CWRU DSCI351-351m-451: Lab Exercise LE2

Timeseries Analysis, ggplot2, Text Mining

Prof.:Roger French, TA: Raymond Wieser, Sameera Nalin Venkat, Mingxuan Li

17 September, 2021

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: 1.5 points

- LE2-2a, LE2-2b: 0.25 points each
- LE2-2c, LE2-2d: 0.5 points each

LE2-3: 3 points

- LE2-3a, LE2-3b: 0.5 point each
- LE2-3c: 1 point
- LE2-3d: 0.5 point
- LE2-3e: 1 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 time-series?

Answer - It is a way by which we can analyse a data over time. Time series decomposition involves thinking of a series as a combination of level, trend, seasonality, and noise components. Decomposition provides a useful abstract model for thinking about time series generally and for better understanding problems during time series analysis and forecasting.

- The AirPassengers data set of airline passengers every month for 12 years

```
## playground
# data(AirPassengers)
# is.ts(AirPassengers)
# class(AirPassengers)
# start(AirPassengers)
# end(AirPassengers)
# frequency(AirPassengers)
# boxplot(AirPassengers ~ cycle(AirPassengers))
# plot(diff(log(AirPassengers)))
# AP <- AirPassengers
# View(AP)

# data.frame(as.integer(rownames(AirPassengers)))

# playground ends
```

```
data(AirPassengers) #Loads specified data sets, or list the available data sets.
AirPassengers
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949 112 118 132 129 121 135 148 148 136 119 104 118
## 1950 115 126 141 135 125 149 170 170 158 133 114 140
## 1951 145 150 178 163 172 178 199 199 184 162 146 166
## 1952 171 180 193 181 183 218 230 242 209 191 172 194
## 1953 196 196 236 235 229 243 264 272 237 211 180 201
## 1954 204 188 235 227 234 264 302 293 259 229 203 229
## 1955 242 233 267 269 270 315 364 347 312 274 237 278
## 1956 284 277 317 313 318 374 413 405 355 306 271 306
## 1957 315 301 356 348 355 422 465 467 404 347 305 336
## 1958 340 318 362 348 363 435 491 505 404 359 310 337
```

```
## 1959 360 342 406 396 420 472 548 559 463 407 362 405
## 1960 417 391 419 461 472 535 622 606 508 461 390 432
```

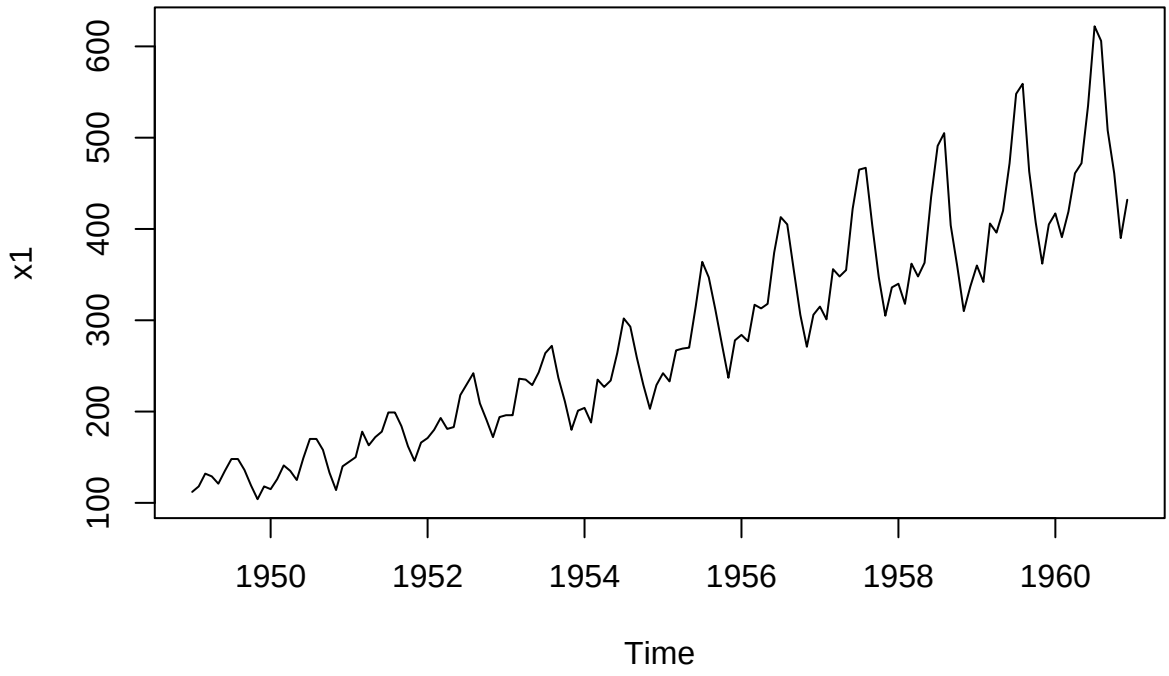
```
AirPassengers <- as.data.frame(AirPassengers)   #converting a ts object to a dataframe ;this means # #
is.data.frame(AirPassengers)
```

```
## [1] TRUE
```

```
# head(AirPassengers)
```

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

```
plot(AirPassengers)
```




```
# length(AirPassengers)
# frequency(AirPassengers)

#
# abline(reg = lm(AP ~ time(AP)))
# plot(aggregate(AP,FUN =mean))
#
# summary(AirPassengers)
```

ANSWER (what do you notice about the plot?) -> There is a periodicity in every year, and the amplitude keeps on increasing every year.

- 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
?ts

ap_y <- ts(data = AirPassengers, frequency = 12)
```

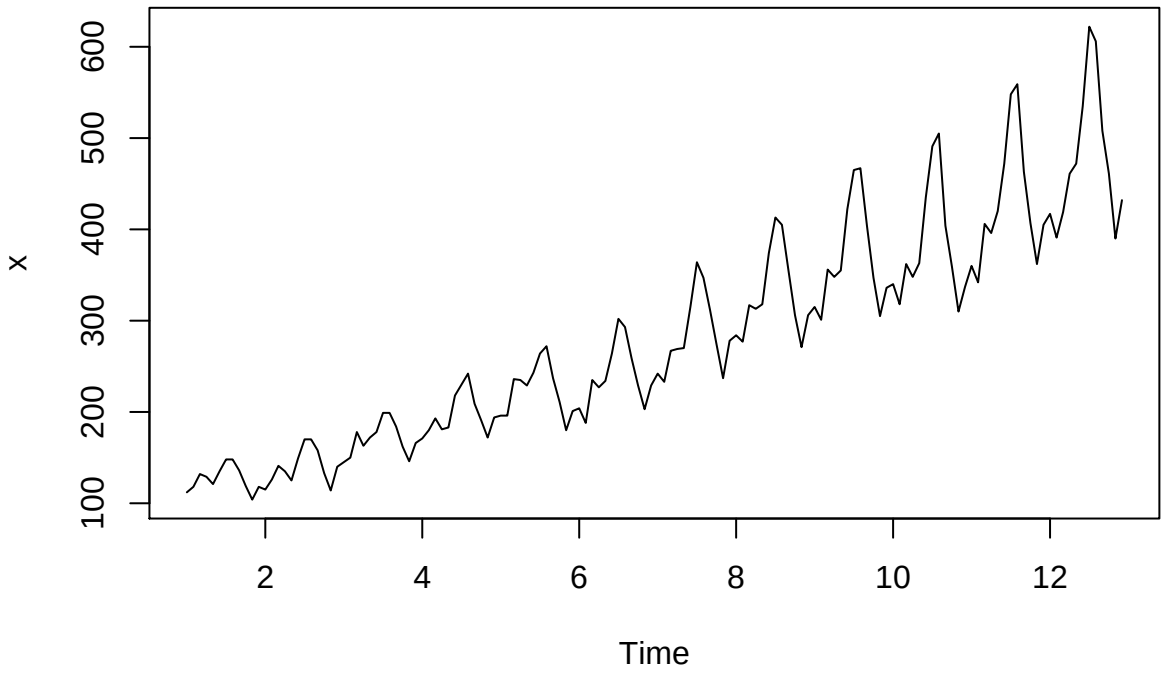
```
ap_y
```

```
##    Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1  112 118 132 129 121 135 148 148 136 119 104 118
## 2  115 126 141 135 125 149 170 170 158 133 114 140
## 3  145 150 178 163 172 178 199 199 184 162 146 166
## 4  171 180 193 181 183 218 230 242 209 191 172 194
## 5  196 196 236 235 229 243 264 272 237 211 180 201
## 6  204 188 235 227 234 264 302 293 259 229 203 229
## 7  242 233 267 269 270 315 364 347 312 274 237 278
## 8  284 277 317 313 318 374 413 405 355 306 271 306
## 9  315 301 356 348 355 422 465 467 404 347 305 336
## 10 340 318 362 348 363 435 491 505 404 359 310 337
## 11 360 342 406 396 420 472 548 559 463 407 362 405
## 12 417 391 419 461 472 535 622 606 508 461 390 432
```

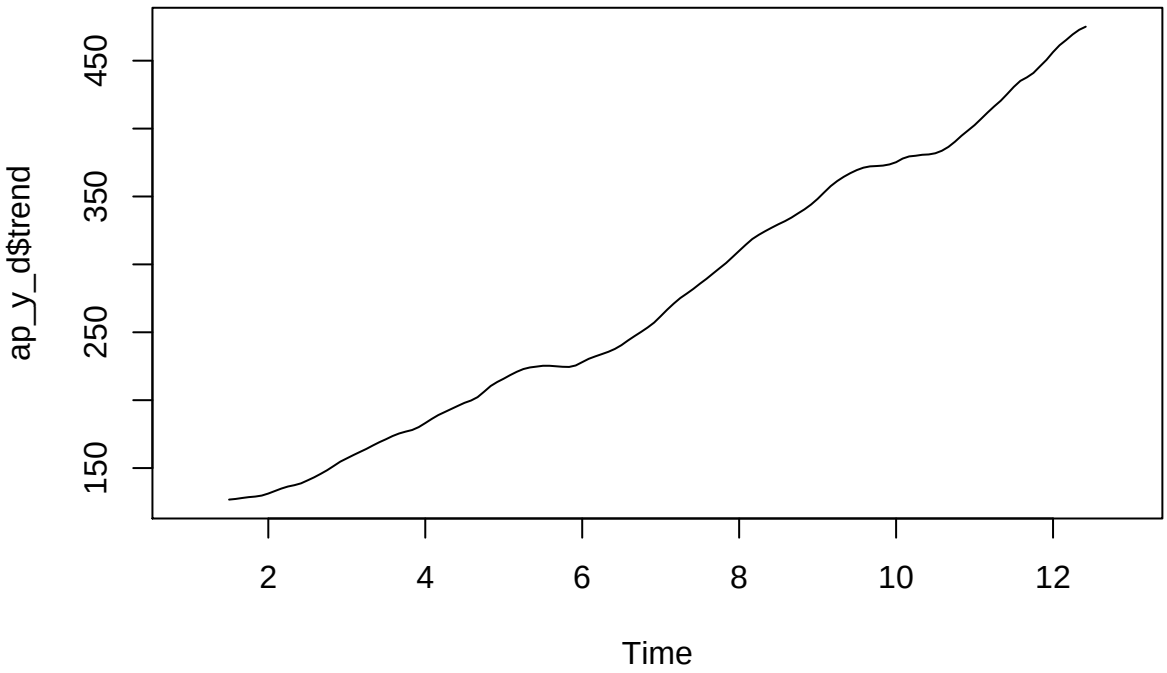
ANSWER (what will the frequency (points per season) of a yearly season be?) -> if there are 4 seasons in a year of 12 months, then each season will have 3 points

- 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?

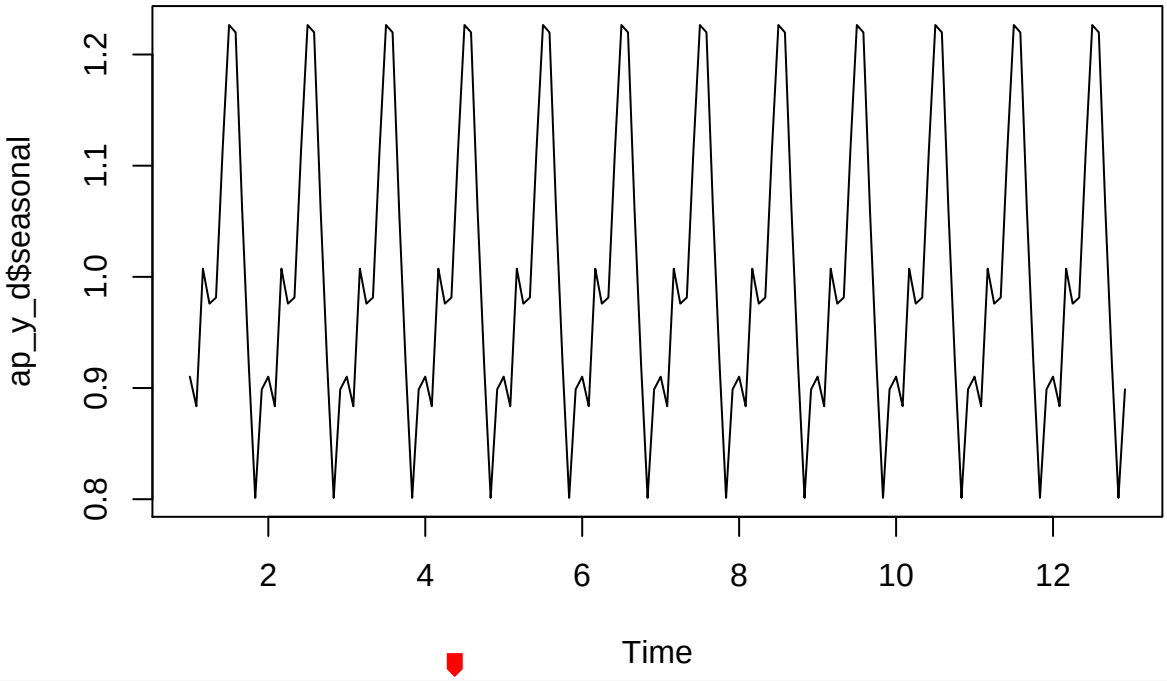
```
# playground
?decompose

# View after decomposition into base components

ap_y_d <-decompose(ap_y,type = "multiplicative")
plot(ap_y_d$x)
```




```
plot(ap_y_d$trend)
```

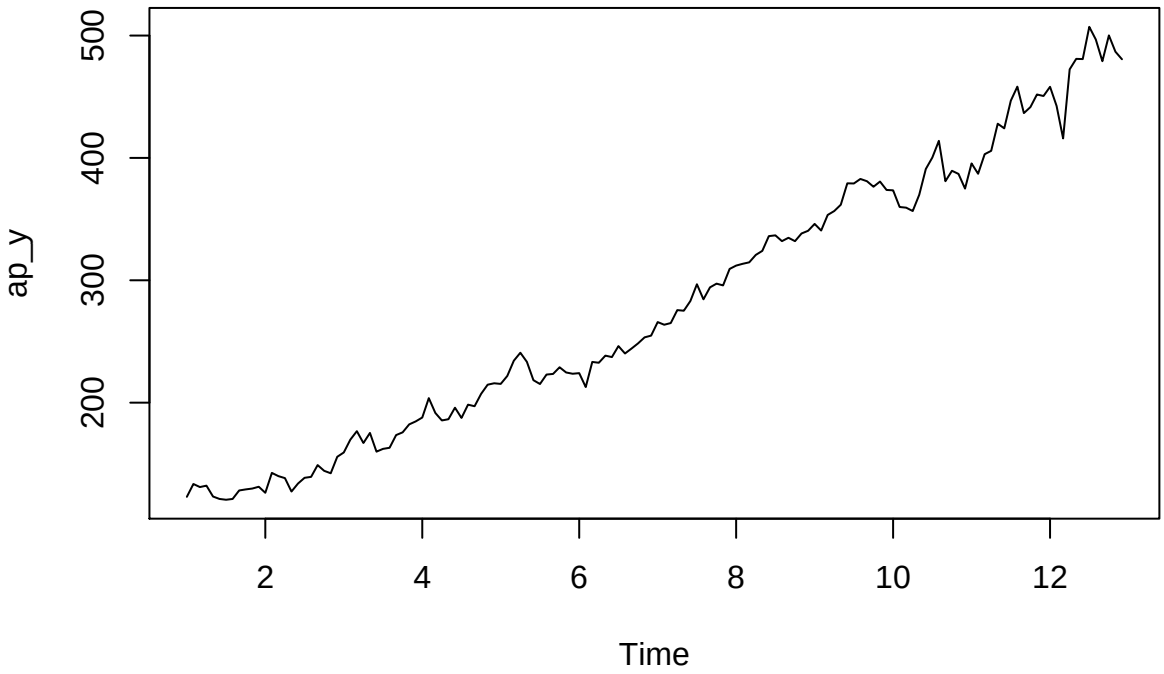



```
plot(ap_y_d$seasonal)
```




```
# ap_y_d2 <- (ap_y / ap_y_d$seasonal)   #is this correct ?
# plot(ap_y_d2)
```

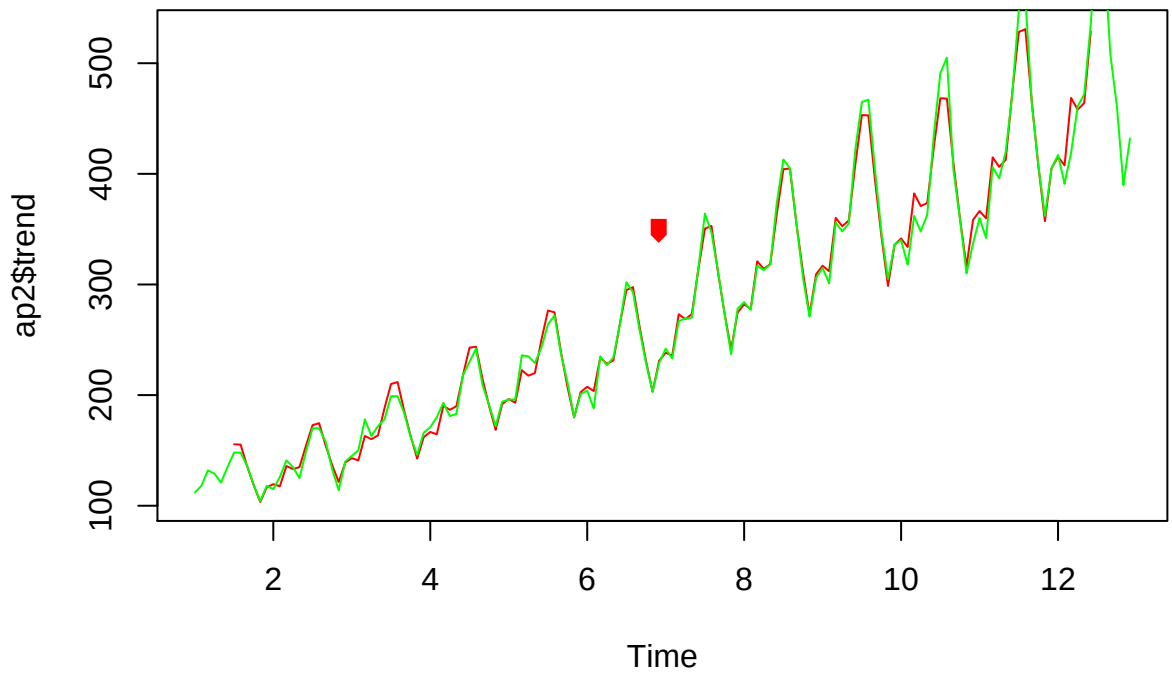
```
# Removing seasonality


ap2 <-decompose(ap_y, type = "multiplicative")
ap3 <- ap_y /ap2$seasonal # removing seasonal

plot(ap3)
```



ANSWER (what do you notice about the trend?) -> The trend of passengers keeps on increasing as the years pass.

- 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?

