Data visualization: Basics

Yufeng Huang

Associate Professor of Marketing, Simon Business School

July 28, 2022

Why do we need data visualization

- A significant part of data science is to let **humans** understand you
- This includes communicating your results:
  - transparently (no mis-representation)
  - efficiently (to the point)

This class

- Examples
- Basic plot() function
- Histogram, density, and boxplot

Example: mpg and hp of cars

```
# mtcars is a built-in dataset
head(mtcars, n = 10)    # shows first n (10) rows of a data frame
```

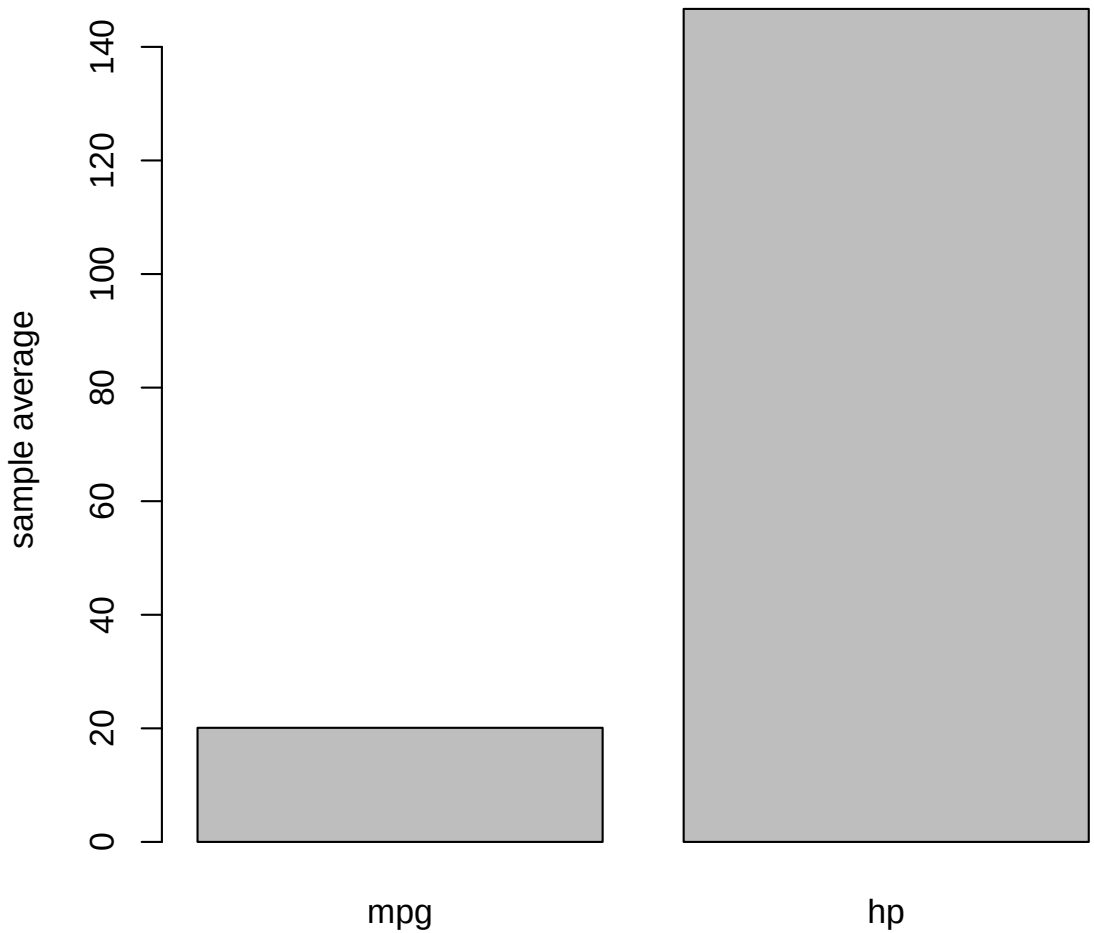
```
##                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
```

Suppose I want to know average mpg and hp[1]

```
# We have not really talked about data frames,
#    but we can use symbol "$" to find a column in a data frame

# average miles per gallon
mean(mtcars$mpg)

## [1] 20.09062

# average horsepower
mean(mtcars$hp)

## [1] 146.6875

# can squeeze them into a vector (with element names)
summary.cars <- c(mpg = mean(mtcars$mpg), hp = mean(mtcars$hp))
summary.cars

##       mpg        hp
##  20.09062 146.68750
```

[1] We're using a lot of vectors in the form of x$y; for now simply understand x$y as column y in dataframe (spreadsheet) x

```r
# Instead, I could produce a graph
#    barplot() produces a bar plot
barplot(summary.cars, ylab = 'sample average')
```

