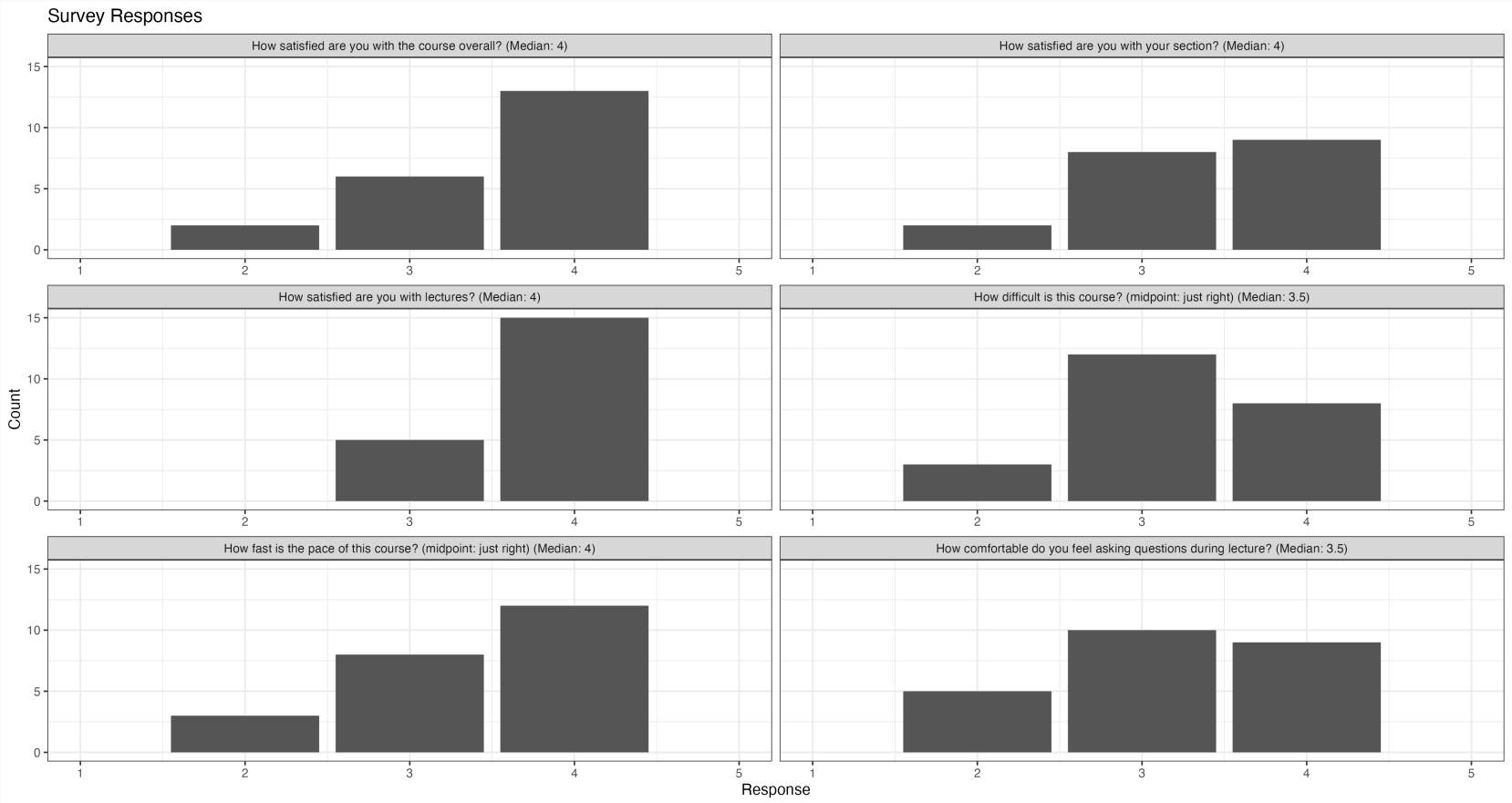
# Lecture 7: Base R Plots

Ingmar Sturm
UCSB
2024-07-03

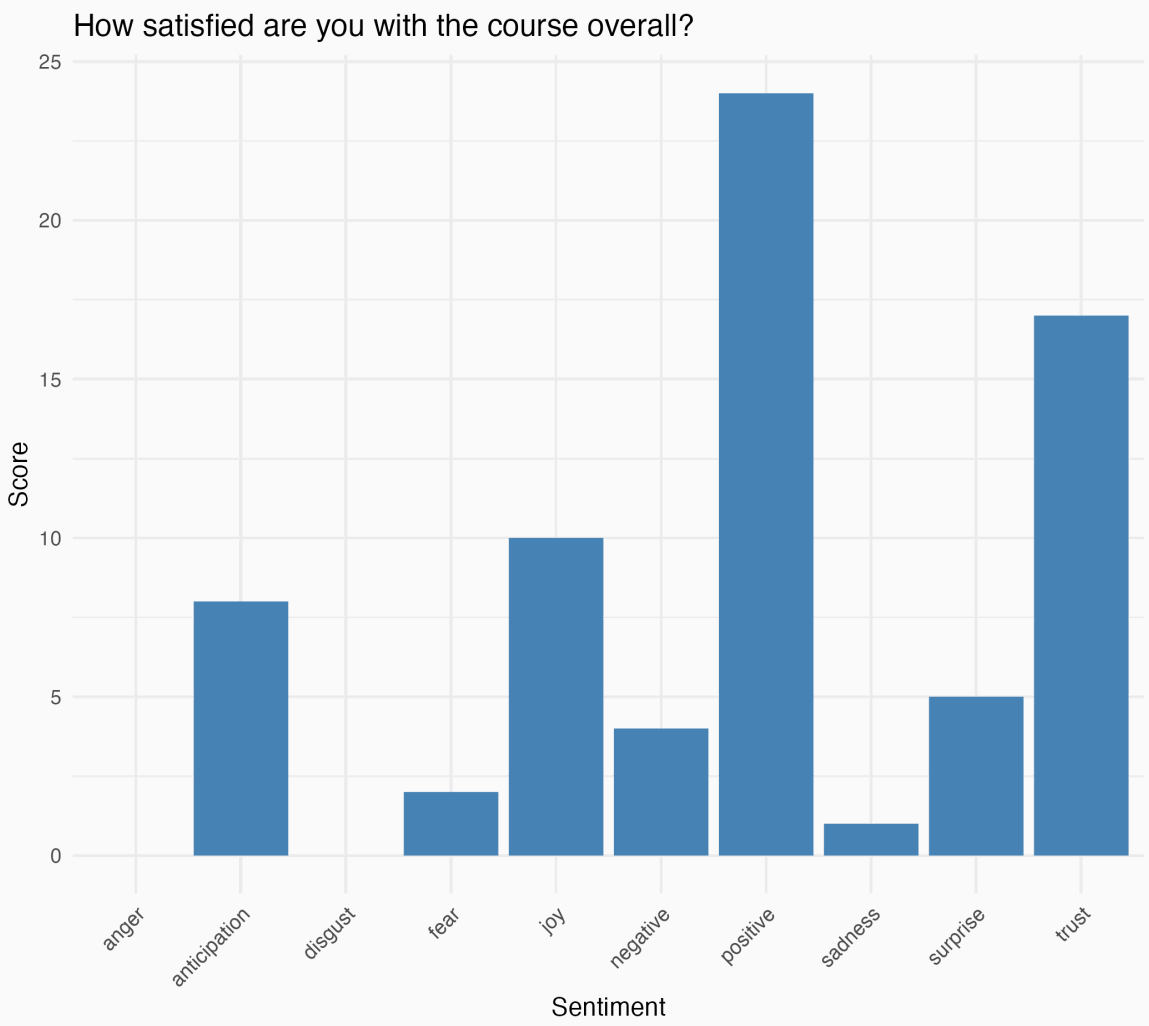
Special thanks to Robin Liu for select course content used with permission.

