IST 3420: Introduction to Data Science and Management

Langtao Chen, Fall 2017

5. Data Summarization and Visualization

Reading

- Appendix: Data Summarization and Visualization
  - http://onlinelibrary.wiley.com/doi/10.1002/9781118874059.app1/pdf
- Comprehensive Guide to Data Visualization in R
  - https://www.analyticsvidhya.com/blog/2015/07/guide-data-visualization-r/
- Chart Suggestions—A Thought-Starter
  - http://extremepresentation.typepad.com/files/choosing-a-good-chart-09.pdf

Learning Objectives

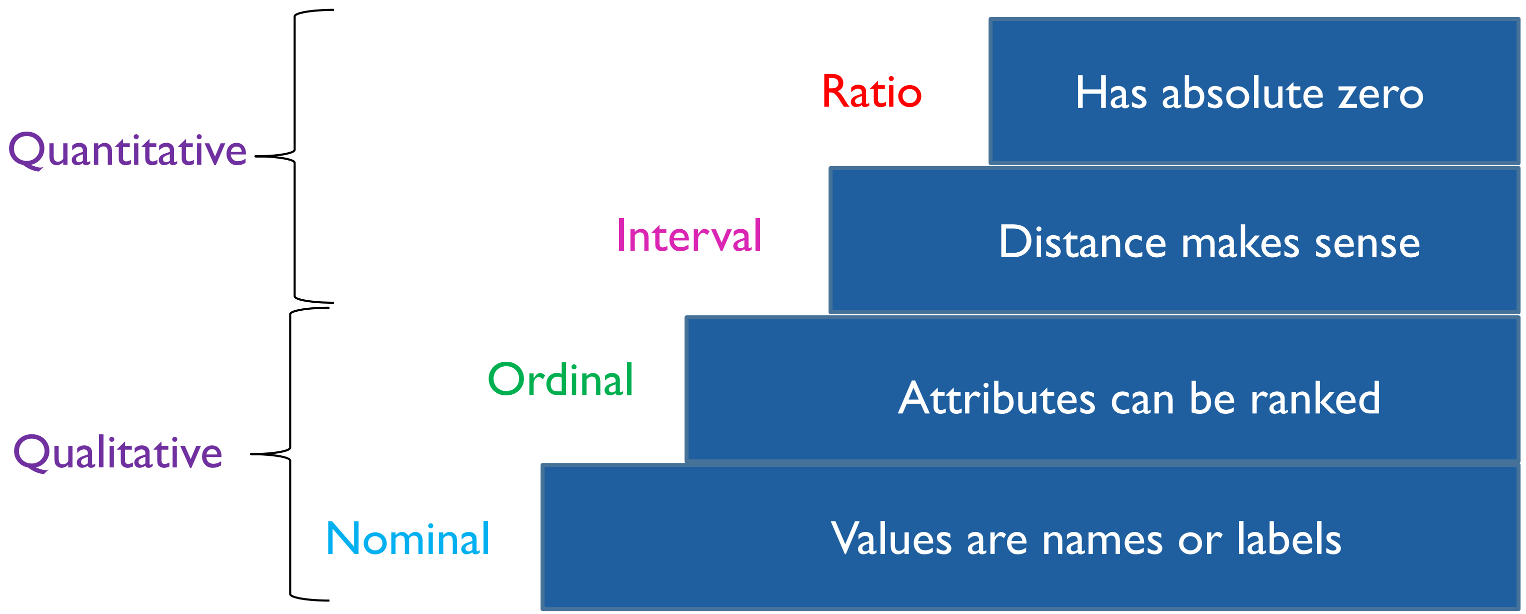
- Be able to choose appropriate tabular and basic graphic methods for different types of data (qualitative vs. quantitative)
- Understand tabular and basic graphic methods and be able to interpret these visualization results
- Be able to use ggplot2 to visualize data
- Understand spatial data structure
- Be able to construct advanced visualization such as spatial plots, hexagon binning, mosaic plot, heat map

Agenda

- Introduction to Data Summarization and Visualization
- Tabular and Basic Graphical Methods
- Graphical Parameters
- Visualizing Data Using ggplot2 Package
- Visualizing Spatial Data
- Some Advanced Visualization Methods

Recap: Hierarchy of Measurement Scales

- A higher level scale contains all properties of its lower scale.
- From lower to higher levels, analysis tends to be more comprehensive. Improper use of lower level scales suffers information loss in the data
- In general, we prefer a higher scale of measurement than a lower one.

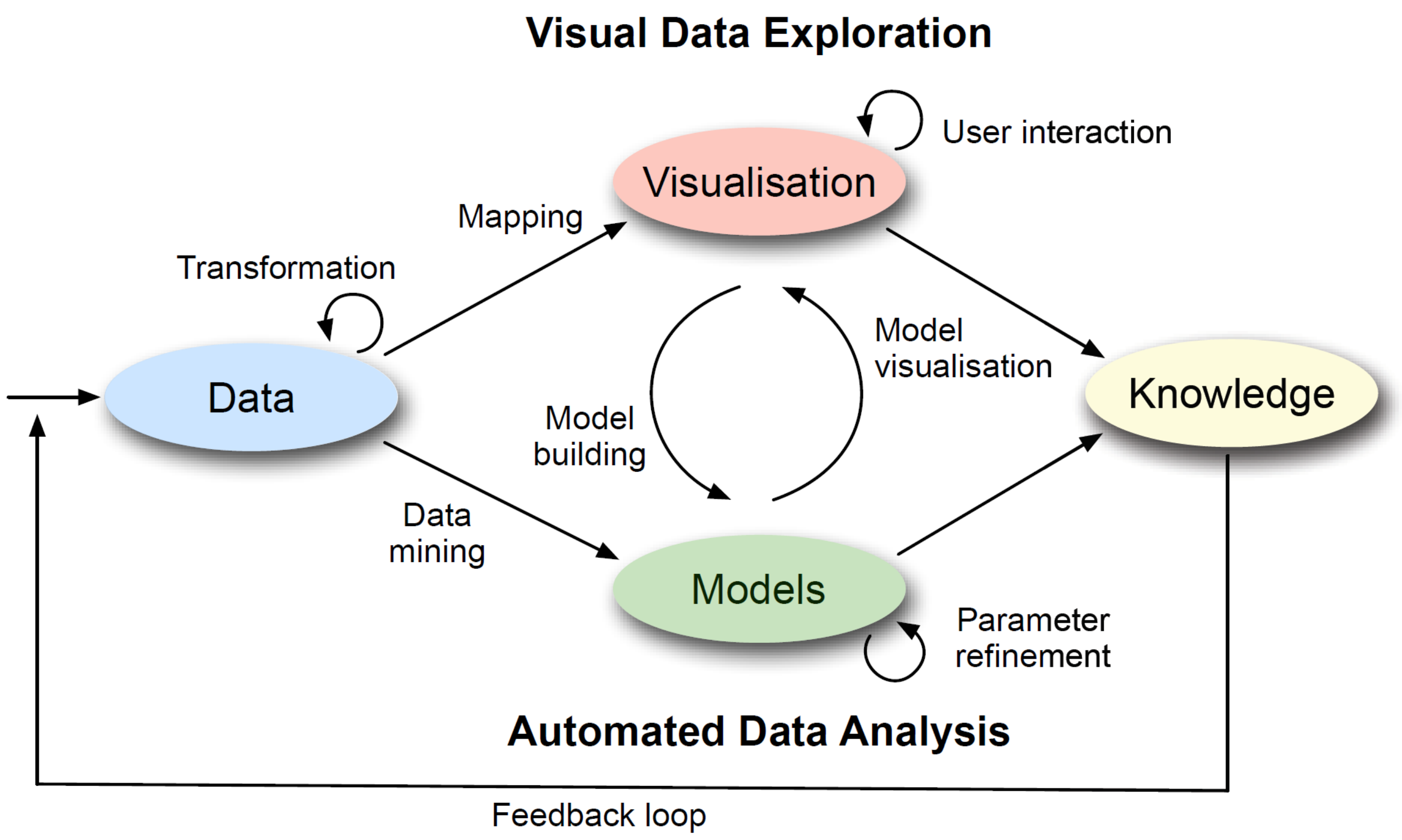


Summarize and Visualize Data

- Summarizing and visualizing data facilitates communication of data analysis to the users or customers.
- Data types
  - Qualitative data (nominal, interval)
  - Quantitative data (ordinal, ratio)
- Approaches
  - Tabular methods
  - Graphical methods

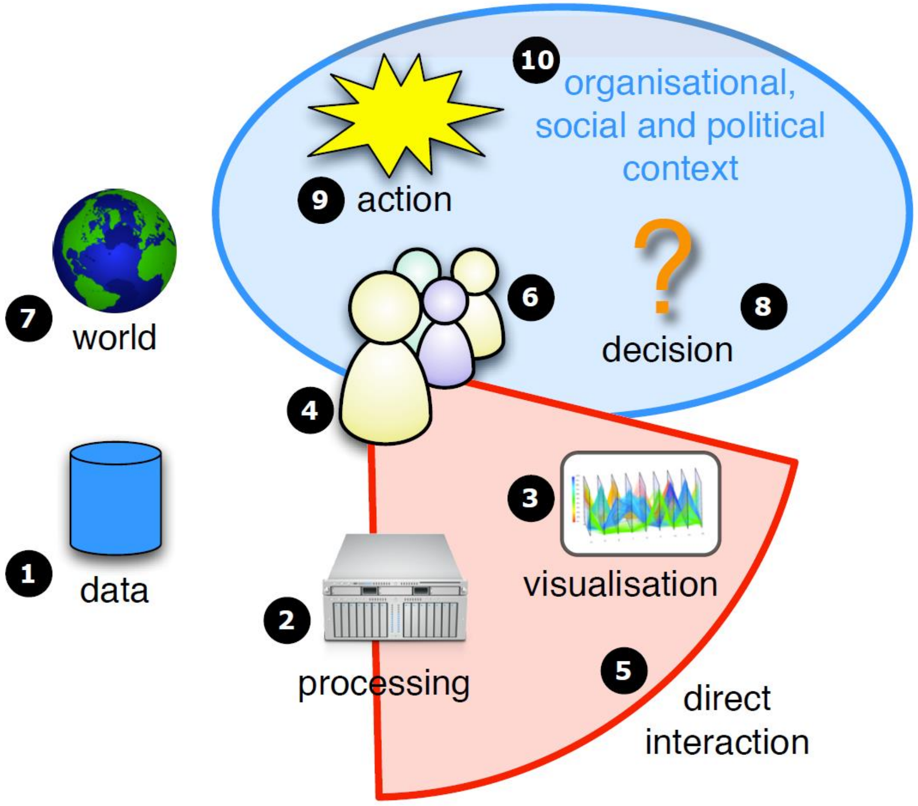
Data Visualization

- "A picture is worth a thousand words"
- "The purpose of visualization is insight, not pictures"



