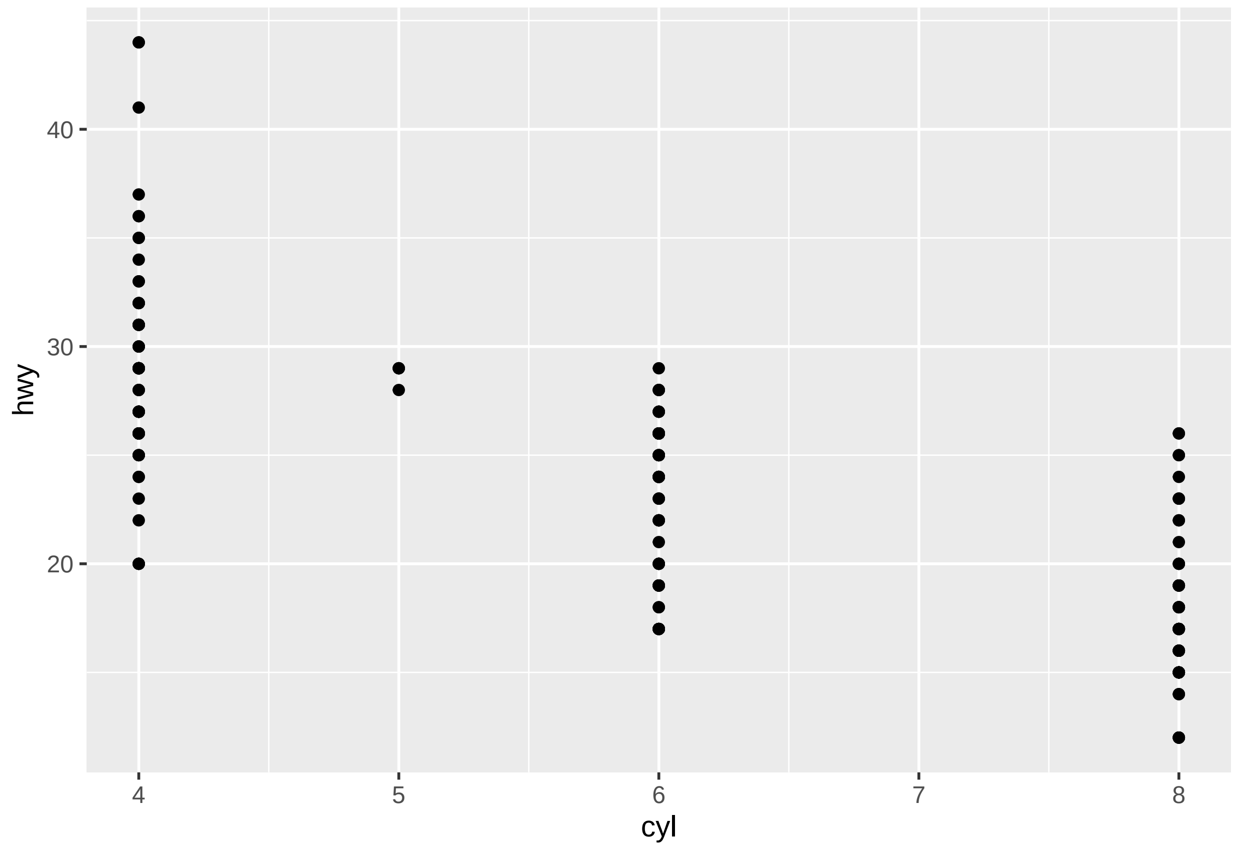
ggplot2 is a package for making plots from data. It provides tools for making complex and detailed graphs. ggplot2 builds graphs in layers, where first the graph must be specified, then layers are added to the plot using the '+' operator. In this example nothing appears in the plot

```r
library(ggplot2)
data("mpg")
ggplot(data = mpg, aes(x = cyl, y = hwy))
```