# Analyzing Poll Results (like a data scientist)

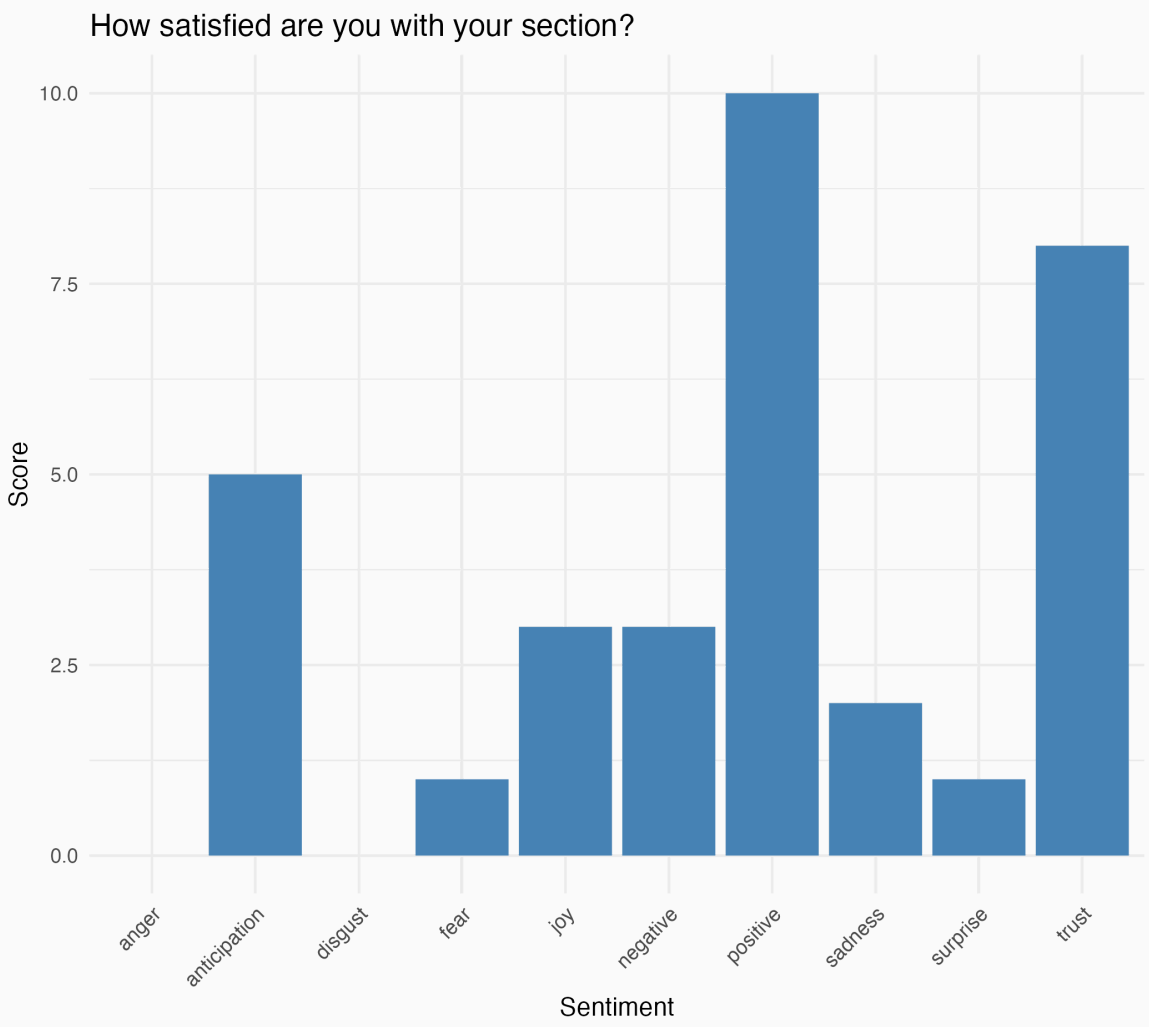
# Numerical Variables



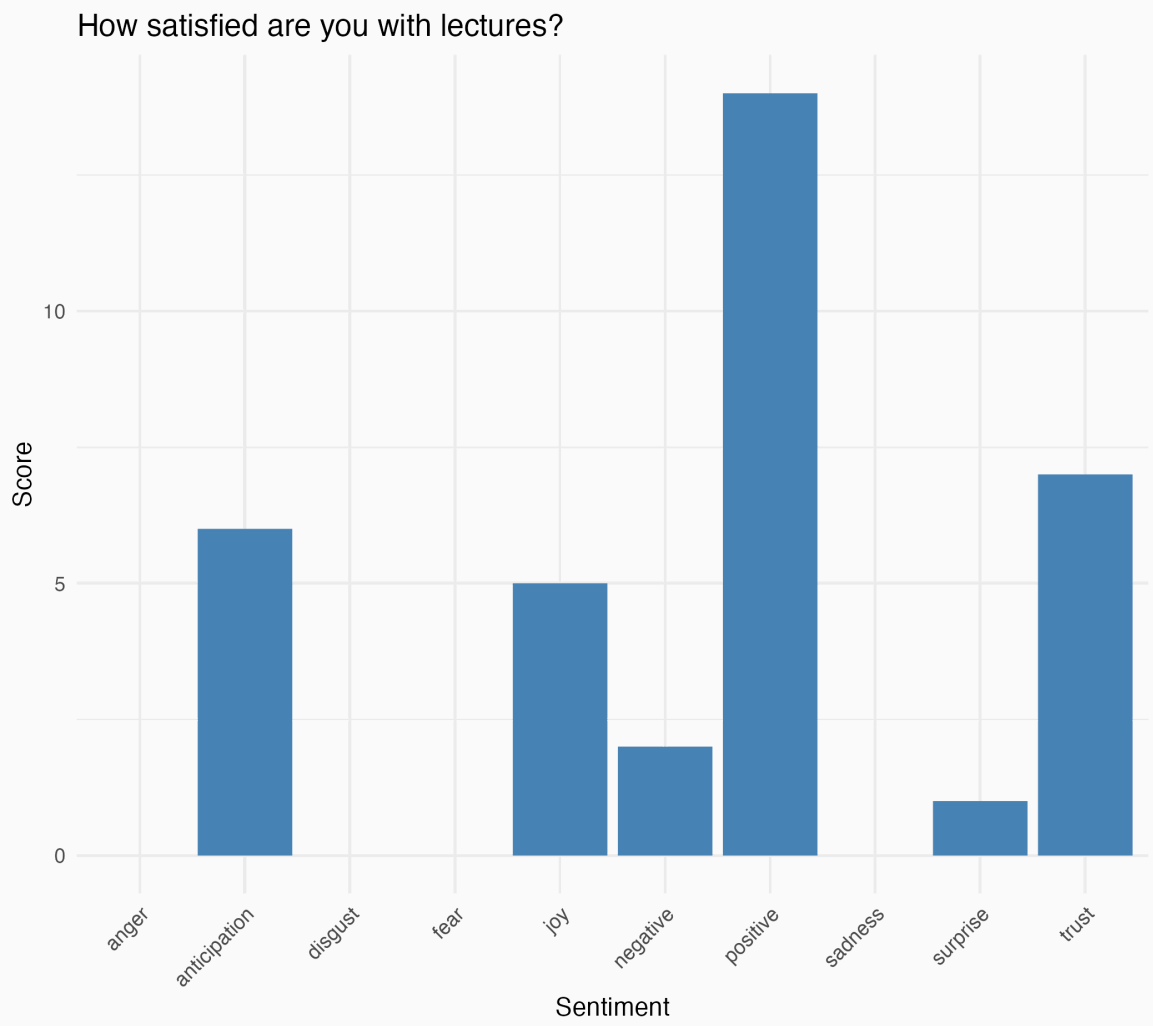
# Sentiment Analysis 1/4



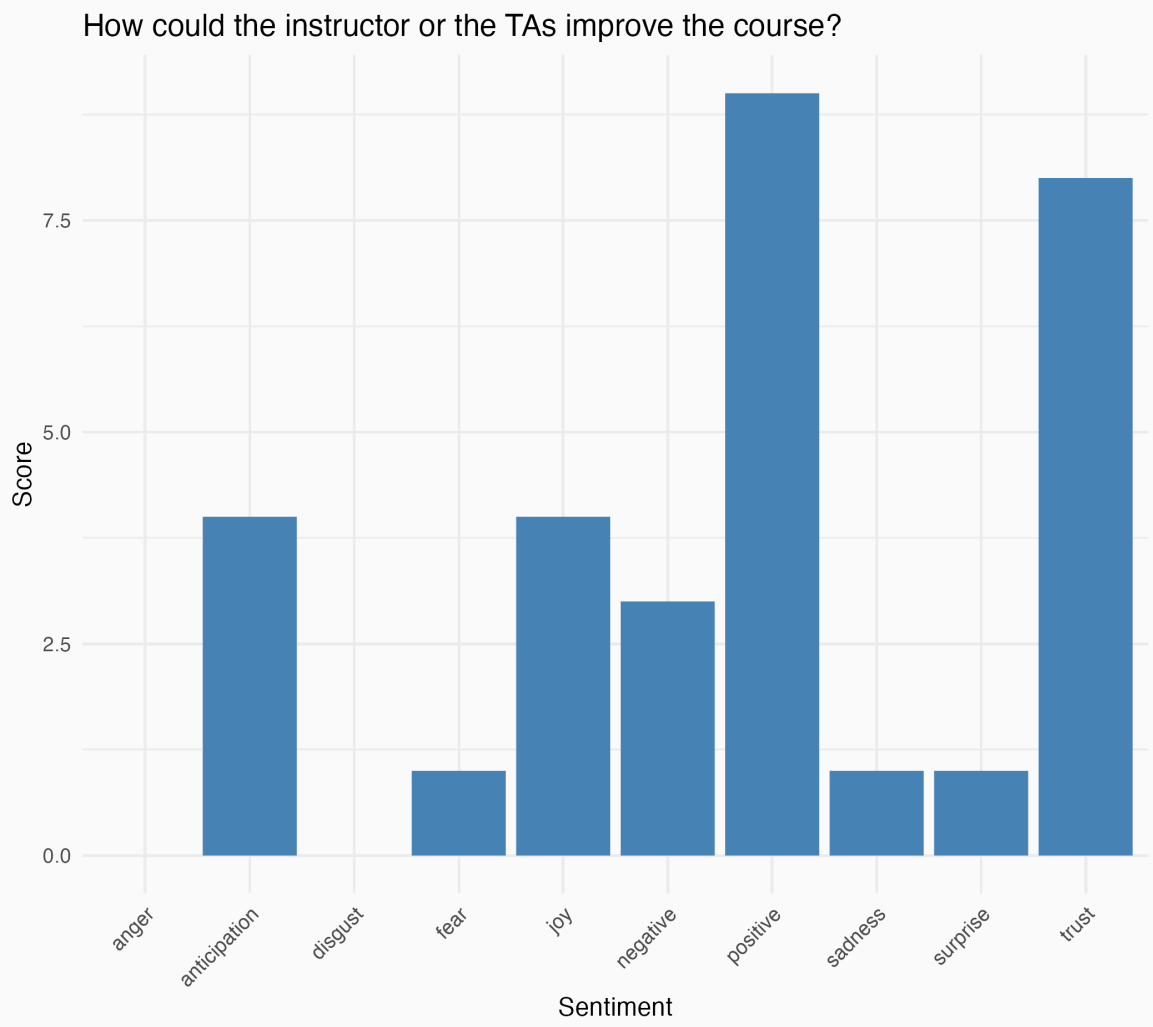
# Sentiment Analysis 2/4



# Sentiment Analysis 3/4



# Sentiment Analysis 4/4

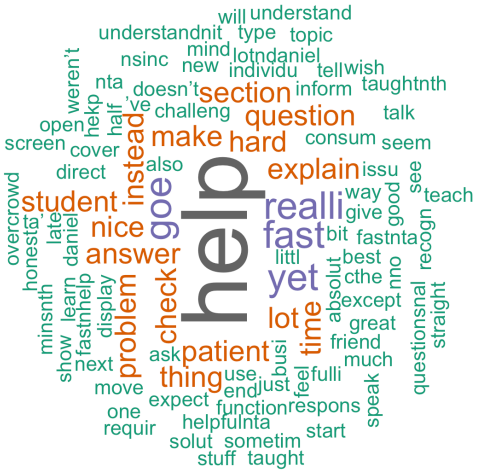


# Word Cloud 1/4

How satisfied are you with the course overall?

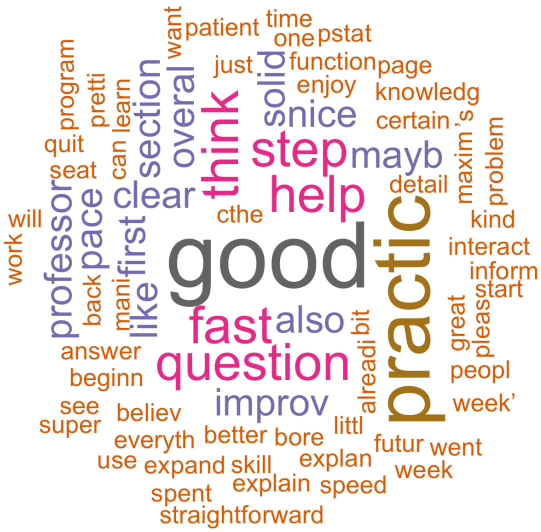
# Word Cloud 2/4

How satisfied are you with your section?



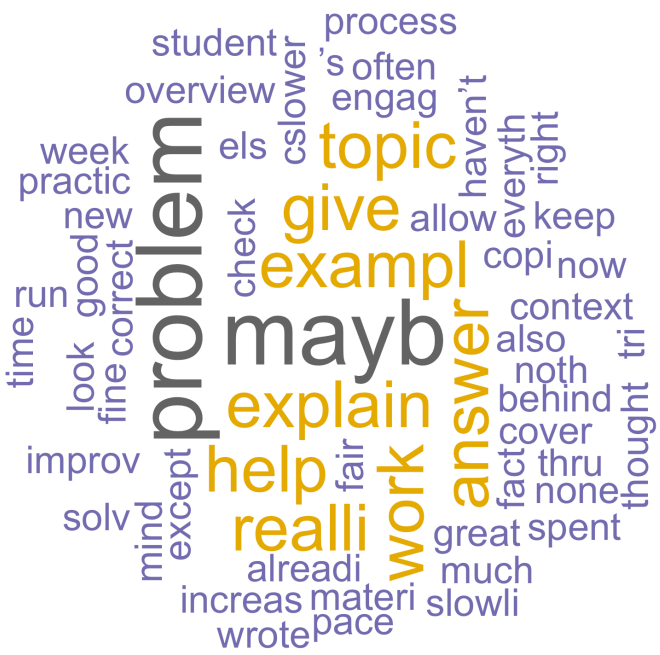
# Word Cloud 3/4

How satisfied are you with lectures?



# Word Cloud 4/4

How could the instructor or the TAs improve the course?