```
# Plotting simulated valued that is trend and seasonal vs actual

simulated <- (ap2$trend * ap2$seasonal)
plot( simulated, type="l", col="red" )
lines(ap_y, col="green")
```



AN-

SWER (how well does the trend represent the data?) ->

The trend very closely follows the actual 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?

```
# input1 <-read.csv("data/2108-351-351m-451-LE1-solectria-data-final.csv", header=TRUE)
#
# is.ts(input1) # check if it is a timeseries
# summary(input1)

# plot(input1$ghi_solargis,type = "l")
# plot(input1$iacp, type ="l")
# plot(input1$ghir, type = "l")
```

```
#########################################################
# Read in the data and get a look at the structure
input1 <-read.csv("data/2108-351-351m-451-LE1-solectria-data-final.csv", header=TRUE)
head(input1)
```

```
##                  time ghir iacp  temp ghi_solargis clear    ratio
## 1 2012-06-01 00:00:00 0.00    0 20.11            0 False 1.451629
## 2 2012-06-01 00:15:00 0.29    0 19.88            0 False 1.455344
## 3 2012-06-01 00:30:00 0.00    0 19.70            0 False 1.460904
## 4 2012-06-01 00:45:00 0.00    0 19.46            0 False 1.468499
## 5 2012-06-01 01:00:00 0.00    0 19.24            0 False 1.478400
## 6 2012-06-01 01:15:00 0.00    0 19.10            0 False 1.490993
```

```
#########################################################
# Find total length of dataset
dim(input1)[0]
```

```
## integer(0)
```

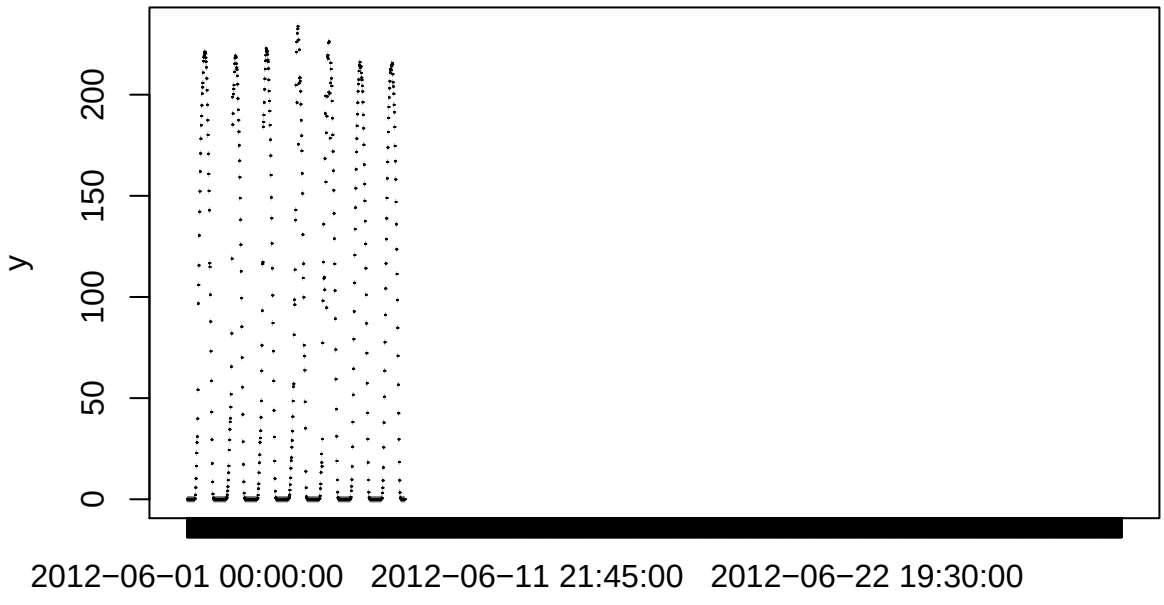
```
#########################################################
# Find the time interval
difftime(input1$time[2], input1$time[1],
         units = "mins")
```

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

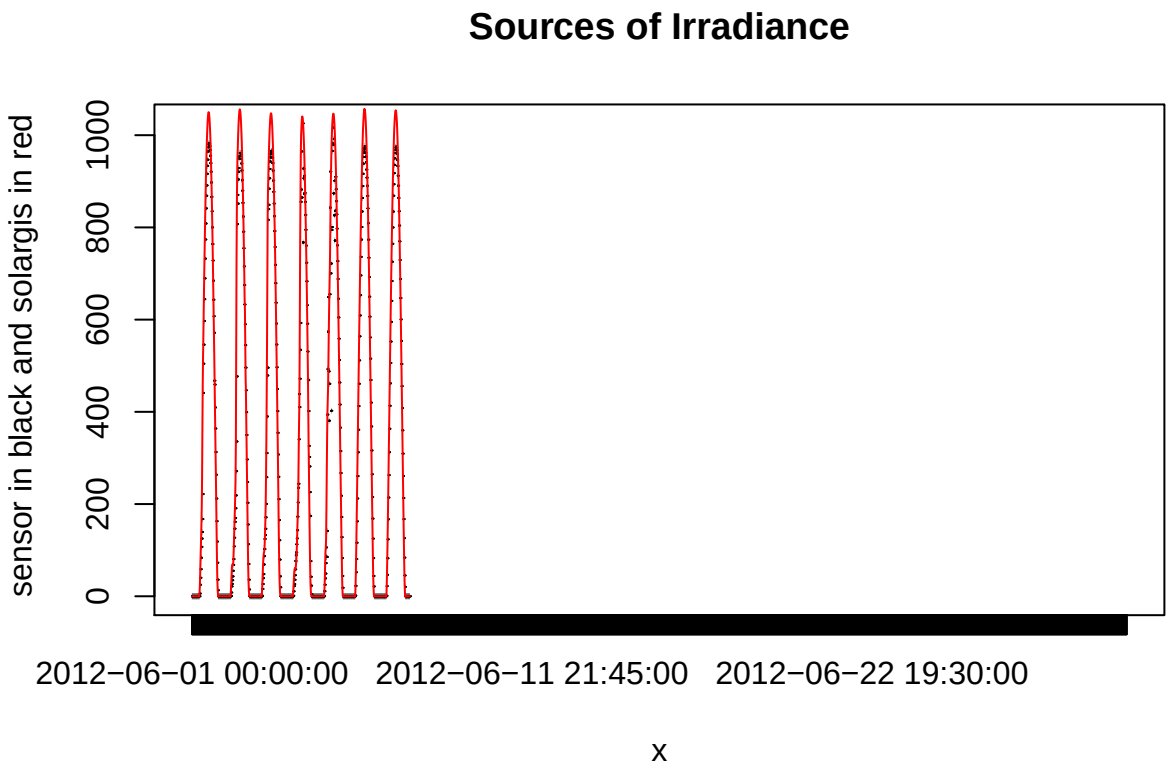
```
#########################################################
# Plot the irradiance and power for the first week
class(input1$time)
```

```
## [1] "factor"
```

```
# as.character(input1$time)
x <- input1$time
# y <- format(input1$time,  "%X" )
# z <- strptime(x ,format = "", tz = "")
z <- as.Date(x, "%Y-%m-%d")
input1$date <-z
# head(input1)

# inputf1 -> input1[]
dt_fst_wk <- subset(input1, date>="2012-06-01" & date<="2012-06-07")









###############


# Entire dataset is in 15-minute increment

# Power plot over time
timedt <- dt_fst_wk$time
pwr <- dt_fst_wk$iacp

plot(timedt,  pwr , col="blue", type ="l",main = "power vs time" ,pch="o", lty=1 )
```




```
# Both sources of irradiance
sen_ghi <- dt_fst_wk$ghir
gis_ghi <- dt_fst_wk$ghi_solargis

plot(timedt,  sen_ghi , col="green", type ="l", main="Sources of Irradiance",
ylab=" sensor in black and solargis in red" )
lines(timedt, gis_ghi , col="red", type ="l")
```

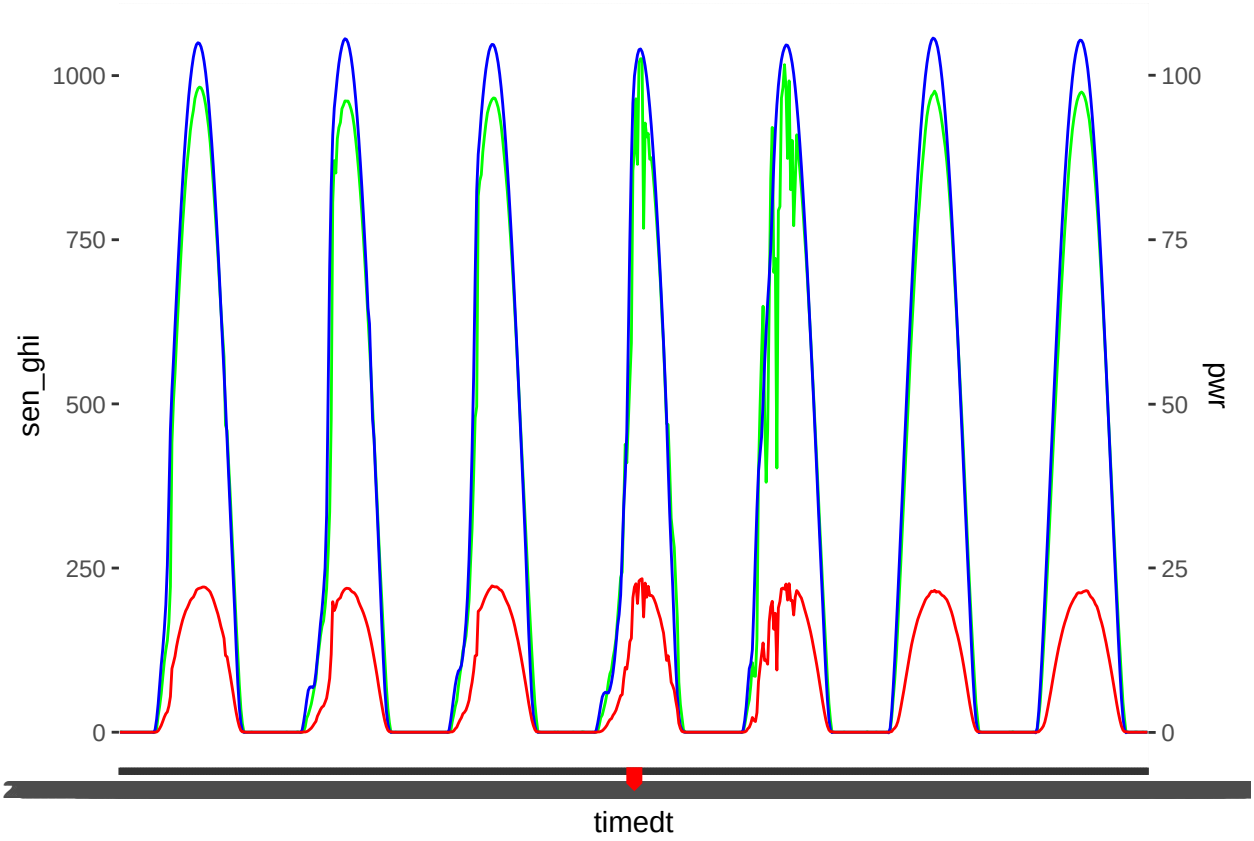



```
# #### Playground
# ## locale-specific version of date()
# format(Sys.time(), " %a %b %d %X %Y %Z")
# # "Mon Sep 13 11:02:51 AM 2021 EDT"
#
# ## time to sub-second accuracy (if supported by the OS)
# format(Sys.time(), "%H:%M:%OS3")
#
# (x <- strptime(c("2006-01-08 10:07:52", "2006-08-07 19:33:02"),
#                "%Y-%m-%d %H:%M:%S", tz = "EST5EDT"))
# attr(x, "tzone")
#
# ## read in date info in format 'ddmmmyyyy'
# ## This will give NA(s) in some non-English locales; setting the C locale
# ## as in the commented lines will overcome this on most systems.
# ## lct <- Sys.getlocale("LC_TIME"); Sys.setlocale("LC_TIME", "C")
# x <- c("1jan1960", "2jan1960", "31mar1960", "30jul1960")
# z <- strptime(x, "%d%b%Y")
# ## Sys.setlocale("LC_TIME", lct)
# class(z)
# x <-"2012-06-01 00:00:00"
# as.character(x)
# y <- strptime(x, format = "", tz = "")
# class(y)
```

ANSWER (how does the irradiance look compared to what you would expect from the trend of sunlight?) -> The irradiance captured by the sensor closely follows the solar gis values.

ANSWER (how well does the power and irradiance match up?) -> The power and irradiance have the same period and they match up well.

```
# All 3 lines on the same plot with a secondary axis

ggplot() +
  geom_line(aes(x= timedt, y = sen_ghi), group =1, color ="green")+
  geom_line(aes(x= timedt, y = gis_ghi), group =1, color ="blue")+
  geom_line(aes(x= timedt, y = pwr), group =1, color ="red")+
  scale_y_continuous(sec.axis = sec_axis(trans =~./10, name ="pwr"))
```




```
# # par(mar = c(5, 4, 4, 4) + 0.3)               # Additional space for second y-axis
# plot(timedt, pwr, pch = 16, col = "blue")              # Create first plot
# par(new = TRUE)                                # Add new plot
# plot(timedt, sen_ghi, pch = 17, col = "green",              # Create second plot without axes
#      axes = FALSE, xlab = "", ylab = "")
# lines(timedt, gis_ghi , col="red", type ="l")
# axis(side = 4, at = pretty(range(sen_ghi)))      # Add second axis
# mtext("sen_ghi", side = 4, line = 3)
```

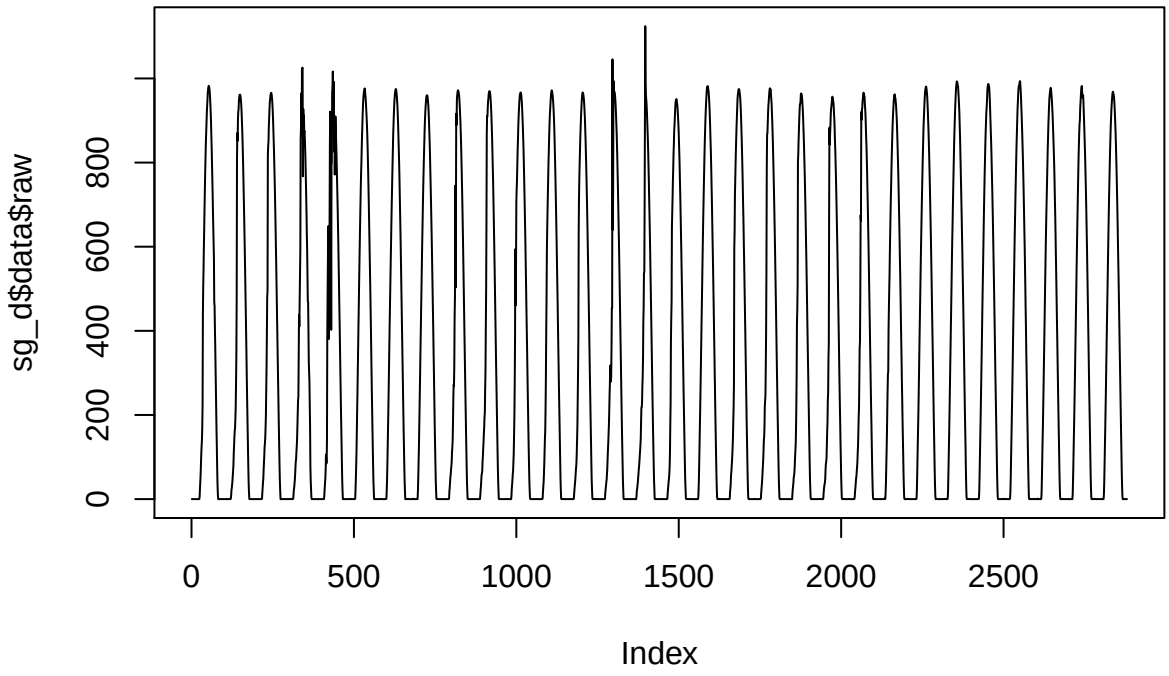
- Use the ts() functions and the stlplus() function from the stlplus package
  - to decompose the sensor and SolarGIS irradiance and the power
    - for the whole month.
  - Plot each of the decompositions, what do you notice?