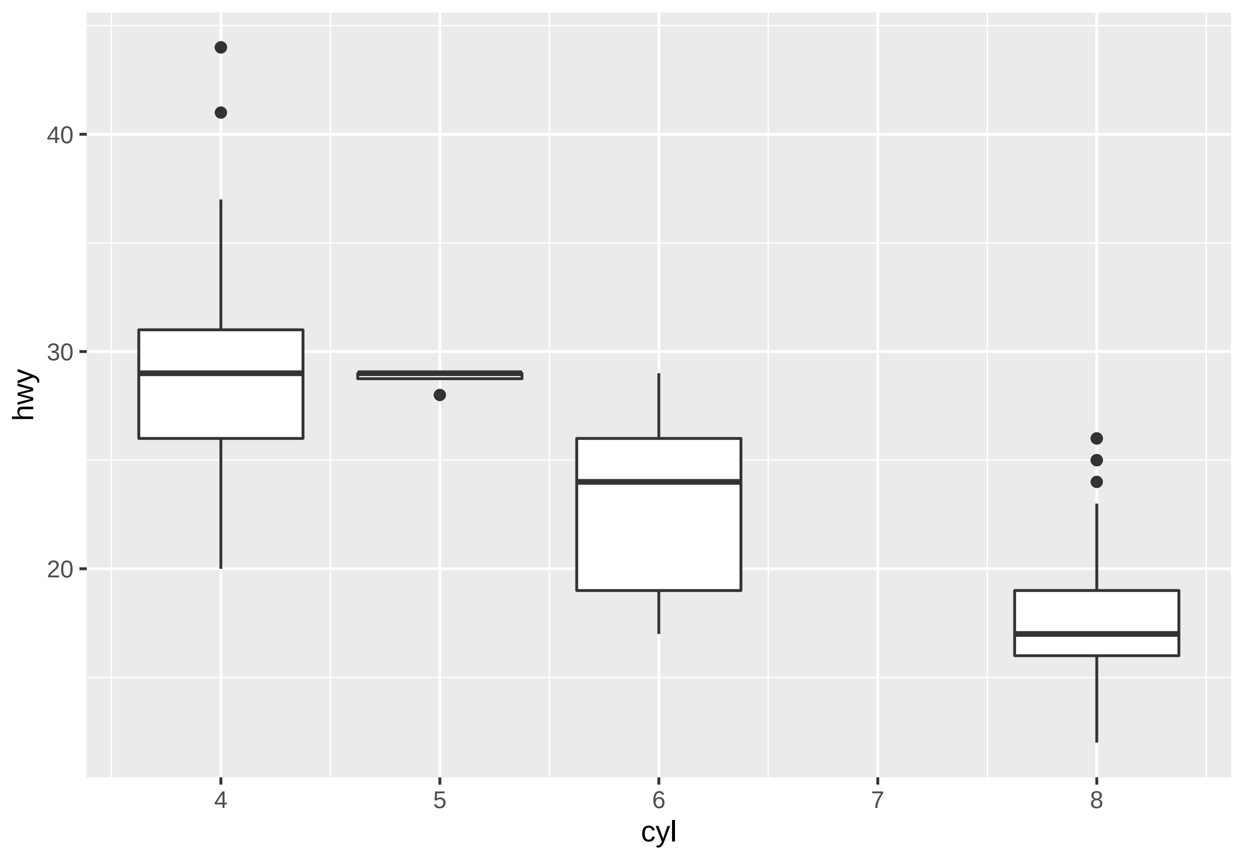


This is because we did not define the next layer, all we did was define some kind of graph between cylinders and highway mpg Since we have already defined the data at the beginning, we don't need to specify it in the layer

```r
ggplot(data = mpg, aes(x = cyl, y = hwy)) + geom_point()
```

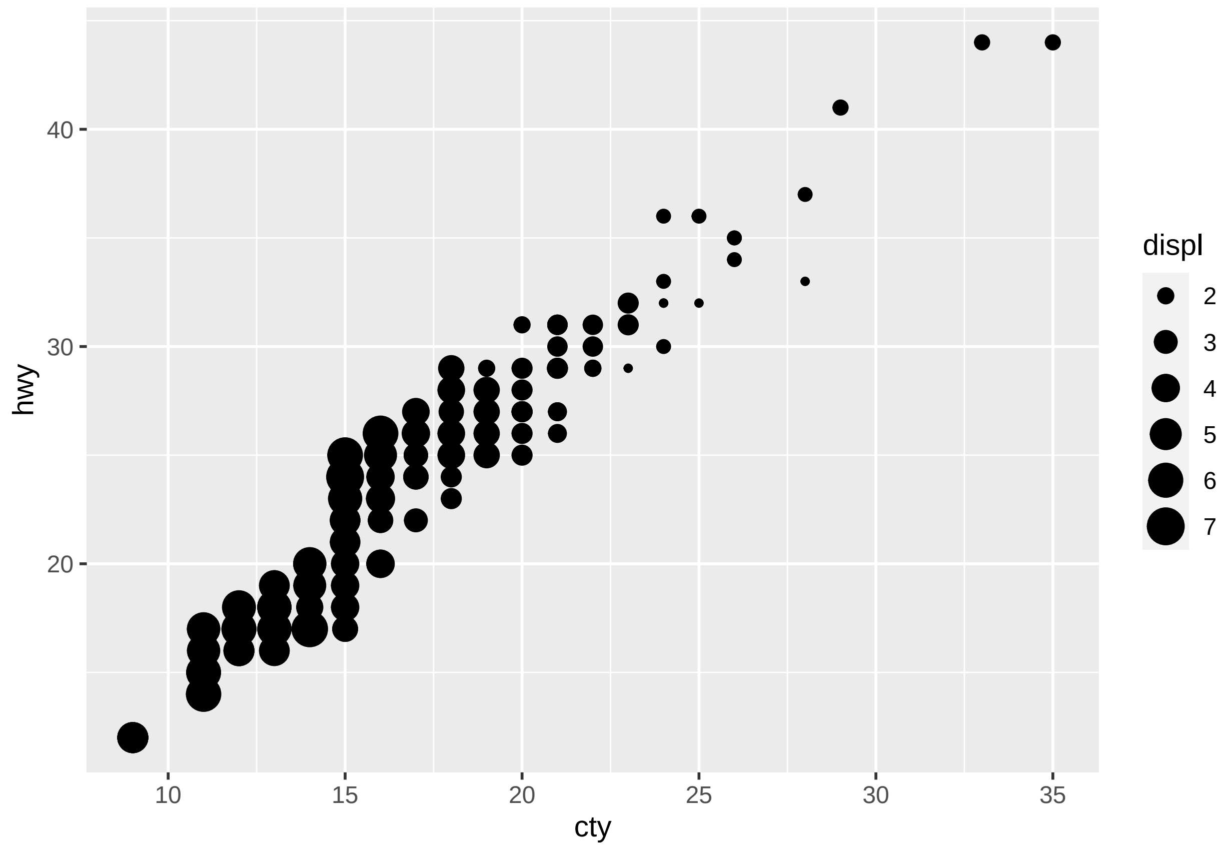


```
# or a different layer
# here we have to define cyl as the group for each box
ggplot(data = mpg, aes(x = cyl, y = hwy)) + geom_boxplot(aes(group = cyl))
```



We can add additional information about showing data in our plot by adding parameters into the aesthetics (aes()) function