Visualization Amplifies Human Cognition

- Information visualization amplifies human cognitive capability in six basic ways:
  - by increasing cognitive resources, such as by using a visual resource to expand human working memory,
  - by reducing search, such as by representing a large amount of data in a small space,
  - by enhancing the recognition of patterns, such as when information is organized in space by its time relationships,
  - by supporting the easy perceptual inference of relationships that are otherwise more difficult to induce,
  - by perceptual monitoring of a large number of potential events, and
  - by providing a manipulable medium that, unlike static diagrams, enables the exploration of a space of parameter values



Why Visualization is Important? Anscombe's Quartet

- In 1973 Francis Anscombe demonstrated four datasets:

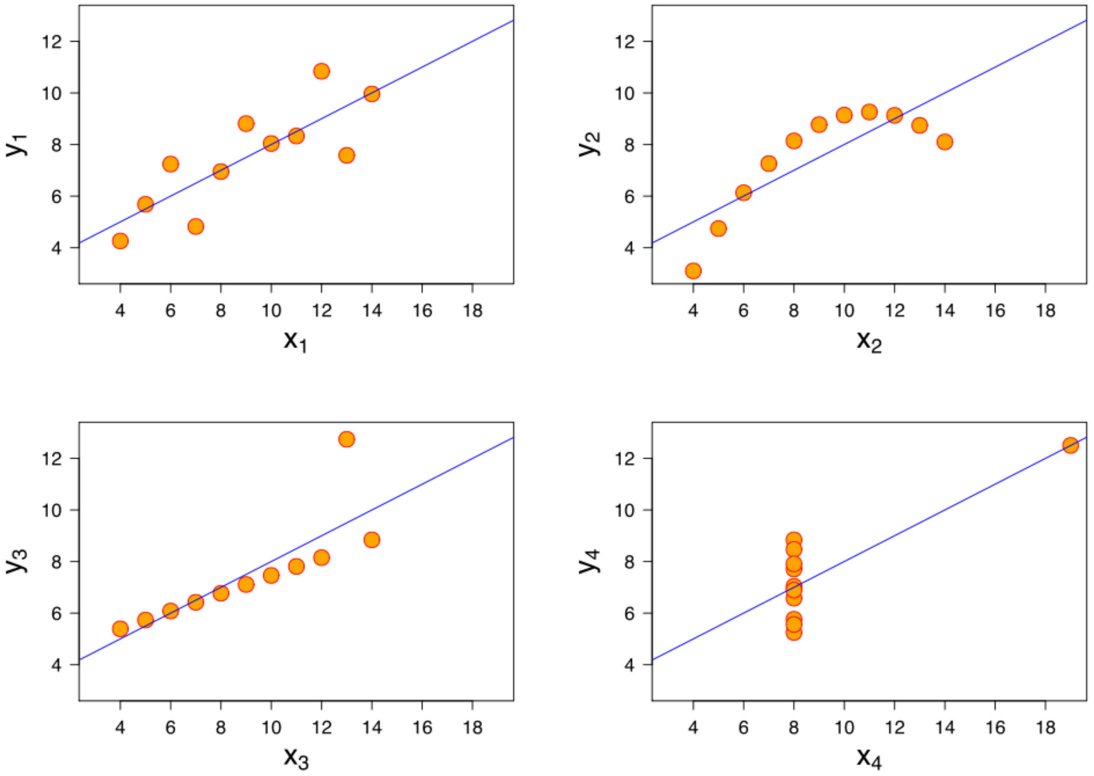
I		II		III		IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

Source: https://en.wikipedia.org/wiki/Anscombe's_quartet

Why Visualization is Important? Anscombe Quartet

- The dangers of summary statistics:
  - All four datasets are identical when examined using simple summary statistics, but vary considerably when graphed.

Property	Value
Mean of *x*	9
Sample variance of *x*	11
Mean of *y*	7.50
Sample variance of *y*	4.125
Correlation between *x* and *y*	0.816
Linear regression line	$y = 3.00 + 0.500x$



Source: https://en.wikipedia.org/wiki/Anscombe's_quartet

Summarize and Visualize Data

Using Tabular and Basic Graphical Methods

Overview of Basic Visualization Methods

Category	Method	Qualitative Data	Quantitative Data
Tabular Methods	Frequency Distribution	Yes	Yes
	Relative Frequency Distribution	Yes	Yes
	Percent Frequency Distribution	Yes	Yes
	Cumulative Frequency Distribution		Yes
	Cumulative Rel. Freq. Distribution		Yes
	Crosstabulation	Yes	Yes
Basic Graphical Methods	Bar Plot	Yes	
	Pie Chart	Yes	
	Dot Plot		Yes
	Box Plot		Yes
	Density Plot		Yes
	Line Chart		Yes
	Scatter Plot		Yes

Tabulate A Single Variable

- Both qualitative and quantitative data
  - Frequency distribution: shows the frequency (or count) of items
  - Relative frequency distribution: shows frequency proportion of items
- Quantitative data
  - Accumulative frequency distribution: shows frequency below a level
  - Accumulative relative frequency distribution: shows frequency proportion below a level

R Code: Tabulate Qualitative Data

```
## Part 1.Tabulate Qualitative Data ##
# Calculate frequency
gear.freq <- table(mtcars$gear)
gear.freq
cbind(gear.freq)  # Print frequency in column format

# Calculate relative frequency
gear.rel.freq <- gear.freq/nrow(mtcars)
gear.rel.freq
cbind(gear.rel.freq)  # Print relative frequency

cbind(gear.freq, gear.rel.freq)  # Print frequency and relative frequency
```

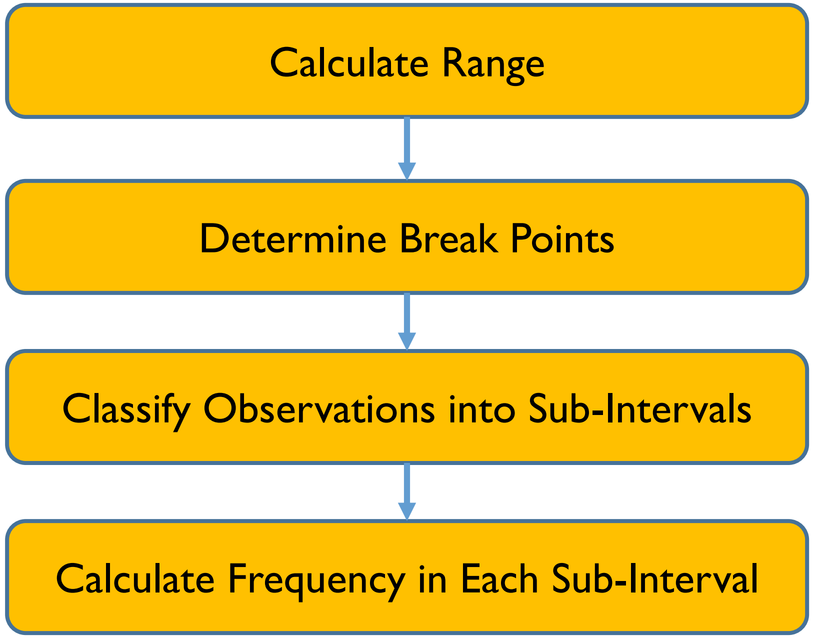
```
> cbind(gear.freq, gear.rel.freq
  gear.freq gear.rel.freq
3        15       0.46875
4        12       0.37500
5         5       0.15625
```

Interpretation: Most cars have 3 or 4 gears.

Tabulate Quantitative Data

- Quantitative data are measured in ordinal and/or ratio scales.
- Directly counting the number of unique values is meaningless.

Procedure of Tabulating Quantitative Data



R Code: Tabulate Quantitative Data

```
mpg <- mtcars$mpg
# Calculate range
range(mpg)
# Determine break points
breaks <- seq(10,35, by = 5)
# Classify observations into sub-intervals
mpg.cut <- cut(mpg, breaks, right = FALSE)
# Calculate frequency in each sub-interval
mpg.freq <- table(mpg.cut)
cbind(mpg.freq)  # Print in column format
# Calculate cumulative frequency
mpg.cum.freq = cumsum(mpg.freq)
cbind(mpg.cum.freq)
# Calculate relative frequency
mpg.rel.freq <- mpg.freq/nrow(mtcars)
cbind(mpg.rel.freq)
# Calculate cumulative relative frequency
mpg.cum.rel = cumsum(mpg.rel.freq)
cbind(mpg.cum.rel)
# Print all in column format
cbind(mpg.freq, mpg.cum.freq, mpg.rel.freq,
mpg.cum.rel)
```

```
> cbind(mpg.freq, mpg.cum.freq, mpg.rel.freq, mpg.cum.rel)
        mpg.freq mpg.cum.freq mpg.rel.freq mpg.cum.rel
[10,15)        5            5      0.15625     0.15625
[15,20)       13           18      0.40625     0.56250
[20,25)        8           26      0.25000     0.81250
[25,30)        2           28      0.06250     0.87500
[30,35)        4           32      0.12500     1.00000
```

Tabulate 2 or More Categorical Variables

```
# Create a two-way frequency table
cyl_gear <- table(mtcars$cyl,mtcars$gear)
cyl_gear

# Calculate table margin
margin.table(cyl_gear,1)
margin.table(cyl_gear,2)

# Create a three-way frequency table
cyl_gear_carb <- table(mtcars$cyl,mtcars$gear, mtcars$carb)
cyl_gear_carb
ftable(cyl_gear_carb) # Print as flat contingency table

# Use xtabs() to create contingency tables
xtabs(~ cyl + gear, data = mtcars)
ftable(xtabs(~ cyl + gear + carb, data = mtcars))
```

```
> xtabs(~ cyl + gear, data = mtcars)
   gear
cyl  3  4  5
  4  1  8  2
  6  2  4  1
  8 12  0  2
> ftable(xtabs(~ cyl + gear + carb,
              data = mtcars))
         carb 1 2 3 4 6 8
cyl gear
4   3         1 0 0 0 0 0
    4         4 4 0 0 0 0
    5         0 2 0 0 0 0
6   3         2 0 0 0 0 0
    4         0 0 0 4 0 0
    5         0 0 0 0 1 0
8   3         0 4 3 5 0 0
    4         0 0 0 0 0 0
    5         0 0 0 1 0 1
```

Cross Tabulate Quantitative Variables

- Like creating frequency table, cross-tabulating quantitative variables involves using a sequence of sub-intervals.

```
# Cross tabulate mpg and cylinder
car1 <- mtcars
range(car1$mpg)
break1 <- seq(10,35, by = 5)
break1
car1$mpg.cut <- cut(car1$mpg, breaks, right = FALSE)
xtabs(~ mpg.cut + cyl, data = car1)
```

```
> xtabs(~ mpg.cut + cyl, data = car1)
         cyl
mpg.cut   4 6 8
  [10,15) 0 0 5
  [15,20) 0 4 9
  [20,25) 5 3 0
  [25,30) 2 0 0
  [30,35) 4 0 0
```

Interpretation: Cars with more cylinders tend to have lower miles per gallon.