# Now: Plotting in Base R

# airquality

We will use the `airquality` data frame in the library `datasets`

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(datasets)
```

# airquality

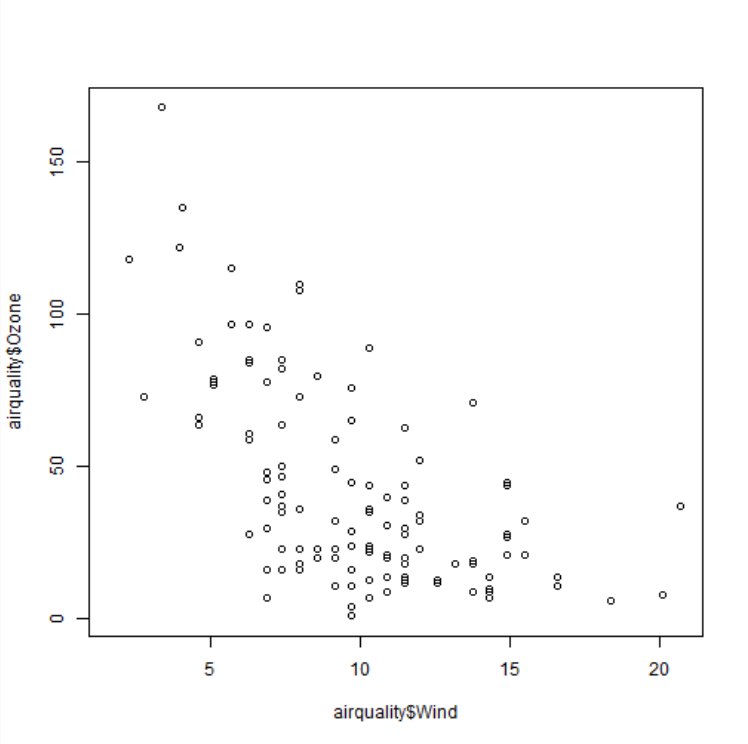
`?airquality` will give you the metadata in the help window.

```
Description
Daily air quality measurements in New York, May to September 1973.

Format
A data frame with 153 observations on 6 variables.

[,1]    Ozone    numeric    Ozone (ppb)
[,2]    Solar.R    numeric    Solar R (lang)
[,3]    Wind    numeric    Wind (mph)
[,4]    Temp    numeric    Temperature (degrees F)
[,5]    Month    numeric    Month (1--12)
[,6]    Day    numeric    Day of month (1--31)
```

# Base R plots

Basic plotting is easy; just call the relevant functions.