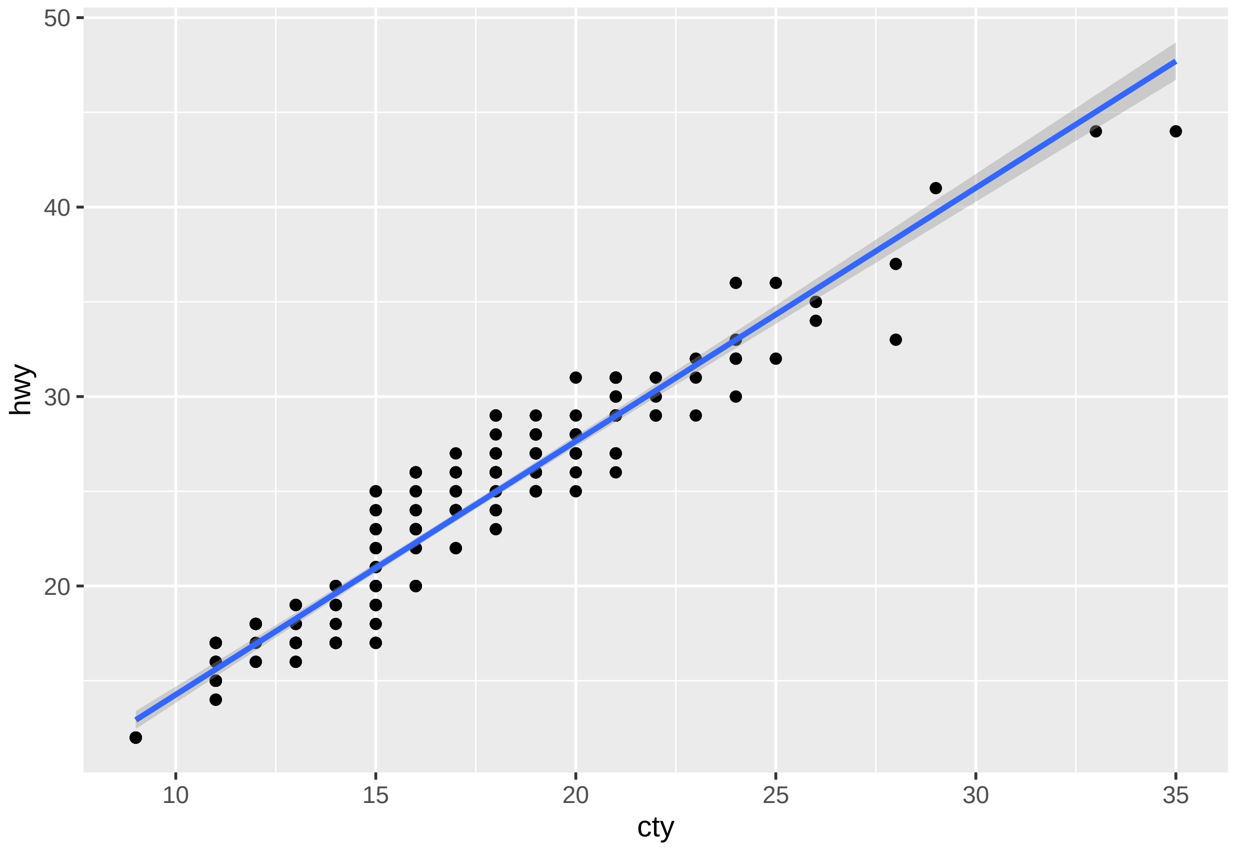
```
ggplot(data = mpg, aes(x = cty, y = hwy)) + geom_point(aes(size = displ))
```



We can also add on additional layers if we want to, keep in mind ordering is important. The data for each layer can be defined per layer, this is important if you're trying to add multiple data sets to a plot

```
ggplot() + geom_point(data = mpg, aes(x = cty, y = hwy)) + geom_smooth(data = mpg, aes(x = cty, y = hwy
```

```
## `geom_smooth()` using formula 'y ~ x'
```