Visualize Data in R

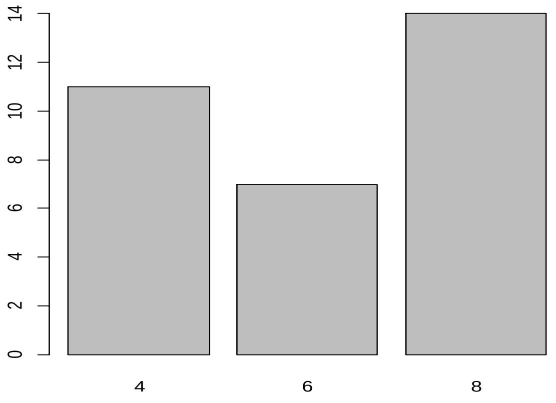
- R provides strong data visualization capabilities
- Basic graphs for qualitative variables
  - bar plots (simple, stacked, grouped)
  - pie charts (simple, annotated)
- Basic graphs for quantitative variables
  - dot plots
  - boxplots
  - density plots (histograms and kernel density plots)
  - line charts
  - scatter plots
- Advanced graphs

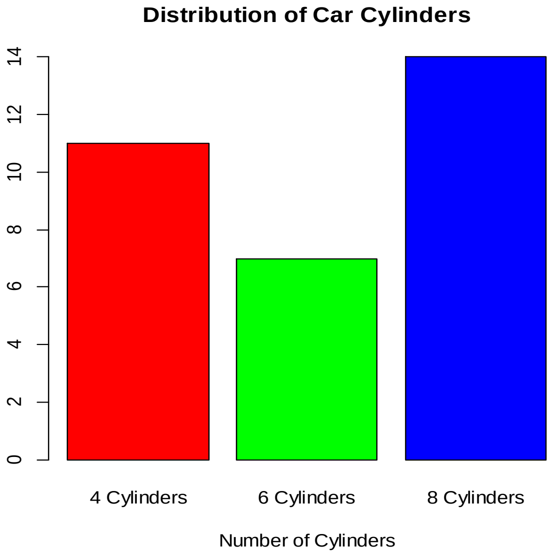
Bar Plots (Qualitative Data)

- Syntax: barplot(height, ...)
- Three types
  - Simple bar plots
  - Stacked bar plots
  - Grouped bar plots

R Code: Simple Bar Plots

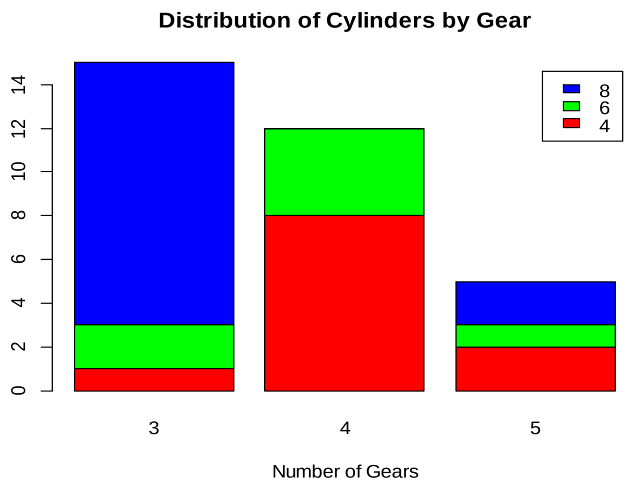
```
## Simple Bar Plots ##
car.cyl <- table(mtcars$cyl)
car.cyl
str(car.cyl)

# A simple bar plot
barplot(car.cyl)

# A simple bar plot with more details
barplot(car.cyl,
        main = "Distribution of Car Cylinders",
        xlab = "Number of Cylinders",
        names.arg = c("4 Cylinders", "6 Cylinders", "8 Cylinders"),
        col = rainbow(3))
```

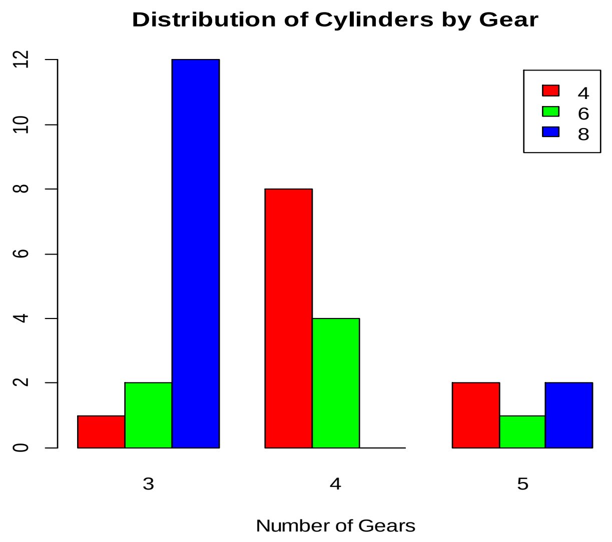




R Code: Stacked and Grouped Bar Plots

```
## Stacked Bar Plots ##
cyl.gear <- table(mtcars$cyl,mtcars$gear)
cyl.gear
str(cyl.gear)

barplot(cyl.gear,
        main="Distribution of Cylinders by Gear",
        xlab="Number of Gears",
        col=rainbow(3),
        legend = rownames(cyl.gear))

## Grouped Bar Plots ##
barplot(cyl.gear,
        main="Distribution of Cylinders by Gear",
        xlab="Number of Gears",
        col=rainbow(3),
        legend = rownames(cyl.gear),
        beside=TRUE)
```



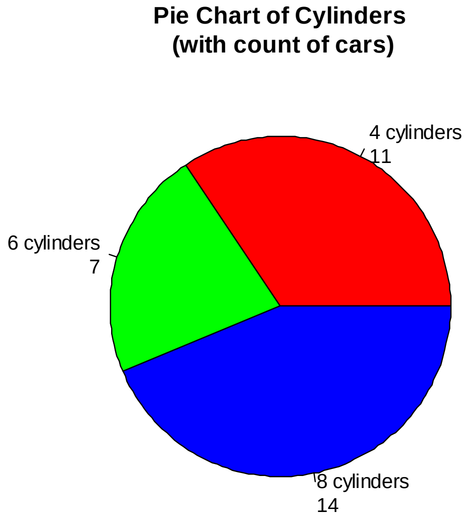


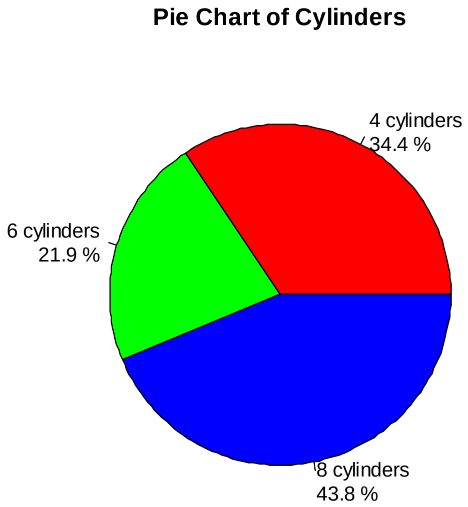
Pie Plots (Qualitative Data)

- Syntax: pie(x, labels = names(x),...)

R Code: Pie Chart

```
## Pie Plot ##
car.cyl <- table(mtcars$cyl)
lb1 <- paste(names(car.cyl), " cylinders\n", car.cyl, sep="")
pie(car.cyl,
    labels = lb1,
    main="Pie Chart of Cylinders\n (with count of cars)",
    col=rainbow(length(lb1))
)

# Calculate percentage
pct <- round(car.cyl/sum(car.cyl)*100, digits = 1)
lb2 <- paste(names(car.cyl), " cylinders\n", paste(pct,"%"),
sep="")
pie(car.cyl,
    labels = lb2,
    main="Pie Chart of Cylinders",
    col=rainbow(length(lb2))
)
```





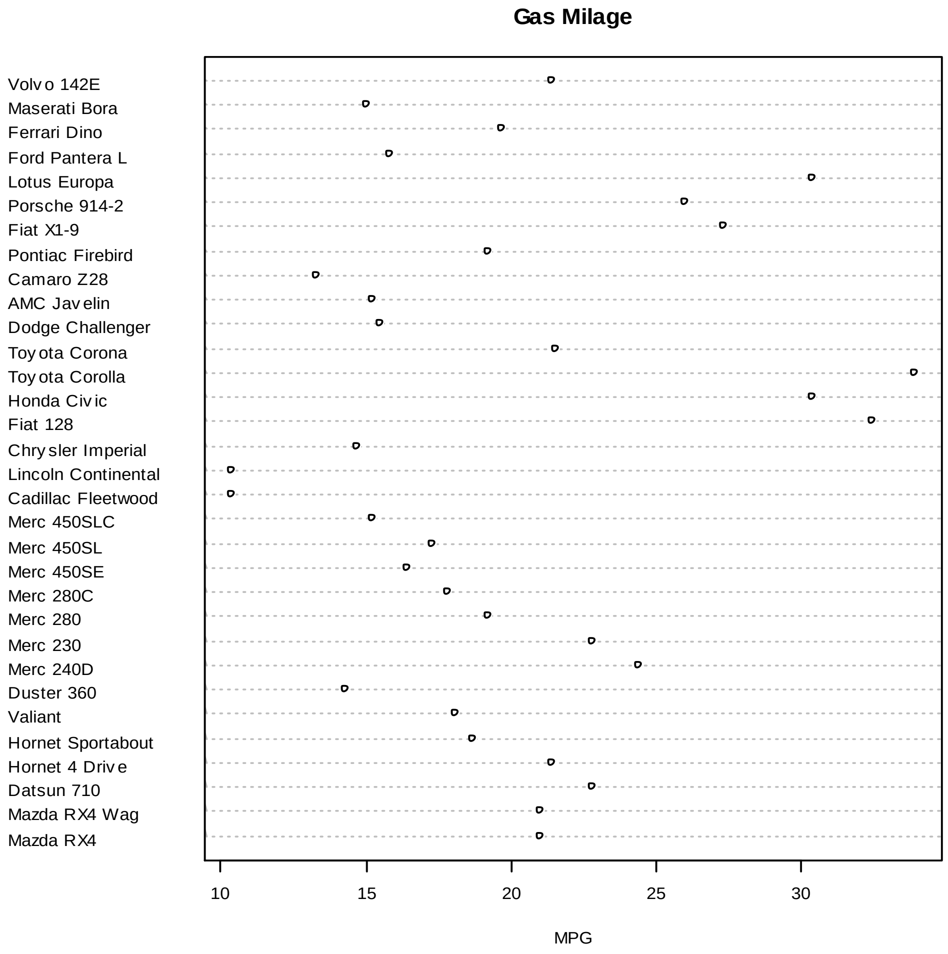
Dot Plots (Quantitative Data)

- Syntax: dotchart(x, labels = ,...)
  - cex: the character size to use (a very useful setting to avoid label overlap).
  - groups: an optional factor indicating how the elements of x are grouped.
  - color: the color(s) to be used for points and labels.
  - gcolor: the single color to be used for group labels and values.