```
# 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?
# use s.window = "periodic" for the stlplus function
library(stlplus)
?stlplus()
```

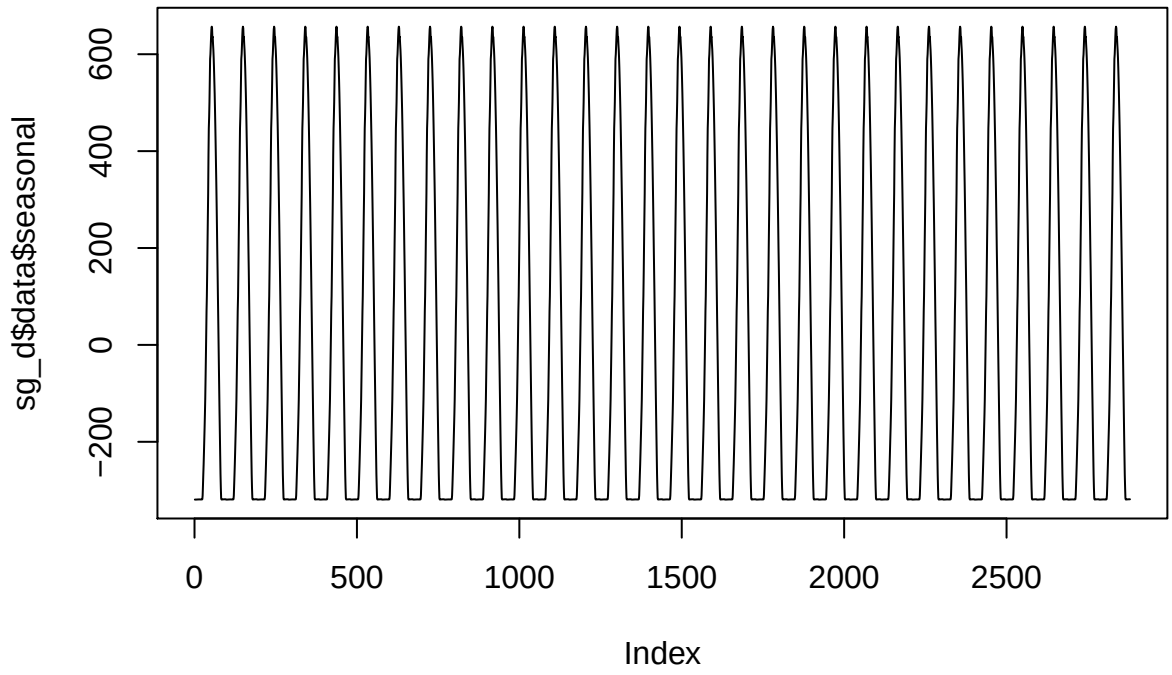
```
#convert data to timeseries objects
sen_ghi <- ts(data = input1$ghir,  frequency =4*24) ## change this
gis_ghi <- ts(data = input1$ghi_solargis , frequency = 4*24)
pwr <- ts(data = input1$iacp, frequency = 4*24)

?
sen_ghi
```

```
## No documentation for 'sen_ghi' in specified packages and libraries:
## you could try '??sen_ghi'
```

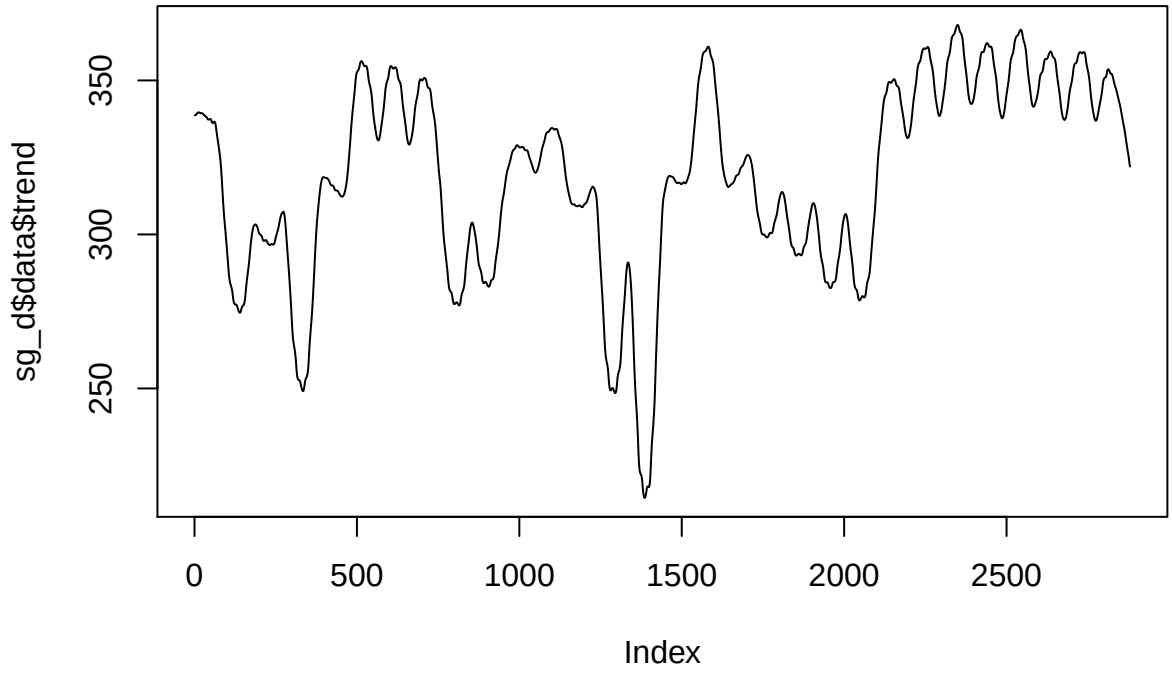
```
#decompose using stlplus
sg_d <- stlplus(sen_ghi,s.window = "periodic")
gg_d <- stlplus(gis_ghi,s.window = "periodic")
pwr_d <- stlplus(pwr,s.window = "periodic")


#plots1 - 4
plot(sg_d$data$raw, type="l")
```



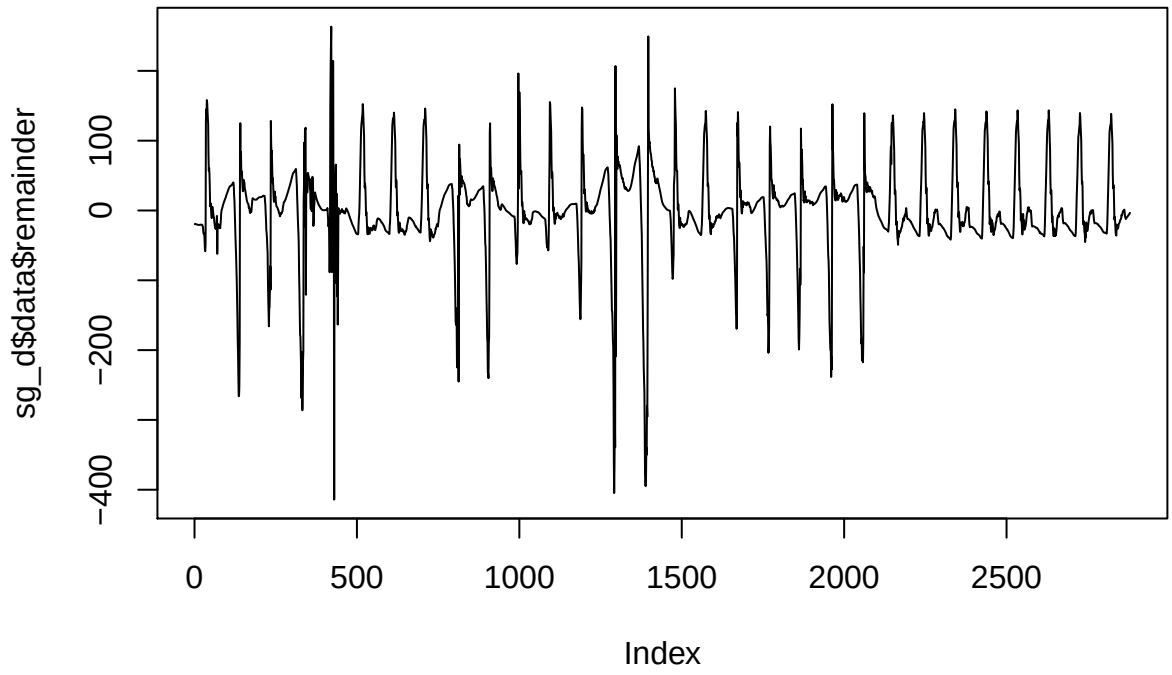

```
plot(sg_d$data$seasonal, type="l")
```



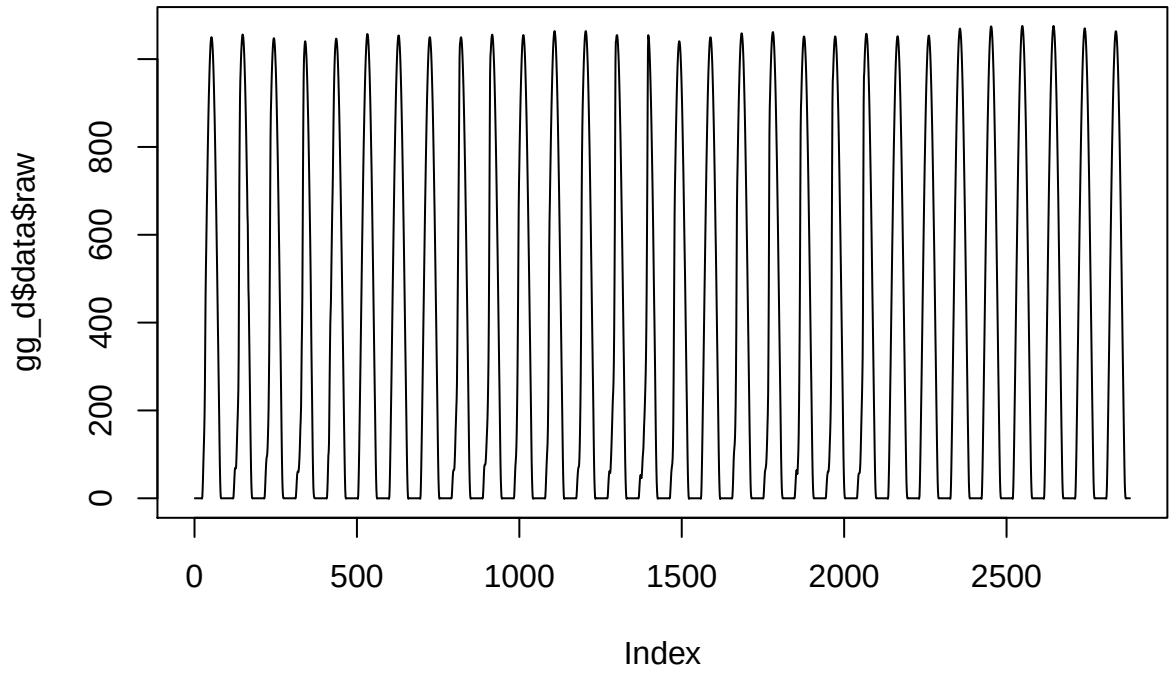

```
plot(sg_d$data$trend, type="l")
```



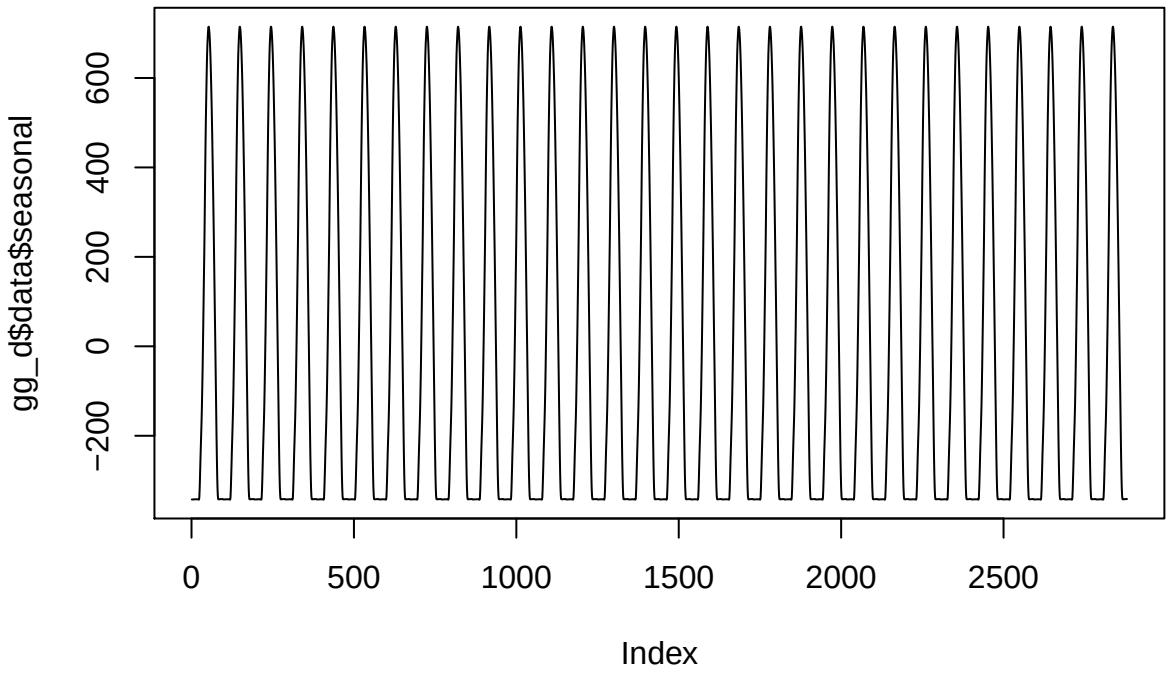

```
plot(sg_d$data$remainder, type="l")
```



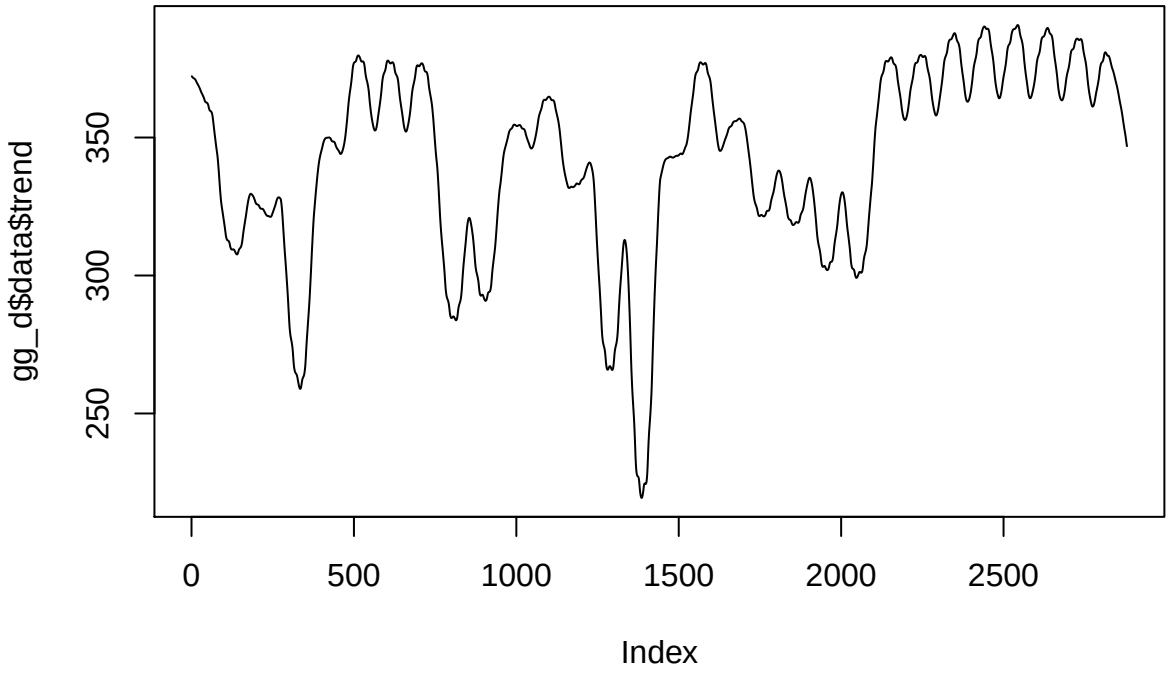