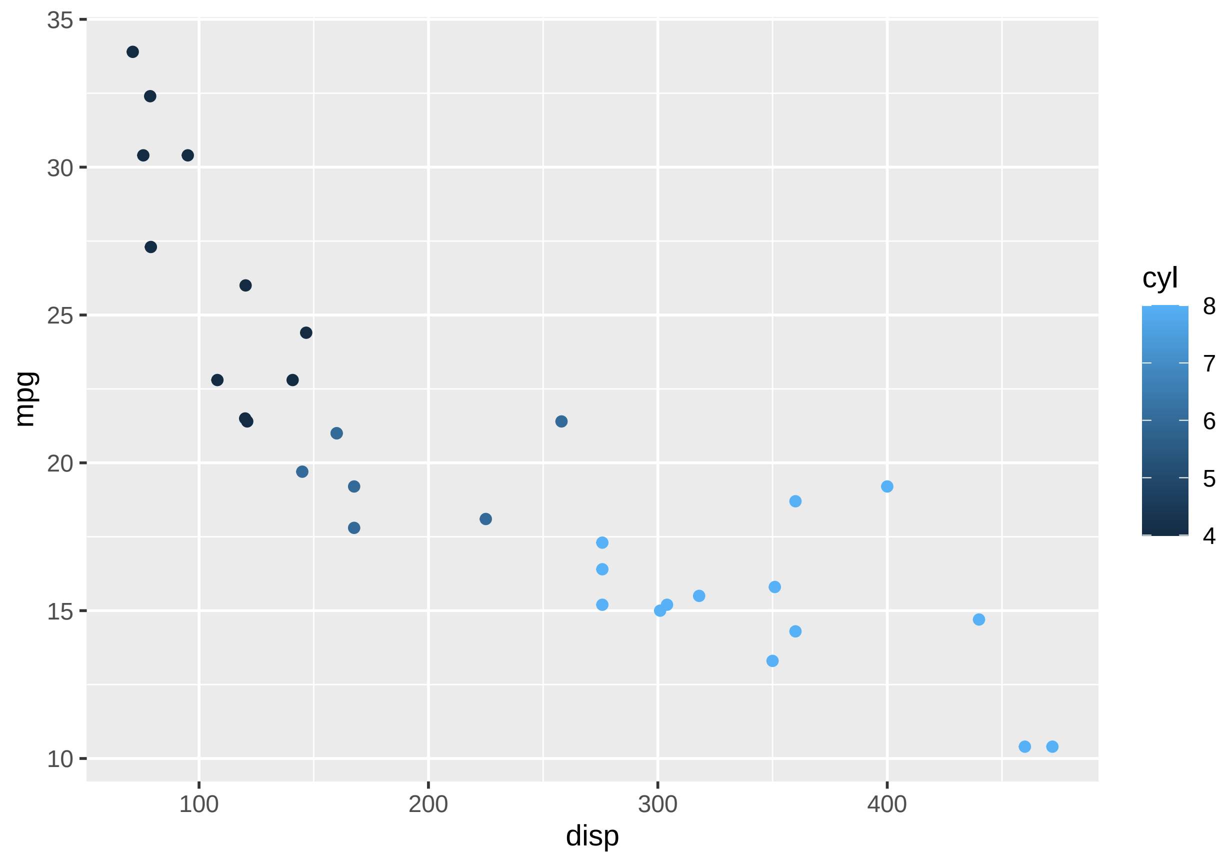


Now it's your turn to make some plots. All plotting must be done using ggplot2. Any data manipulation must be done with dplyr pipelines running into the ggplot functon

2.2.1 LE2-2a (0.25 points)

2a: Use the mtcars data set, plot mpg vs displacement and color by cylinders

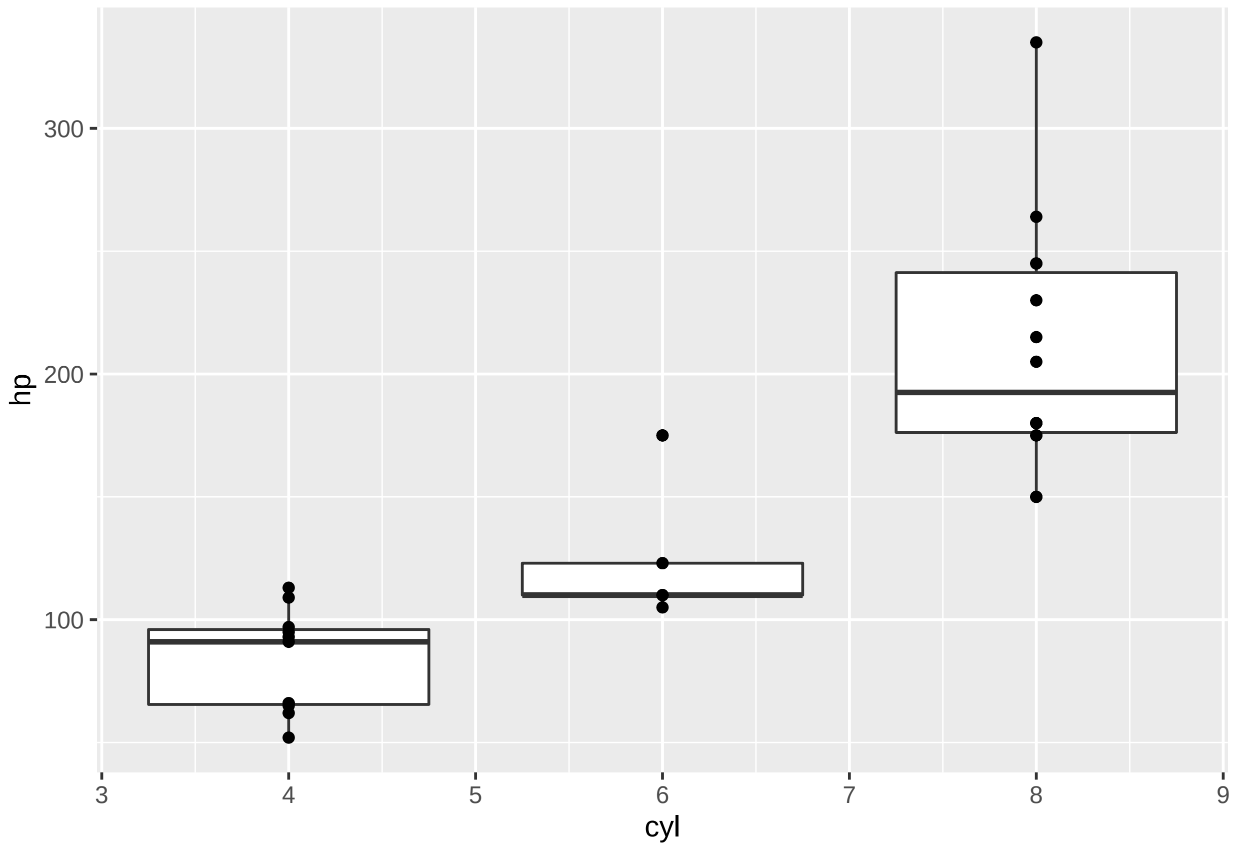
```
data("mtcars")
ggplot(data = mtcars, aes(x = disp, y = mpg, color = cyl)) + geom_point()
```



2.2.2 LE2-2b (0.25 points)

2b: Create a boxplot of the horsepower readings for each cylinder count, show the data points on top of the plot