R Code: Dot Plots

- A simple dot plot

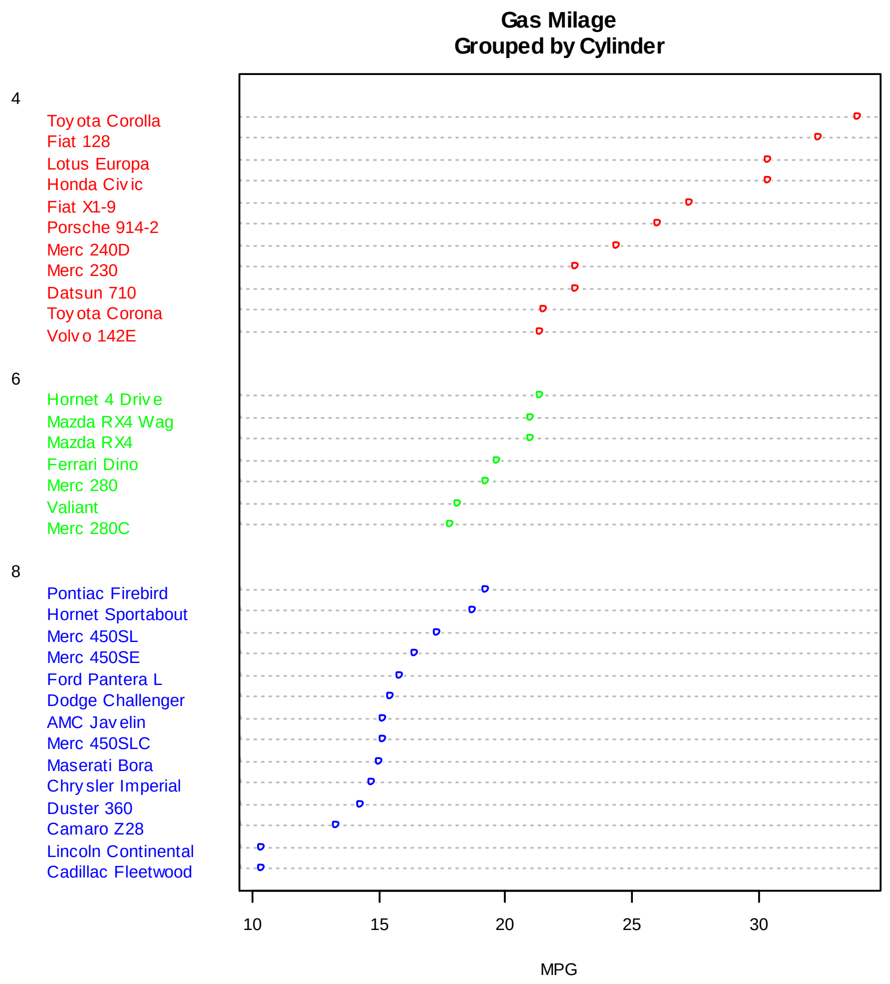
```
#A simple dot plot
dotchart(mtcars$mpg,
        labels = row.names(mtcars),
        cex = 0.6,
        main="Gas Milage",
        xlab = "MPG")
```



R Code: Dot Plots

- A dot plot with grouped elements

```
# A dot plot with grouped elements
x <- mtcars[c("mpg","cyl")]
x <- x[order(x$mpg),]

x$color <- x$cyl
x$color <- dplyr::recode(x$color,`4` = "red",
`6` = "green", `8` = "blue")
x$cyl <- as.factor(x$cyl) # In order to show
factor label in dotchart
str(x)

dotchart(x$mpg,
        labels = row.names(x),
        cex = 0.6,
        main="Gas Milage\nGrouped by
Cylinder",
        xlab = "MPG",
        groups = x$cyl,
        color = x$color)
```



A Five-Number Summary (Quantitative Data)

- Use the following 5 numbers to summarize data
  - 1. Smallest value
  - 2. First quartile (Q1)
  - 3. Median (Q2)
  - 4. Third quartile (Q3)
  - 5. Largest value

Procedure:
Step 1: Sort the data in ascending order;
Step 2: Calculate the smallest value, the three quartiles, and the largest value.

23 12 24 55 60 34 20 17 19 37 43 51

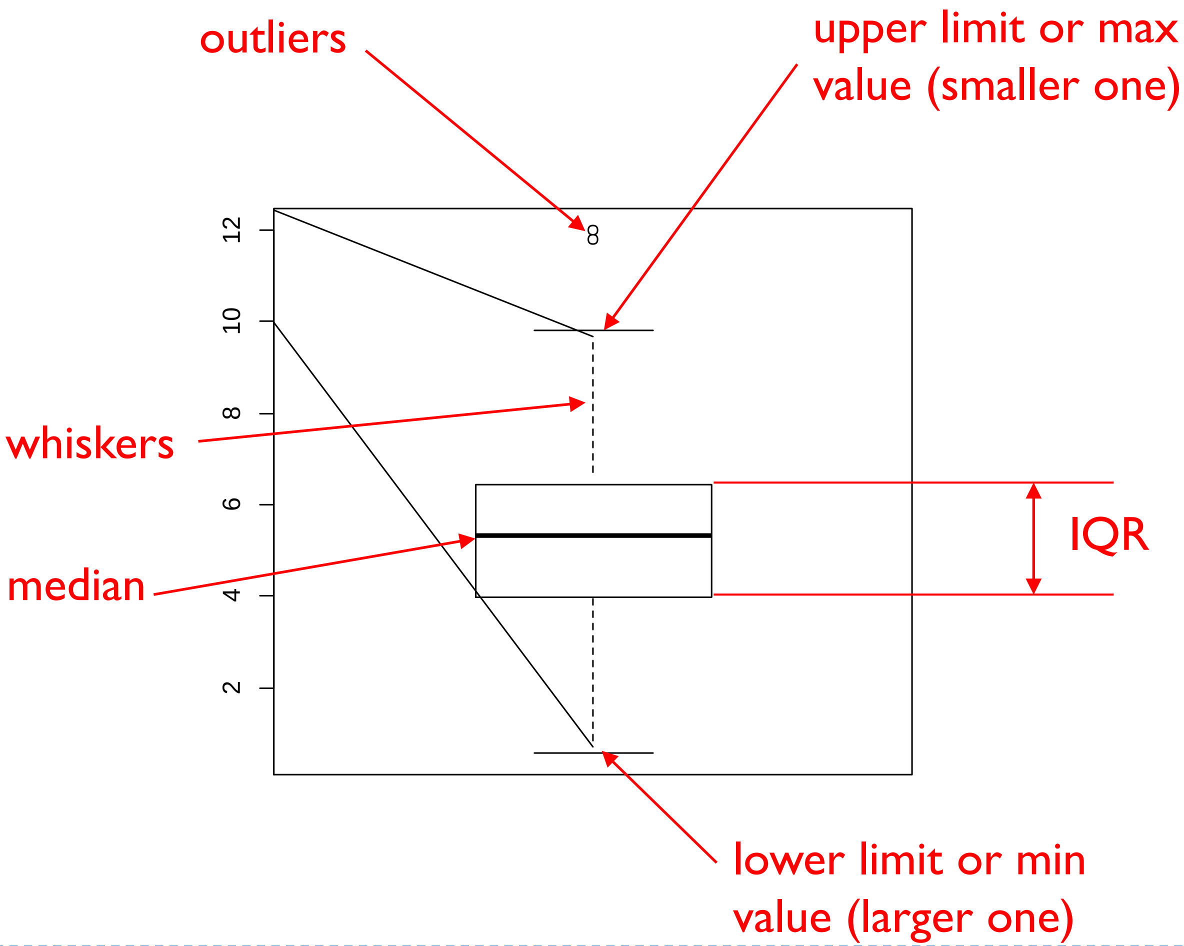
↓

12 17 19 | 20 23 24 | 34 37 43 | 51 55 60

Q1=19.75 Q2=29 Q3=45 (Sample Quantile Type 7)

Boxplots (a.k.a. Box-and-Whisker Plots)

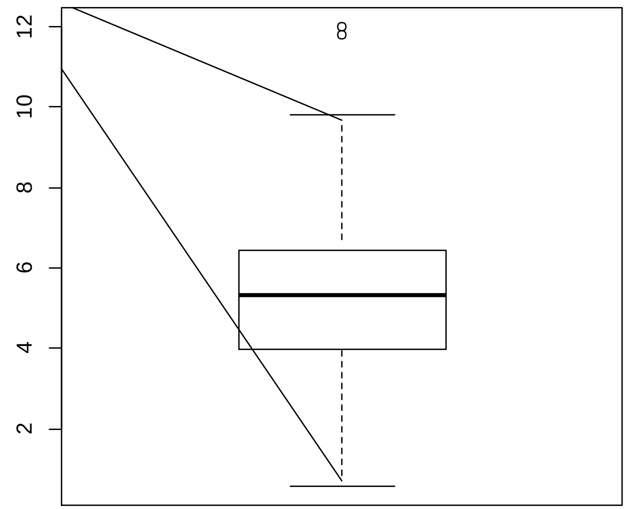
- Box-and-whisker plots summarize data based on a five- number summary.
  - First quartile (Q1)
  - Median
  - Third quartile (Q3)
  - Interquartile range
    IQR = Q3 - Q1
  - Upper limit = Q3 + 1.5IQR
  - Lower limit = Q1 – 1.5IQR
  - Outliers: beyond the range
    [Q1 – 1.5IQR, Q3 + 1.5IQR]

