

Lecture 7: Base R Plots

Ingmar Sturm

UCSB

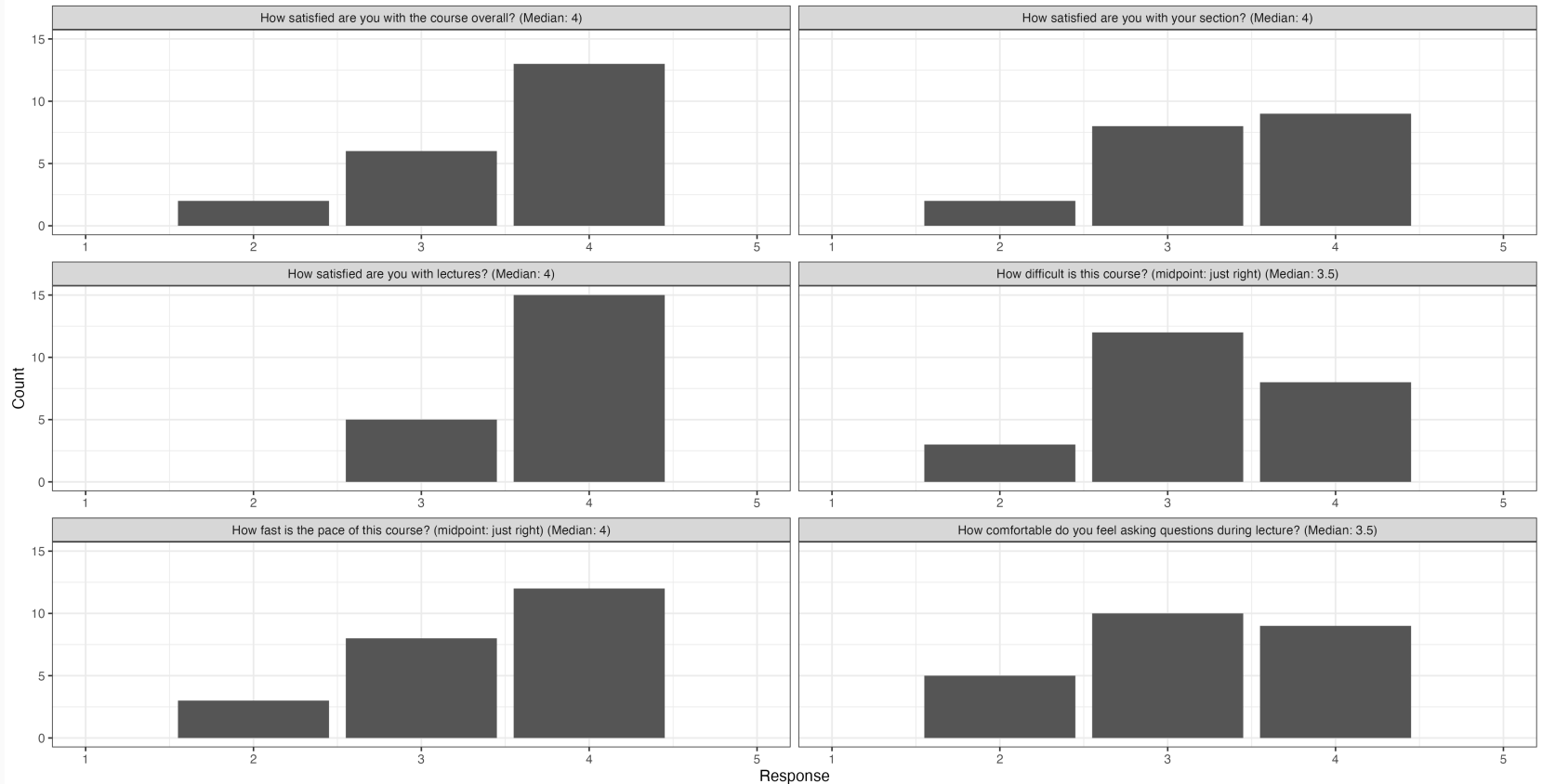
2024-07-03

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

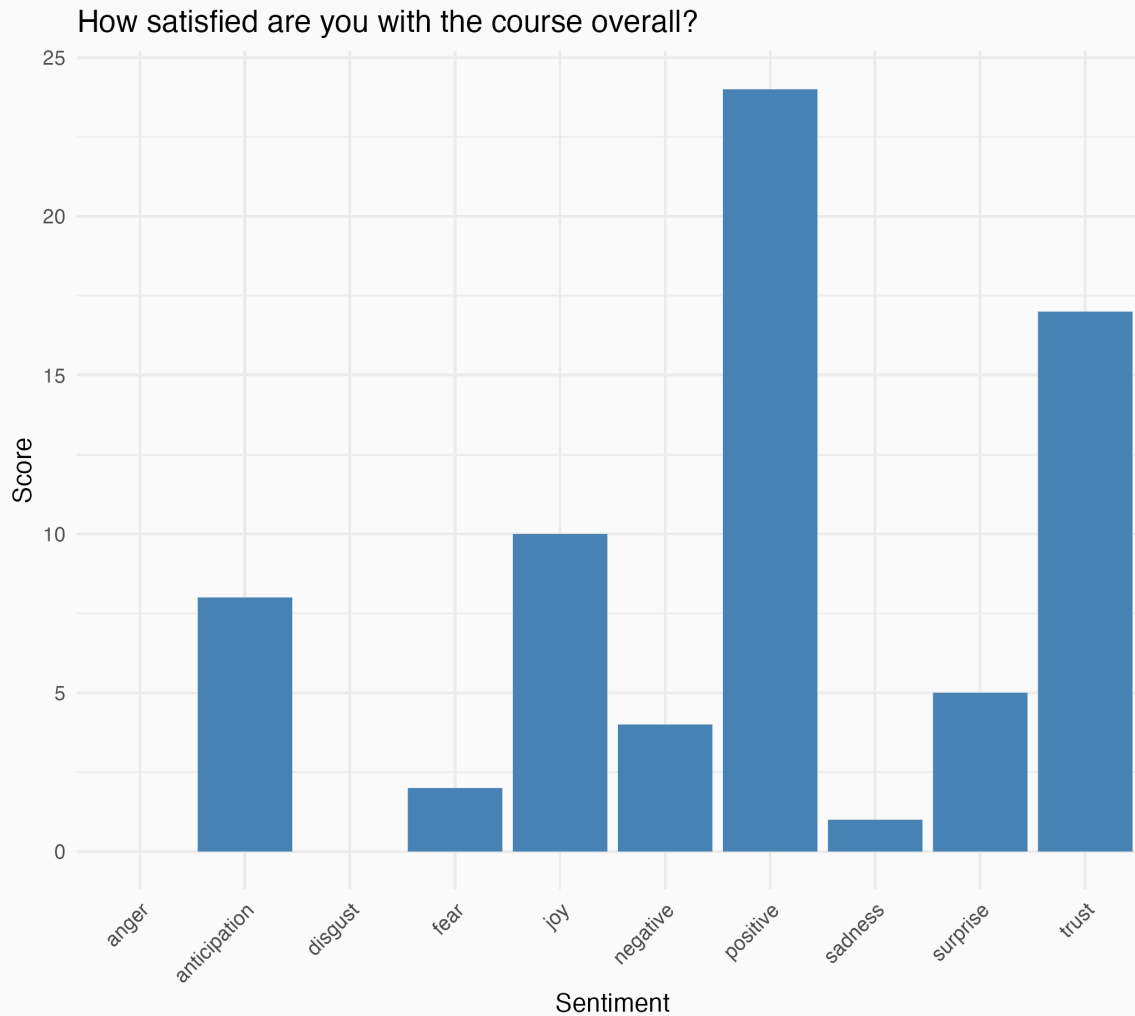
Analyzing Poll Results (like a data scientist)