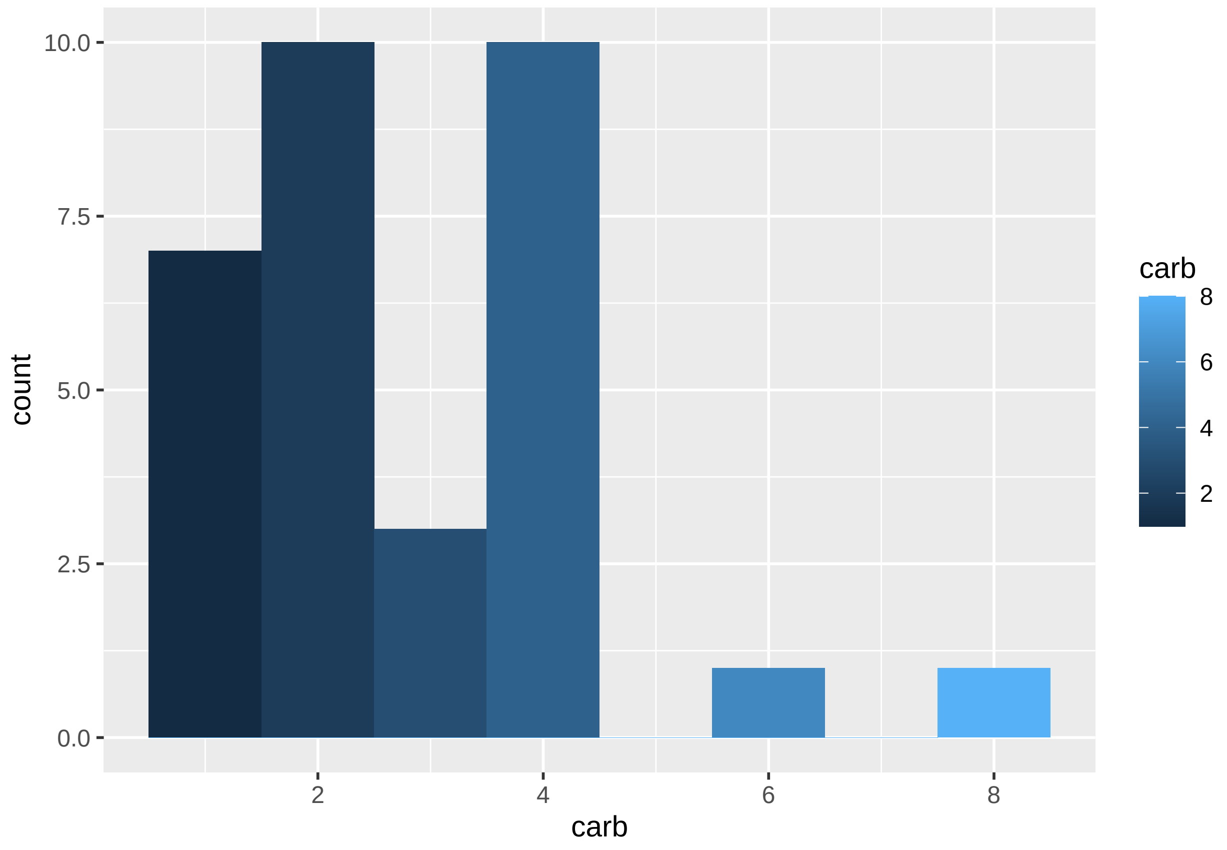
```
ggplot(data = mtcars, aes(x = cyl, y = hp, group = cyl)) + geom_boxplot() + geom_point()
```



2.2.3 LE2-2c (0.5 points)

2c: Plot a histogram of the number cars in each carburetor count group

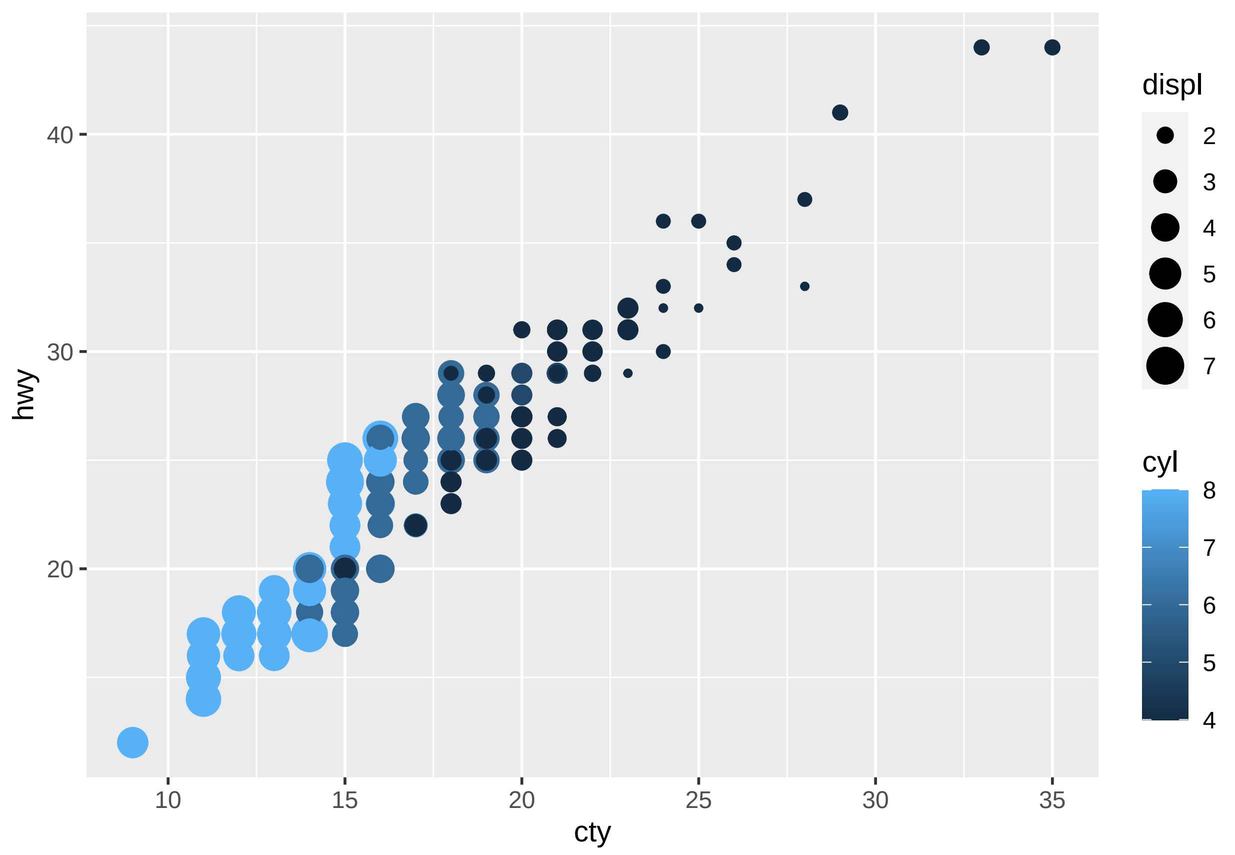
```
ggplot(data = mtcars, aes(x = carb, group = carb, fill = carb)) + geom_histogram(binwidth = 1)
```