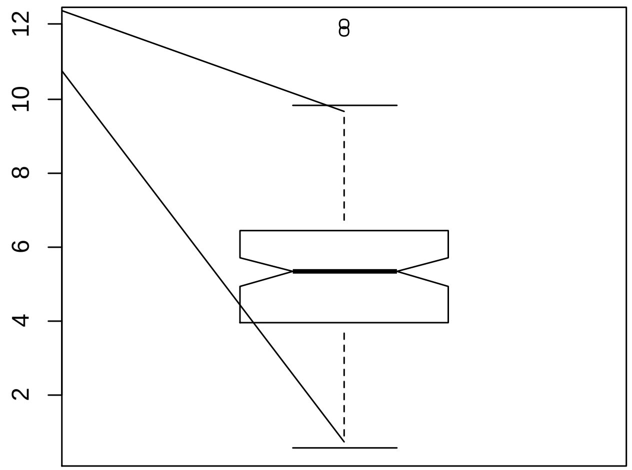


R Code: Box Plot

- Simple box plot

```
# Box plot of a variable
set.seed(1)
x1 <- rnorm(100, mean = 5, sd = 2)
x1[88] <- 12; x1[89] <- 11.8  # Set two outliers
quantile(x1)
boxplot(x1)
boxplot(x1,notch=TRUE)  # Draw a notch in each side of the box
```

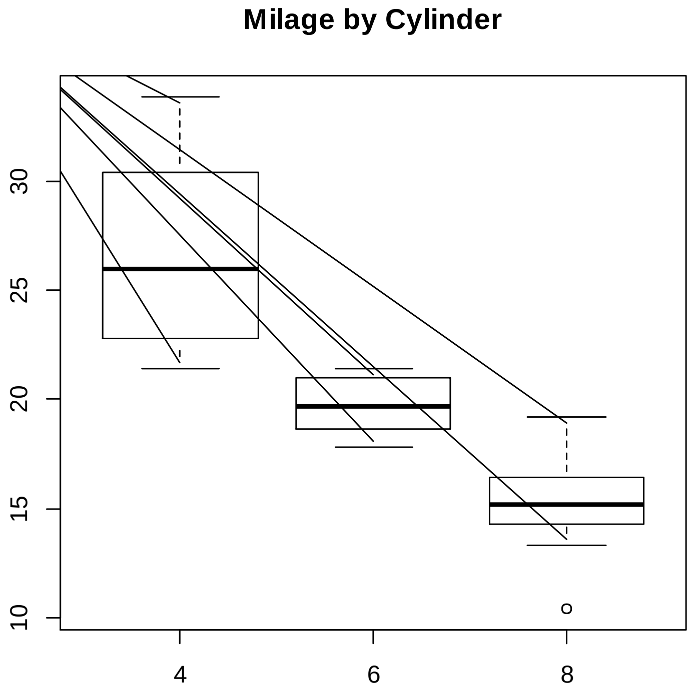


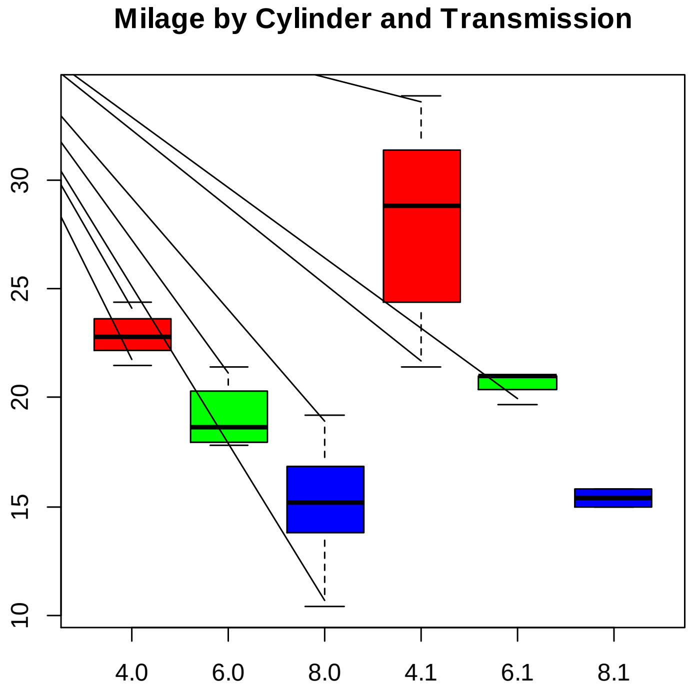


R Code: Box Plot

- Box plot by group

```
# Box plot of a variable by group
boxplot(mpg ~ cyl, data = mtcars, main = "Mileage by Cylinder")

boxplot(mpg ~ cyl*am, data = mtcars, main = "Mileage by Cylinder and Transmission", col = rainbow(3))
```





Density Plots (Quantitative Data)

- Histogram
  - A histogram visualize the distribution of a variable in terms of frequency, relative frequency, or percent frequency.
  - A histogram can be thought of as a simplified kernel density plot.

- Kernel Density Plot
  - A kernel density plot uses a kernel algorithm to smooth frequencies over the bins.
  - This yields a smoother probability density function, which will in general more accurately reflect distribution of the underlying variable.

Source: https://en.wikipedia.org/wiki/Histogram

Histograms

- A histogram reveals the shape of distribution.

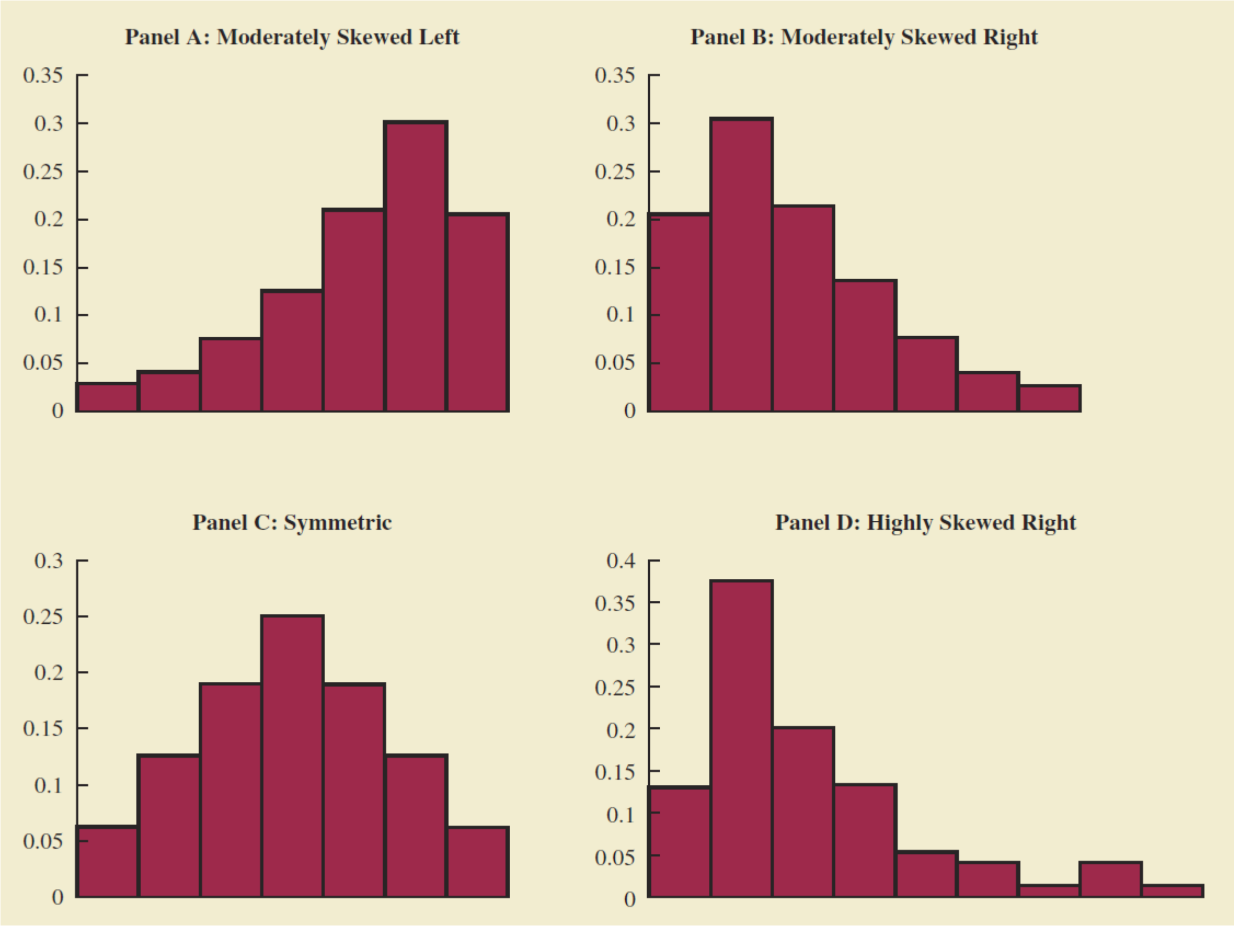
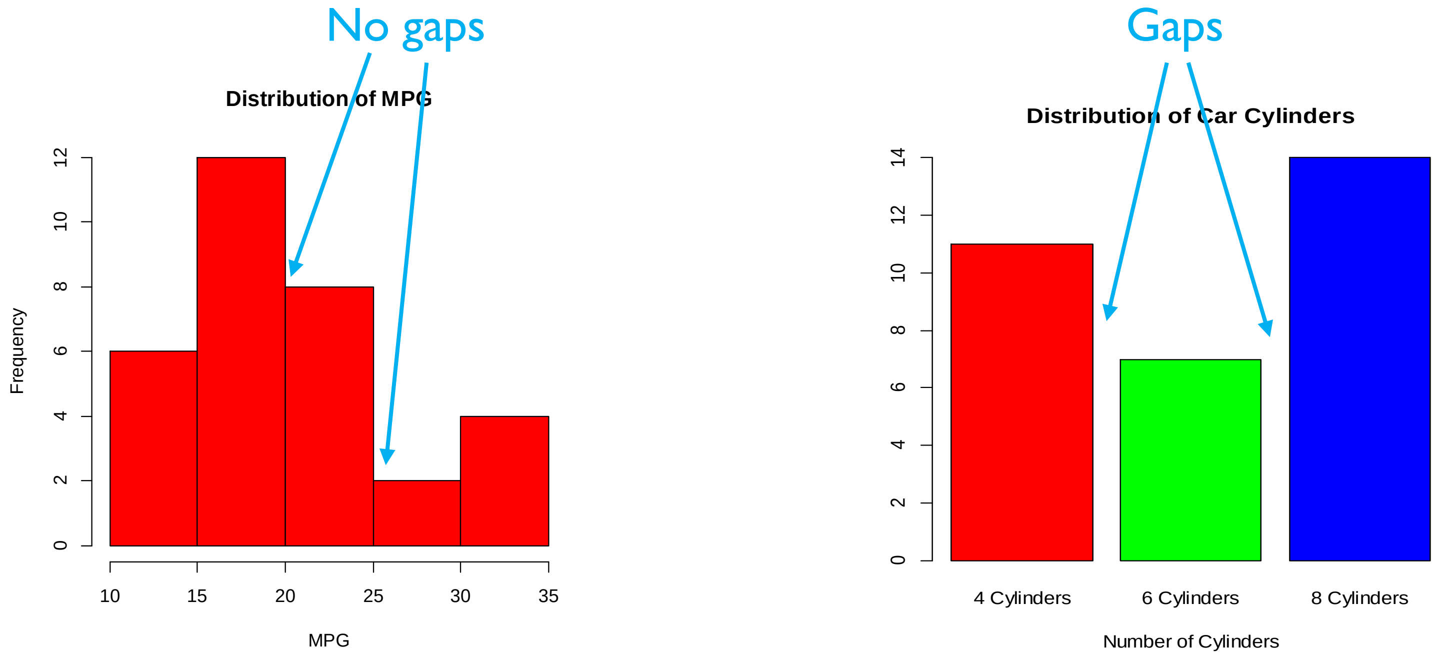