Numerical Variables

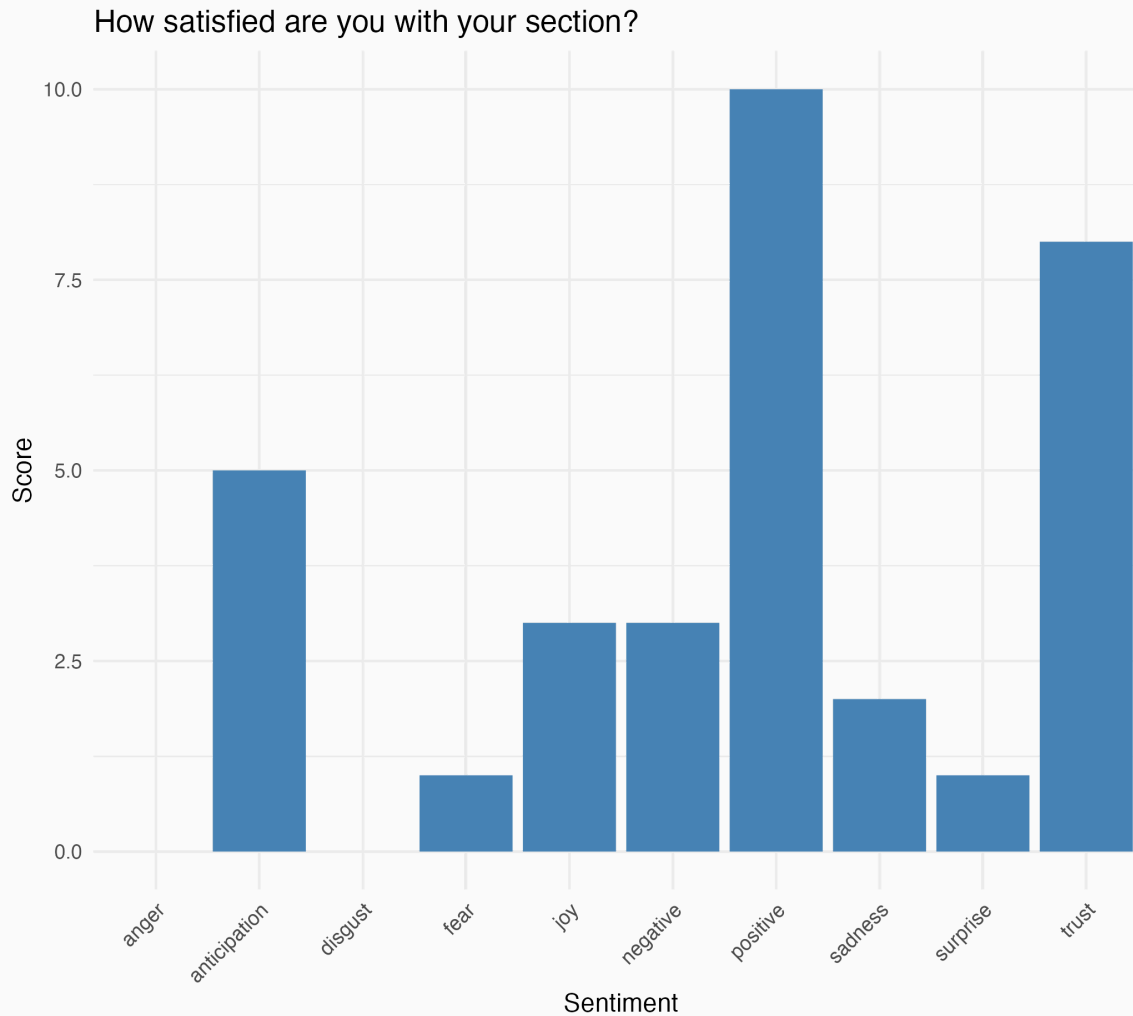
Survey Responses



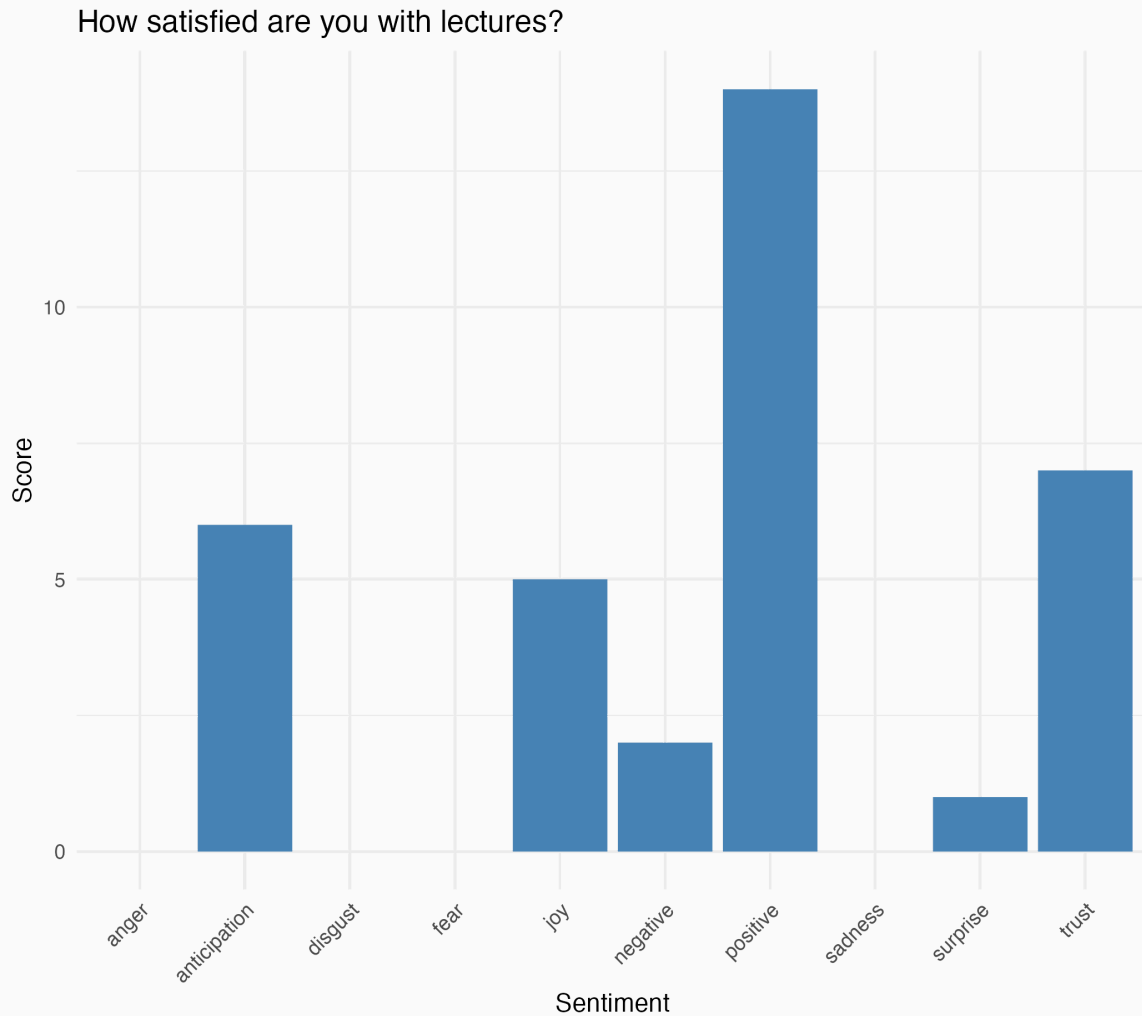
Sentiment Analysis 1/4



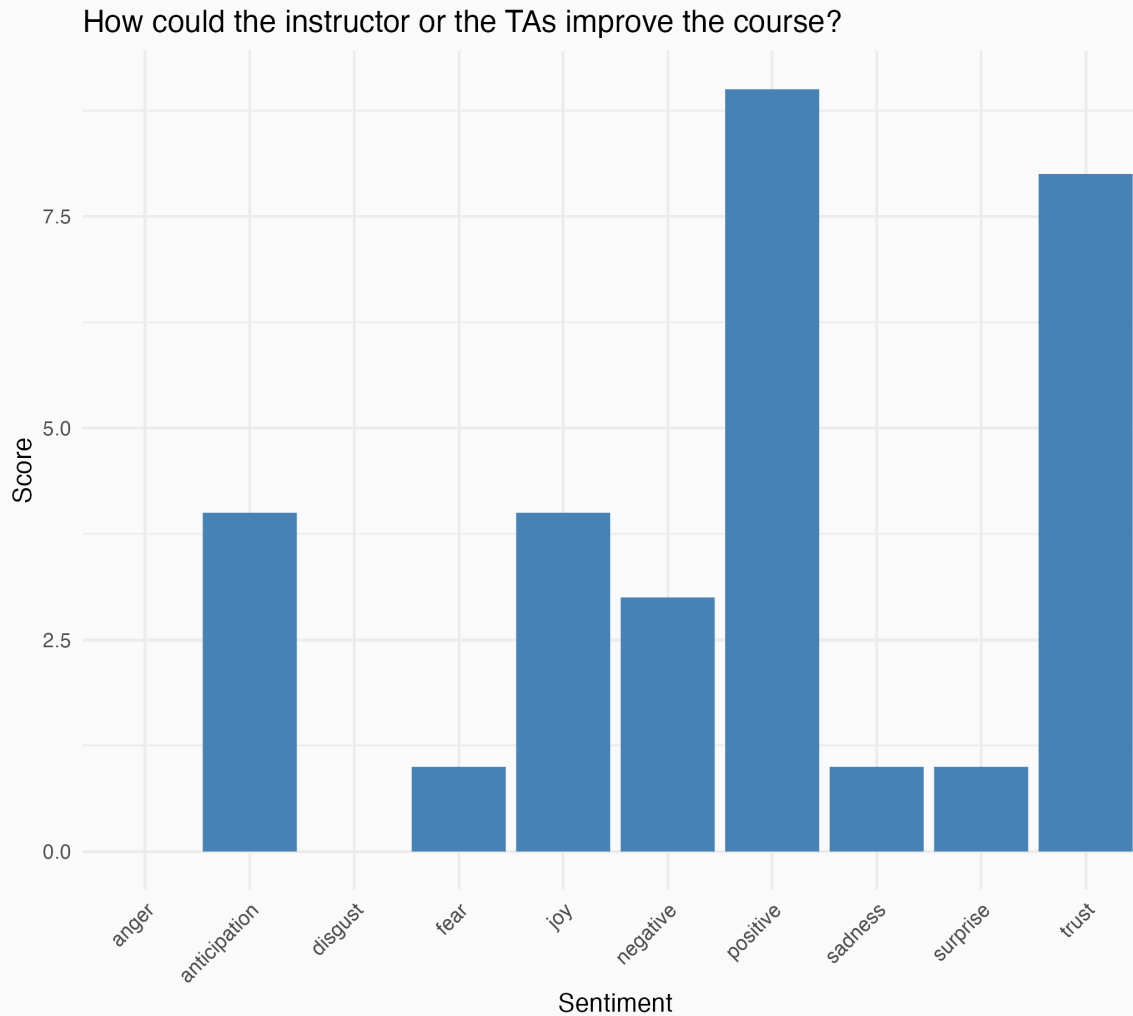
Sentiment Analysis 2/4



Sentiment Analysis 3/4



Sentiment Analysis 4/4



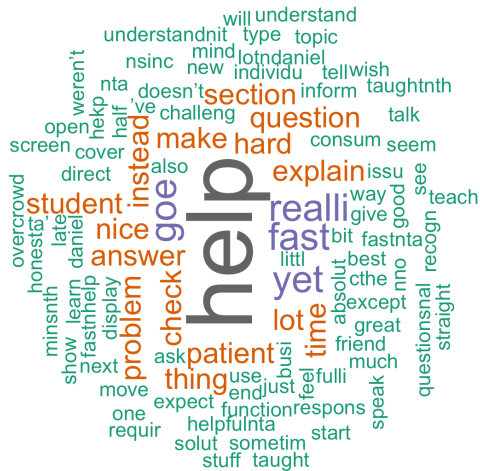
Word Cloud 1/4

How satisfied are you with the course overall?



Word Cloud 2/4