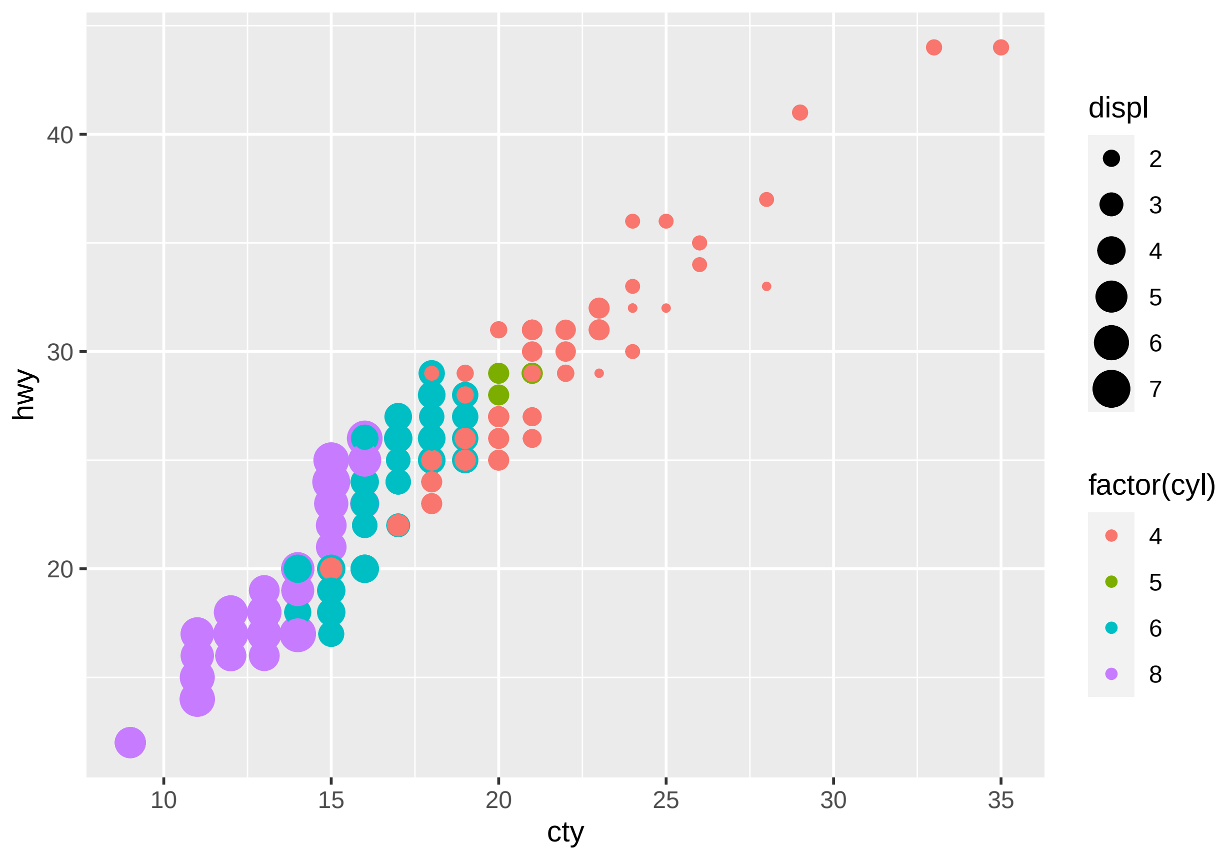
2.2.4 LE2-2d (0.5 points)

2d: Explain why these two plots look different, why does the color and key change between them?

```r
ggplot(data = mpg, aes(x = cty, y = hwy)) + geom_point(aes(color = cyl, size = displ))
```



```r
ggplot(data = mpg, aes(x = cty, y = hwy)) + geom_point(aes(color = factor(cyl), size = displ))
```



ANSWER (explain why these two plots look different) -> color coding of the cylinder. first is standard, second is the factor of the standard, otherwise they represent the same thing. ANSWER (why does the color and key change between them?) -> the color and the legend/key representing the color changes. factors represent completely different levels, otherwise break down the spectrum into separate components, which makes sense as to why the colors change.