```
plot(airquality$Wind, airquality$Ozone)
```

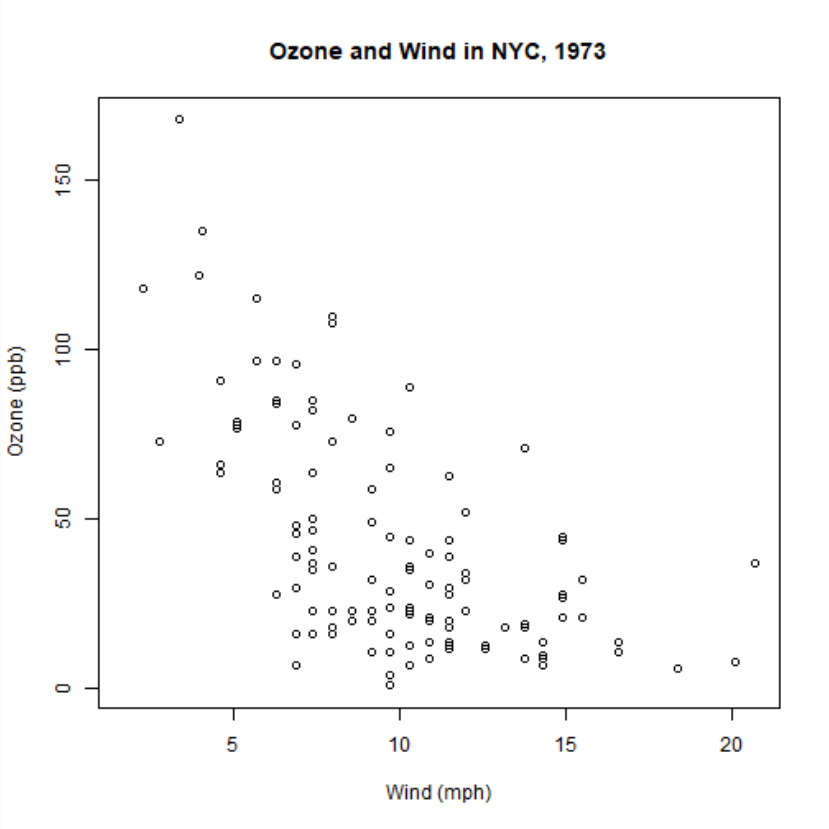



But we should always add a meaningful title and axis labels...

# Base R plots

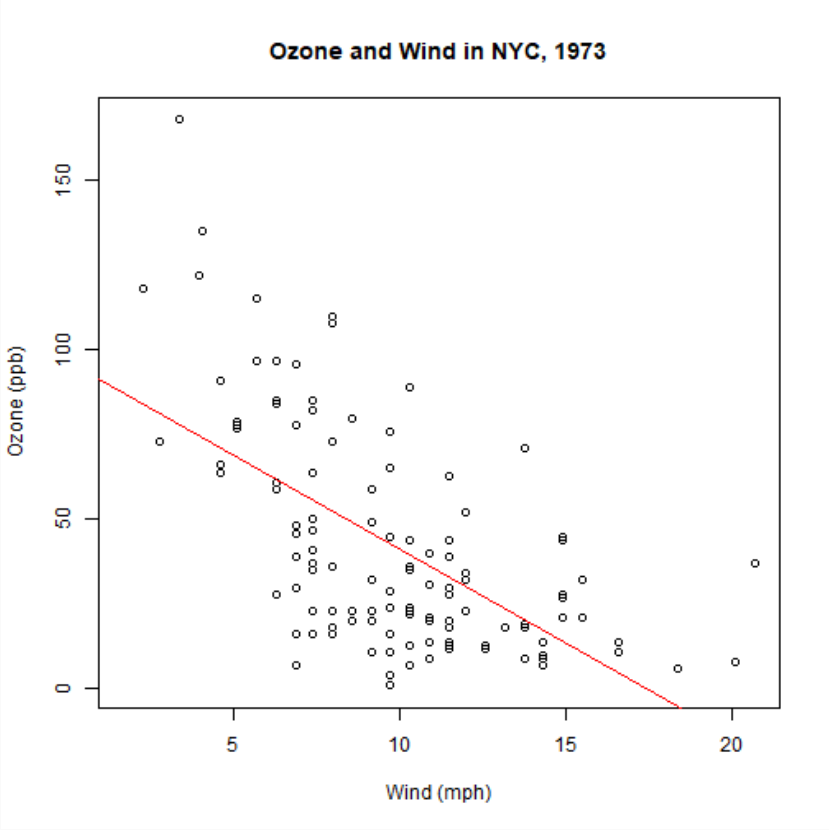
Basic plotting is easy; just call the relevant functions.

```
plot(airquality$Wind, airquality$Ozone, main="Ozone and Wind in NYC, 1973",
     xlab="Wind (mph)", ylab="Ozone (ppb)")
```



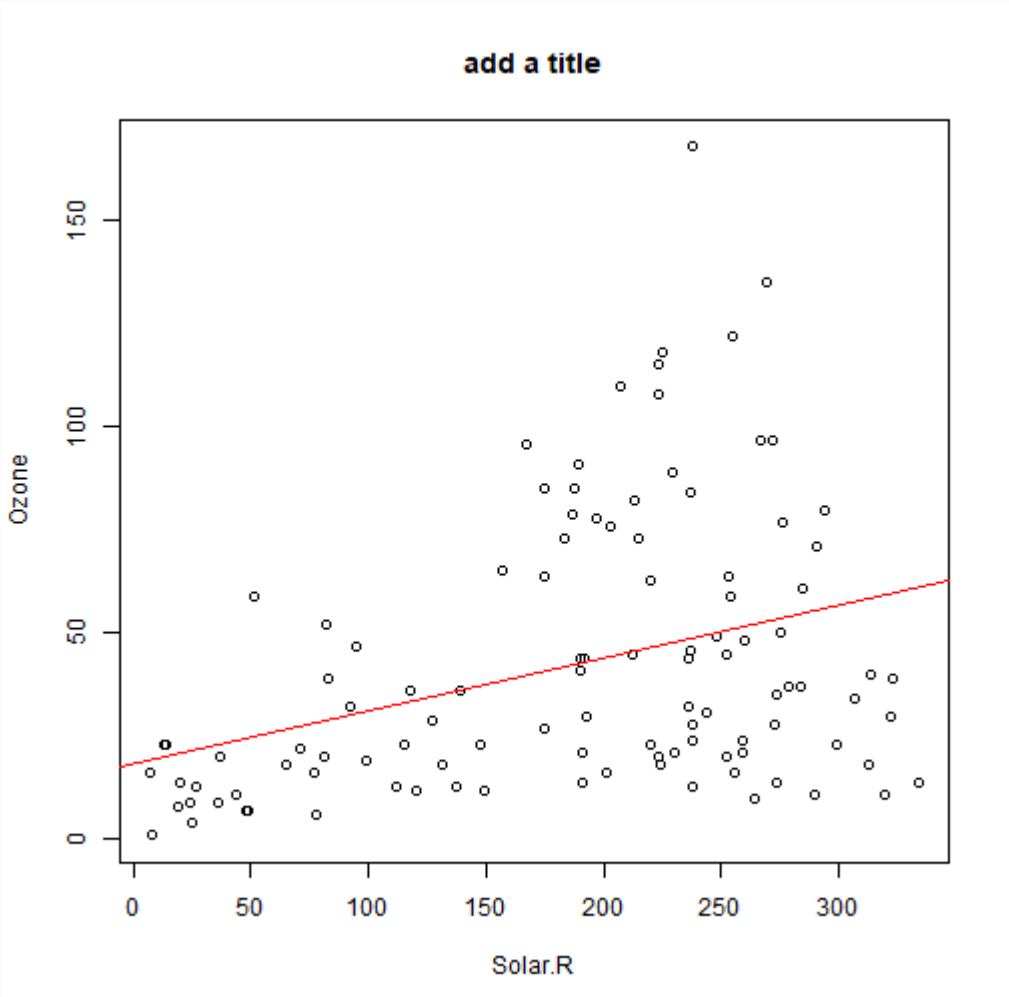
# Adding a trendline

```
plot(airquality$Wind, airquality$Ozone, main="Ozone and Wind in NYC, 1973",
     xlab="Wind (mph)", ylab="Ozone (ppb)")
air_trend <- lm(airquality$Ozone ~ airquality$Wind)
abline(air_trend, col = "red")
```



# Your turn

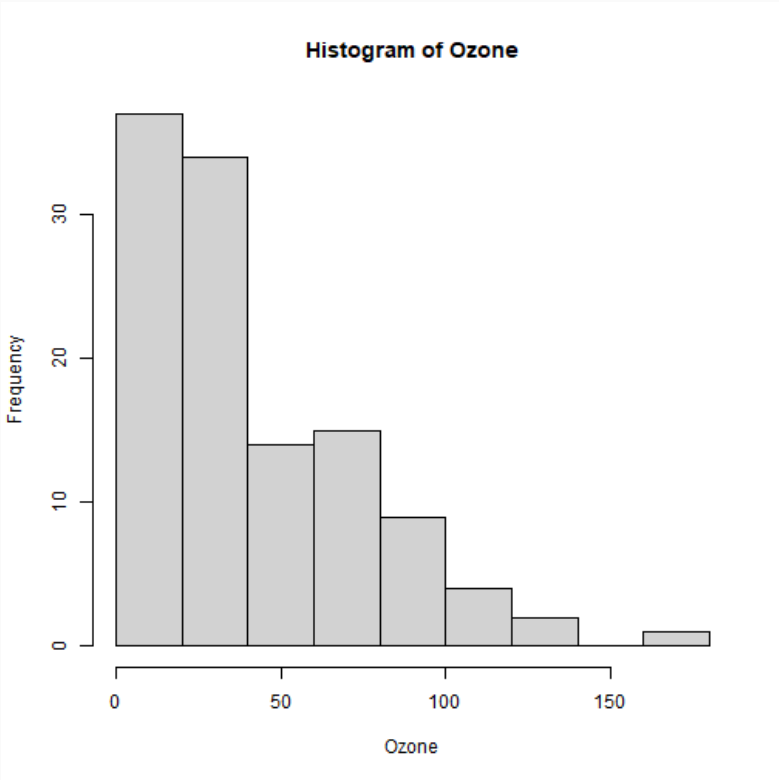
Create a scatter plot of Ozone vs. Solar Radiation with a trendline. Add a meaningful title and axis labels to the plot.



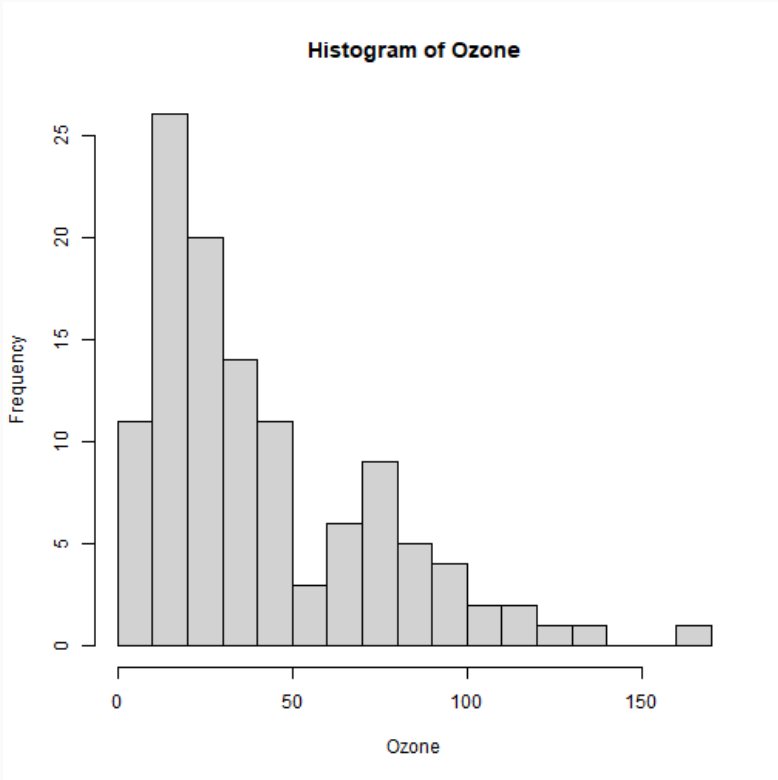

02:00

# Histograms