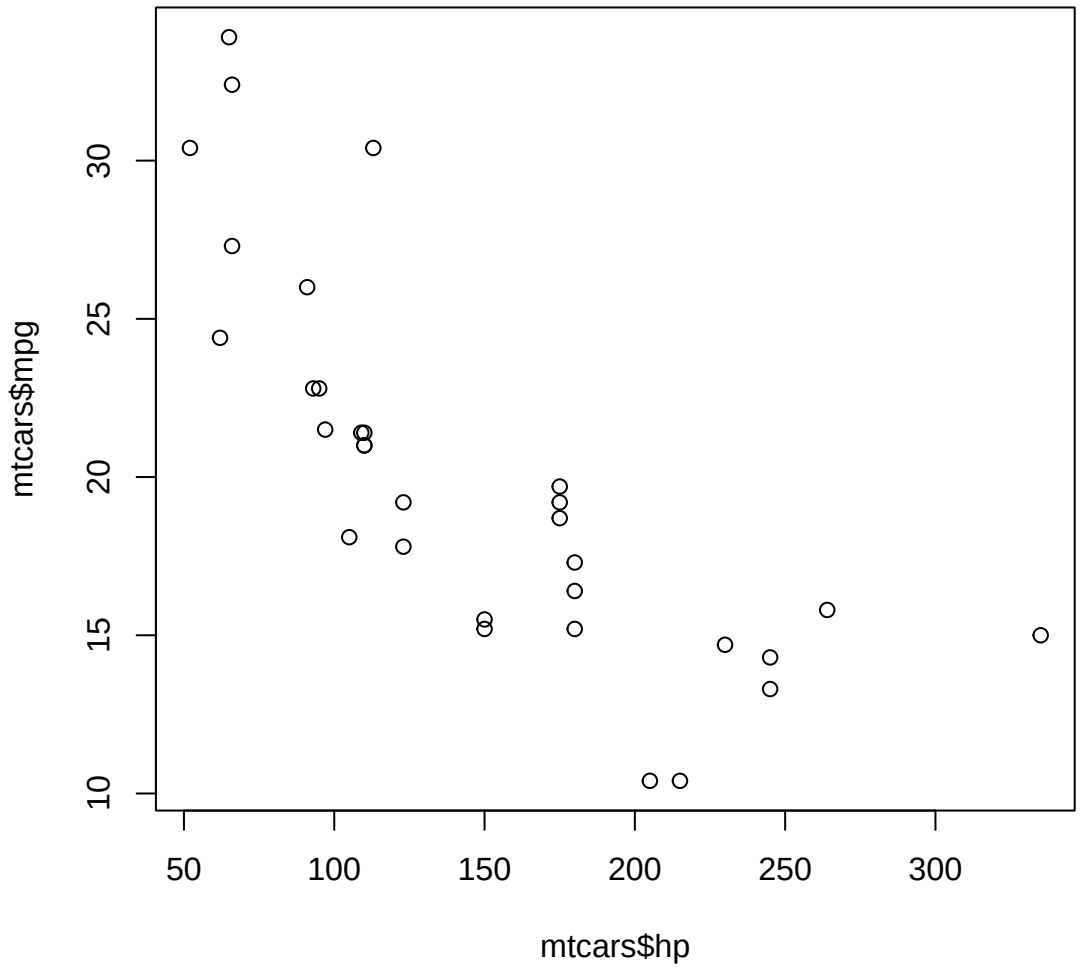


What does this plot tell us beyond the mean? Do you like this plot and why?

In a different scenario, suppose I care about correlation between the two

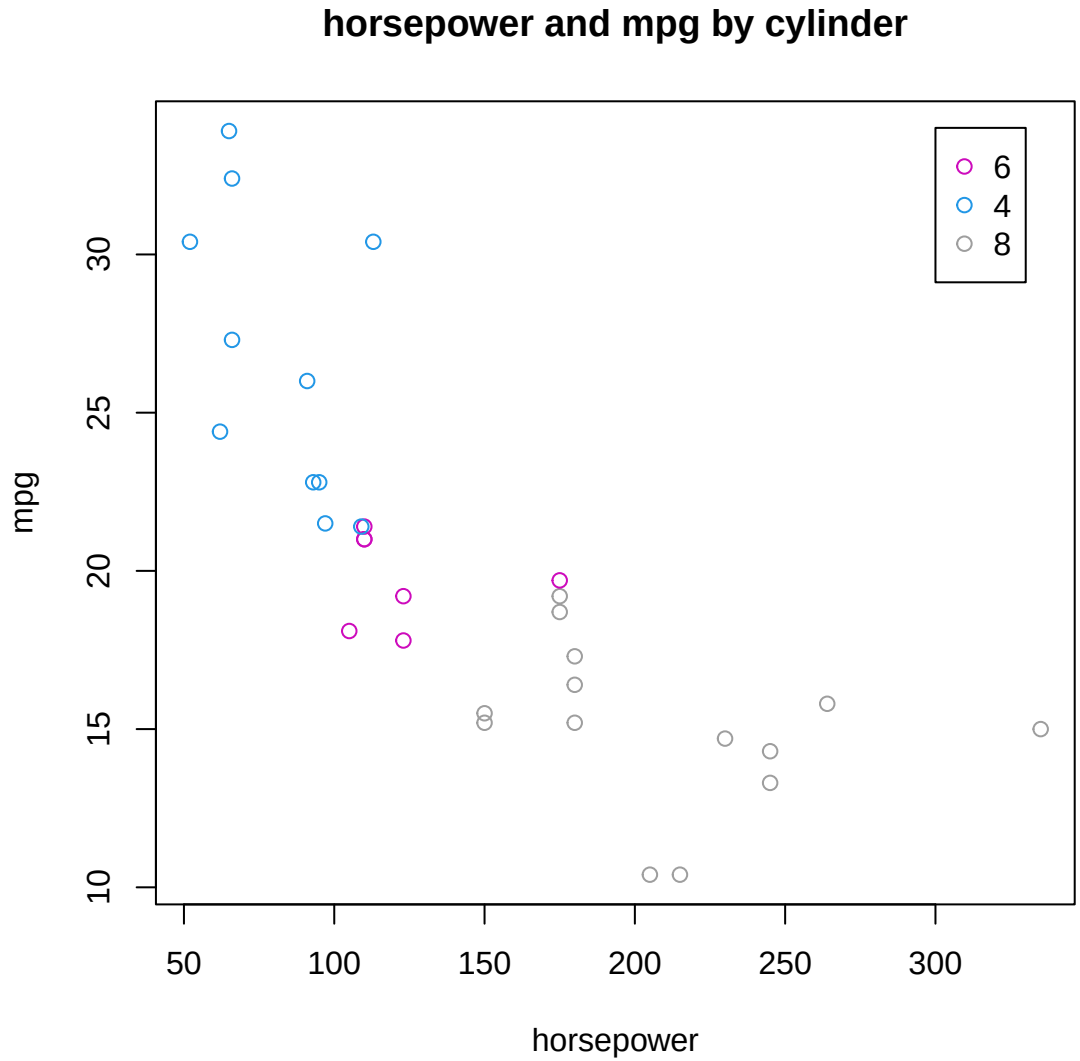
```
# I could summarize the correlation
cor(mtcars$mpg, mtcars$hp)

## [1] -0.7761684

# Or I can run a linear regression if I think hp is the cause
fit <- lm(mpg ~ hp, data = mtcars)

# below shows the linear relationship that
#    MPG = 30 - 0.07 * HP
fit$coefficients

## (Intercept)          hp
## 30.09886054 -0.06822828
```

```r
# Instead, I could produce a graph on this
plot(mtcars$hp, mtcars$mpg)
```



What does this plot tell us beyond the correlation?