How satisfied are you with your section?



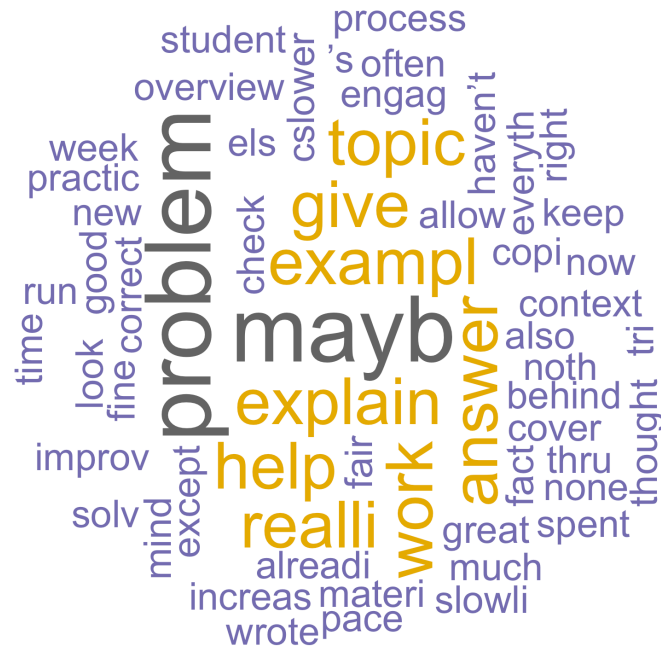
Word Cloud 3/4

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

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` will give you the metadata in the help window.

Description

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

Format

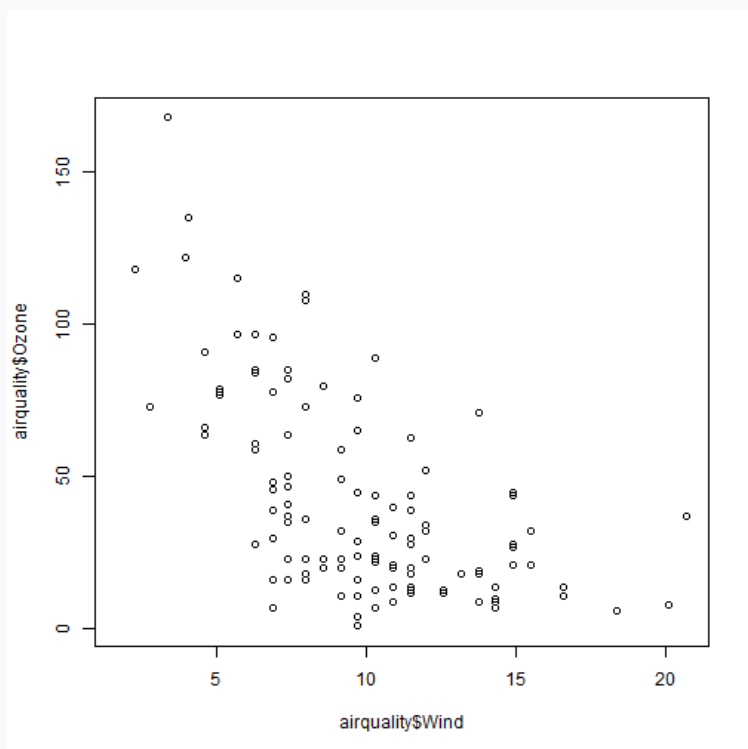
A data frame with 153 observations on 6 variables.