2.3 LE2-3. Text Mining of Song Lyrics: (2.5 points)

- Complete the given problems
- dplyr, ggplot, and pipes and pipelines are highly recommended
- We will be using the Tidytext package to aid with our text mining
- The dataset for this assignment is a collection of the information and lyrics from every top 100 billboard song since 1965

2.3.1 LE2-3a (0.5 points)

- Load in the data set
- Print the lyrics of the #4 song from 1988
- Show the top 20 artists with the most hits, which artist has the most total top 100 hits?

```
lyrics <- read_csv("data/billboard_lyrics_1964-2015.csv")
```

```
## Rows: 5100 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Song, Artist, Lyrics
## dbl (3): Rank, Year, Source
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
#print lyrics
#lyrics[lyrics$Rank == 4, ]$Lyrics


artists <- as.data.frame(table(lyrics$Artist))
names(artists) <- c("artist", "freq")
artists <- artists[order(artists$freq, decreasing = T), ]
row.names(artists) <- NULL

topTwenty <- artists$artist[1:20]
topTwenty
```

```
##  [1] madonna            elton john         mariah carey
##  [4] janet jackson      michael jackson    stevie wonder
##  [7] rihanna            taylor swift       whitney houston
## [10] kelly clarkson     pink               the beatles
## [13] britney spears     the black eyed peas chicago
## [16] aretha franklin    katy perry         rod stewart
## [19] usher              boyz ii men
## 2473 Levels: 100 proof aged in soul 10000 maniacs 10cc ... zz top
```

```r
topSongs <- lyrics[1:100, ]
topArtists <- as.data.frame(table(topSongs$Artist))
names(topArtists) <- c("artist", "freq")
topArtists <- topArtists[order(topArtists$freq, decreasing = T), ]
row.names(topArtists) <- NULL
artistWMostTopHundredHits <- topArtists$artist[1]
artistWMostTopHundredHits
```

```
## [1] hermans hermits
## 76 Levels: barbara lewis barbara mason barry mcguire ... we five
```

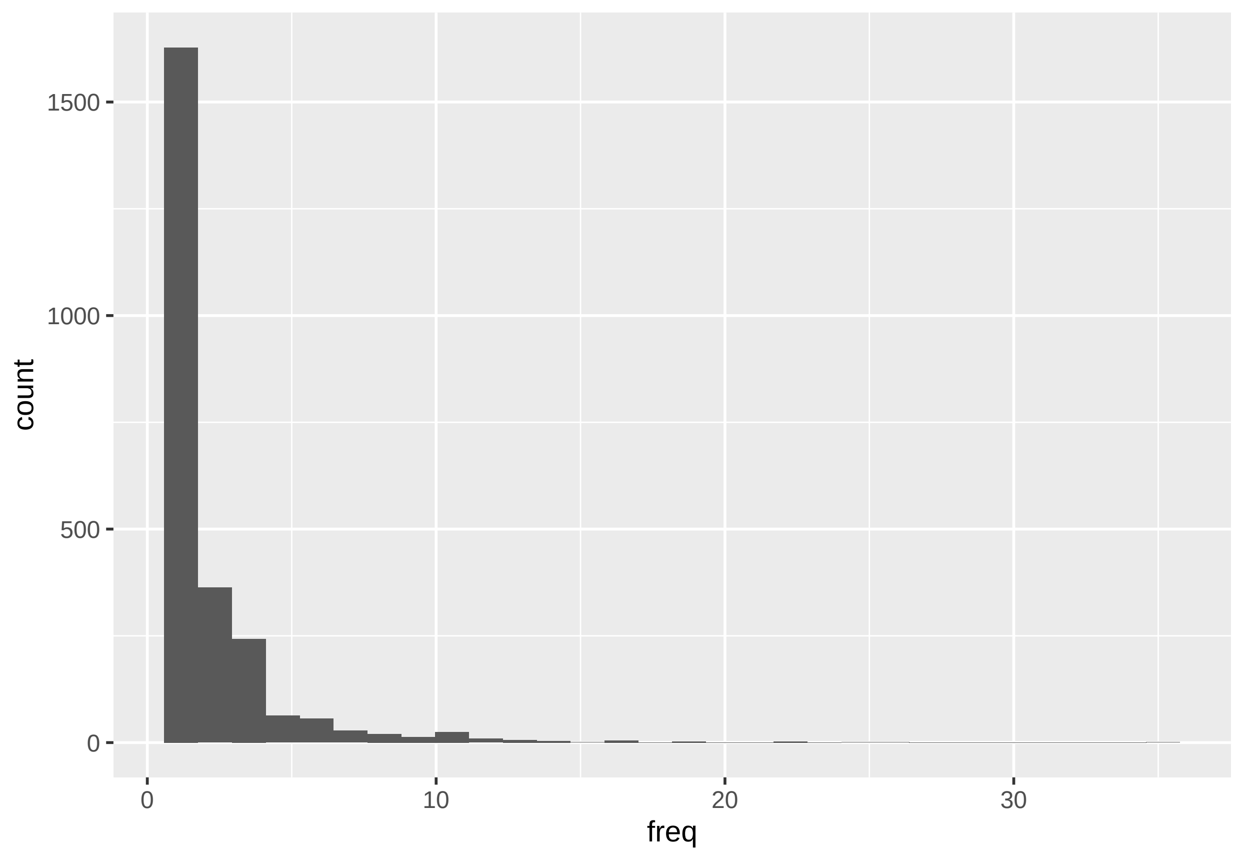
ANSWER -> The artist that has the most total top 100 hits is hermans hermits