Highlighting cylinder by color reveals the driver behind this



So why do we need data visualization?

- More like, **when** do we need data visualization?
- Data visualization is not meant to "look fancy," but to convey your message in a more efficient way
- Visualize data if and only if it conveys the point:
  - transparently, and
  - efficiently
- Today: some basic plot types in R
  - on the to-do-list: more discussions and examples about visualization

Examples

Example 1: use plot() to show the relationship between two variables

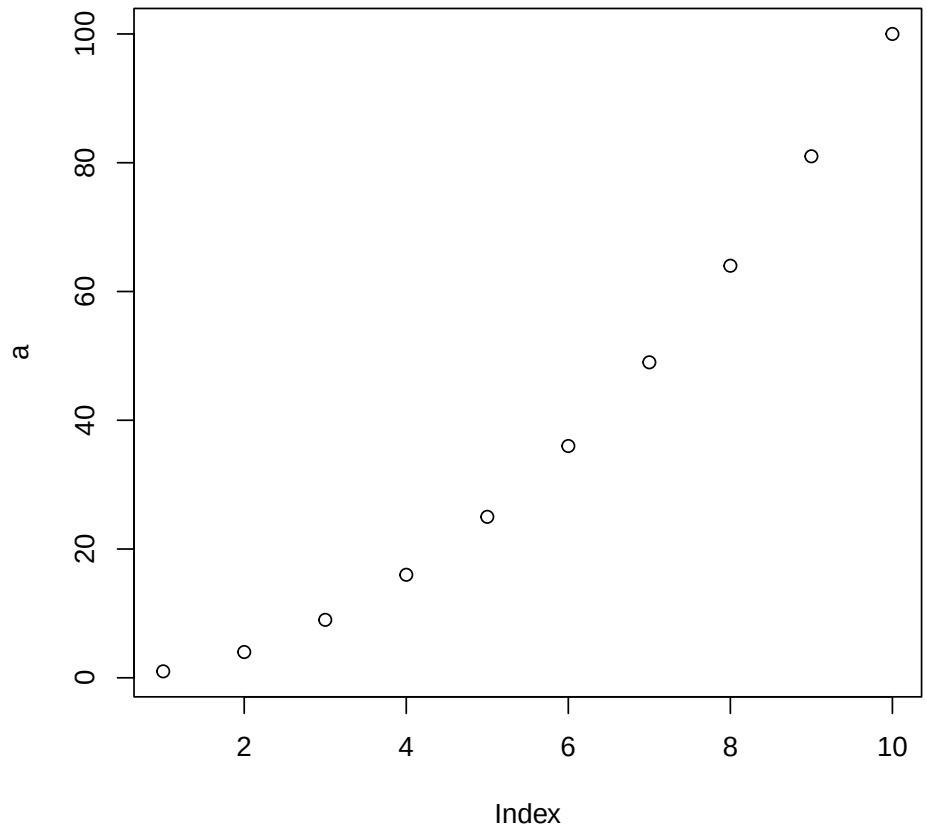
- plot() is a very basic yet flexible function to generate plots
- Depending on the data you throw at it, it generates different types of plots
- In one-variable case:

Function	Data	Description
plot()	numeric	scatterplot
plot()	factor	barplot
...	...	...

- Two-variable case:

Function	X data	Y data	Description
plot()	numeric	numeric	scatter plot
plot()	factor	numeric	box plot
plot()	2-D table		mosaic plot
...	...		...

```r
# plot a numeric vector
a <- (1:10)^2
plot(a)
```



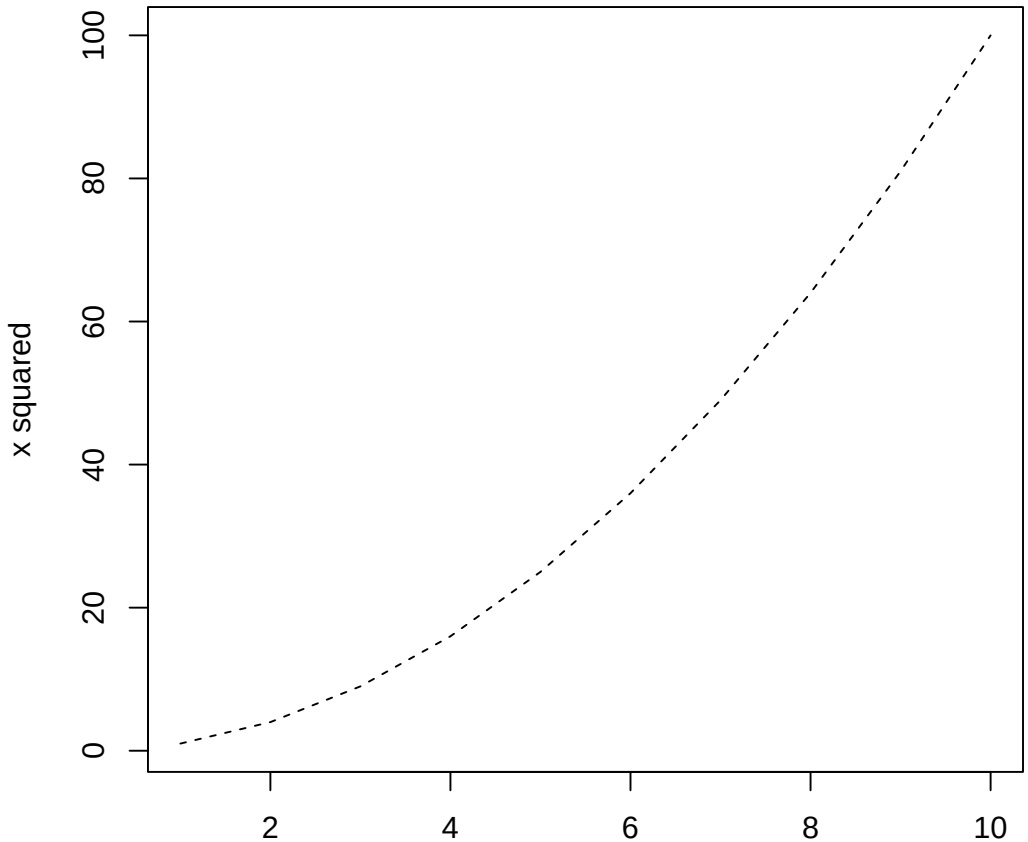
Note: plot(a) where a is numeric is implicitly understood as the scatter plot between a's index and a's value