[,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)
```

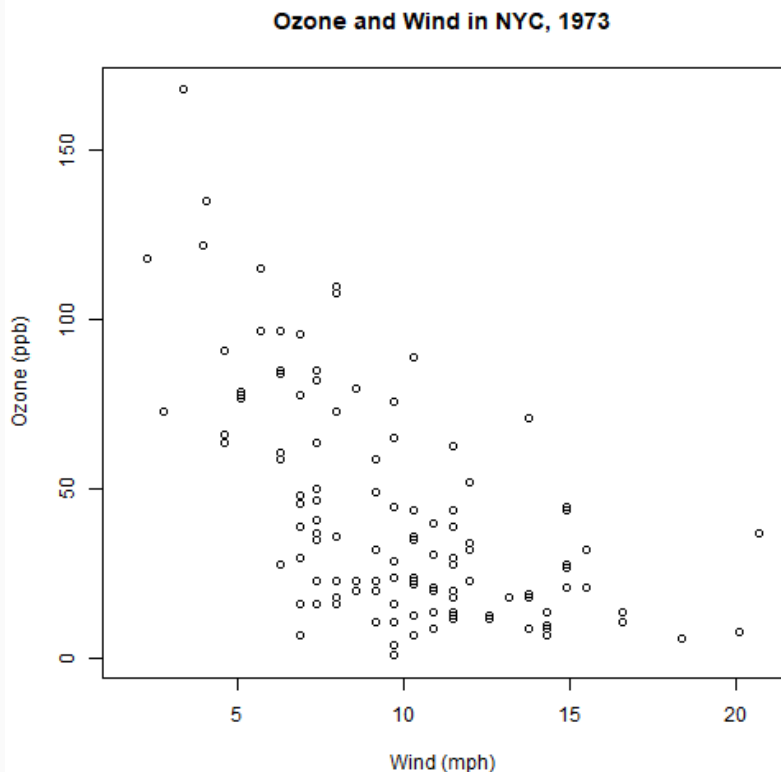


But we should always add a meaningful title and axis labels...

Base R plots

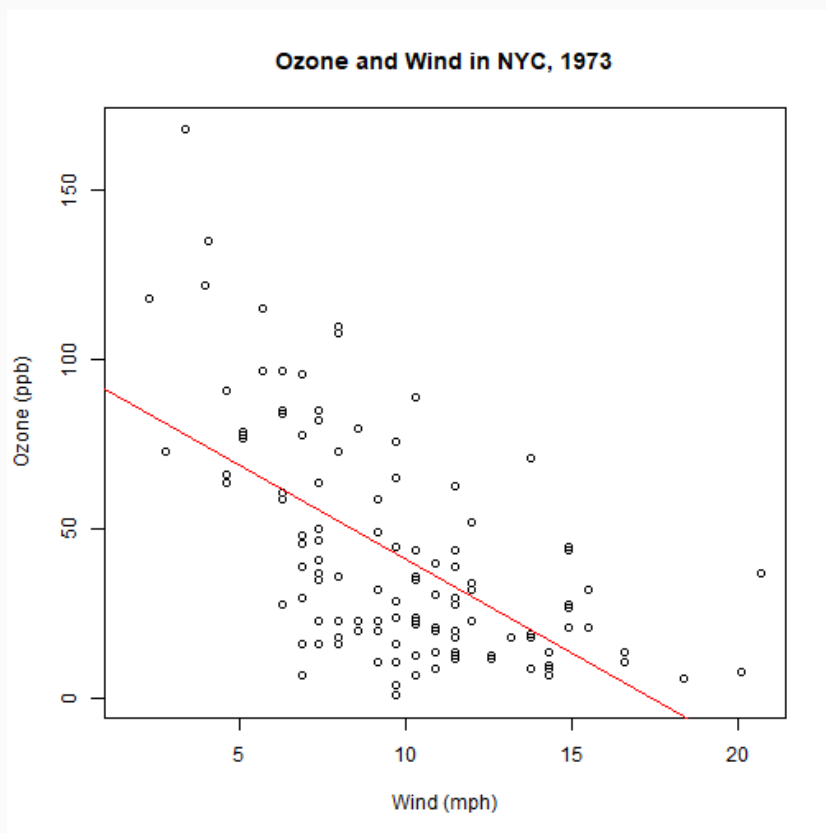
Basic plotting is easy; just call the relevant functions.

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



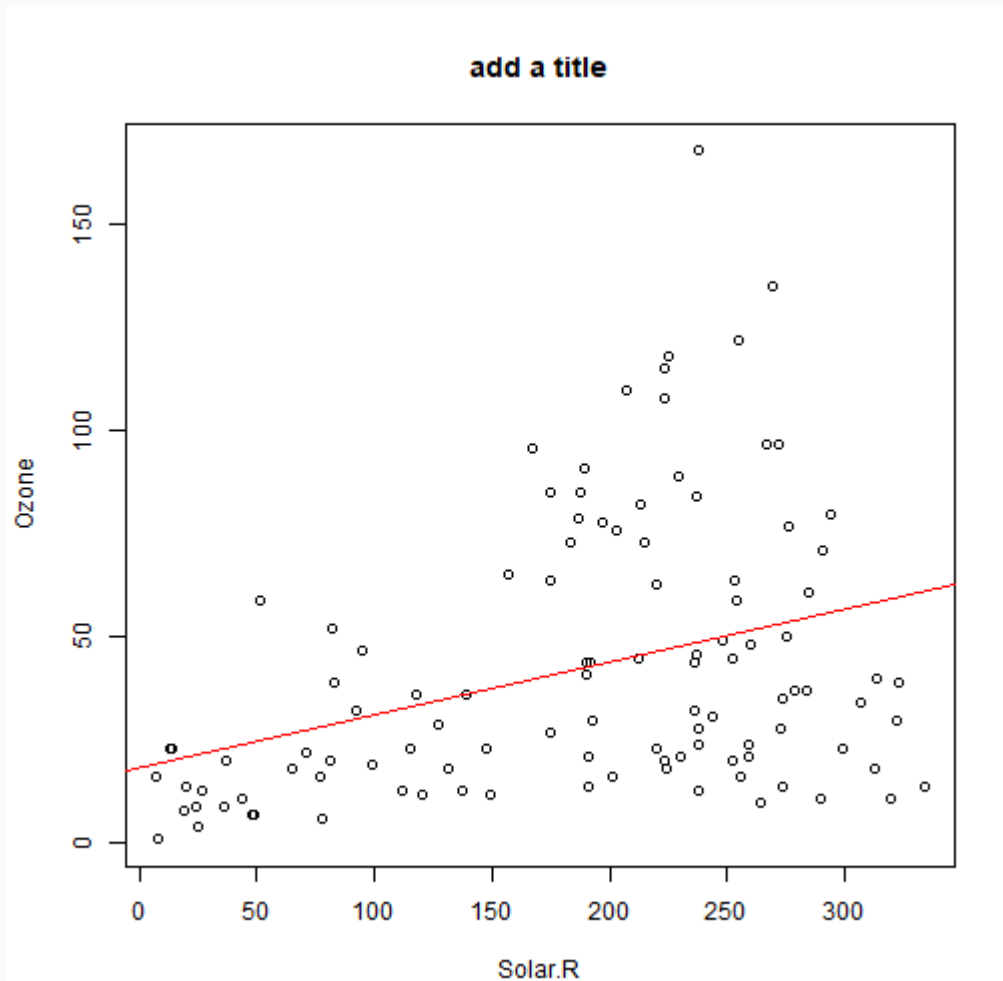
Adding a trendline

```
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

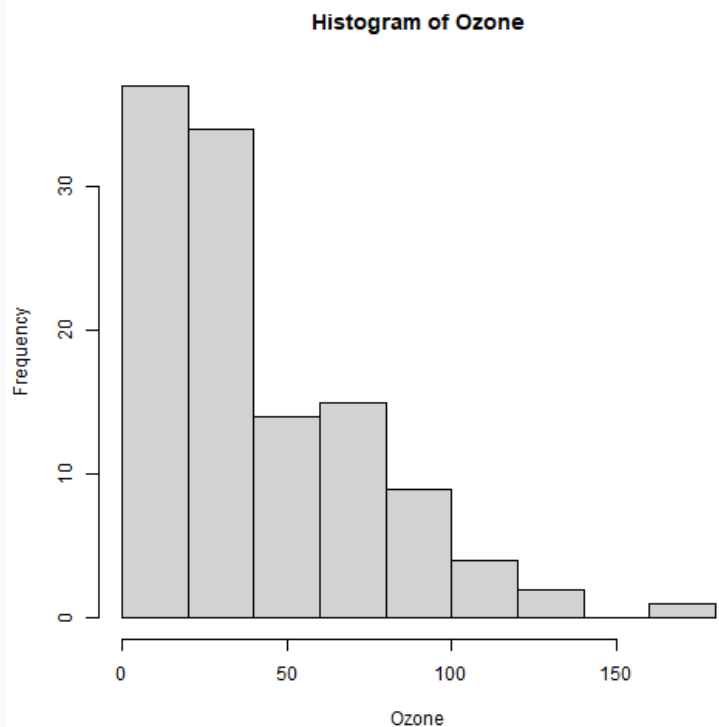
Create a scatter plot of Ozone vs. Solar Radiation with a trendline. Add a meaningful title and axis labels to the plot.