Image source: Anderson et al. "*Statistics for business & economics*" *(11e)*

Histograms vs. Bar Charts

- Histograms are used to visualize the distribution of quantitative data.
- Bar charts are used to summarize qualitative data.

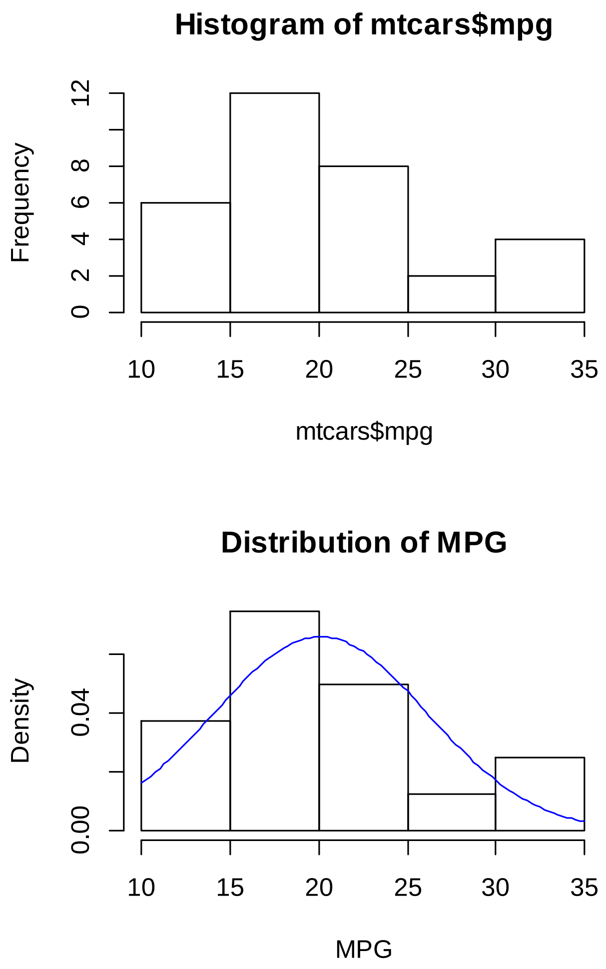


Adjacent bins (usually equal size)

Categories

R Code: Histograms

```
# Only show 2*1 plots on a page
par(mfrow=c(2,1))
hist(mtcars$mpg)

hist(mtcars$mpg,
    main = "Distribution of MPG",
    xlab = "MPG",
    probability = TRUE)
# Add a normal curve
curve(dnorm(x, mean=mean(mtcars$mpg),
          sd=sd(mtcars$mpg)),
     col="blue",
     add=TRUE)

par(mfrow=c(1,1))
```



Interpretation: The distribution of mpg is moderately right skewed.

Determine the Number of Bins

- We can set the breaks parameter to other numbers.

- Different bin sizes can reveal different features of the data:
  - Less bins lead to oversmoothing and bias
  - More bins result in imprecise estimation due to extra noise

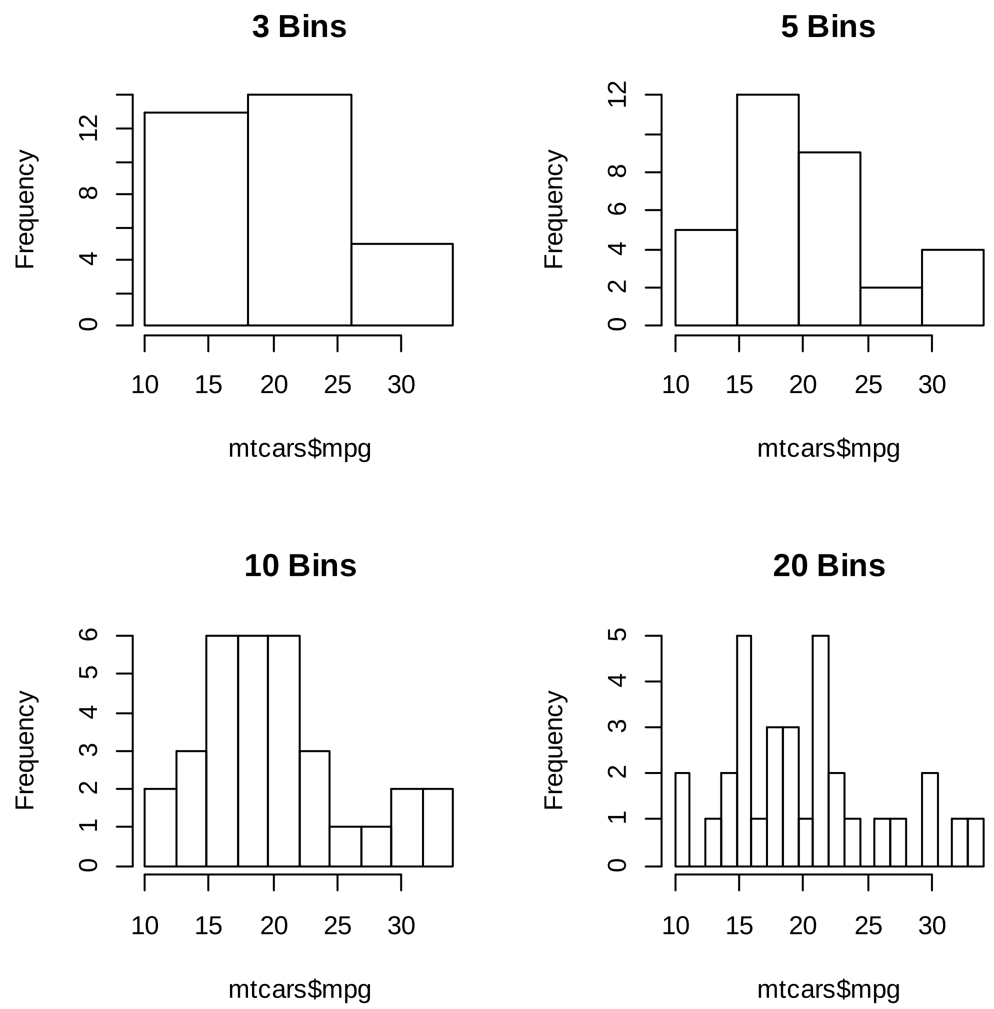
- There is no "best" number of bins.

- By default, R hist() function use Sturges' Rule to determine the number of breaks:

$$k = \text{ceiling}(\log_2 n+1)$$

R Code: Set the Number of Bins

```
# Range of mpg: 10.4~33.9
range(mtcars$mpg)

# Only show 2*2 plots on a page
par(mfrow=c(2,2))
hist(mtcars$mpg, main = "3 Bins",
     breaks = seq(10,34, l= 4))

hist(mtcars$mpg, main = "5 Bins",
     breaks = seq(10,34, l= 6))

hist(mtcars$mpg, main = "10 Bins",
     breaks = seq(10,34, l= 11))

hist(mtcars$mpg, main = "20 Bins",
     breaks = seq(10,34, l= 21))

par(mfrow=c(1,1))
```

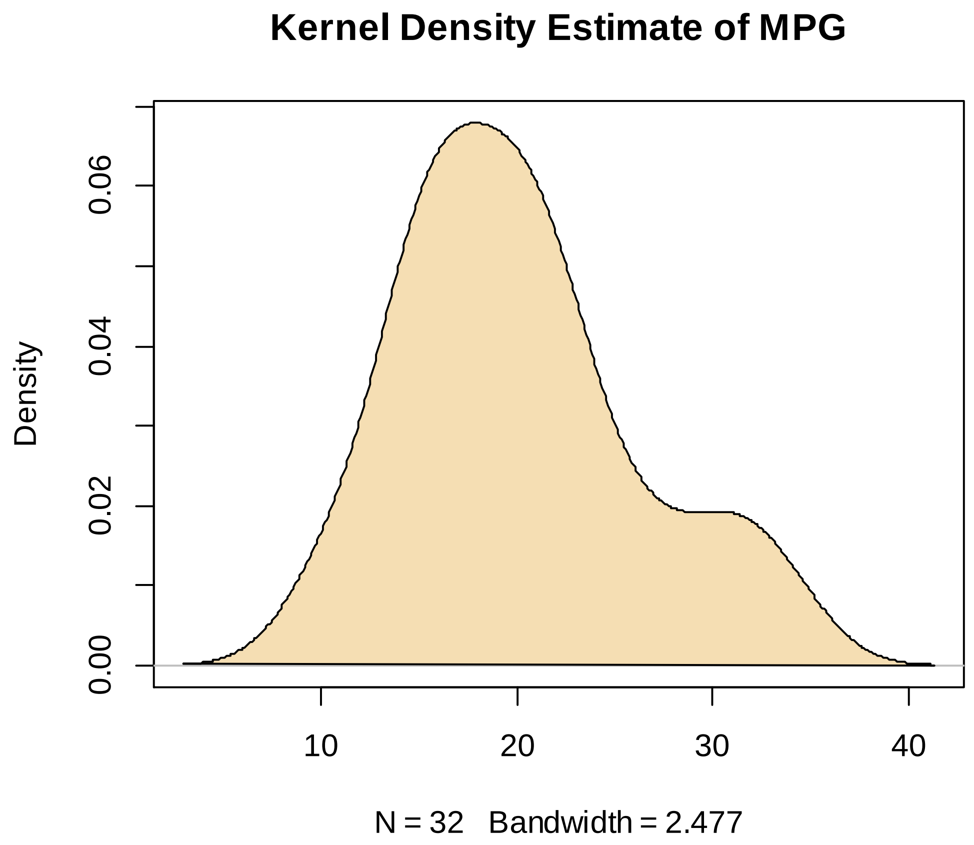


Kernel Density Plots

- Density, or probability density function (PDF), is a function that describes the relative probability for a variable to hold a given value.
- Density is usually a better way to describe the distribution of a quantitative variable.
- Use R function density() to calculate univariate kernel density estimation