Additional arguments in plot()

- "Arguments" here refers to additional instructions we put in a function
- Plot() is a very flexible function:
  - its output not only depends on the data type but also on what we "instruct" it to do
  - this is yet another case where **human input** is very important
- For example, if we want to plot a quadratic curve rather than a few scatter points, we should use a connected plot rather than a scatter plot

```
# plot a, but also connect the points, and make the line dashed
#    in addition, change x and y axis label to be more informative
#    type means type of plot ('l' stands for 'line')
#    lty means "line type", 2 refers to the second type (see help file)
plot(a, type = 'l', lty = 2, xlab = "x", ylab = "x squared")
```

Additional arguments in plot()

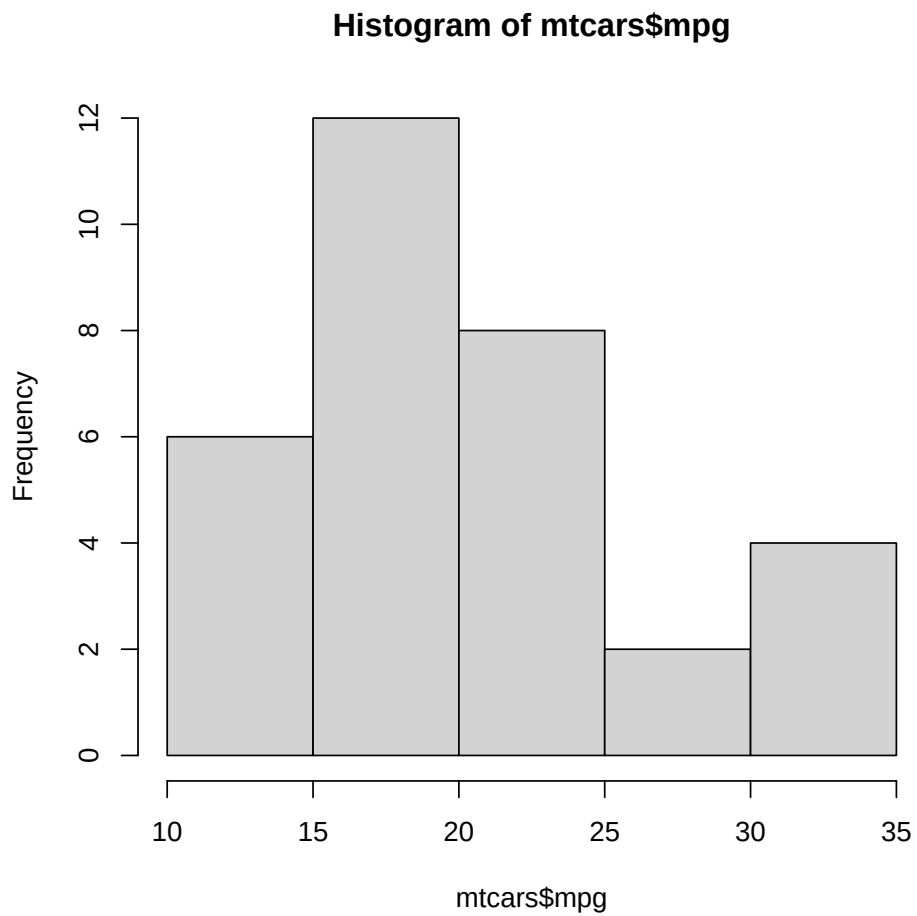


Note: usually I'll add an x-variable$_x$ to make the code clear, so e.g.

```
b <- 1:10
plot(b, b^2, ...)
```