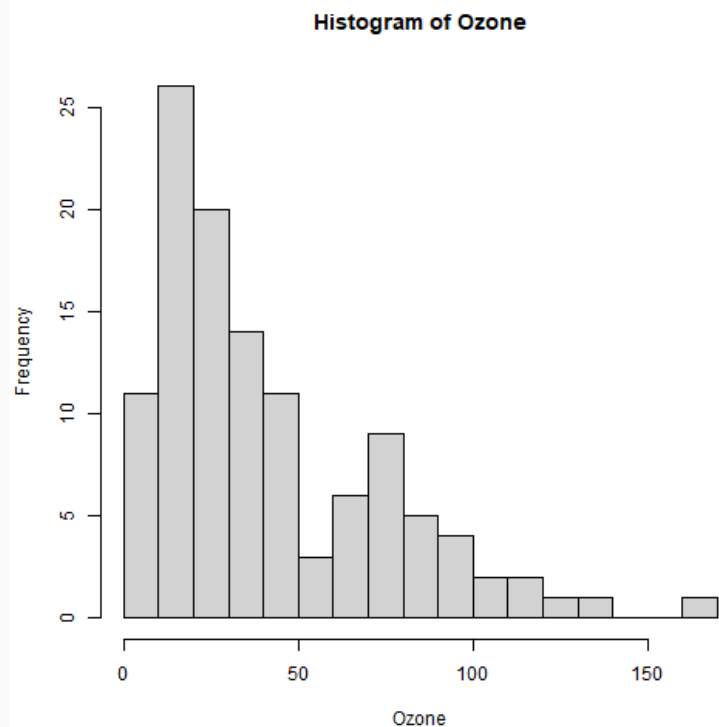
02:00

Histograms

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



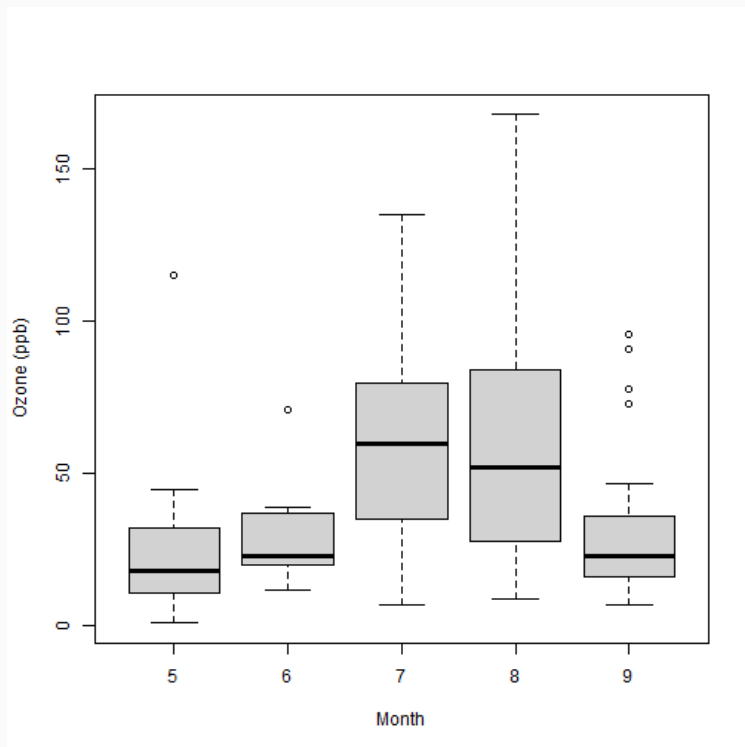
```
hist(airquality$Ozone, breaks = 20,  
     main = "Histogram of Ozone",  
     xlab = "Ozone")
```



Boxplots

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

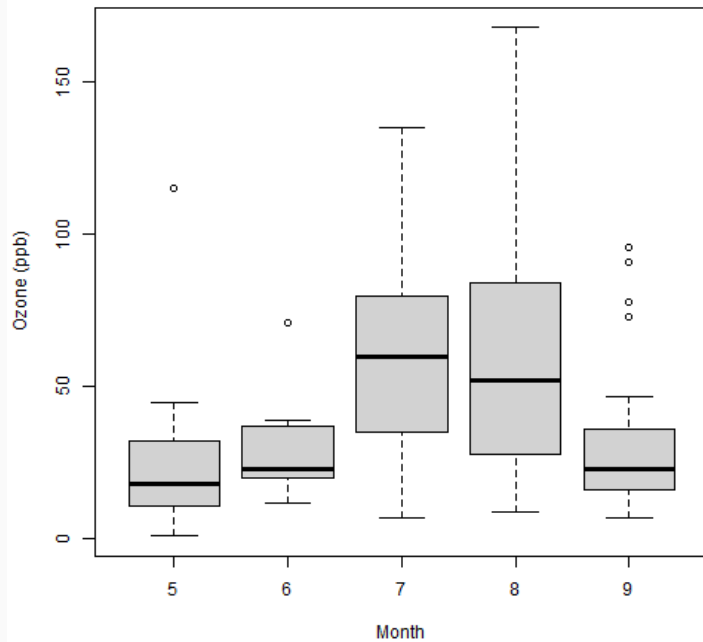
```
boxplot(Ozone ~ Month, airquality, xlab = "Month", ylab = "Ozone (ppb)")
```



Median, percentiles, quartiles, IQR, outliers

Formulas

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



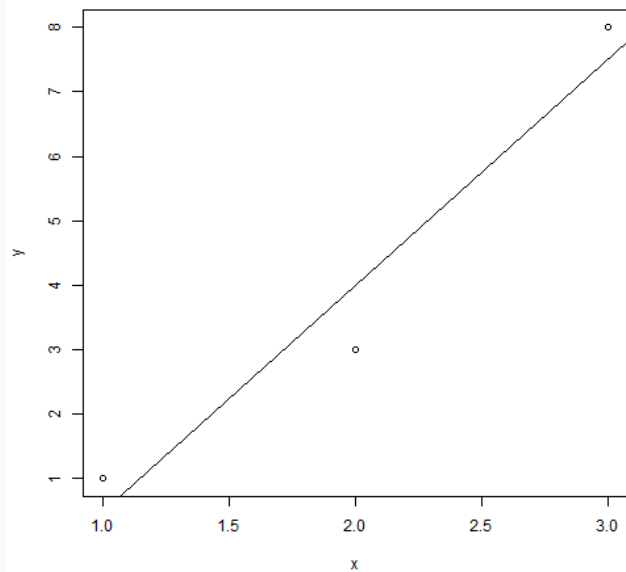
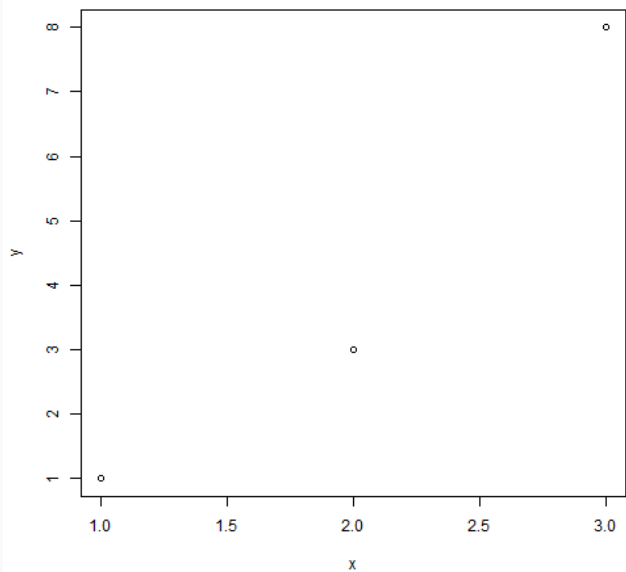
```
class(y ~ x)
```