2.3.2 LE2-3b (0.5 points)

- Build a histogram of the amount of times artists appear on the top hits billboard, what does the trend look like, what does it suggest about 1 hit wonders?

```r
ggplot(artists, aes(x = freq)) + geom_histogram()
```

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



ANSWER -> Histogram indicates that the majority of the artists in the top 100 are 1 hit wonders

2.3.3 LE2-3c (1 point)

- Lets do a lyrical comparison between 2 top artists, Elton John and Eminem
- First, filter the main data set to only the 2 given artists (you can make them seperate data frames or keep them together, your call)
- Use the unnest_tokens() function from tidytext to split the lyrics up so that each word has its own row
- Show the top 10 most commonly used words for each artist

```
library(tidytext)
eminem <- lyrics[lyrics$Artist == "eminem", ]$Lyrics
row.names(eminem) <- NULL
mwords <- tibble(text = eminem)
mwords<- mwords %>% unnest_tokens(word, text)
emwords <- as.data.frame(table(mwords))
names(emwords) <- c("word", "freq")
emwords <- emwords[order(emwords$freq, decreasing = T), ]
row.names(emwords) <- NULL

topTenEm <- emwords$word[1:10]
topTenEm
```

```
##  [1] you  the  i    to   and  a    it   me   im   just
## 1802 Levels: 12yearold 2002 4th 73 8 90 a able about absurd accents ... zone
```

```r
ej <- lyrics[lyrics$Artist == "elton john", ]$Lyrics
row.names(ej) <- NULL
jwords <- tibble(text = ej)
jwords<- jwords %>% unnest_tokens(word, text)
ejwords <- as.data.frame(table(jwords))
names(ejwords) <- c("word", "freq")
ejwords <- ejwords[order(ejwords$freq, decreasing = T), ]
row.names(ejwords) <- NULL

topTenEj <- ejwords$word[1:10]
topTenEj
```

```
##  [1] the  you  i    and  in   to   a    oh   your me
## 1150 Levels: 2016 27th 47th a about absurd acrobat after again ... zero
```

- It should be clear that most of these are not significantly meaningful words and should be removed, we can remove them using the stop_words dataframe provided with the tidytext package (hint: look at dplyr anti_join())
- Remove them and show the top ten remaining words for each artist

```r
data("stop_words")
mwords <- mwords %>% anti_join(stop_words)
```

```
## Joining, by = "word"
```

```r
emwords <- as.data.frame(table(mwords))
names(emwords) <- c("word", "freq")
emwords <- emwords[order(emwords$freq, decreasing = T), ]
row.names(emwords) <- NULL

topTenEm <- emwords$word[1:10]
topTenEm
```

```
##  [1] im    ah    dont  baby  stand shady girl  hes   sing  slim
## 1484 Levels: 12yearold 2002 4th 73 8 90 absurd accents accusates ... zone
```

```r
jwords <- jwords %>% anti_join(stop_words)
```

```
## Joining, by = "word"
```

```r
ejwords <- as.data.frame(table(jwords))
names(ejwords) <- c("word", "freq")
ejwords <- ejwords[order(ejwords$freq, decreasing = T), ]
row.names(ejwords) <- NULL

topTenEj <- ejwords$word[1:10]
topTenEj
```

```
##  [1] love     dont     life     im       time     sad      lucy     saved
##  [9] sky      diamonds
## 884 Levels: 2016 27th 47th absurd acrobat againblue againmaybe ageless ... zapped
```

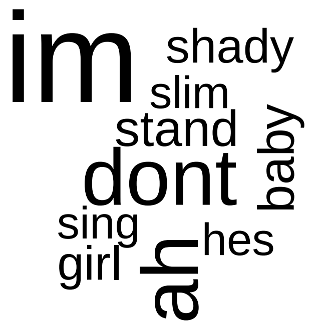
2.3.4 LE2-3d (0.5 points)

- Build a word cloud of the lyrics for each artist, find an R package that will help you with this
- Compare and contrast the word clouds
- Did the stop_words dataframe work well to remove non-meaningful words?

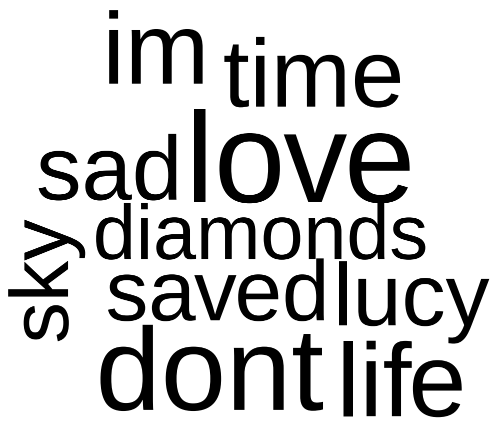
```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(RColorBrewer)
wordcloud(emwords$word[1:10], freq = emwords$freq[1:10])
```



```
wordcloud(ejwords$word[1:10], freq = ejwords$freq[1:10])
```



ANSWER (compare and contrast the word clouds) -> From the size of each word in the wordclouds, it seems that elton john uses his top words more frequently than eminem using his top words. Eminem being a rapper and Elton John being a rock artist, it makes more sense that eminem uses less words, but more words less frequently. That said, words like "slim", "shady", "im", and "girl" reflect eminem the most, while words like "love", "life", "lucy", and "sad" reflect elton the most. It's interesting how they both use "im" pretty frequently.

ANSWER (did stop_words work well to remove non-meaningful words?) -> Yes it did, it gave different outputs, and the words that were picked after the anti_join better reflect the characteristics and personality of each artist.

2.3.4.1 Links http://www.r-project.org

http://rmarkdown.rstudio.com/