```
#plots5-8
plot(gg_d$data$raw, type="l")
```



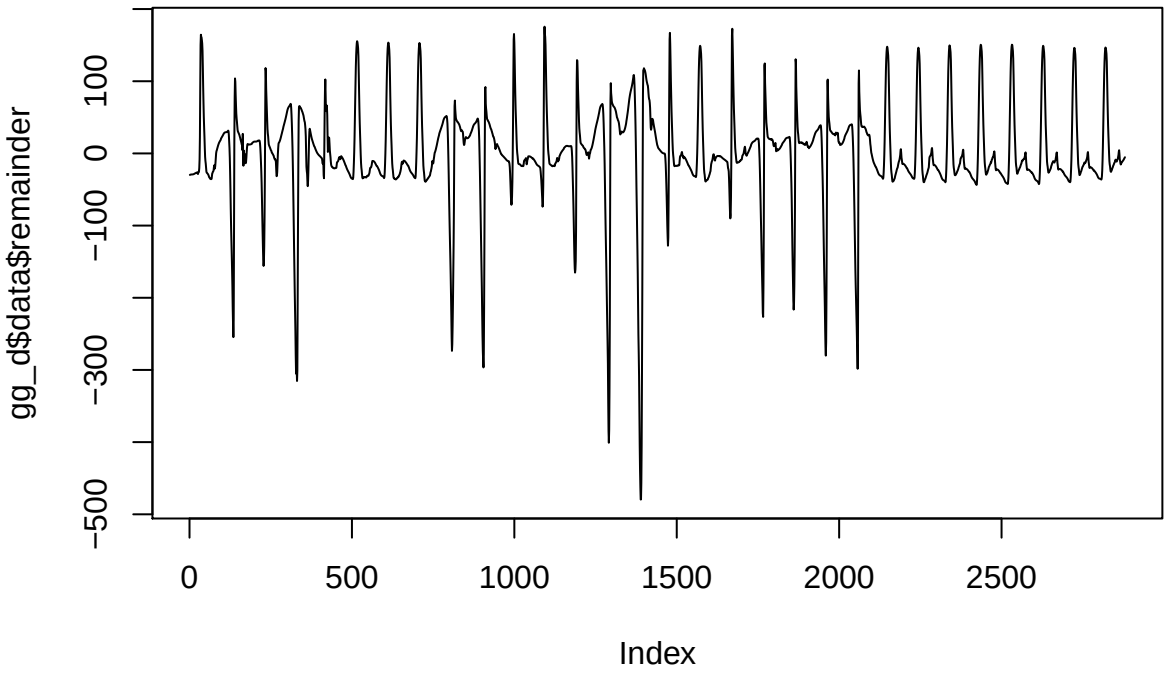

```
plot(gg_d$data$seasonal, type="l")
```



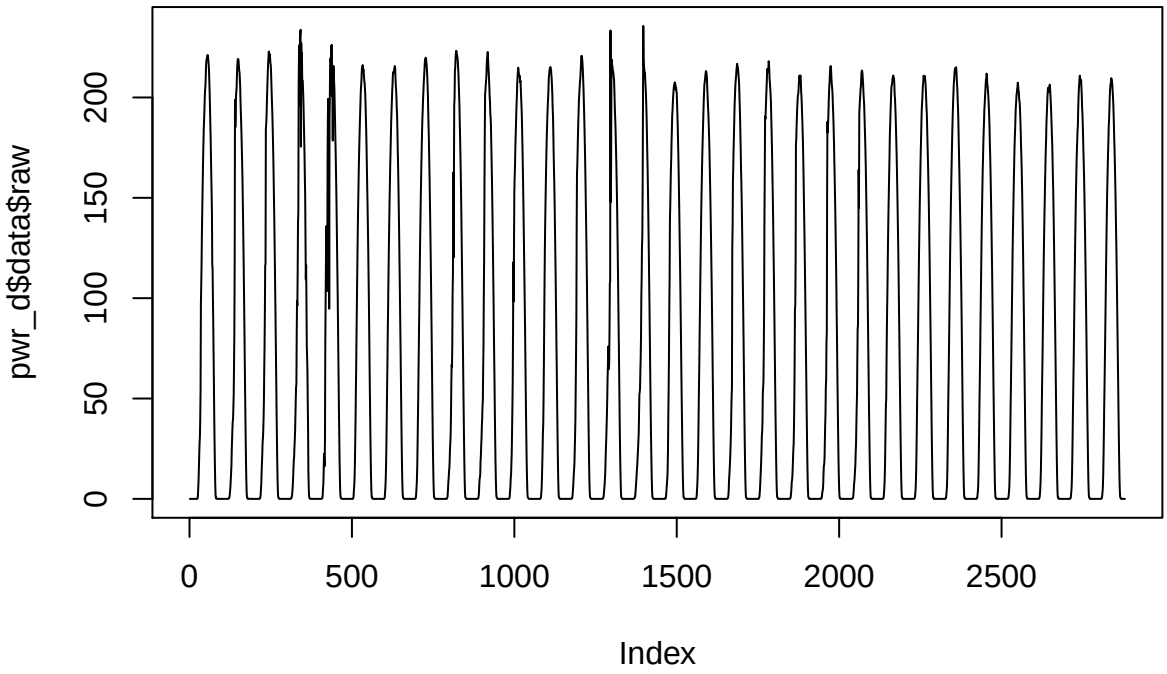

```
plot(gg_d$data$trend, type="l")
```



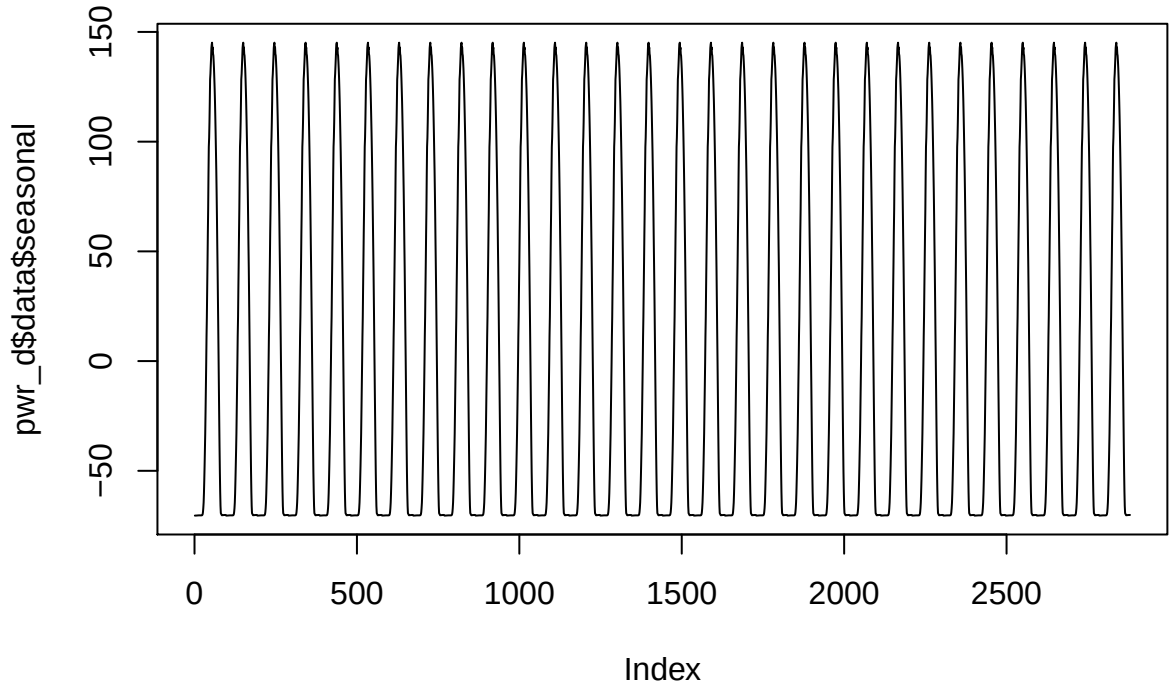

```
plot(gg_d$data$remainder, type="l")
```



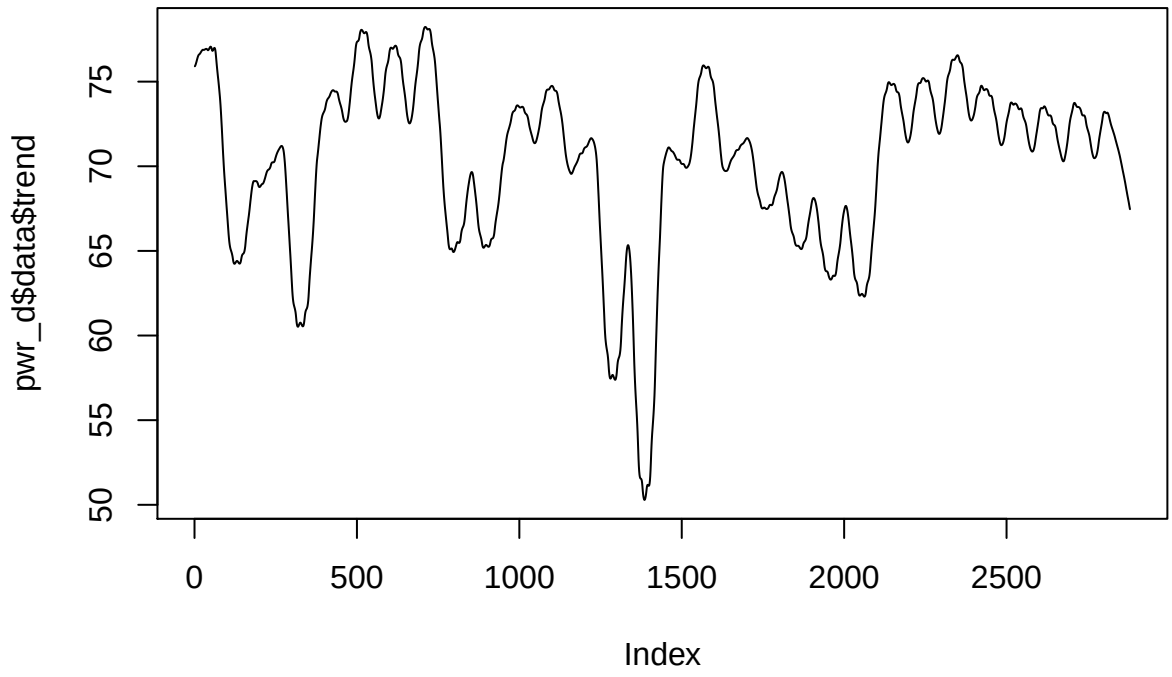

```
#plots5-8
plot(pwr_d$data$raw, type="l")
```



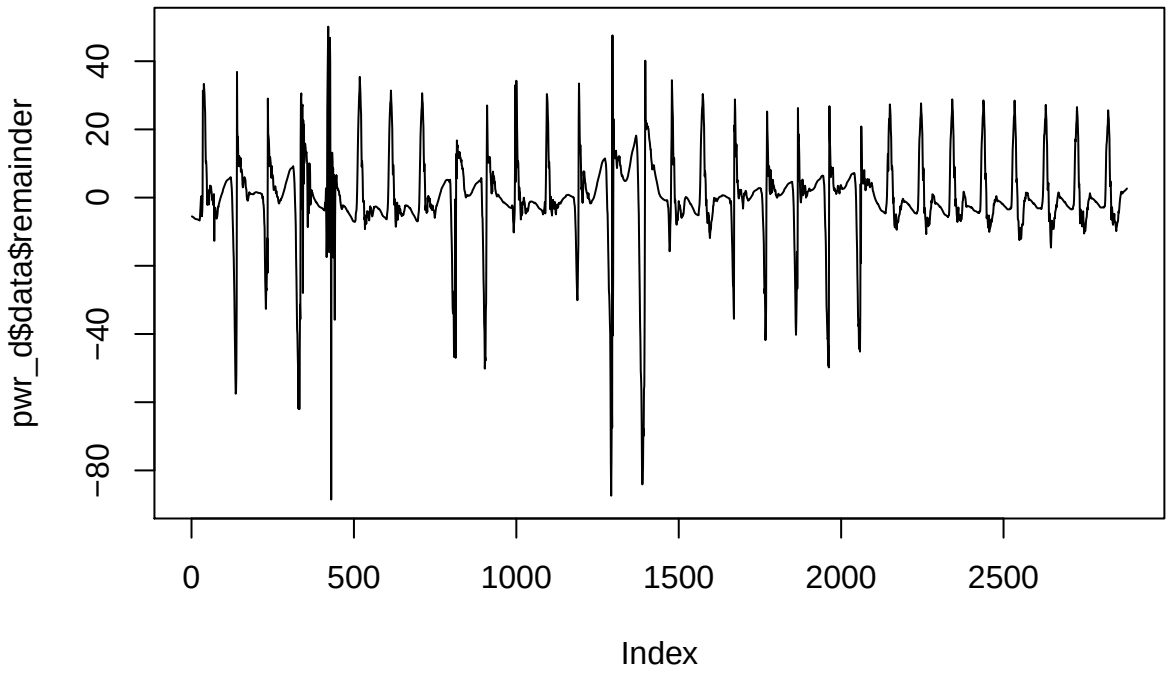

```
plot(pwr_d$data$seasonal, type="l")
```




```
plot(pwr_d$data$trend, type="l")
```




```
plot(pwr_d$data$remainder, type="l")
```

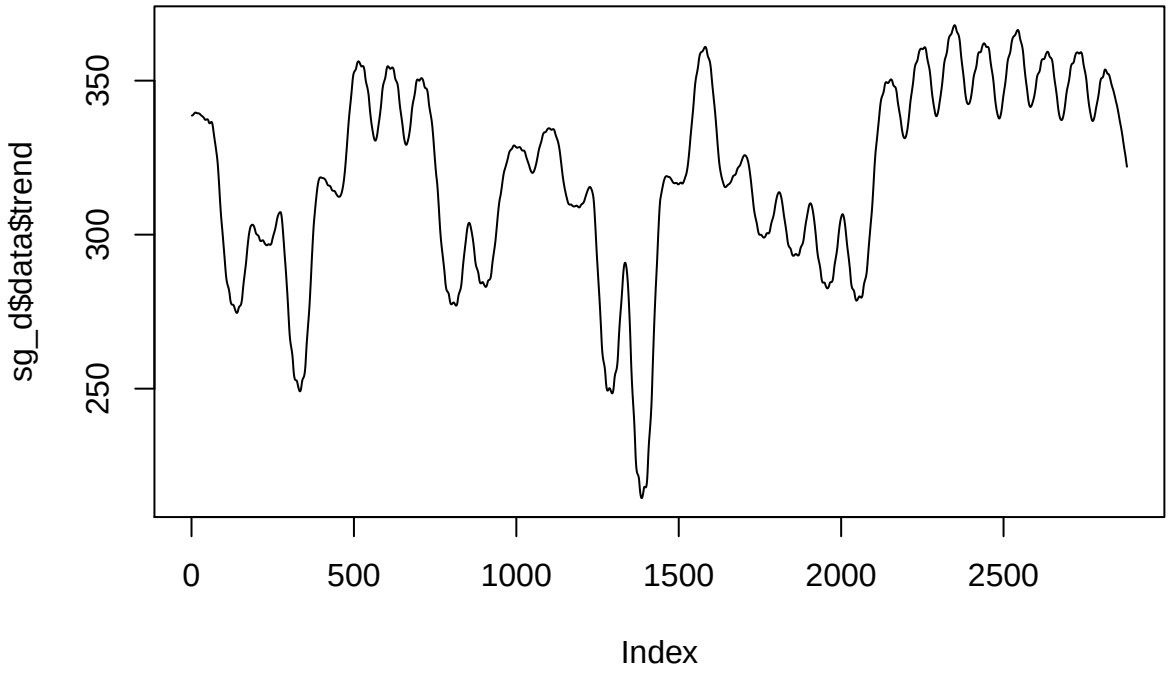


AN-

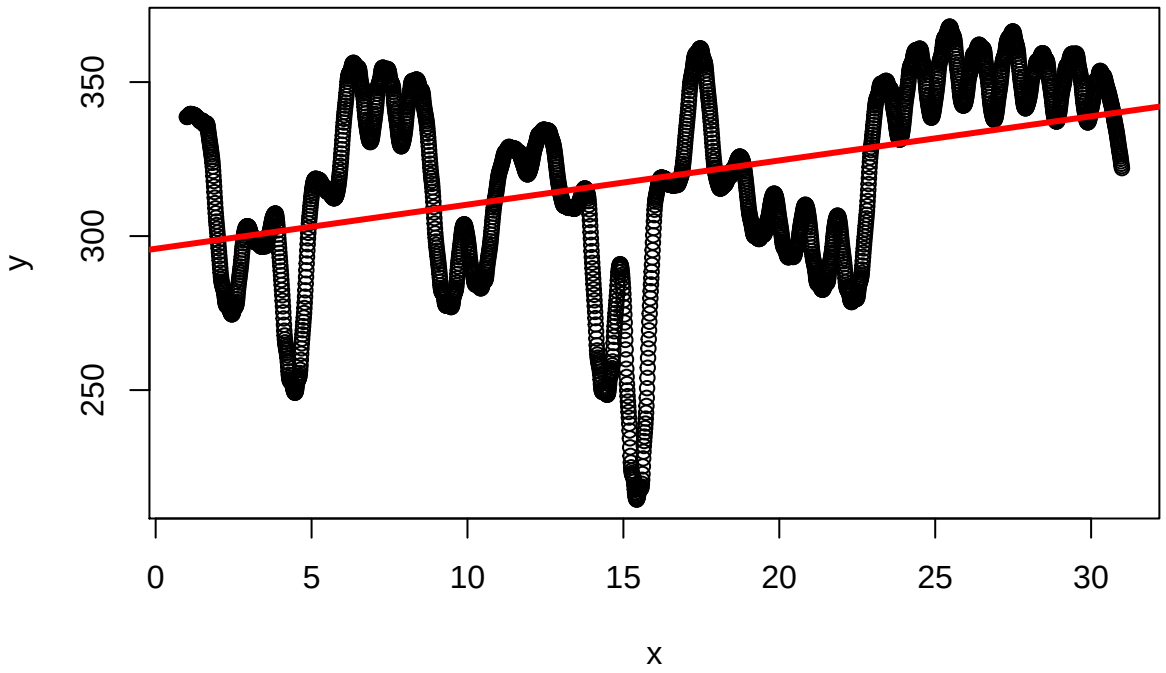
SWER (what do you notice about the decompositions?) ->

- 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?