```
hist(airquality$Ozone,
     main = "Histogram of Ozone",
     xlab = "Ozone")
```



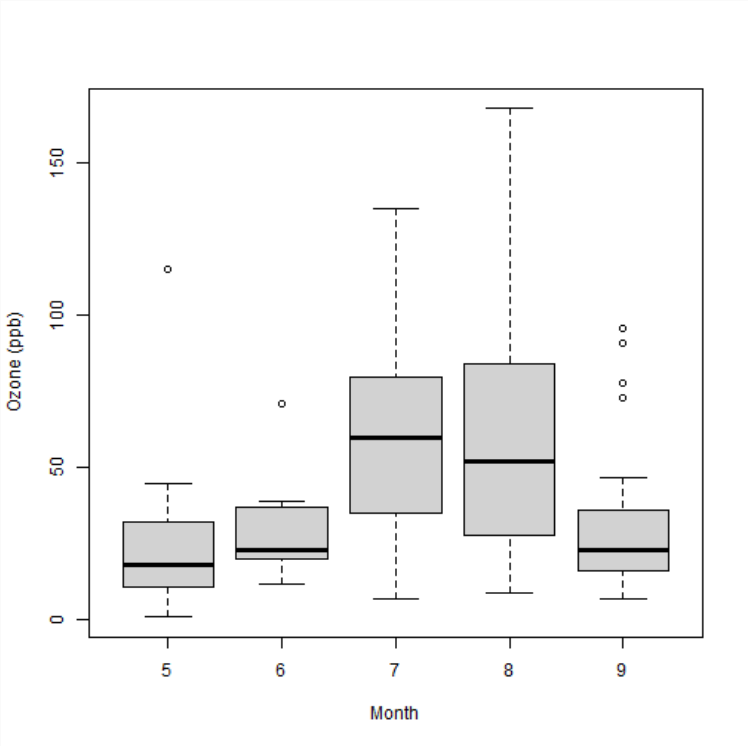

```
hist(airquality$Ozone, breaks = 20,
     main = "Histogram of Ozone",
     xlab = "Ozone")
```



# Boxplots

Boxplots use the `y ~ x` notation. In R, this construct is called a `formula`

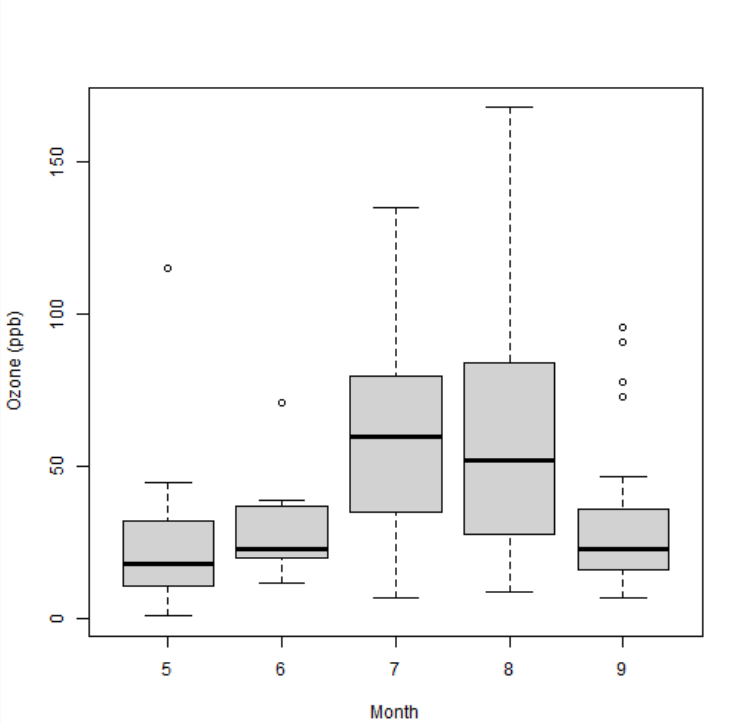
```
boxplot(Ozone ~ Month, airquality, xlab = "Month", ylab = "Ozone (ppb)")
```




Median, percentiles, quartiles, IQR, outliers

# Formulas

```
boxplot(Ozone ~ Month, airquality, xlab = "Month", ylab = "Ozone (ppb)")
```



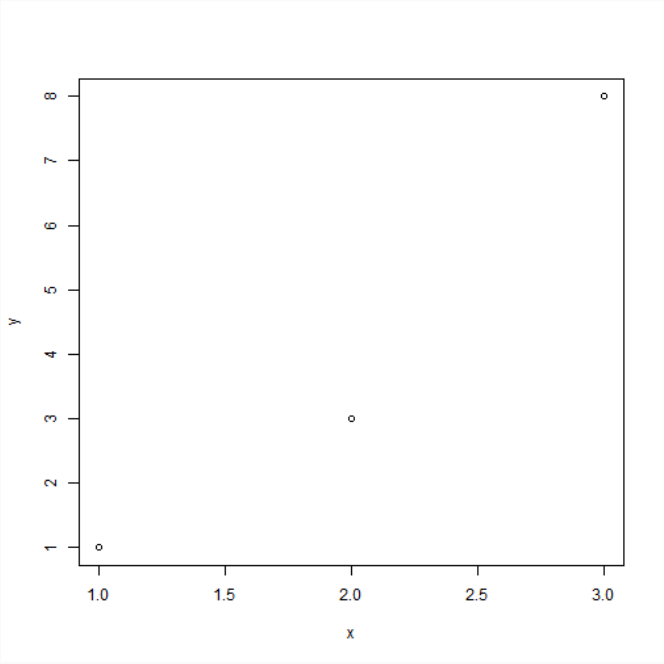

```
class(y ~ x)
```

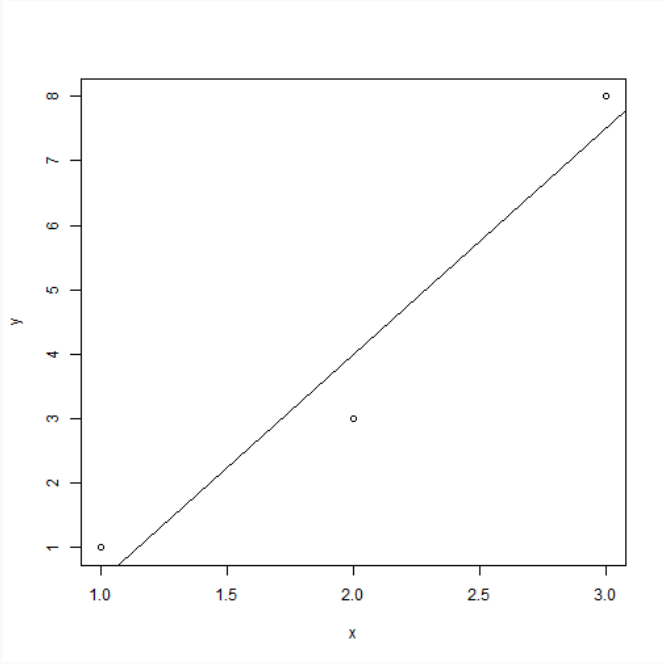
```
## [1] "formula"
```

# Building plots

It is common to first create a plot and then layer stuff on top.

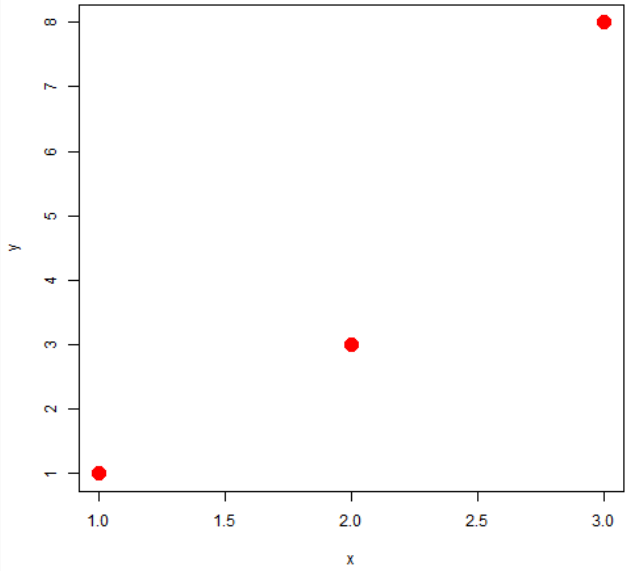
```
x <- 1:3
y <- c(1, 3, 8)
plot(x, y) # Creates a new plot
lmout <- lm(y ~ x)
abline(lmout) # adds a line "fitting" the points
```

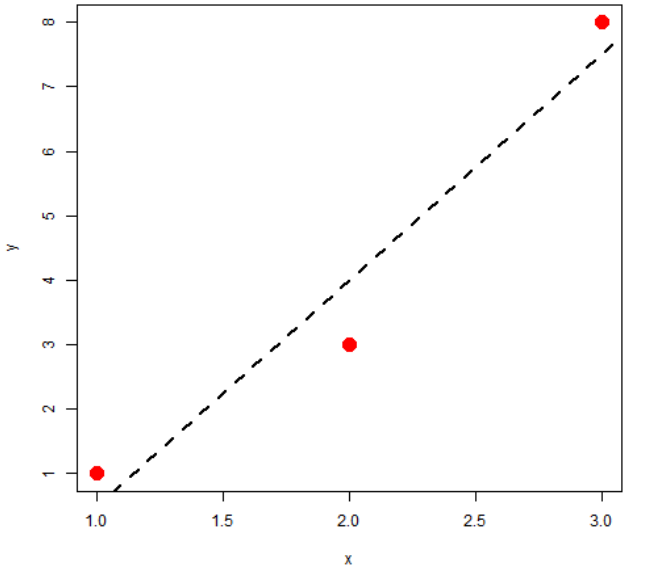


# Plot parameters