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

Building plots

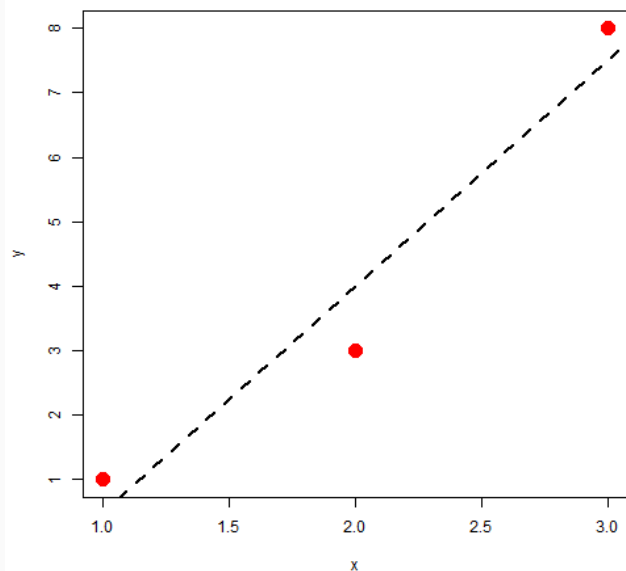
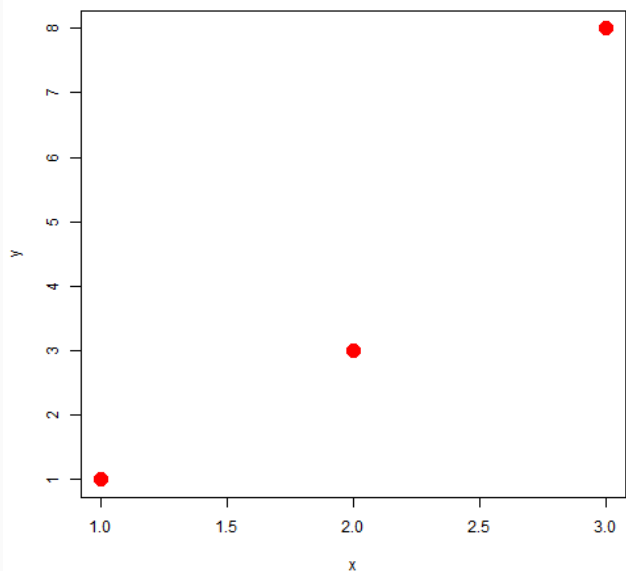
It is common to first create a plot and then layer stuff on top.

```
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

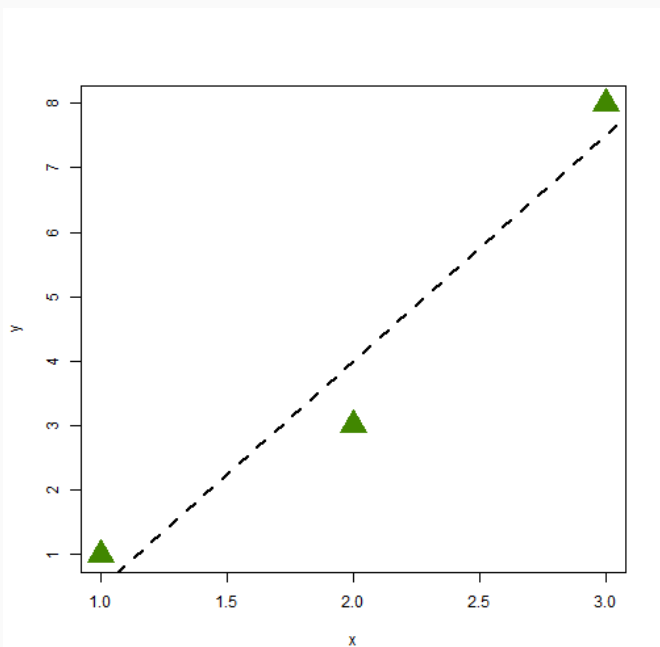
```
x <- 1:3  
y <- c(1, 3, 8)  
plot(x, y, cex=2, col="red", pch=19) # Creates a new plot  
lmout <- lm(y ~ x)  
abline(lmout, lty=2, lwd=2) # adds a line "fitting" the points
```



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
```

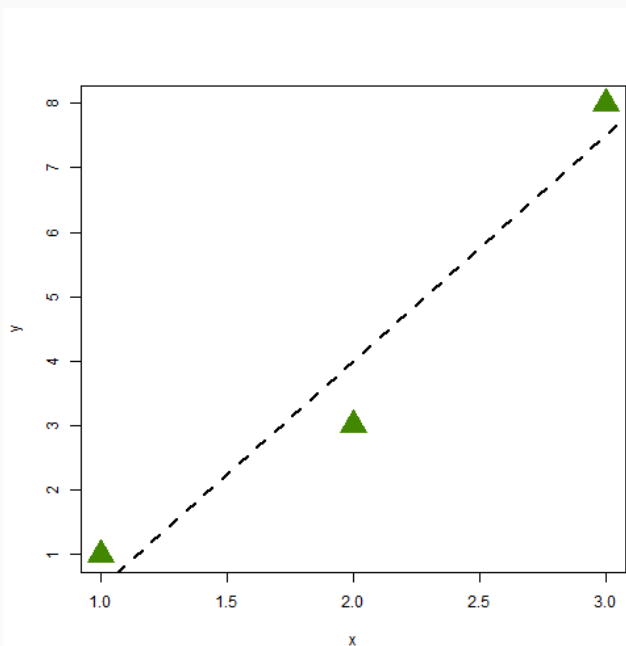


0	1	2	3	4	
□	○	△	+	×	
5	6	7	8	9	
◇	▽	⊠	✱	⬠	
10	11	12	13	14	
⊕	⊗	▣	⊗	▤	
15	16	17	18	19	
■	●	▲	◆	●	
20	21	22	23	24	25
●	●	■	◆	▲	▼