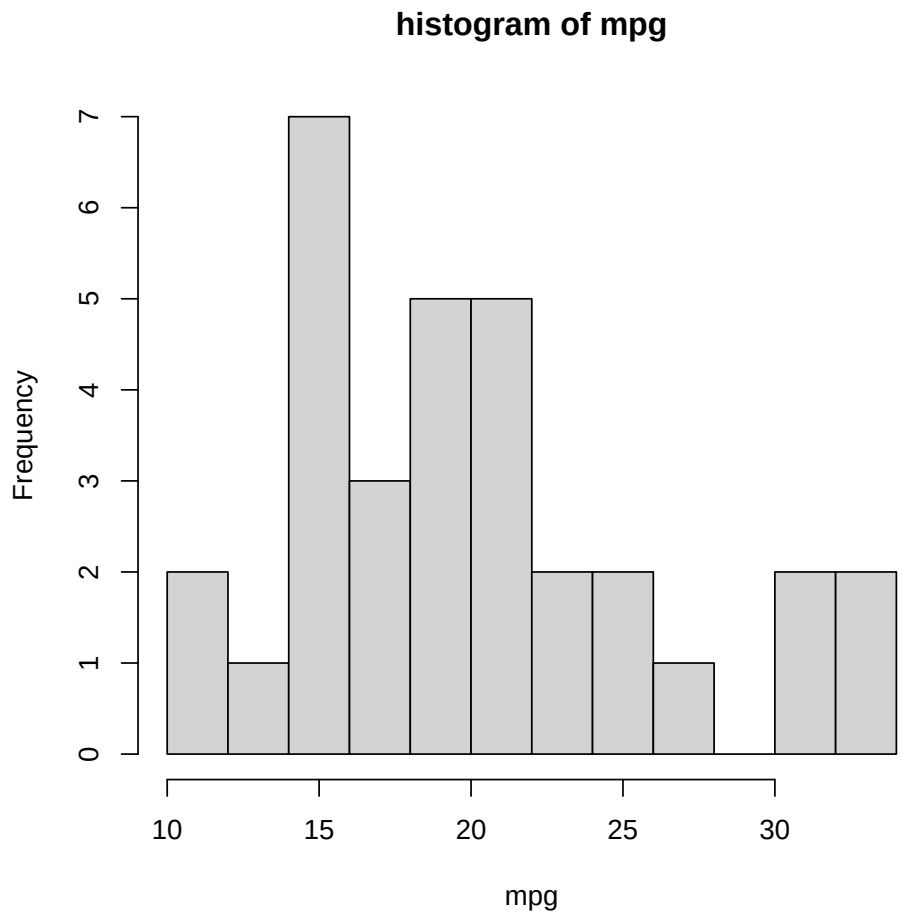
Example 2: use histogram to show the distribution of one variable

```
# histogram of mpg among cars
hist(mtcars$mpg)
```



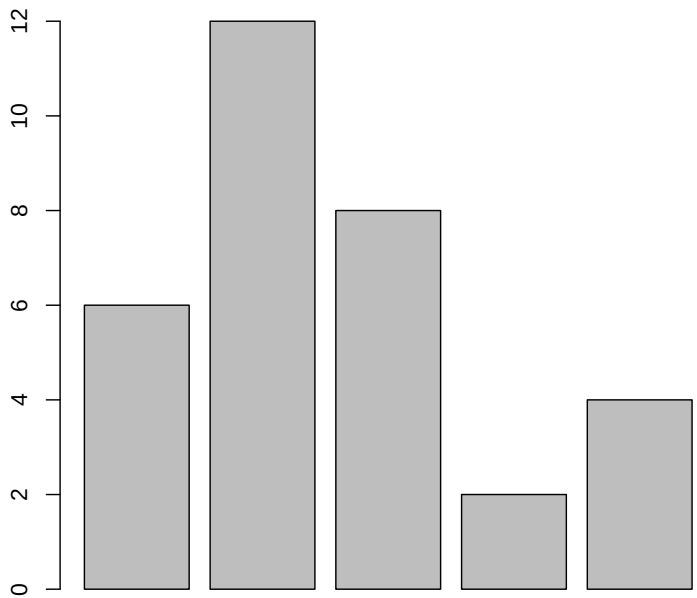
Additional arguments

```
# use more granular bins to see the distribution more clearly
#    'nclass' specifies approximately how many bins
#    can use 'breaks' to specify the exact break points for the bins
#    'main' specifies the title of the plot
hist(mtcars$mpg, nclass = 15, xlab = "mpg", main = "histogram of mpg")
```



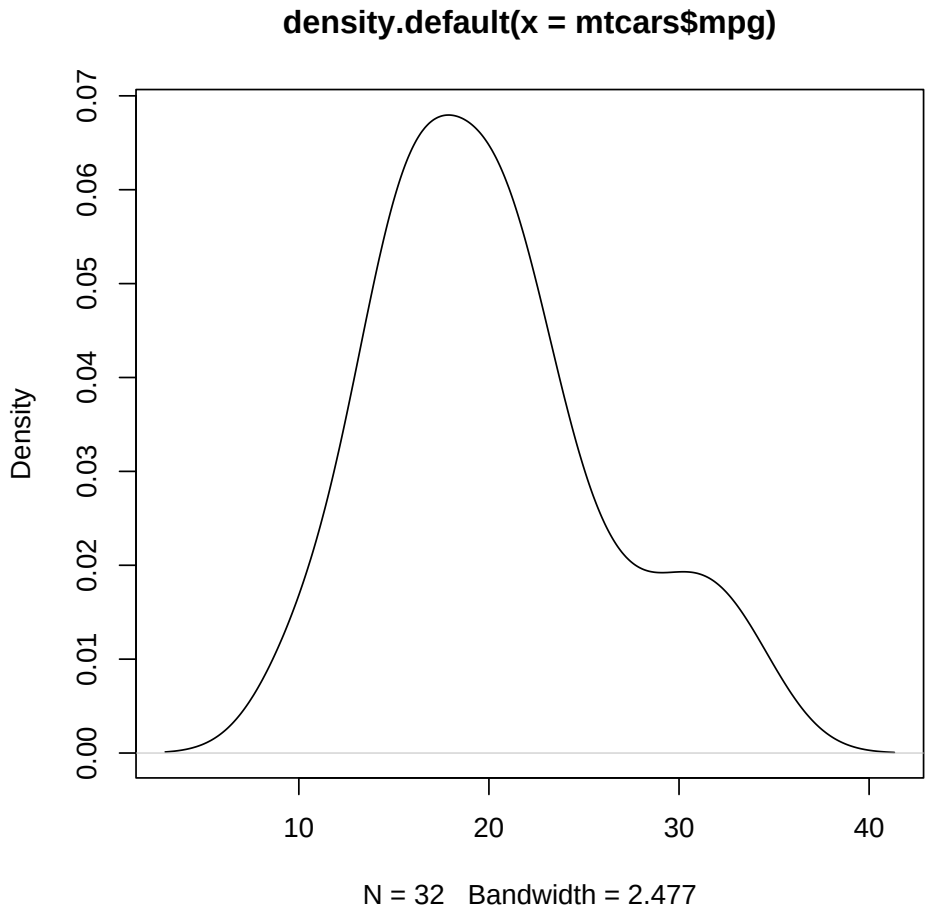
Question: what is the difference between hist() and barplot()?

```r
# hist() can also export frequency table
freq.df <- hist(mtcars$mpg, plot = F)   # freq.df is a list
head(freq.df, n = 2)

## $breaks
## [1] 10 15 20 25 30 35
##
## $counts
## [1]  6 12  8  2  4

barplot(freq.df$counts)
```