```
y <- sg_d$data$trend
x <- sg_d$time
plot(sg_d$data$trend, type="l")
```



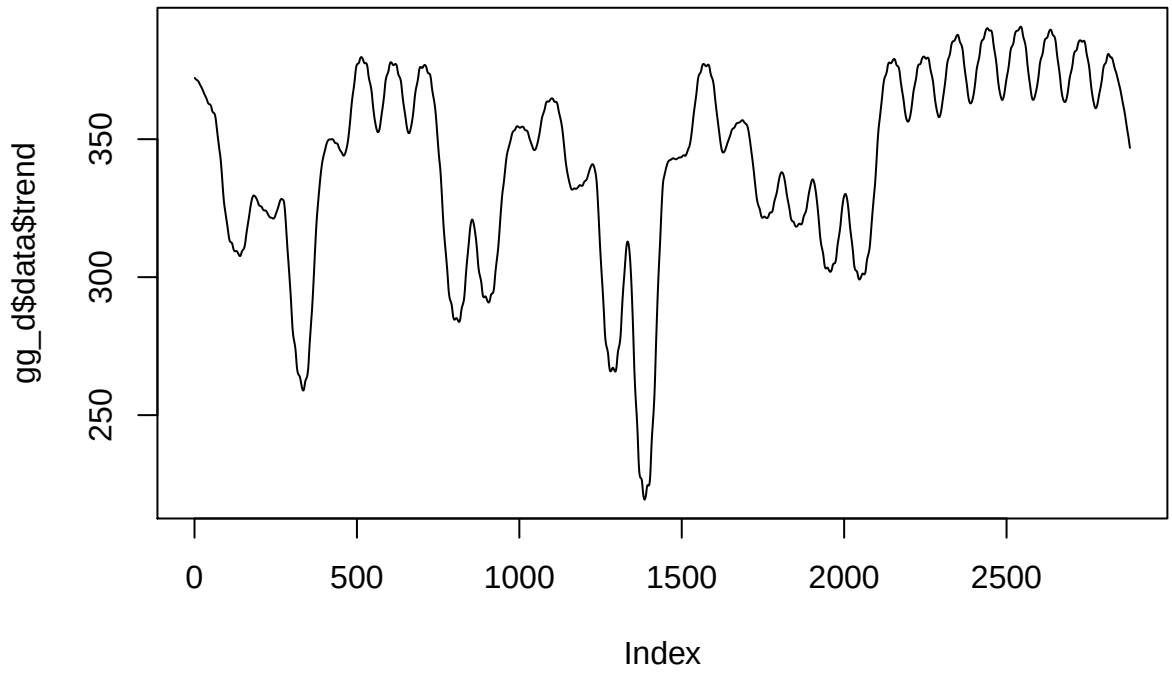

```
model1 <- lm(y ~ x)
plot(x,y)
abline(model1, lwd =3, col ="red")
```



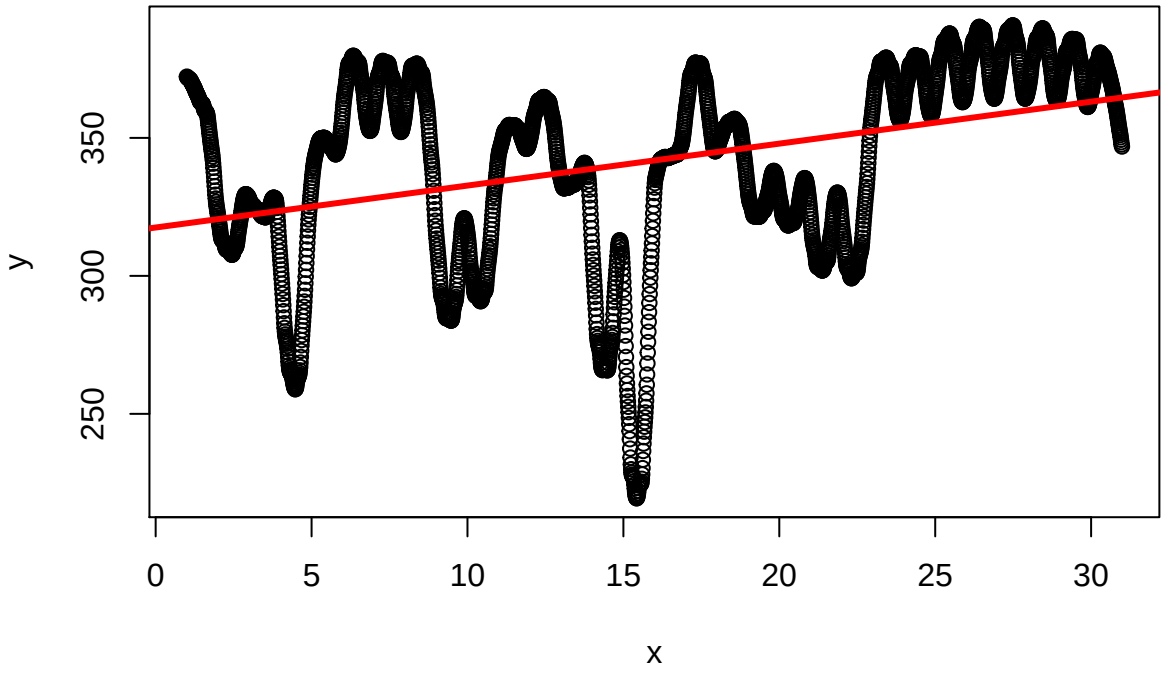

```
summary(model1)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -103.543  -18.987    3.705   20.206   51.279
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 295.88174    1.13373     261   <2e-16 ***
## x             1.43376    0.06233      23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.97 on 2878 degrees of freedom
## Multiple R-squared:  0.1553, Adjusted R-squared:  0.155
## F-statistic: 529.1 on 1 and 2878 DF,  p-value: < 2.2e-16
```

```
y <- gg_d$data$trend
x <- gg_d$time
plot(gg_d$data$trend, type="l")
```




```
model2 <- lm(y ~ x)
plot(x,y)
abline(model2, lwd =3, col ="red")
```

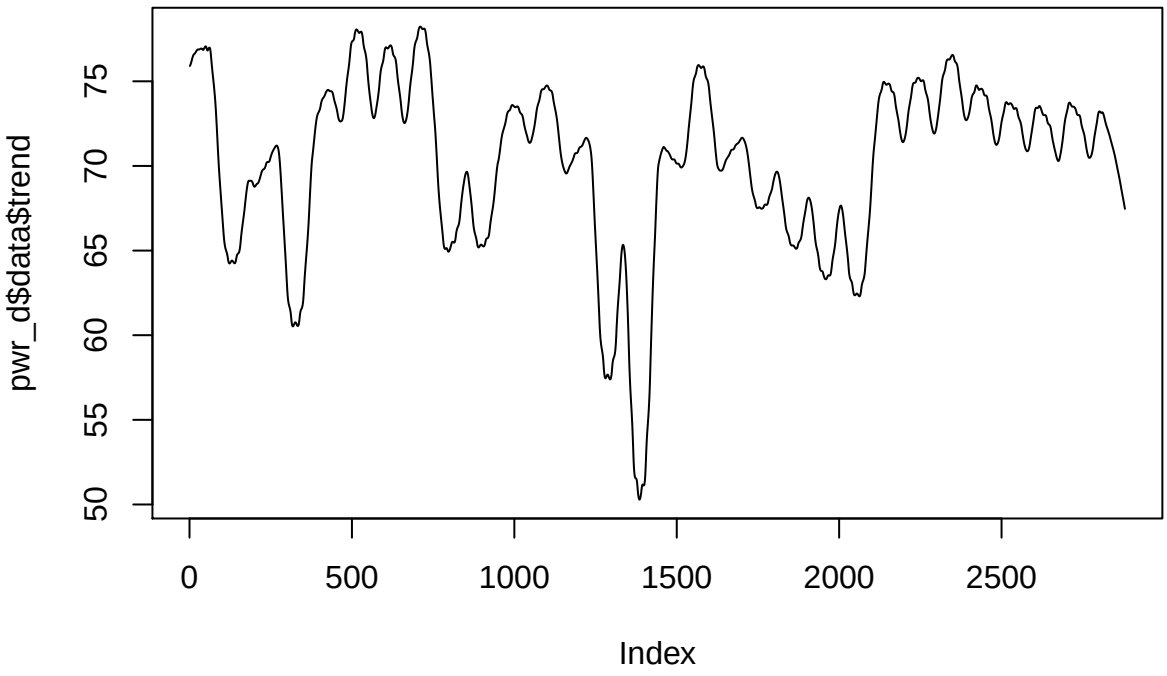



```
summary(model2)
```

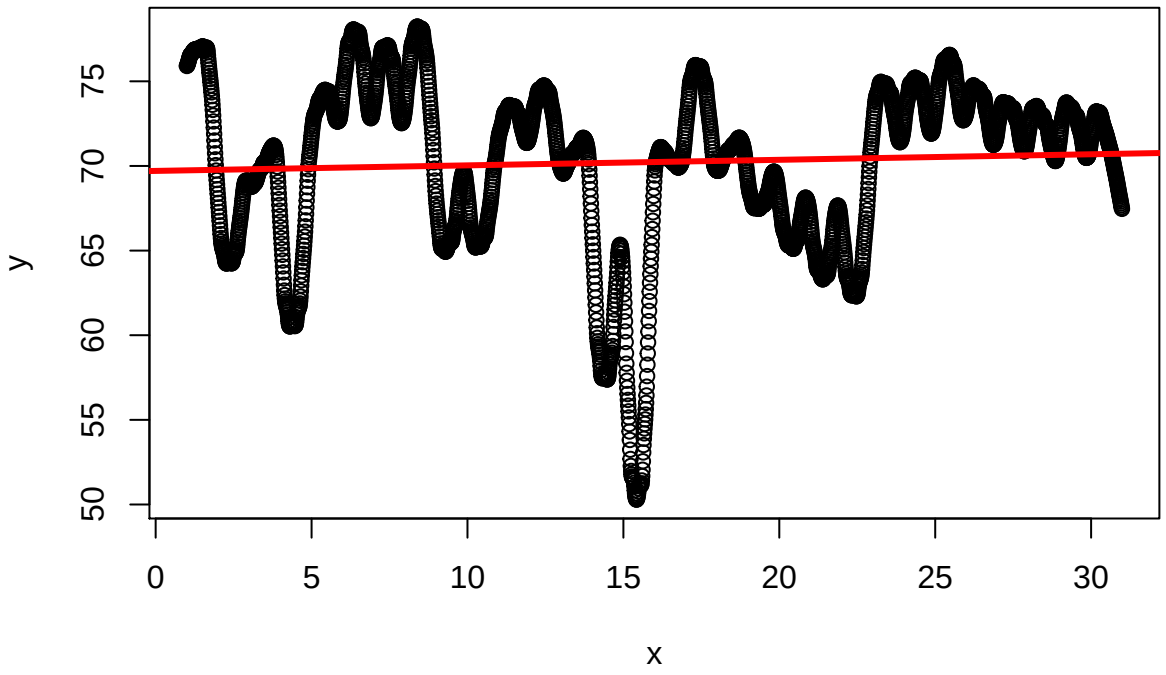
```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -121.569  -18.965    6.223   23.220   53.151
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 317.52385    1.26607  250.79   <2e-16 ***
## x             1.51993    0.06961   21.84   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 32.35 on 2878 degrees of freedom
## Multiple R-squared:  0.1421, Adjusted R-squared:  0.1418
## F-statistic: 476.8 on 1 and 2878 DF,  p-value: < 2.2e-16
```

```
y <- pwr_d$data$trend
x <- pwr_d$time
plot(pwr_d$data$trend, type="l")
```




```
model3 <- lm(y ~ x)
plot(x,y)
abline(model3, lwd =3, col ="red")
```




```
summary(model3)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.9218  -2.8153   0.9651   3.4244   8.2455
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 69.70711    0.19894 350.391  < 2e-16 ***
## x            0.03276    0.01094   2.995  0.00277 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.083 on 2878 degrees of freedom
## Multiple R-squared:  0.003107,   Adjusted R-squared:  0.002761
## F-statistic: 8.971 on 1 and 2878 DF,  p-value: 0.002766
```

ANSWER (compare the models to each other and how are they different?) -> both sensor sunlight data and Gis show an increase in the years with almost the same slope, even though power has increased over the years, it has risen slowly.

- 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?) -> Yes, since power has risen slower than the growth rate of irradiance, it is likely that the system has degraded over time.

ANSWER (how do the sensor GHI and the SolarGIS GHI compare to power?) -> The sensor GHI and Solar

GIS GHI, have grown the same way, it is likely that the sensors are working well.

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,

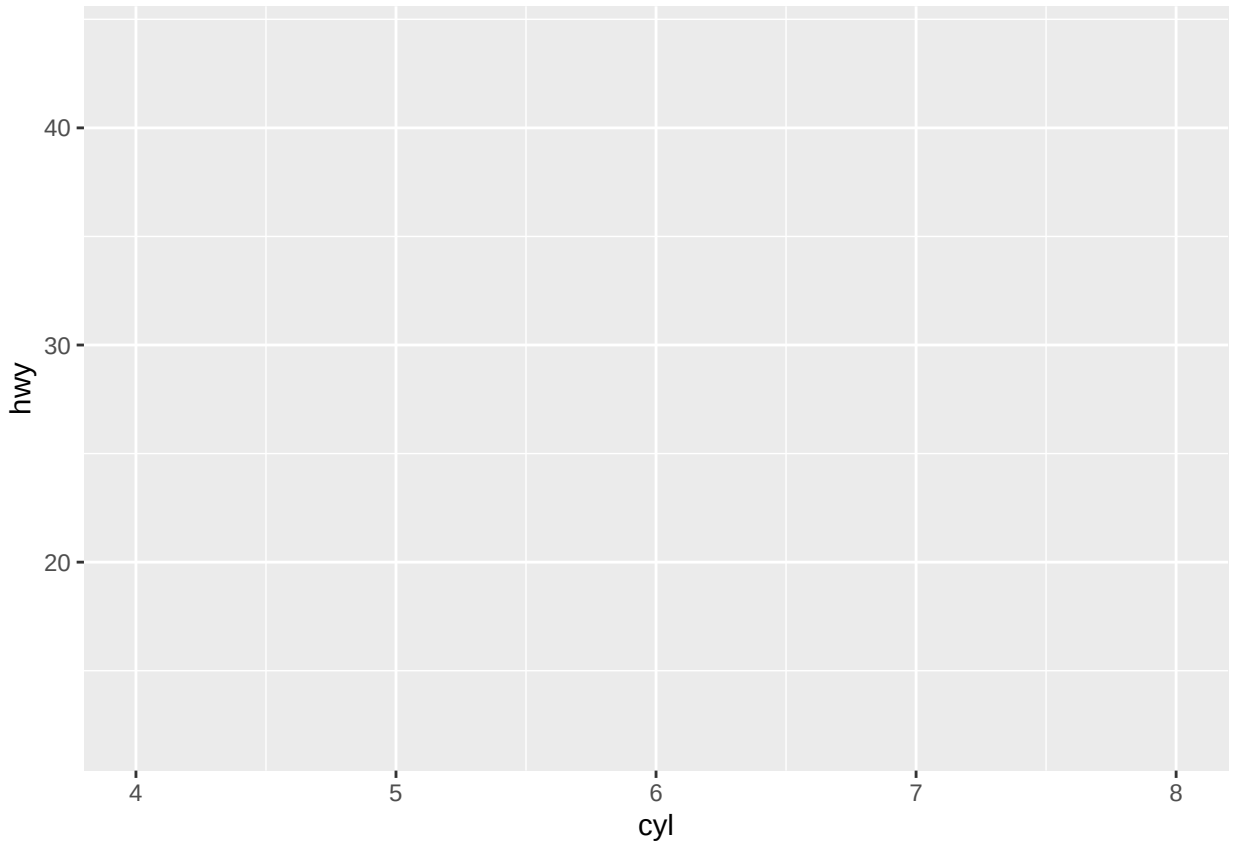
- where first the graph must be specified,
- then layers are added to the plot using the '+' operator.

In this example nothing appears in the plot