```r
x <- 1:3
y <- c(1, 3, 8)
plot(x, y, cex=2, col="red", pch=19) # Creates a new plot
lmout <- lm(y ~ x)
abline(lmout, lty=2, lwd=2) # adds a line "fitting" the points
```

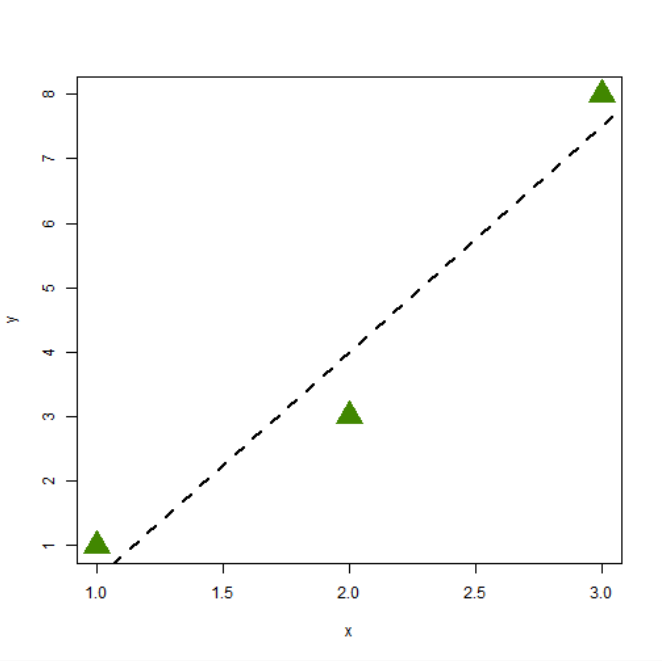


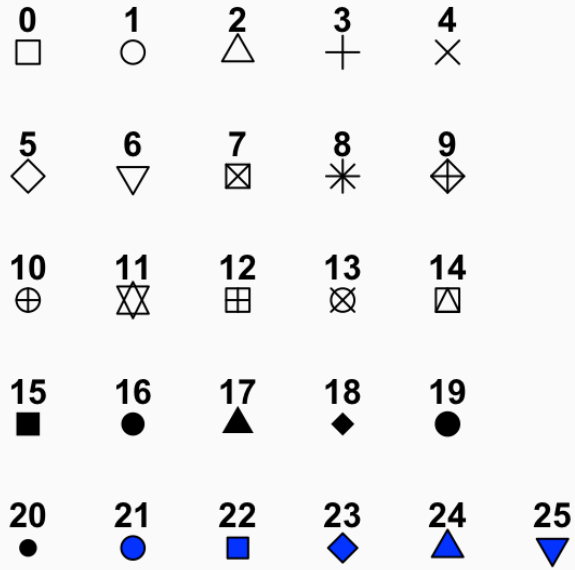



# Plot parameters

Plotting is an art and you should use Google and `?par` to get a plot to look nice.

```
x <- 1:3
y <- c(1, 3, 8)
plot(x, y, cex=3, col="chartreuse4", pch=17) # Creates a new plot
lmout <- lm(y ~ x)
abline(lmout, lty=2, lwd=2) # adds a line "fitting" the points
```

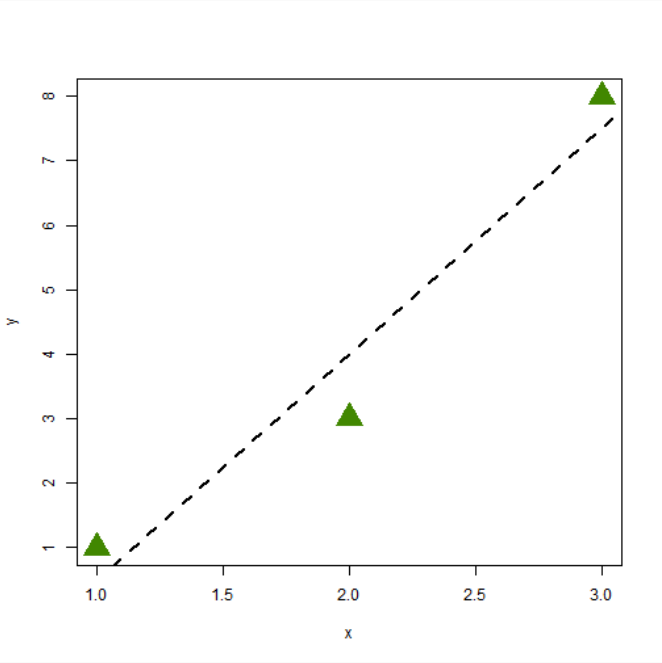


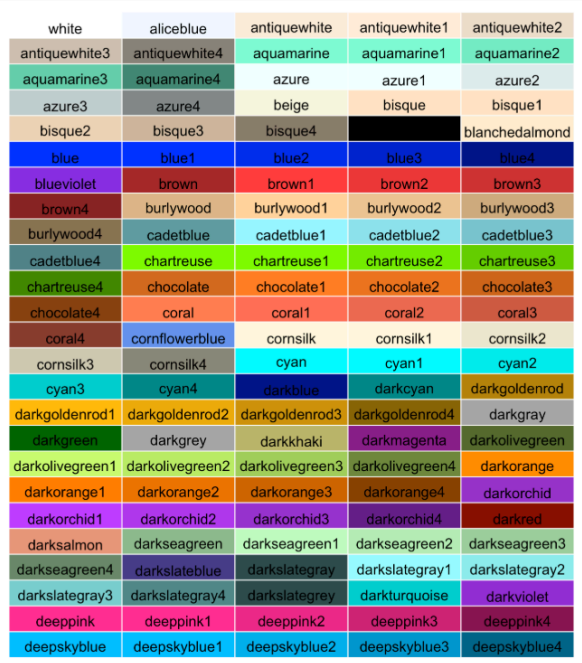



# Plot parameters

Plotting is an art and you should use Google and `?par` to get a plot to look nice.

```
x <- 1:3
y <- c(1, 3, 8)
plot(x, y, cex=3, col="chartreuse4", pch=17) # Creates a new plot
lmout <- lm(y ~ x)
abline(lmout, lty=2, lwd=2) # adds a line "fitting" the points
```

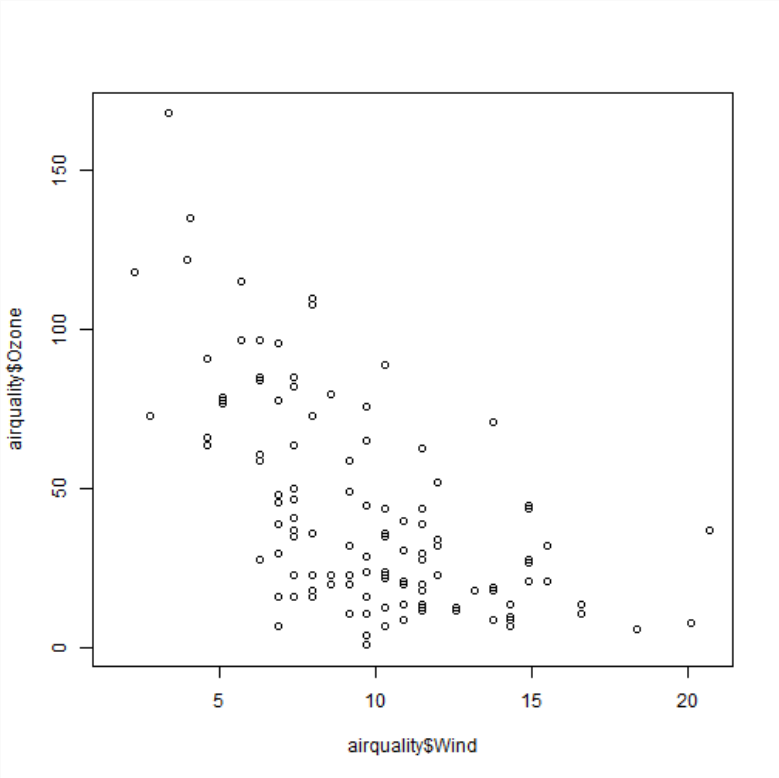


# Building plots

We see a negative trend with wind and ozone.