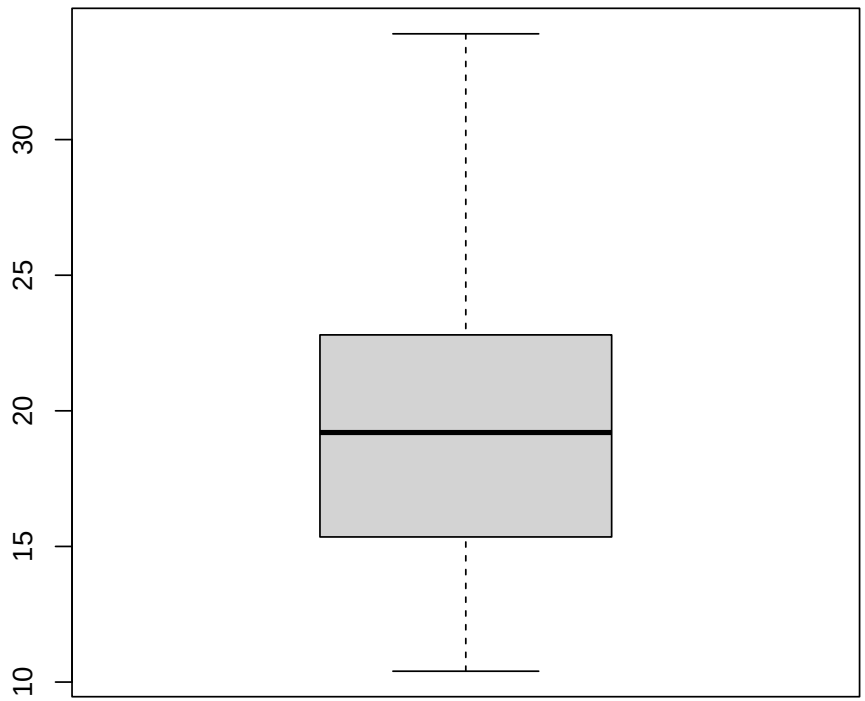
Plot density using density()

```
# plot density estimates
#    note: density() does not do the plotting
plot(density(mtcars$mpg))
```



Visualize quartiles in a distribution using boxplot()

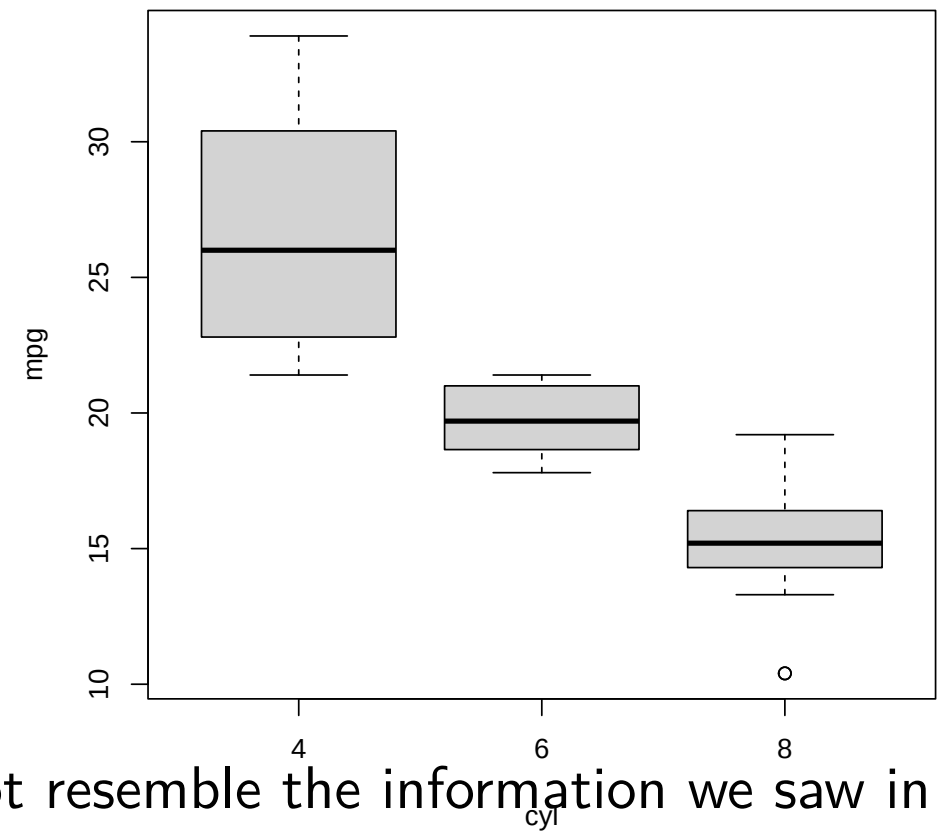
```
# Quartiles of a distribution (median, 25/75%, and some "extreme values")
boxplot(mtcars$mpg)
```



But think of efficiency: how much information are you conveying compared to the previous figure?