```
library(ggplot2)
data("mpg")
head(mpg)
```

```
## # A tibble: 6 x 11
##   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
##   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
## 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
## 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
## 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
## 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
## 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
```

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



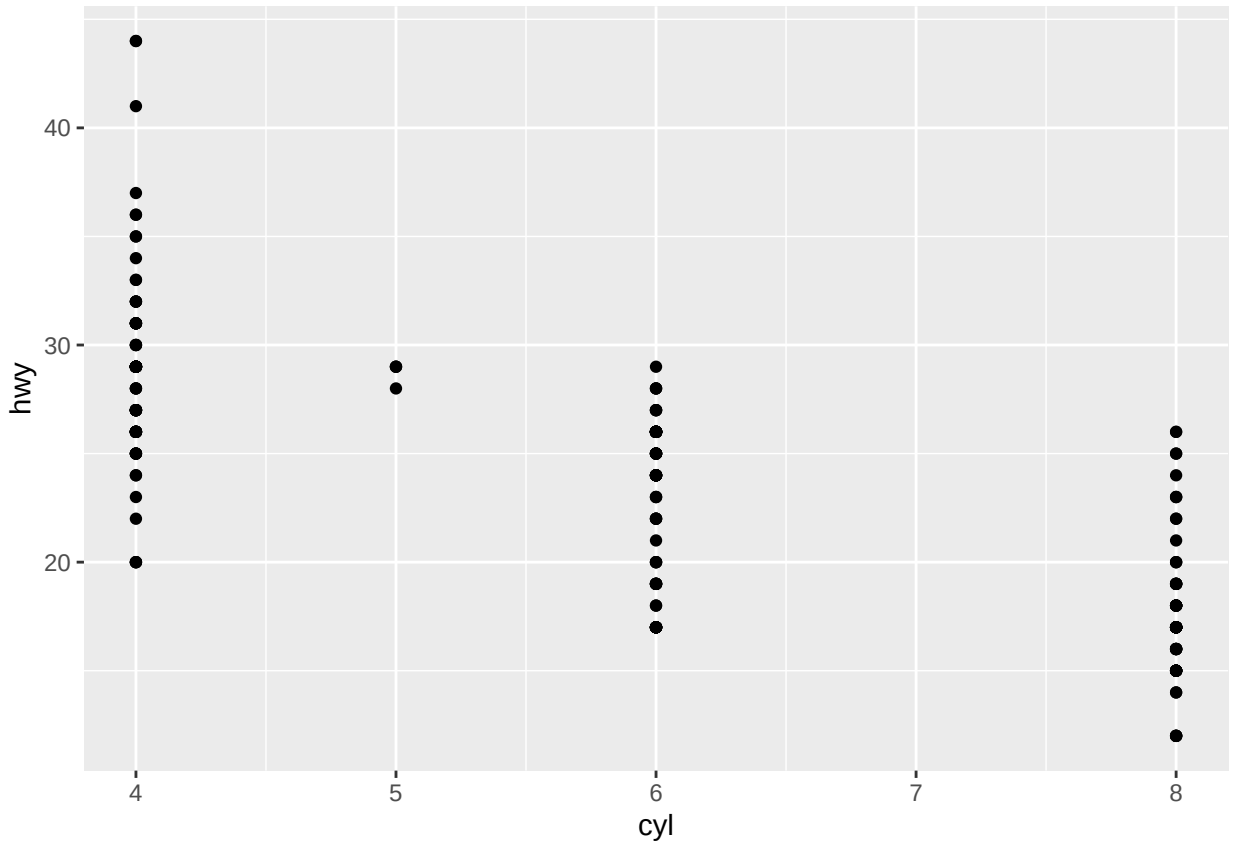
This is because we did not define the next layer,

- all we did was define some kind of graph between cylinders and highway mpg

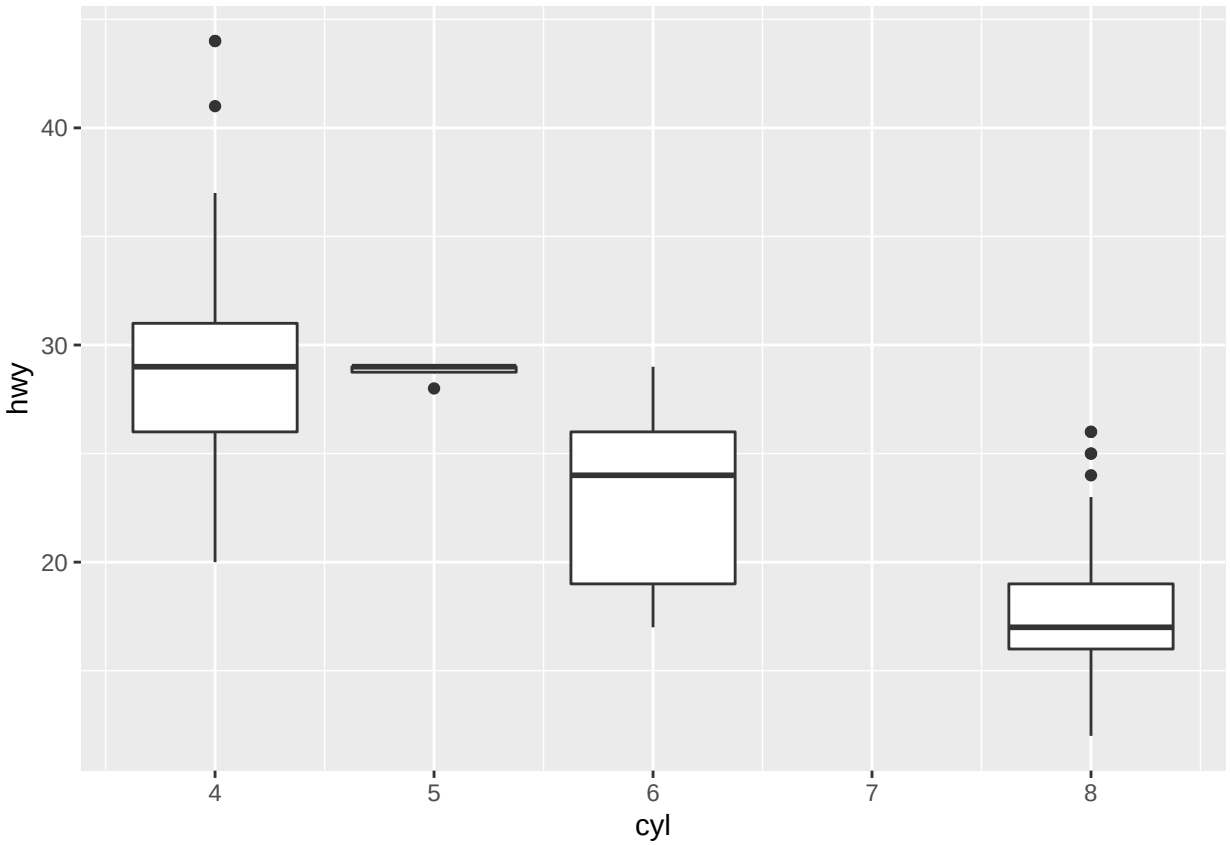
Since we have already defined the data at the beginning,

- we don't need to specify it in the layer

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



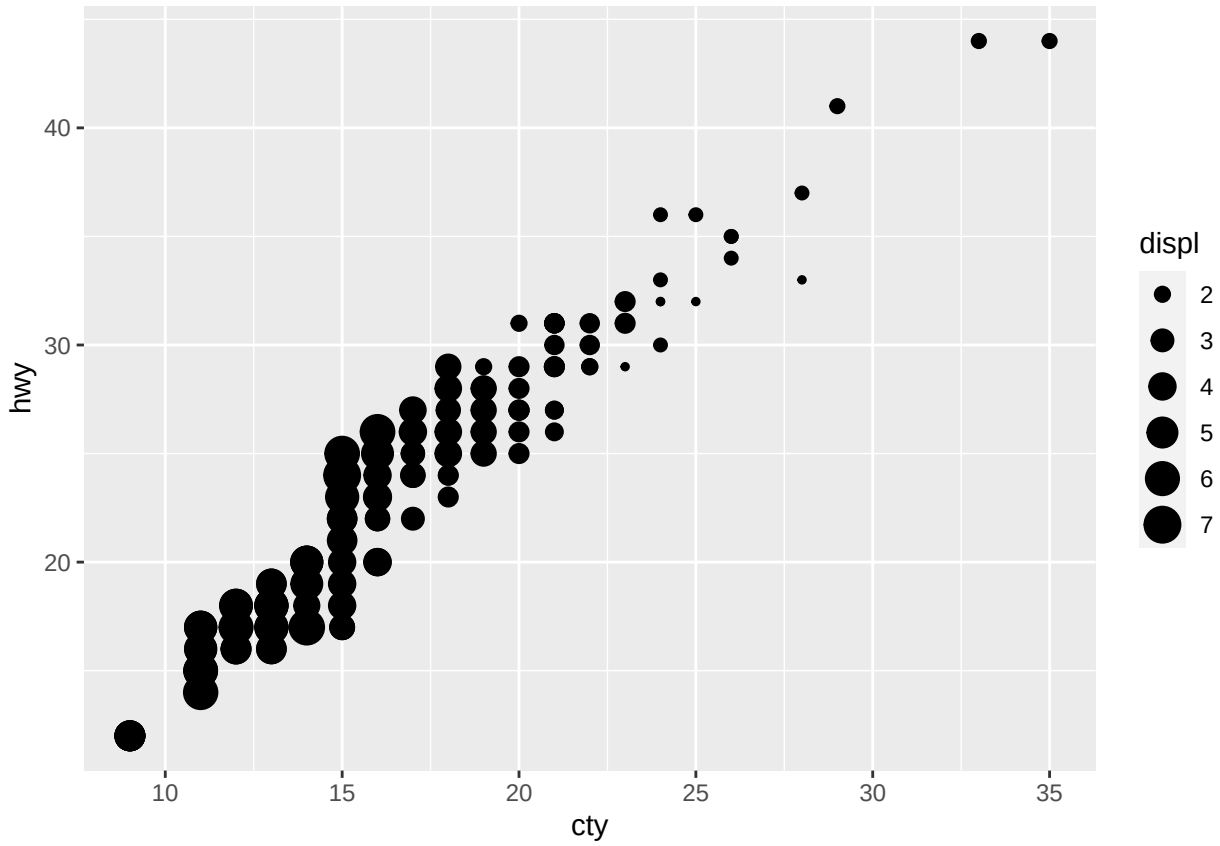

```
# or a different layer

# 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,

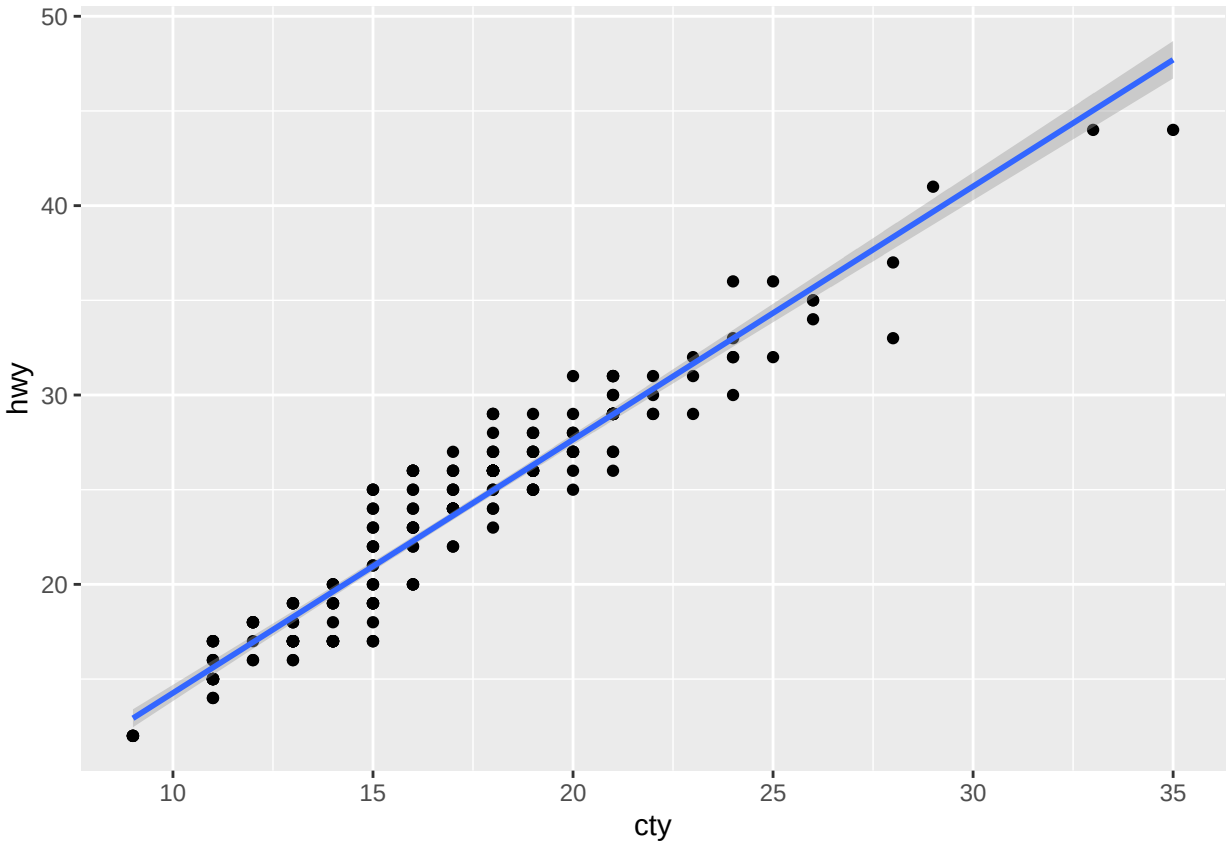
- keep in mind ordering is important.

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), method = "lm")
```

```
## `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 funciton

2.2.1 LE2-2a (0.25 points)

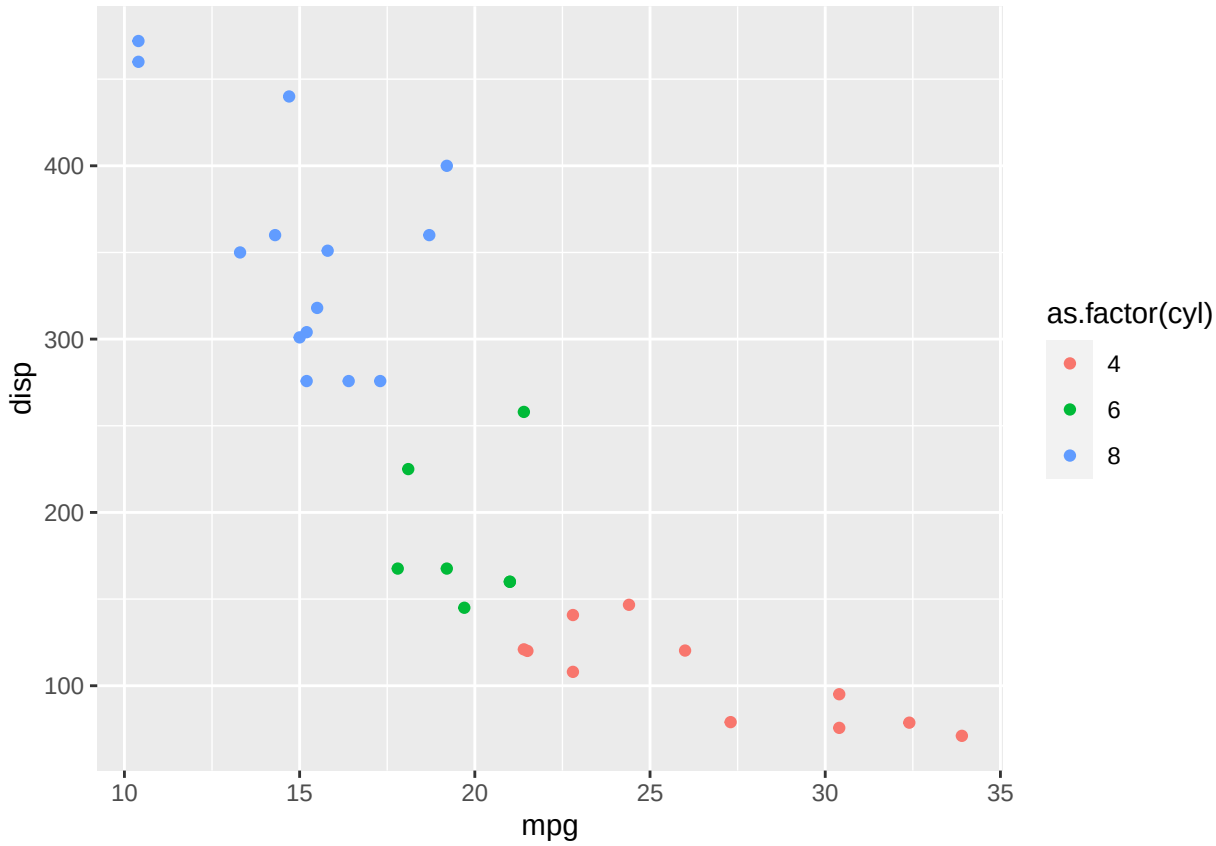
2a: Use the mtcars data set,

- plot mpg vs displacement and color by cylinders

```
data("mtcars")
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

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



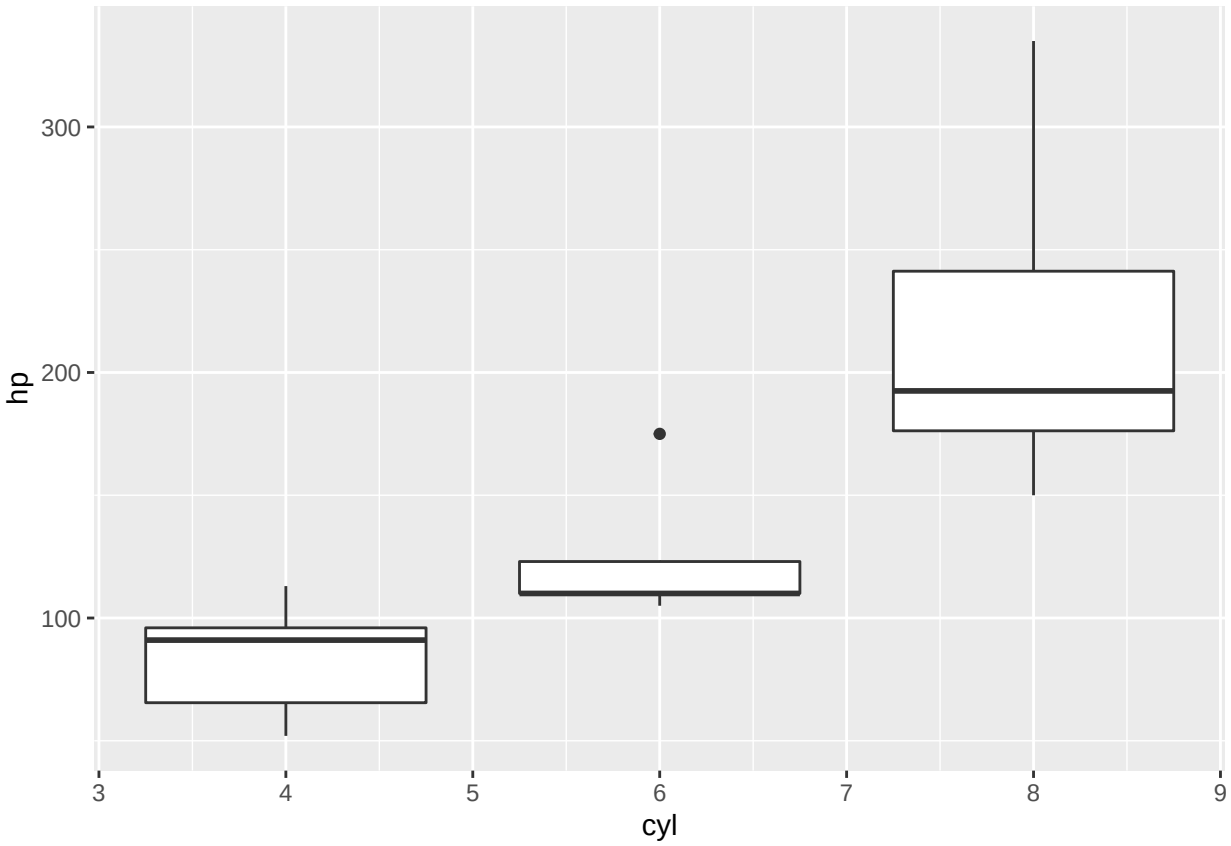