```
plot(airquality$Wind, airquality$Ozone)
```




A flat screen can only show two variables.

unless...? 😳

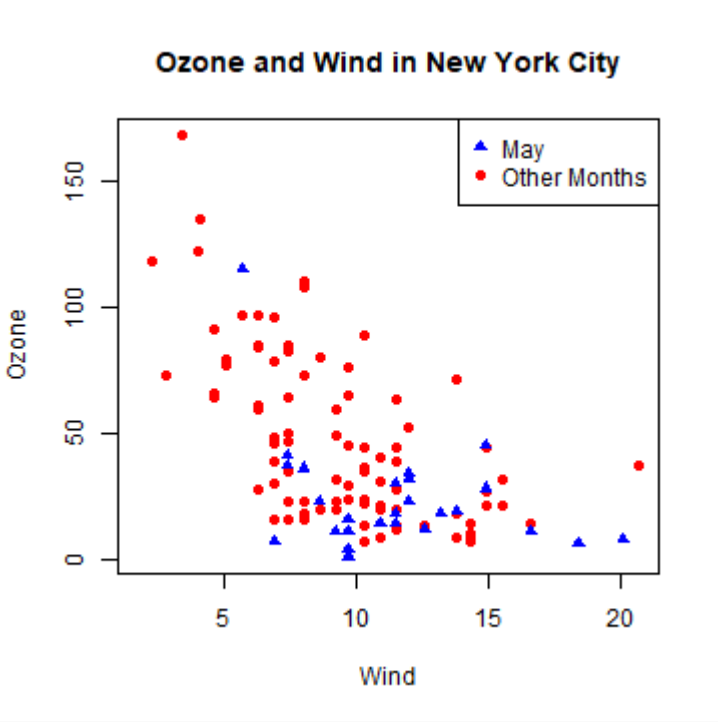
We can add another dimension of information by employing *shape* and *color*.

# Layering Plots Demo 1

It is common for complex plots to start with an *empty* plot and layer stuff on top.

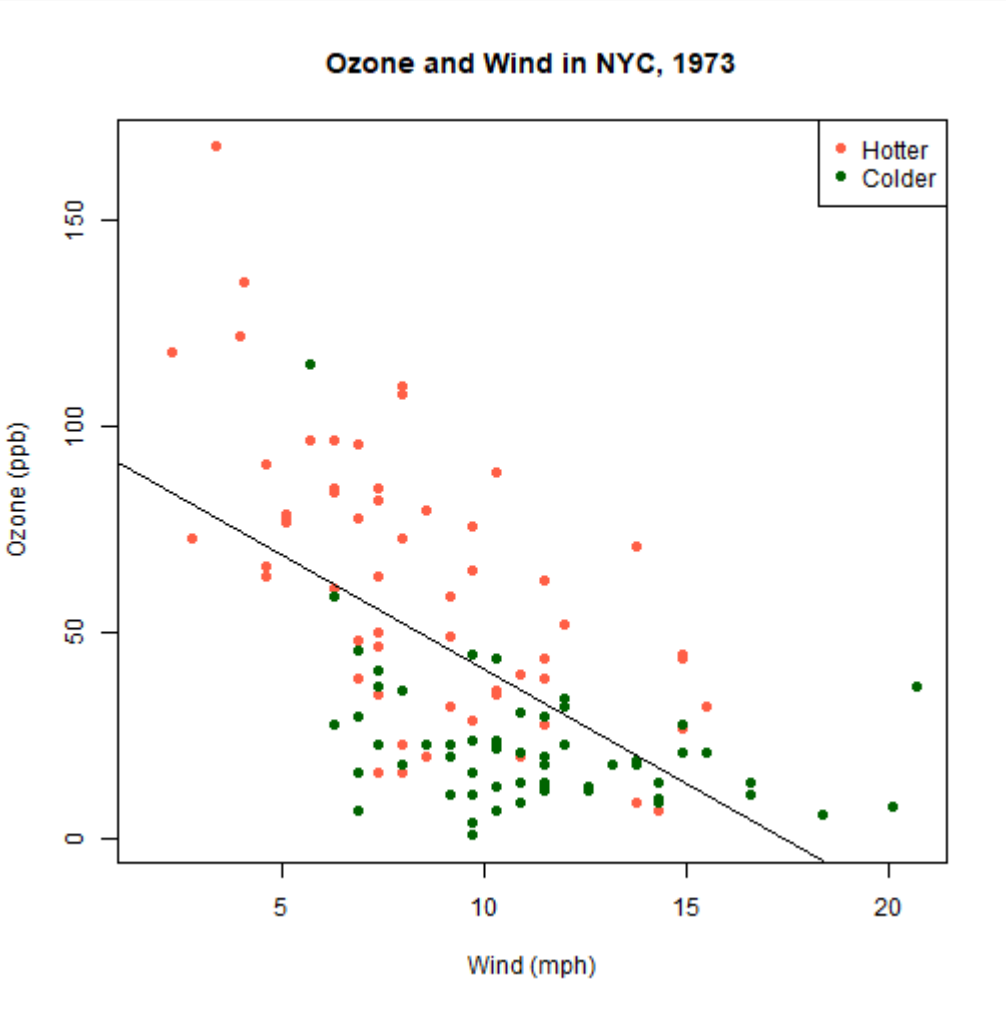
# Layering Plots Demo 1