R Code: Kernel Density Plots

```
# Kernel density plot
dst <- density(mtcars$mpg) # Calculate kernel density estimate
plot(dst, type = "n", main = "Kernel Density Estimate of MPG")
polygon(dst, col = "wheat")
```



Line Charts (Generic Plots)

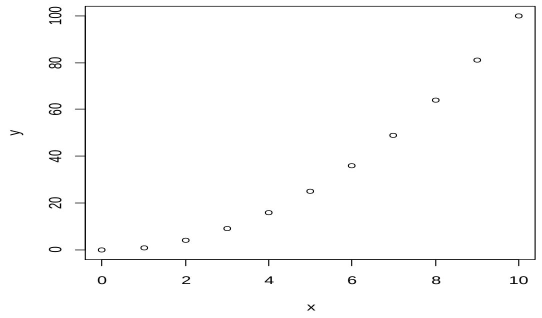
- Syntax: plot(x, y, ...)
- Parameter type
  - "p" for points,
  - "l" for lines,
  - "b" for both,
  - "c" for the lines part alone of "b"
  - "o" for both ‘overplotted’
  - "h" for ‘histogram’ like (or ‘high-density’) vertical lines,
  - "s" for stair steps,
  - "S" for other steps
  - "n" for no plotting

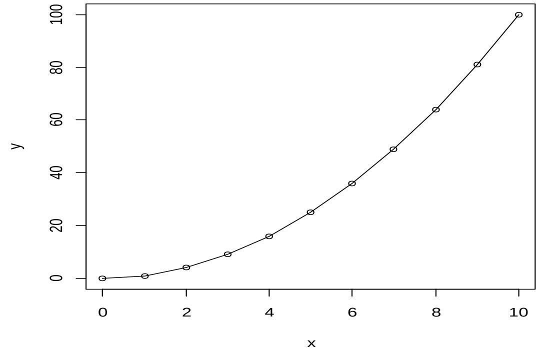
Two Ways to Plot Line Charts

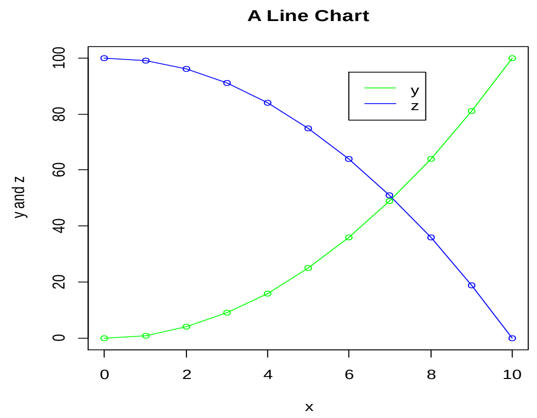
- Use plot(x, y,...) to directly plot x, y data in a graph
- Use plot(x,y,type="n",...) to create a plot without points and lines, then use lines(x,y,...) to add points and lines
  - This method allows plotting multiple lines

R Code

```
## Line charts ##
x <- c(0:10); y <- x^2; z <- 100-y # create data to plot
par(mfrow=c(1,1)) # Only show one plot on a page

plot(x,y) # Create a plot with default parameters

plot(x,y,type = "o") # Create a plot with overplotted points and
lines

# Use plot() to create a plot without points and lines, then use
lines(x,y, ...) to add points and lines
plot(x,y,type = "n", main = "A Line Chart", ylab = "y or z")
lines(x,y,type = "o", col = "green")
lines(x,z,type = "o", col = "blue")
legend(6,95,legend = c("y","z"),lty=c(1,1),col=c("green","blue"))
```







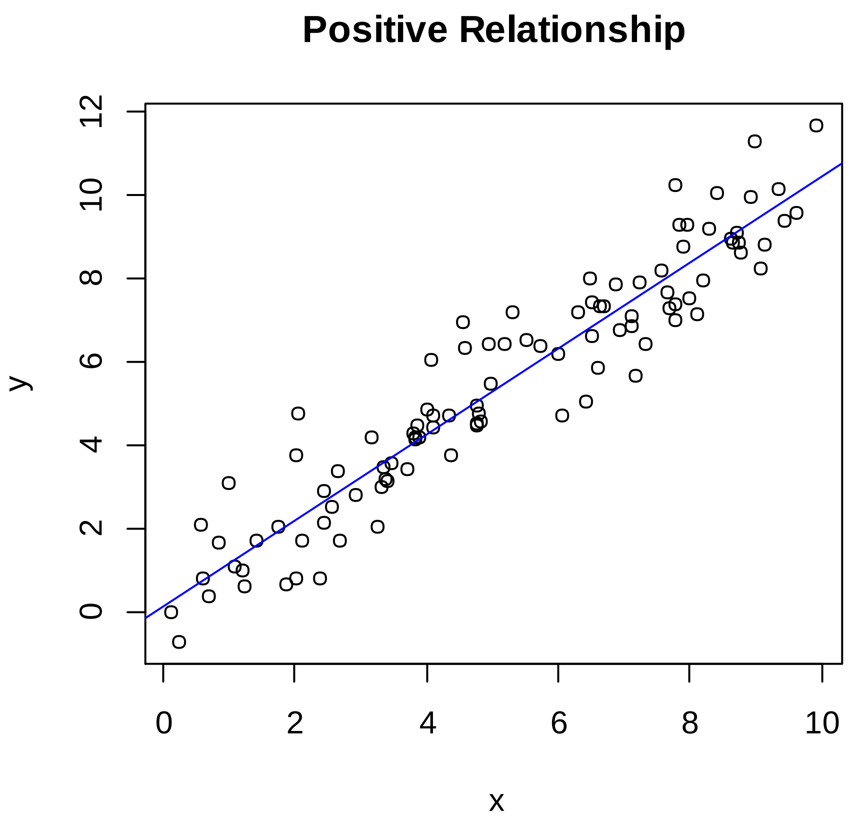
Scatter Plots

- A scatter plot presents the relationship between two quantitative variables.

- A trendline can be used to approximate the relationship.
- A scatterplot matrix is preferred when we want to roughly determine the relationship between multiple variables.