```
?mtcars
```

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

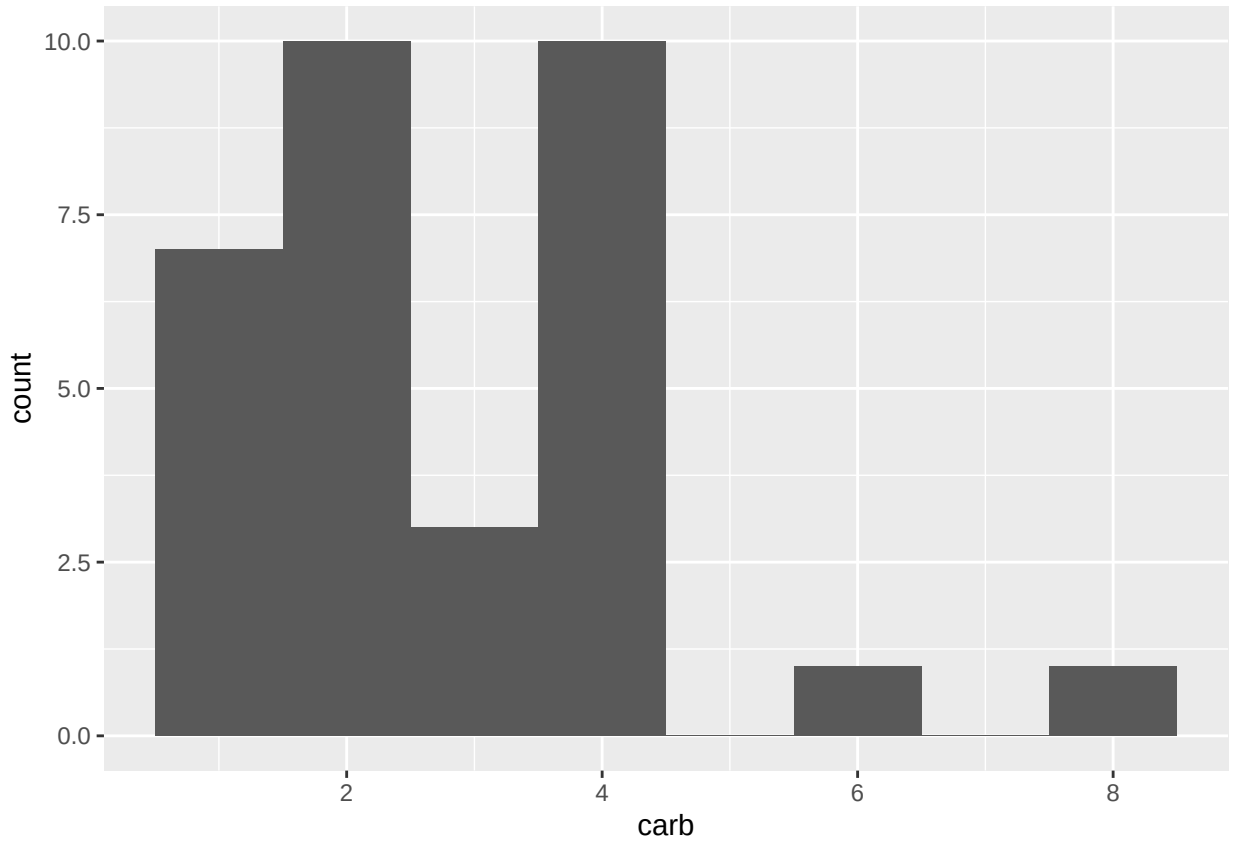
```
ggplot(data = mtcars, aes(x = cyl, y = hp)) +
  geom_boxplot(aes(group = cyl))
```



2.2.3 LE2-2c (0.5 points)

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

```
ggplot(mtcars, aes(carb)) +
  geom_histogram(binwidth =1)
```

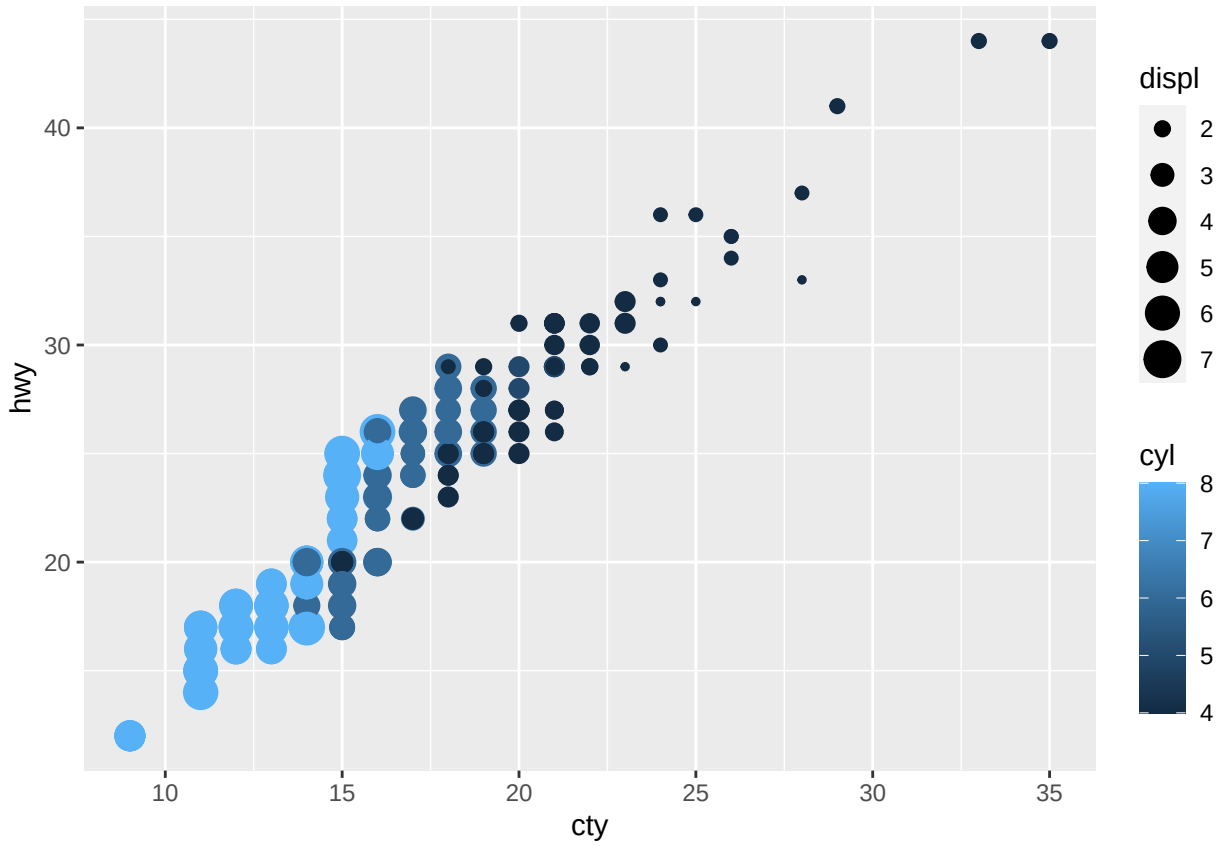


2.2.4 LE2-2d (0.5 points)

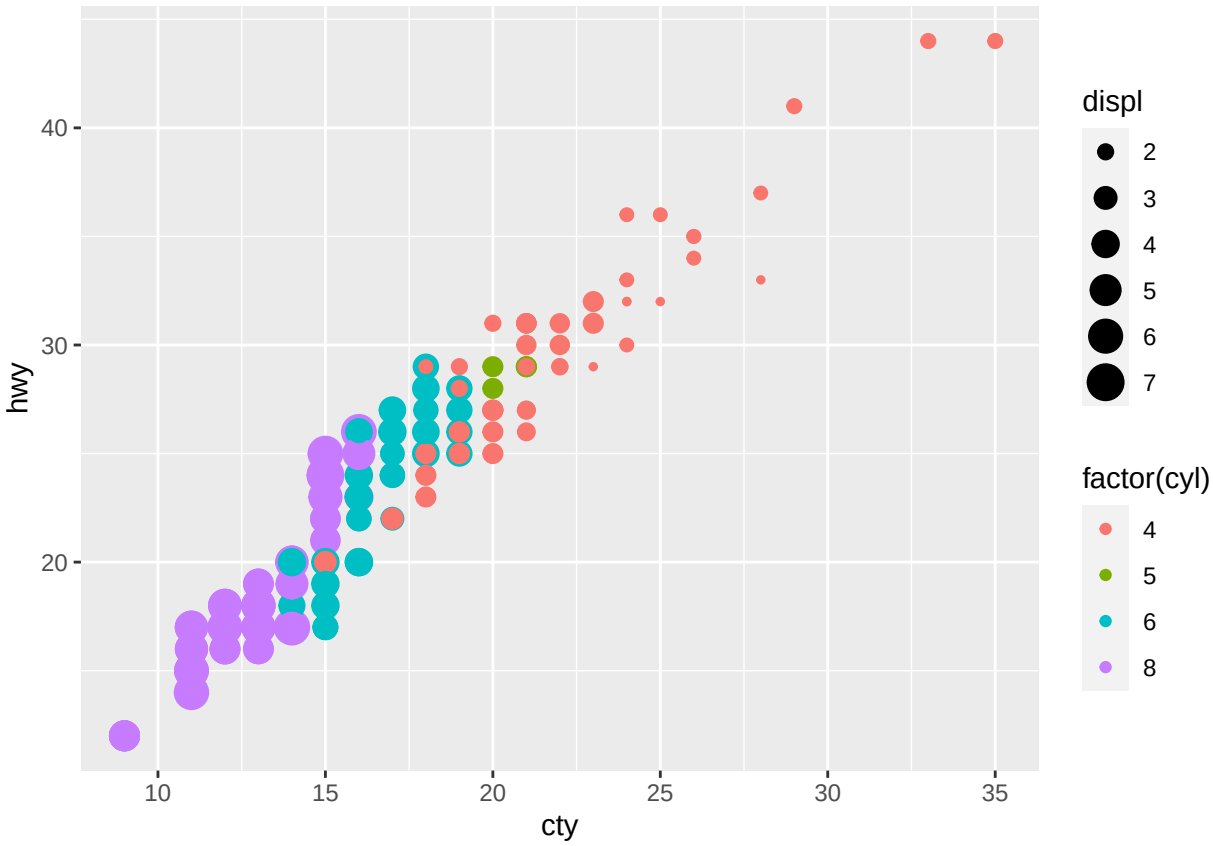
2d: Explain why these two plots look different,

- why does the color and key change between them?

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




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



ANSWER (explain why these two plots look different) -> plot 1 : Shows gradual change in the cty - indicating it as a continuous value plot 2 : treats cty as a discreet values and gives each value of cty a different color to emphasise demarcation

ANSWER (why does the color and key change between them?) -> The first one is more for a continuos cty and the second one is for a discreet cty

---

2.3 LE2-3. Text Mining of Song Lyrics: (3 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?

```
################################################ load in data
musi_inp1 = read.csv("data/billboard_lyrics_1964-2015.csv")
# View(musi_inp1)
################################################\
```

```
library(dplyr)
sr <-musi_inp1[musi_inp1$Rank == 4 & musi_inp1$Year == 1988,]
# filter(musi_inp1, Rank == 4 , Year == 1988)
sr$Lyrics
```

```
## [1]  ooh oohwere no strangers to love you know the rules and so do i a full commitments what im think
## 4633 Levels:   ... zwei drei vier one two three its easy to see but its not that i dont care so cause
```

```
##############################################

music <- tbl_df(musi_inp1)
```

```
## Warning: `tbl_df()` was deprecated in dplyr 1.0.0.
## Please use `tibble::as_tibble()` instead.
```

```
top_artist <-music %>%
  group_by(Artist) %>%
  tally()%>%
  arrange(desc(n))

top_artist$Artist[1:20]
```

```
##  [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:   the mysterians 100 proof aged in soul 10000 maniacs 10cc ... zz top
```

```
top_artist$Artist[1]
```

```
## [1] madonna
## 2473 Levels:   the mysterians 100 proof aged in soul 10000 maniacs 10cc ... zz top
```

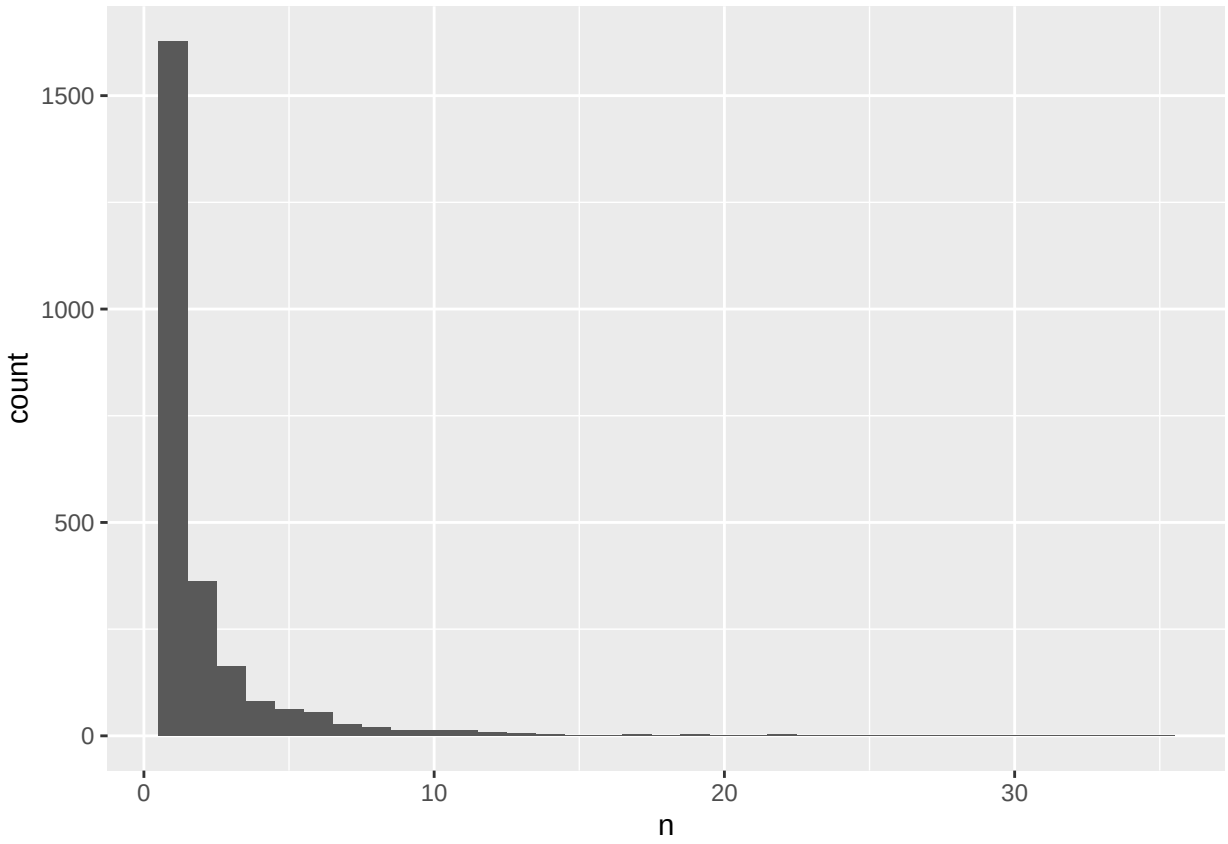
ANSWER -> The artist that has the most total top 100 hits is ***madonna***_____

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?

ANSWER -> There are many artists, more than 1500, who appear very few times in the top 100 hits. There is only 1 artist who has appeared upto 35 times.