```
with(airquality, plot(Wind, Ozone, main = "Ozone and Wind in New York City", type = "n"))
with(filter(airquality, Month != 5), points(Wind, Ozone, col="red", pch=19))
with(filter(airquality, Month == 5), points(Wind, Ozone, col="blue", pch=17))
legend("topright", pch = c(17, 19), col = c("blue", "red"),
       legend = c("May", "Other Months"))
```



# Layering Plots Demo 2

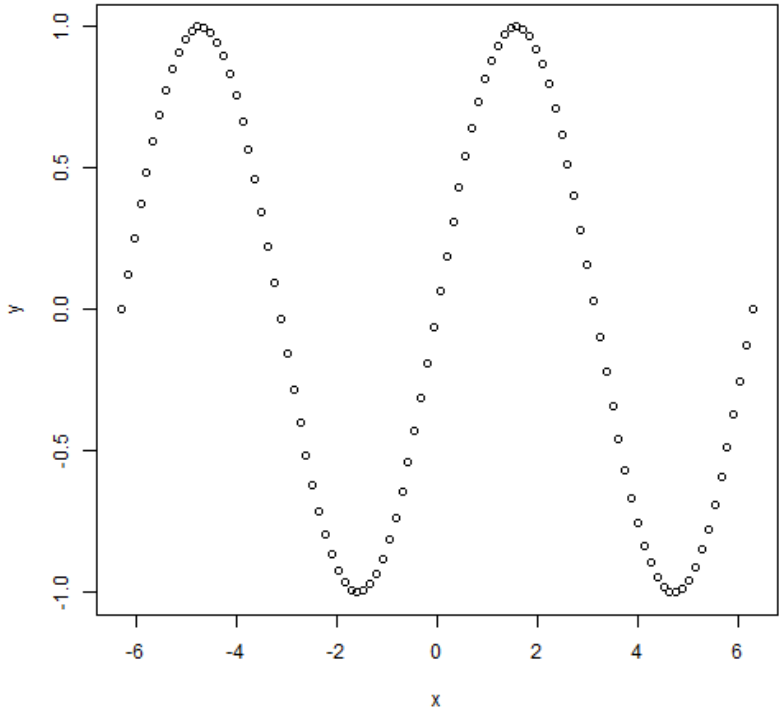
How does *temperature* relate to ozone and wind?



# Plotting functions

To plot a function, *discretize* the domain into `x` and *evaluate* the function into `y`.

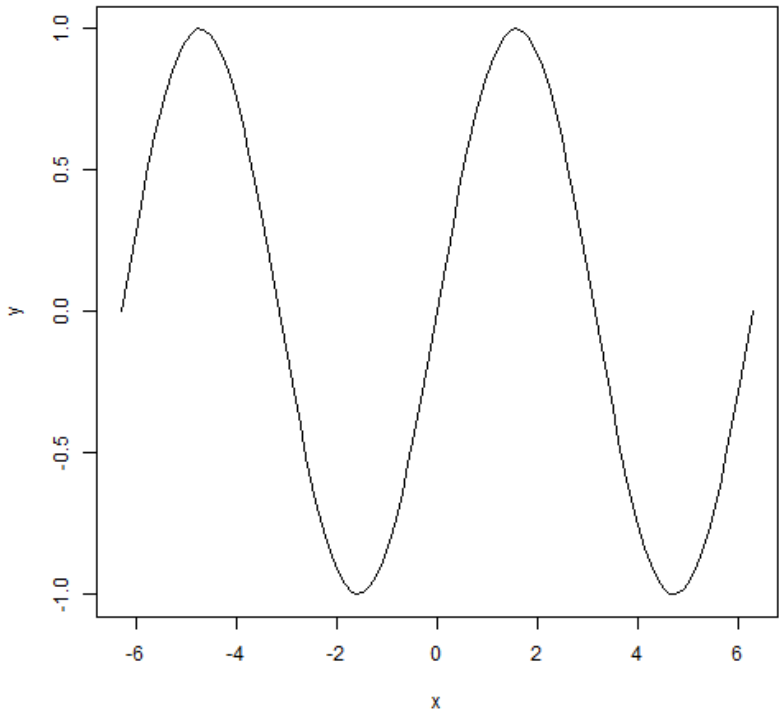
```
x <- seq(-2*pi, 2*pi, len = 100)
y <- sin(x)
plot(y ~ x)
```



# Plotting functions

To plot a function, *discretize* the domain into `x` and *evaluate* the function into `y`.

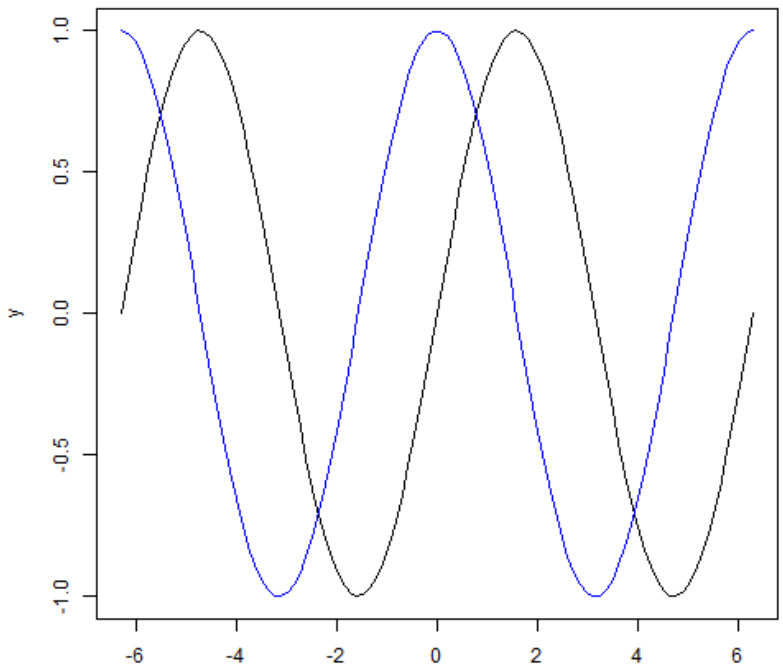
```
x <- seq(-2*pi, 2*pi, len = 100)
y <- sin(x)
plot(y ~ x, type="l") # type="1" smooths the line
```



# Plotting functions

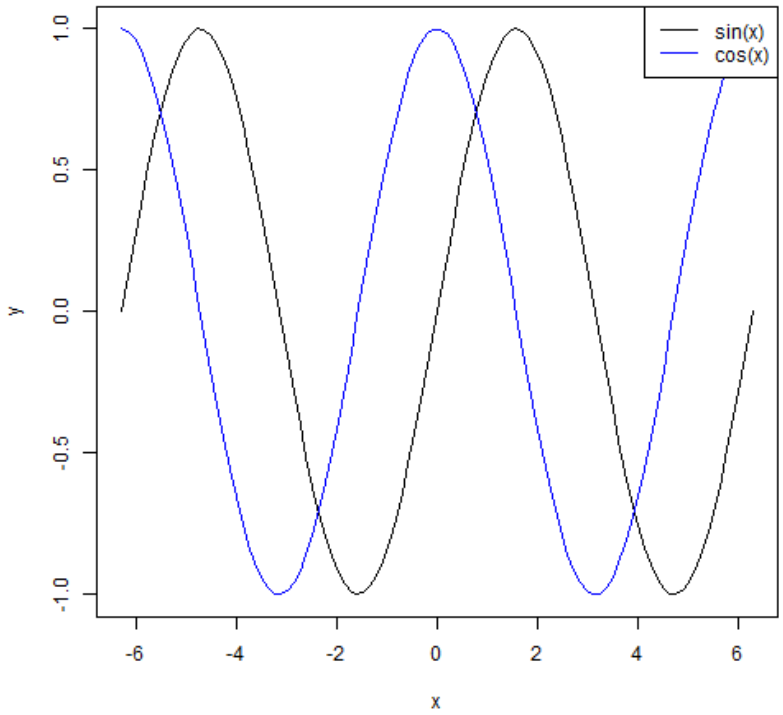
To plot a function, *discretize* the domain into `x` and *evaluate* the function into `y`.

```
x <- seq(-2*pi, 2*pi, len = 100)
y <- sin(x)
plot(y ~ x, type="l") # type="1" smooths the line
lines(cos(x) ~ x, type="l", col="blue") # "lines" adds to an existing plot
```



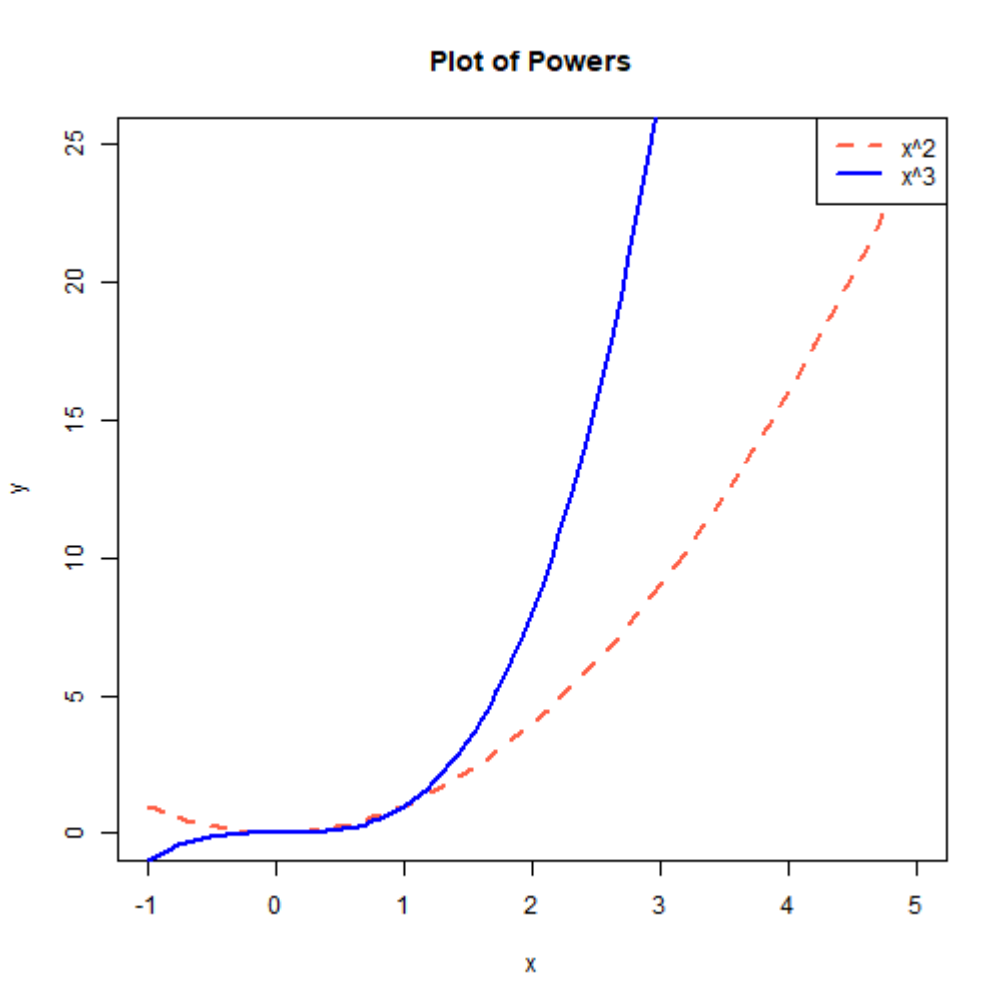
# Plotting functions

```r
x <- seq(-2*pi, 2*pi, len = 100)
y <- sin(x)
plot(y ~ x, type="l") # type="1" smooths the line
lines(cos(x) ~ x, type="l", col="blue") # "lines" adds to an existing plot
legend("topright", lty = 1, col = c("black", "blue"), legend = c("sin(x)", "cos(x)"))
```



# Your turn

*Hint*: Use `type`, `lty`, `lwd`, and `col` parameters. Remember to check out `?par`.




06:00