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
```

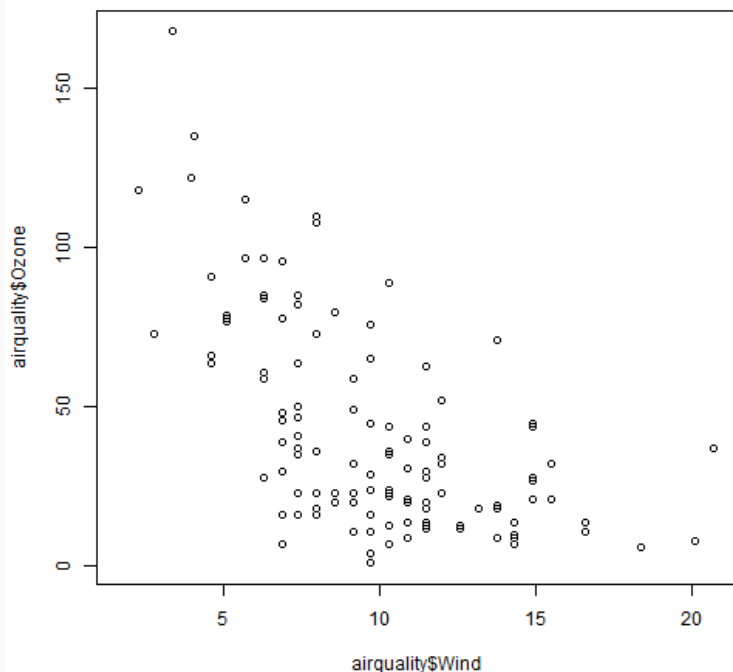


white	aliceblue	antiquewhite	antiquewhite1	antiquewhite2
antiquewhite3	antiquewhite4	aquamarine	aquamarine1	aquamarine2
aquamarine3	aquamarine4	azure	azure1	azure2
azure3	azure4	beige	bisque	bisque1
bisque2	bisque3	bisque4	black	blanchedalmond
blue	blue1	blue2	blue3	blue4
blueviolet	brown	brown1	brown2	brown3
brown4	burlywood	burlywood1	burlywood2	burlywood3
burlywood4	cadetblue	cadetblue1	cadetblue2	cadetblue3
cadetblue4	chartreuse	chartreuse1	chartreuse2	chartreuse3
chartreuse4	chocolate	chocolate1	chocolate2	chocolate3
chocolate4	coral	coral1	coral2	coral3
coral4	cornflowerblue	cornsilk	cornsilk1	cornsilk2
cornsilk3	cornsilk4	cyan	cyan1	cyan2
cyan3	cyan4	darkblue	darkcyan	darkgoldenrod
darkgoldenrod1	darkgoldenrod2	darkgoldenrod3	darkgoldenrod4	darkgray
darkgreen	darkgrey	darkkhaki	darkmagenta	darkolivegreen
darkolivegreen1	darkolivegreen2	darkolivegreen3	darkolivegreen4	darkorange
darkorange1	darkorange2	darkorange3	darkorange4	darkorchid
darkorchid1	darkorchid2	darkorchid3	darkorchid4	darkred
darksalmon	darkseagreen	darkseagreen1	darkseagreen2	darkseagreen3
darkseagreen4	darkslateblue	darkslategray	darkslategray1	darkslategray2
darkslategray3	darkslategray4	darkslategray	darkturquoise	darkviolet
deeppink	deeppink1	deeppink2	deeppink3	deeppink4
deepskyblue	deepskyblue1	deepskyblue2	deepskyblue3	deepskyblue4

Building plots

We see a negative trend with wind and ozone.

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



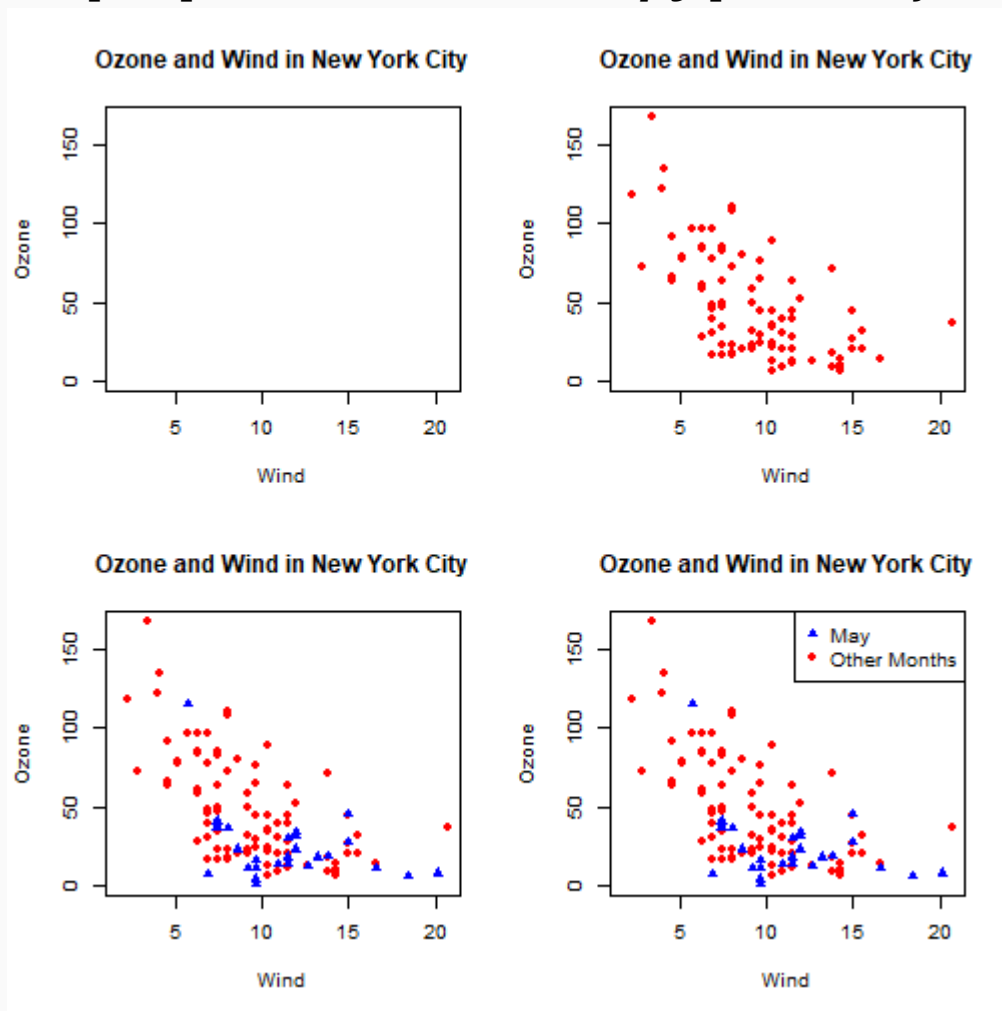
A flat screen can only show two variables.

unless...? 🤔

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

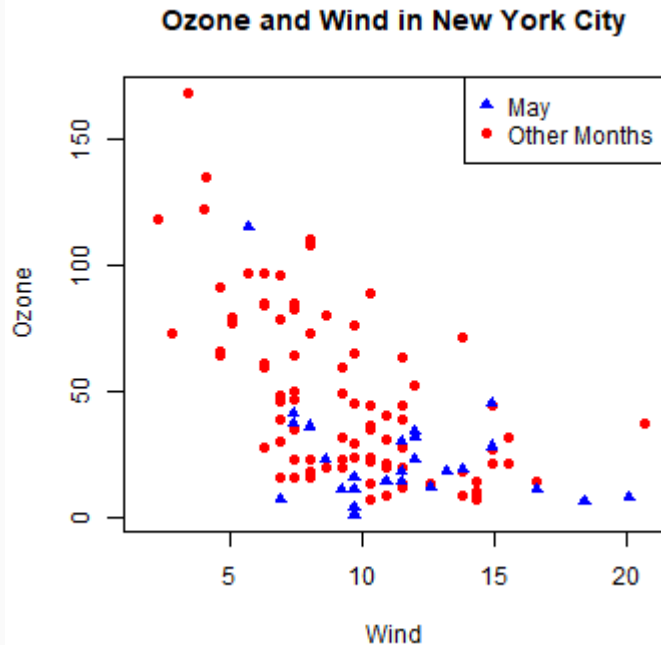
Layering Plots Demo 1

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



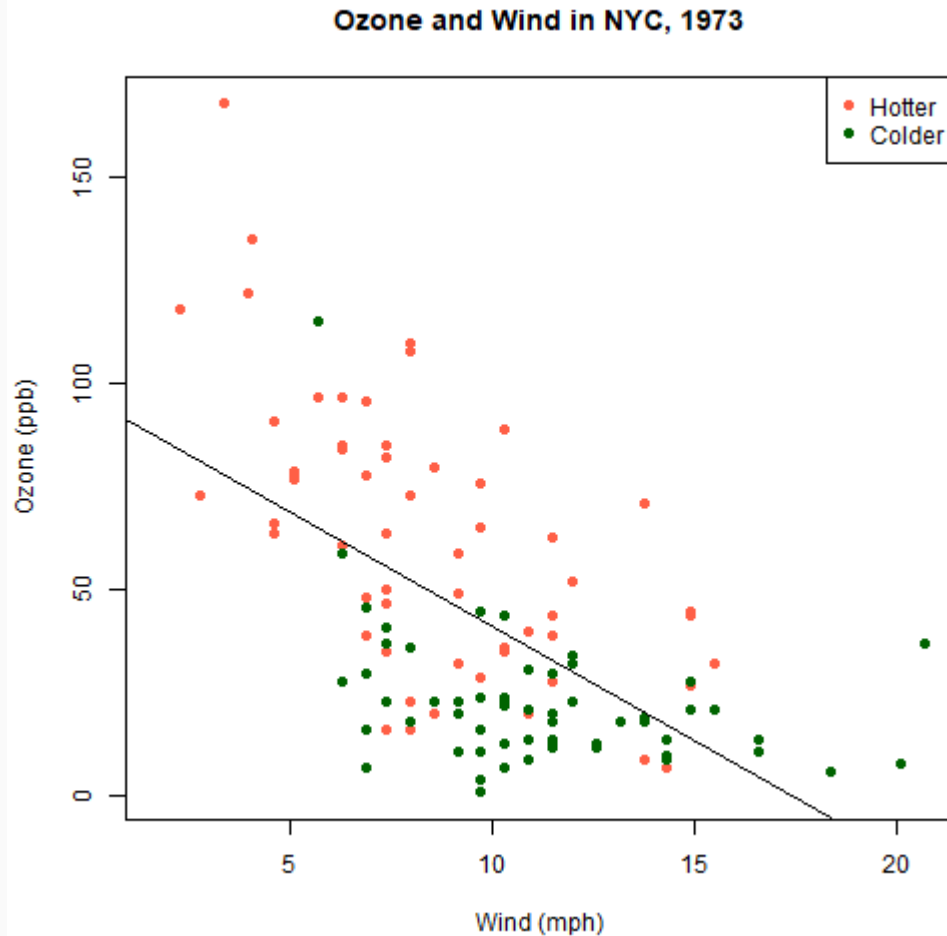
Layering Plots Demo 1

```
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

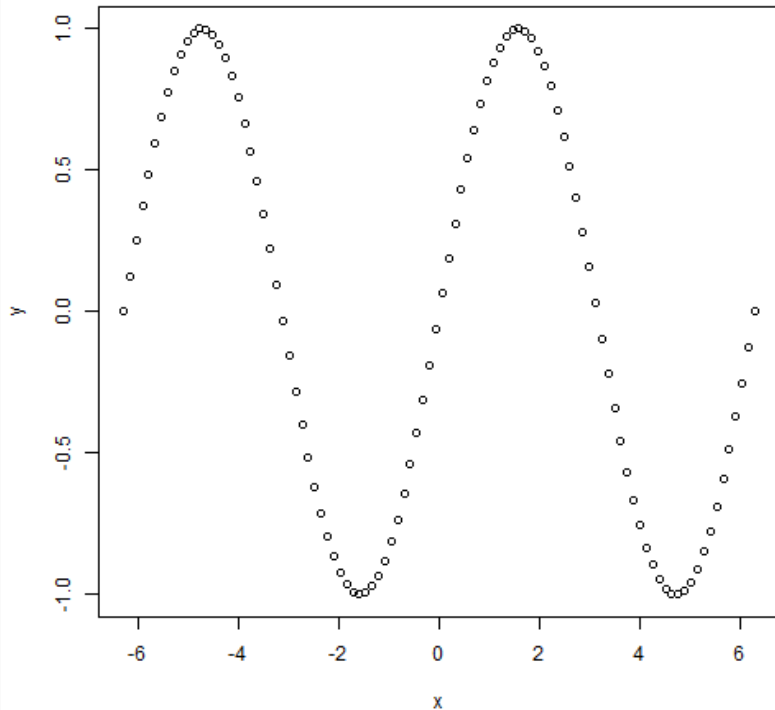
How does *temperature* relate to ozone and wind?