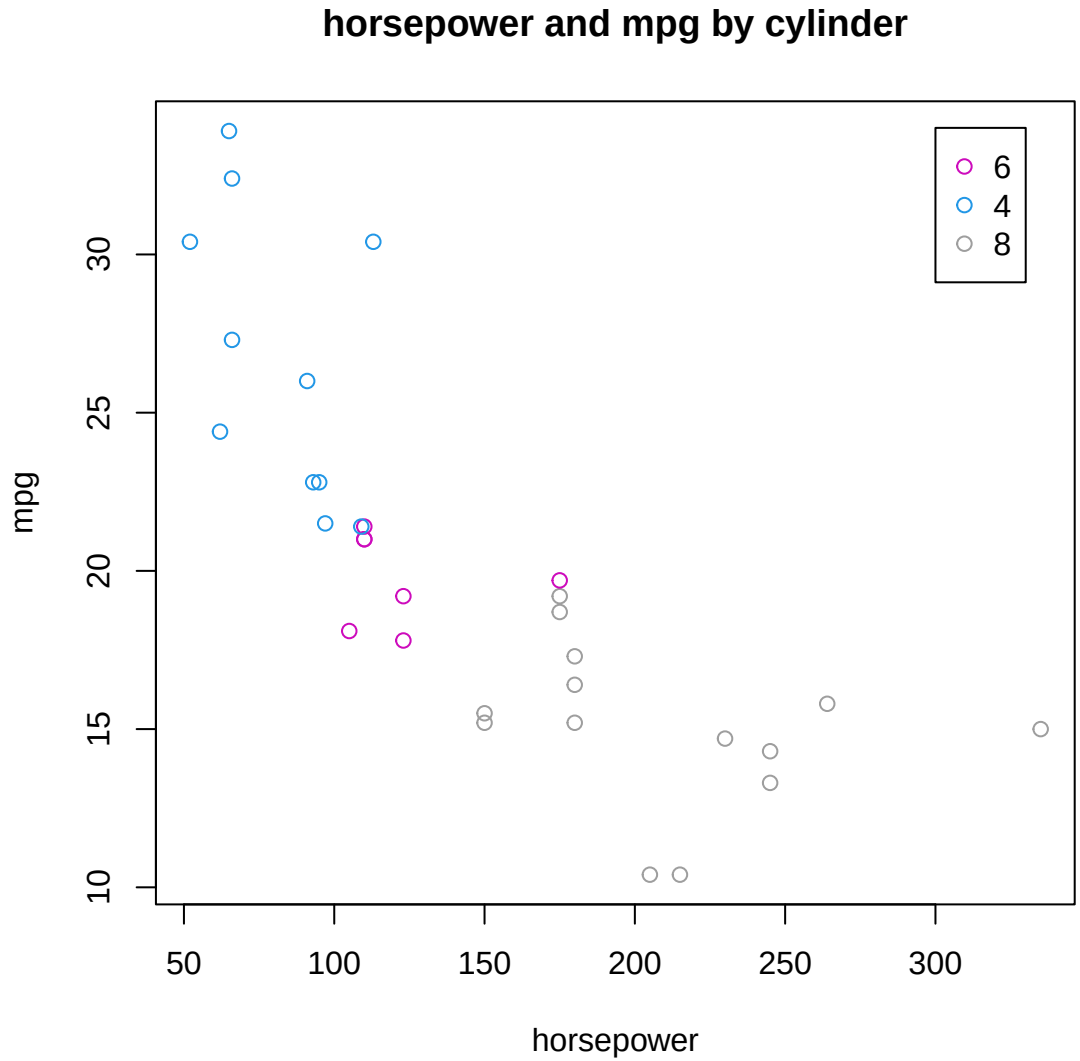
However, real use of barplot() is in combination with an x-variable

```
# Quartiles of a distribution (median, 25/75%, and some "extreme values")
#    the "~" notation refers to a "formula"
#    you'll see more of these in the following weeks
boxplot(mpg ~ cyl, data = mtcars)
```



Does this plot resemble the information we saw in the earlier scatter plot?

Back to the earlier example: what can still be improved?



```r
# I could even enhance this graph to provide more information
plot(
        mtcars$hp, mtcars$mpg,  # data: x and y
        xlab = "horsepower", ylab = "mpg",      # axis label
        col = mtcars$cyl,                       # color scheme
        mar = c(1,1,1,1),                       # margin
        main = "horsepower and mpg by cylinder"
)
legend(
        300, 34,                                # location
        legend = unique(mtcars$cyl),    # legend levels
        col = unique(mtcars$cyl),       # color scheme
        pch = 1                                 # marker type
)
```

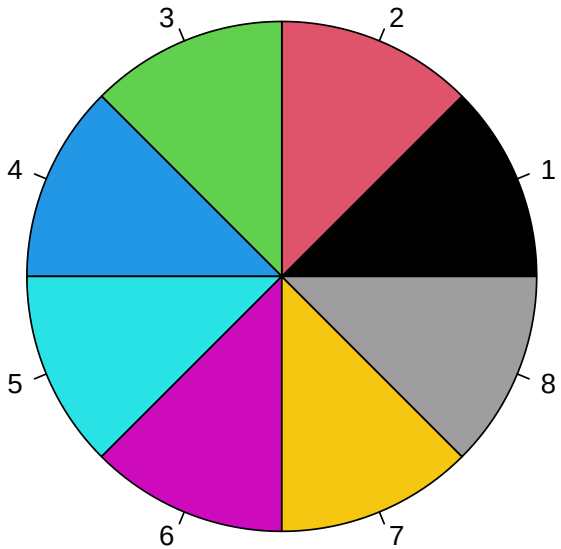
The color parameter

Setting up the color parameter

- There are different systems of defining 'col = '
  - color code
  - rgb(R, G, B, A)
  - RColorBrewer
- We'll give a brief walkthrough of how to use each system