```
hitsvsartist <- top_artist %>%
  group_by(n)%>%
  tally(name = "n2")



ggplot(top_artist, aes(n)) +
  geom_histogram(binwidth =1)
```



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)
# convert factor to character
music$Lyrics <- as.character(music$Lyrics)


mus_ej <- music %>%
  filter(Artist =="elton john" )
head(mus_ej)
```

```
## # A tibble: 6 x 6
##    Rank Song            Artist    Year Lyrics                              Source
##   <int> <fct>           <fct>    <int> <chr>                                <int>
## 1    40 rocket man      elton j~  1972 " she packed my bags last night p~      1
## 2     7 crocodile rock  elton j~  1973 " i remember when rock was young ~      1
## 3    48 daniel          elton j~  1973 " daniel is traveling tonight on ~      1
## 4     9 bennie and the~ elton j~  1974 " hey kids shake it loose togethe~      1
## 5    72 goodbye yellow~ elton j~  1974 " when are you gonna come down wh~      1
## 6    78 dont let the s~ elton j~  1974 " i cant light no more of your da~      1
```

```
mus_emnm <- music %>%
  filter(Artist =="eminem")
head(mus_emnm)
```

```
## # A tibble: 6 x 6
##    Rank Song          Artist  Year Lyrics                                  Source
##   <int> <fct>         <fct>  <int> <chr>                                    <int>
## 1    51 the real sli~ eminem  2000 " may i have your attention please ma~      1
## 2    21 without me    eminem  2002 " obie trice real name no gimmickstwo~      1
## 3    47 cleanin out ~ eminem  2002 " wheres my snare i have no snare in ~      1
## 4    63 lose yourself eminem  2002 " look if you had one shot or one opp~      1
## 5    28 lose yourself eminem  2003 " look if you had one shot or one opp~      1
## 6    89 sing for the~ eminem  2003 " these ideas are nightmares to white~      1
```

```
# break lyrics to words
ej_words <- mus_ej %>%
  unnest_tokens(output = word, input = Lyrics)

head(ej_words)
```

```
## # A tibble: 6 x 6
##    Rank Song       Artist      Year Source word
##   <int> <fct>      <fct>      <int>  <int> <chr>
## 1    40 rocket man elton john  1972      1 she
## 2    40 rocket man elton john  1972      1 packed
## 3    40 rocket man elton john  1972      1 my
## 4    40 rocket man elton john  1972      1 bags
## 5    40 rocket man elton john  1972      1 last
## 6    40 rocket man elton john  1972      1 night
```

```
emnm_words <- mus_emnm %>%
  unnest_tokens(output = word, input = Lyrics)

head(emnm_words)
```

```
## # A tibble: 6 x 6
##    Rank Song                Artist  Year Source word
##   <int> <fct>               <fct>  <int>  <int> <chr>
## 1    51 the real slim shady eminem  2000      1 may
## 2    51 the real slim shady eminem  2000      1 i
## 3    51 the real slim shady eminem  2000      1 have
## 4    51 the real slim shady eminem  2000      1 your
## 5    51 the real slim shady eminem  2000      1 attention
## 6    51 the real slim shady eminem  2000      1 please
```

```
# top 10 words for each artist
ej_top_10 <- ej_words %>%
  group_by(word)%>%
  tally(name = "word_count")%>%
  arrange(desc(word_count))

print("Elton john uses thesa a lot")
```

```
## [1] "Elton john uses thesa a lot"
```

```
ej_top_10$word[1:10]
```

```
##  [1] "the"  "you"  "i"    "and"  "in"   "to"   "a"    "oh"   "your" "me"
```

```
em_top_10 <- emnm_words %>%
  group_by(word)%>%
  tally(name = "word_count")%>%
  arrange(desc(word_count))

print("eminem uses thesa a lot")
```

```
## [1] "eminem uses thesa a lot"
```

```
em_top_10$word[1:10]
```

```
##  [1] "you"  "the"  "i"    "to"   "and"  "a"    "it"   "me"   "im"   "just"
```

- 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

```
data("stop_words")

usable_ej_10 <- ej_top_10 %>%
  anti_join(stop_words, by = "word")

print("elton john uses these words most")
```

```
## [1] "elton john uses these words most"
```

```
usable_ej_10$word[1:10]
```

```
##  [1] "love"     "dont"     "life"     "im"       "time"     "sad"
##  [7] "lucy"     "saved"    "sky"      "diamonds"
```

```
print("eminem uses these words most")
```

```
## [1] "eminem uses these words most"
```

```
usable_em_10 <- em_top_10%>%
  anti_join(stop_words, by = "word")

usable_em_10$word[1:10]
```

```
##  [1] "im"    "ah"    "dont"  "baby"  "stand" "shady" "girl"  "hes"   "sing"
## [10] "slim"
```

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
```

```r
usable_ej <- ej_words %>%
  anti_join(stop_words, by = "word")
usable_em <- emnm_words %>%
  anti_join(stop_words, by = "word")

wordcloud(words = usable_ej$word,  min.freq = 1,           max.words=200, random.order=FALSE, rot.per=0
```

```
## Loading required namespace: tm
```

```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
## drops documents
```

```
## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,
## tm::stopwords())): transformation drops documents
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## knocking could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## gentle could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## counting could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## line could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## heart could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## whispered could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## buy could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## blues could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## night could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## holding could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## bennie could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## boots could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## head could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## grow could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## suddenly could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## ohlucy could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## sweet could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## darling could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## gardener could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## pieces could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## grace could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## horses could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## collide could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## shadows could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## wouldnt could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## aloneand could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## shocking could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## friday could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## wore could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## dresses could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## sightla could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## change could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## bbbbennie could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## makes could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## friends could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## future could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## times could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## breathe could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## people could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## kaleidoscope could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## hooks could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## altarbound could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## butterflies could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## worldhe could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## lovemama could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## stepped could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## shiny could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## grew could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## calling could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## survivor could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## feeling could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## spent could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## children could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## thunder could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## cap could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## woah could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## jean could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## knowing could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## vagabonds could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## lives could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## footsteps could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## greenest could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## hills could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = usable_ej$word, min.freq = 1, max.words = 200, :
## candles could not be fit on page. It will not be plotted.
```


```
wordcloud(words = ej_words$word,  min.freq = 1,           max.words=200, random.order=FALSE, rot.per=0.3

## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
## drops documents

## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,
## tm::stopwords())): transformation drops documents

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## around could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## laughing could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
```

```
## soldiers could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## burned could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## legend could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## high could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## burning could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## suzie could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## something could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## nights could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## goodbye could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## pain could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## rolling could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## babys could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## knocking could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## gentle could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## counting could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## englands could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## brings could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## place could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## theres could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## knew could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## really could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## though could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## known could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## yellow could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## road could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## sunset could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## others could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## living could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## whispered could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## world could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## anything could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, : buy
## could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## wild could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## night could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## died could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## brick could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## finally could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## behind could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## choose could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## suddenly could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## ohlucy could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## room could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## hardest could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## darling could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## dream could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## side could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## gardener could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, : now
## could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## pieces could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## second could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## feels could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## grace could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## fields could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## block could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = ej_words$word, min.freq = 1, max.words = 200, :
## wouldnt could not be fit on page. It will not be plotted.
```

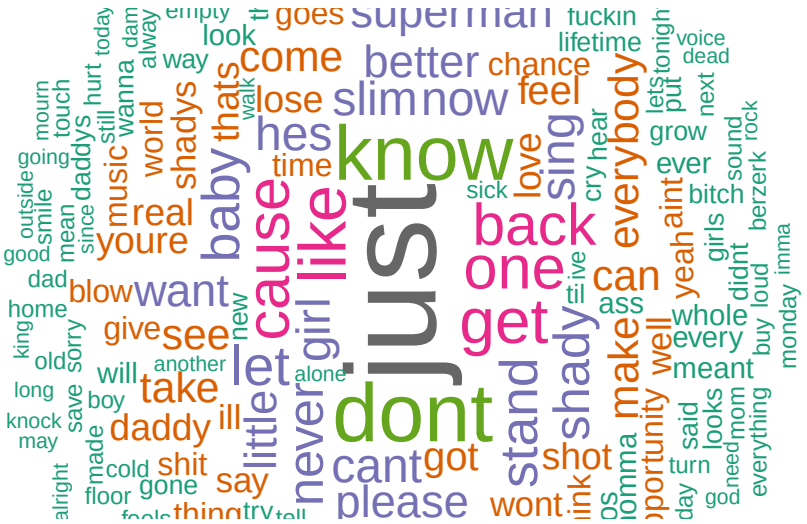

```
wordcloud(words = usable_em$word,  min.freq = 1,           max.words=200, random.order=FALSE, rot.per=0
```

```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
## drops documents

## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,
## tm::stopwords())): transformation drops documents
```


```
wordcloud(words = emnm_words$word,  min.freq = 1,           max.words=200, random.order=FALSE, rot.per=
```

```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
## drops documents
```

```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
## drops documents
```




```
# wordcloud(words = usable_em_10$word, max.words = 10 )
```

ANSWER (compare and contrast the word clouds) -> Both Eminem and Elton john use “don’t” a lot.

ANSWER (did stop_words work well to remove non-meaningful words?) -> The stop words well to remove non- meaningful words. Even though word cloud tries to remove non- meanigful words, in the case of eminem, if we use his raw words without stop words filter a lot of non-meaningful words like “get, he’s, yeah” appear

2.3.5 LE2-3e (1 point)

- The tidytext package gives us the ability to run sentiment analysis as well
- There are multiple sentiment methods available but let’s just use positive and negative sentiment

```
# library(tidytxt)
bing <- get_sentiments("bing")
head(bing)
```

```
## # A tibble: 6 x 2
##   word       sentiment
##   <chr>      <chr>
## 1 2-faces    negative
## 2 abnormal   negative
## 3 abolish    negative
## 4 abominable negative
## 5 abominably negative
## 6 abominate  negative
```

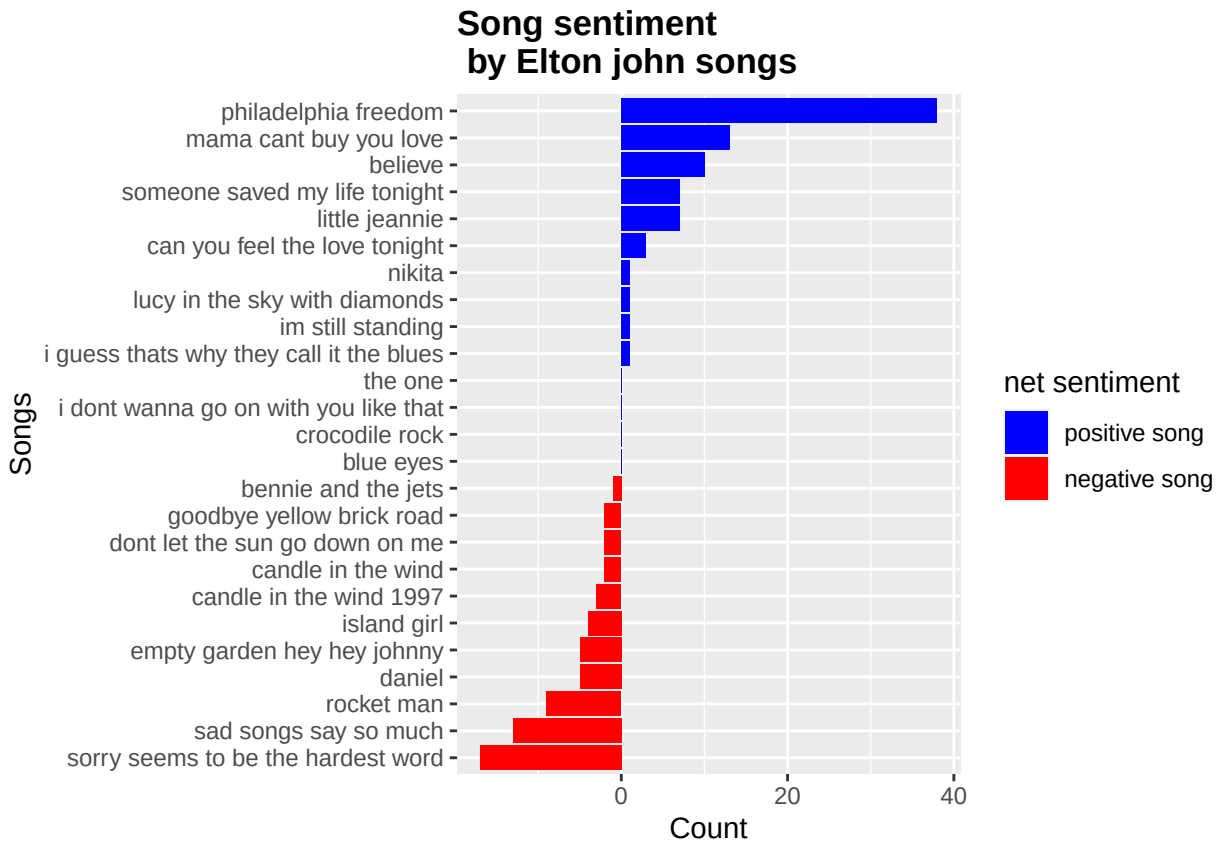
- What is the bing dataframe?

ANSWER (definition of bing dataframe) -> The bing dataframe : is a data set of many words witha a sentiment attached to each word.

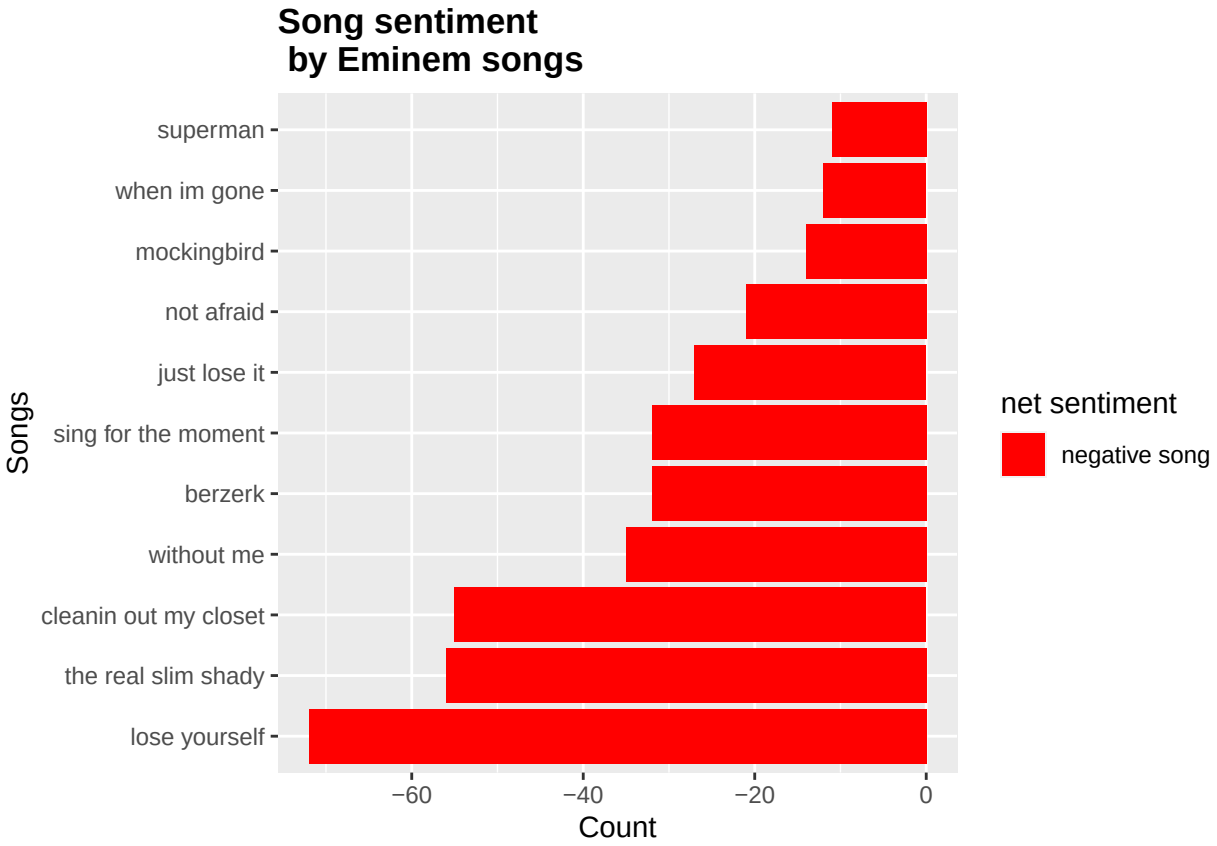
- Pull out the "positive" and "negative" words for each artist (hint: dplyr inner_join())
- For each song per artist, determine the next positive or negative sentiment, there are several ways to do this - the most straightforward being assign a 1 to a positive lyric and -1 to a negtive lyric then sum them together for each song
- Make a bar plot of the net sentiment of each song for each artist, make these plots high quality as well
  - Properly name the axes and title each plot
  - Color by whether the song is overall positive or negative
  - rotate the x-axis names so long song titles are legible
  - Arrange the bars in ascending or descending order

```
## words used by each artist, and assigning it with a sentiment using bing
usable_ej_sent <-  merge(x = usable_ej, y = bing, by = "word")
usable_ej_sent <- usable_ej_sent %>%
  mutate(sen_value = if_else( usable_ej_sent$sentiment == "negative", -1 , 1, 0))

usable_ej_sent <-usable_ej_sent %>%
  group_by(Song)%>%
  summarise(net_Sen = sum(sen_value))

usable_ej_sent <- usable_ej_sent %>%
  arrange(desc(net_Sen))

usable_em_sent <-  merge(x = usable_em, y = bing, by = "word")
usable_em_sent <- usable_em_sent %>%
  mutate(sen_value = if_else( usable_em_sent$sentiment == "negative", -1 , 1, 0))

usable_em_sent <- usable_em_sent %>%
  group_by(Song)%>%
   summarise(net_Sen = sum(sen_value))


p1 <- ggplot(usable_ej_sent, aes( x = reorder(Song, net_Sen), y = net_Sen, fill = net_Sen < 0)) +
  geom_bar(stat = "identity") + ggtitle("Song sentiment \n by Elton john songs") + theme(plot.title=ele
   xlab("Songs") + ylab("Count")+
    scale_fill_manual(values = c("blue", "red"),
                      labels=c('positive song','negative song'))+
  labs(fill='net sentiment')+
```

```
  coord_flip()
p1
```




```
p2 <- ggplot(usable_em_sent, aes( x = reorder(Song, net_Sen), y = net_Sen, fill = net_Sen < 0)) +
  geom_bar(stat = "identity") + ggtitle("Song sentiment \n by Eminem songs") +   theme(plot.title=eleme
   xlab("Songs") + ylab("Count")+
    scale_fill_manual(values = c( "red"),
                      labels= c('negative song')) +
  labs(fill='net sentiment')+
  coord_flip()
p2
```



- What differences do you notice between the 2?

ANSWER -> Elton John has songs that have both positive and negative sentiments. wheras Eminem has songs that are mostly negative.

- Based on what you might know about some of these songs, do you think the sentiment analysis gives a good indication of the positive of negative tone for each song or not?

ANSWER -> It tallies with the general trend of most songs from Eminem but there are some songs like Lose yourself - which are widely known as motivational song, but due to the wording lose, it might be perceived as negative by the algorithem.

- If you're interested, play around with this data set, see if your favorite artist is in here and see if you can find anything else

Just a reminder: if you have questions, please ask them on Slack.

2.3.5.1 Links

http://www.r-project.org

http://rmarkdown.rstudio.com/