Plotting functions

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

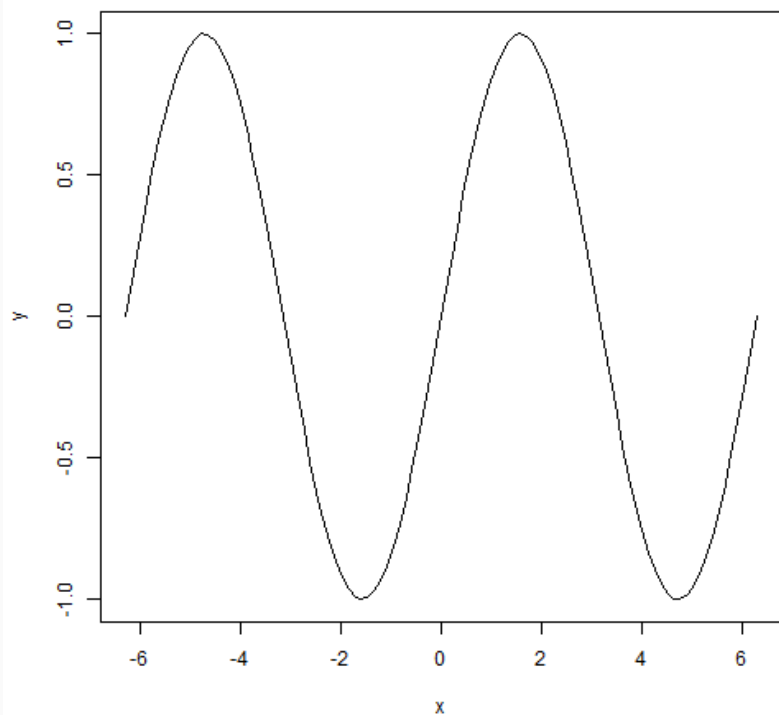
```
x <- seq(-2*pi, 2*pi, len = 100)
y <- sin(x)
plot(y ~ x)
```



Plotting functions

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

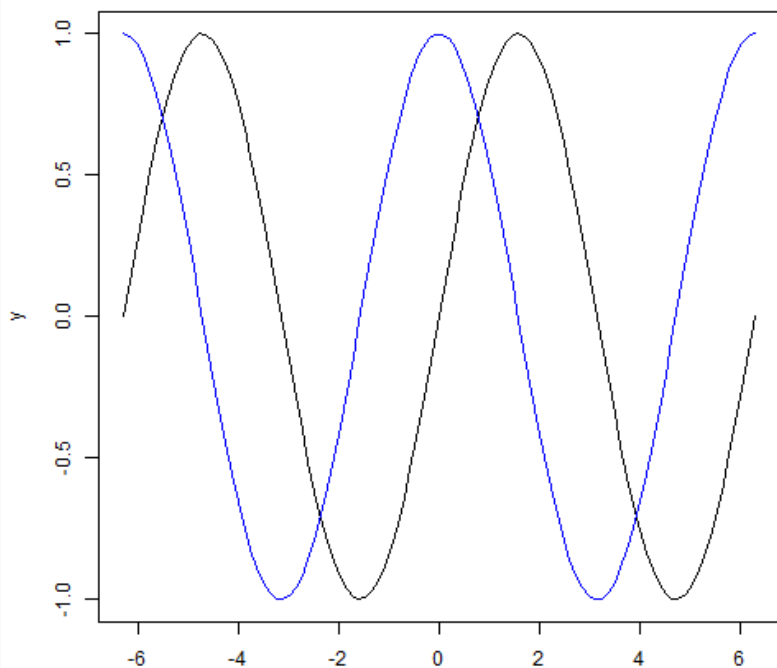
```
x <- seq(-2*pi, 2*pi, len = 100)
y <- sin(x)
plot(y ~ x, type="l") # type="l" smooths the line
```



Plotting functions

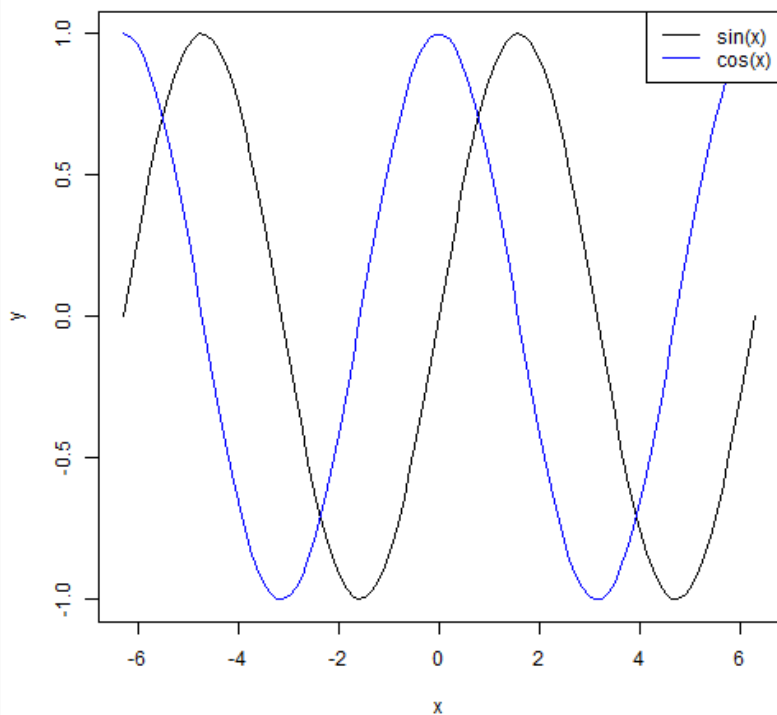
To plot a function, *discretize* the domain into `x` and *evaluate* the function into `y`.

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



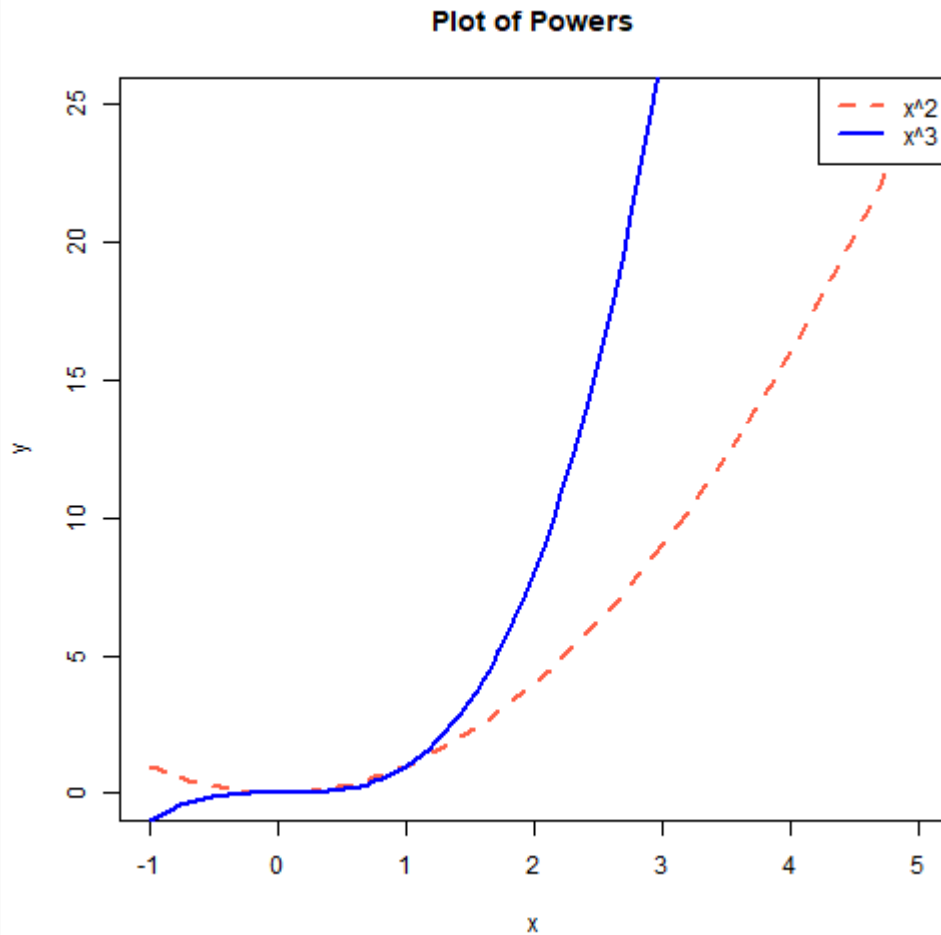
Plotting functions

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