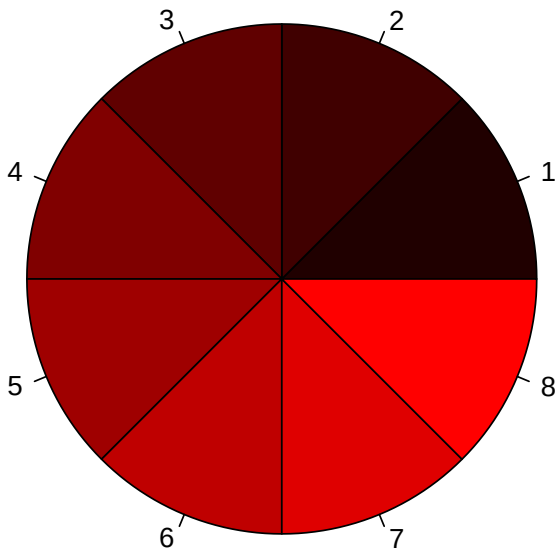
8 basic colors, coded 1-8

```
# create a pie chart that illustrate colors
pie(rep(1, 8), col = 1:8)
```



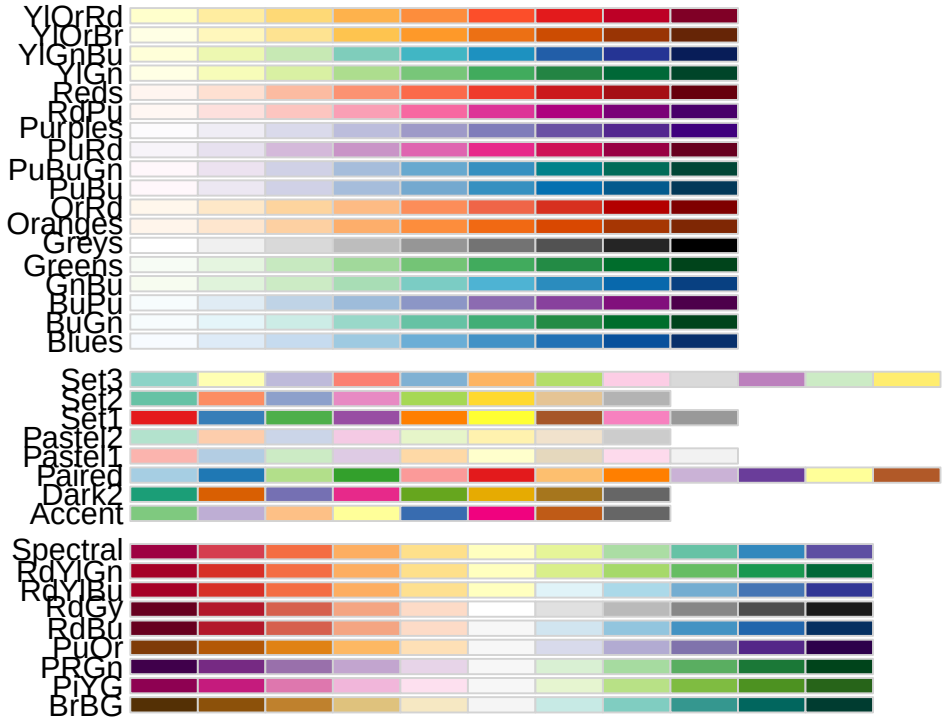
Function rgb(R, G, B, A)

```
# function rgb() specifies red, green, blue and alpha
#    first take different levels of red
pie(rep(1, 8), col = rgb(1:8/8, 0, 0, 1))       # higher is lighter
```

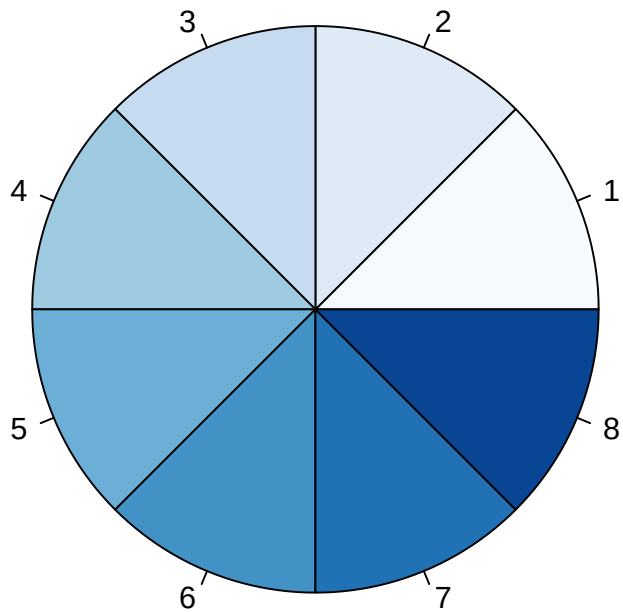


RColorBrewer

```
library("RColorBrewer") # load R Color Brewer
display.brewer.all()    # show palettes
```



```r
# easiest use of R color brewer is to call a class of colors within a palette
blues_vec <- brewer.pal(n = 8, name = "Blues")
pie(rep(1, 8), col = blues_vec)
```



Better: use colors that display the ordering of the value

```
# I could even enhance this graph to provide more information
plot(
        mtcars$hp, mtcars$mpg,  # data: x and y
        xlab = "horsepower", ylab = "mpg",      # axis label
        col = blues_vec[mtcars$cyl],    # color scheme (now uses blues_vec of elements 4, 6, and 8)
        mar = c(1,1,1,1),               # margin
        main = "horsepower and mpg by cylinder"
)
legend(
        300, 34,                        # location
        legend = sort(unique(mtcars$cyl)),      # legend levels
        col = blues_vec[sort(unique(mtcars$cyl))],      # color scheme (same adjustments as before)
        pch = 1                         # marker type
)
```

Better: use colors that display the ordering of the value