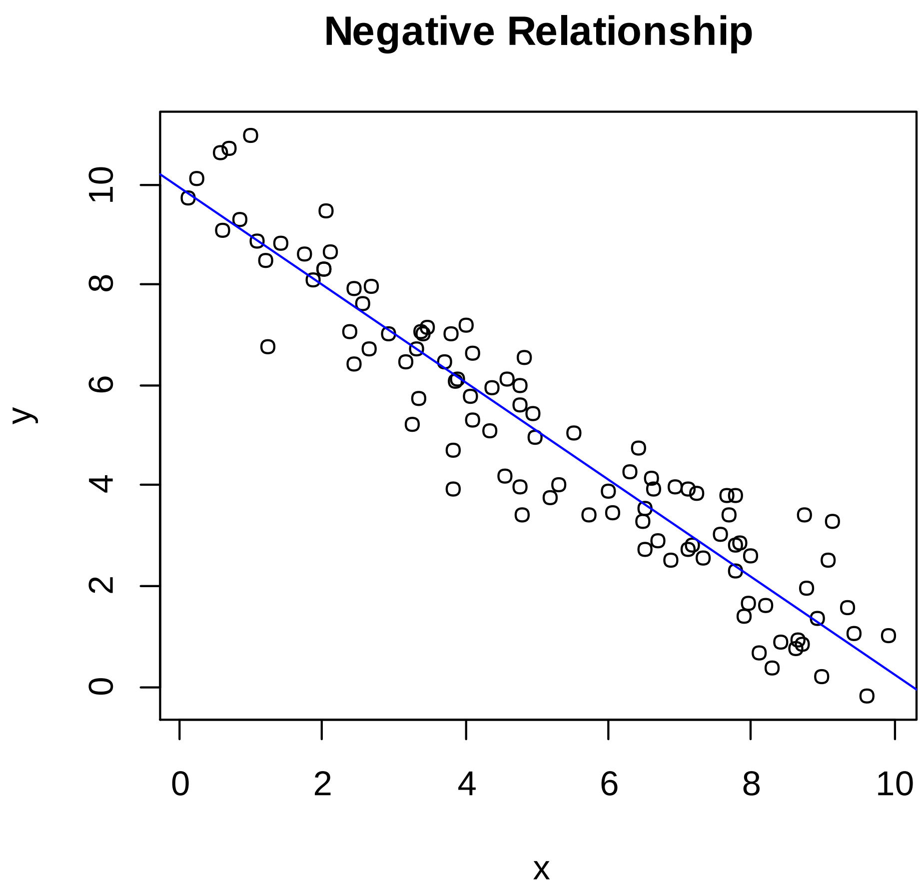
Interpreting Scatter Plots

- A positive relationship



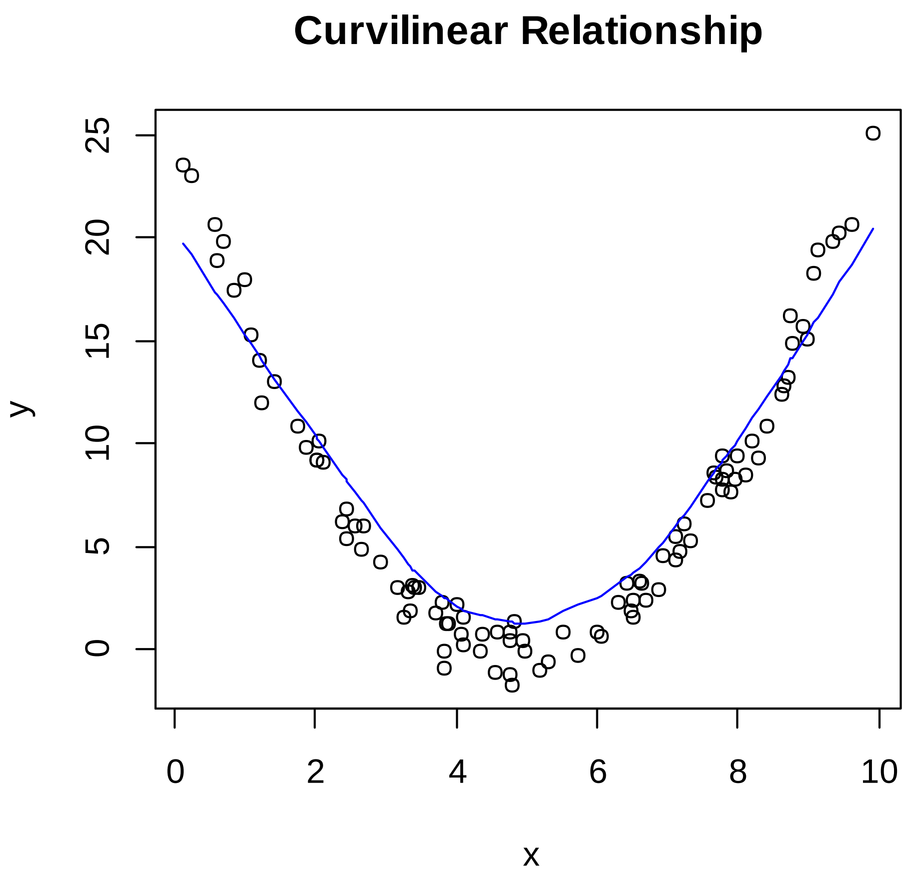
Interpreting Scatter Plots

- A negative relationship



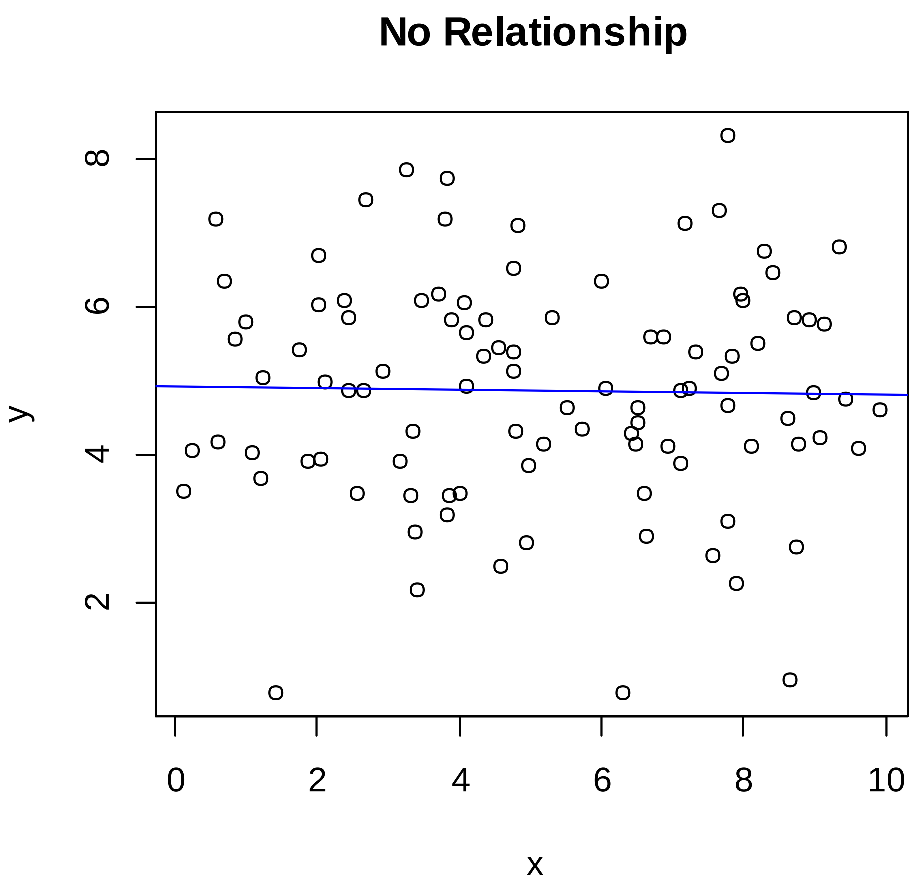
Interpreting Scatter Plots

- A curvilinear relationship



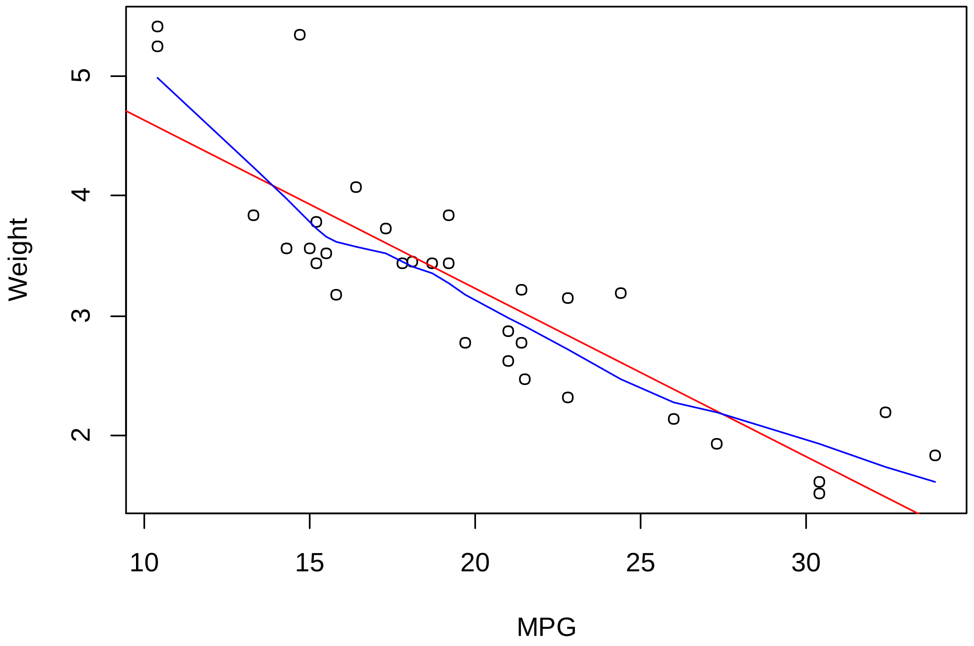
Interpreting Scatter Plots

- No relationship



R Code: Scatter Plots

```
plot(mtcars$mpg, mtcars$wt, xlab="MPG", ylab="Weight")
abline(lm(mtcars$wt~mtcars$mpg), col="red") # Add regression fit line (y~x)
lines(lowess(mtcars$mpg,mtcars$wt), col="blue") # Add lowess fit line (x,y)
```



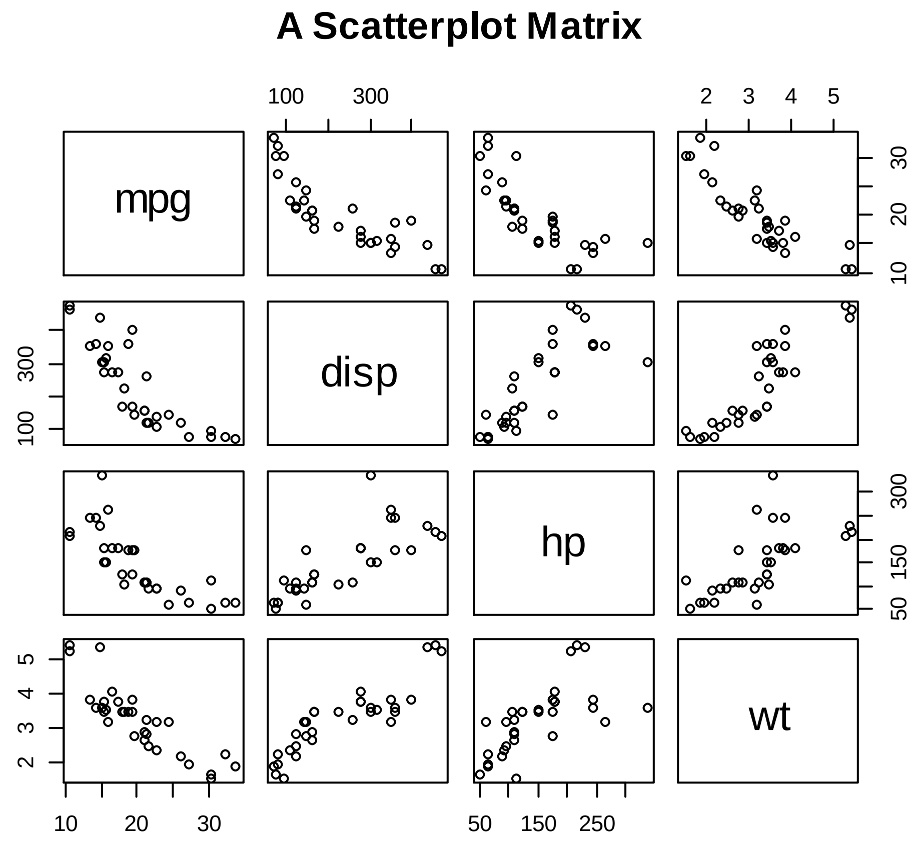
Interpretation: Cars with higher weight tend to have less miles per gallon.

LOWESS = Locally Weighted Scatterplot Smoothing

R Code: Scatter Plot Matrix

- A better way to roughly show the relationship between multiple variables

```
pairs(~mpg+disp+hp+wt, data=mtcars, main="A Scatterplot Matrix")
```



Graphical Parameters

Graphical Parameters

- R uses graphical parameters to control the display of graphs.
  - Font
  - Color
  - Line
  - Symbol
  - Title
  - …

- Type "?par" to learn more.

- In this section, we'll learn some commonly used parameters.

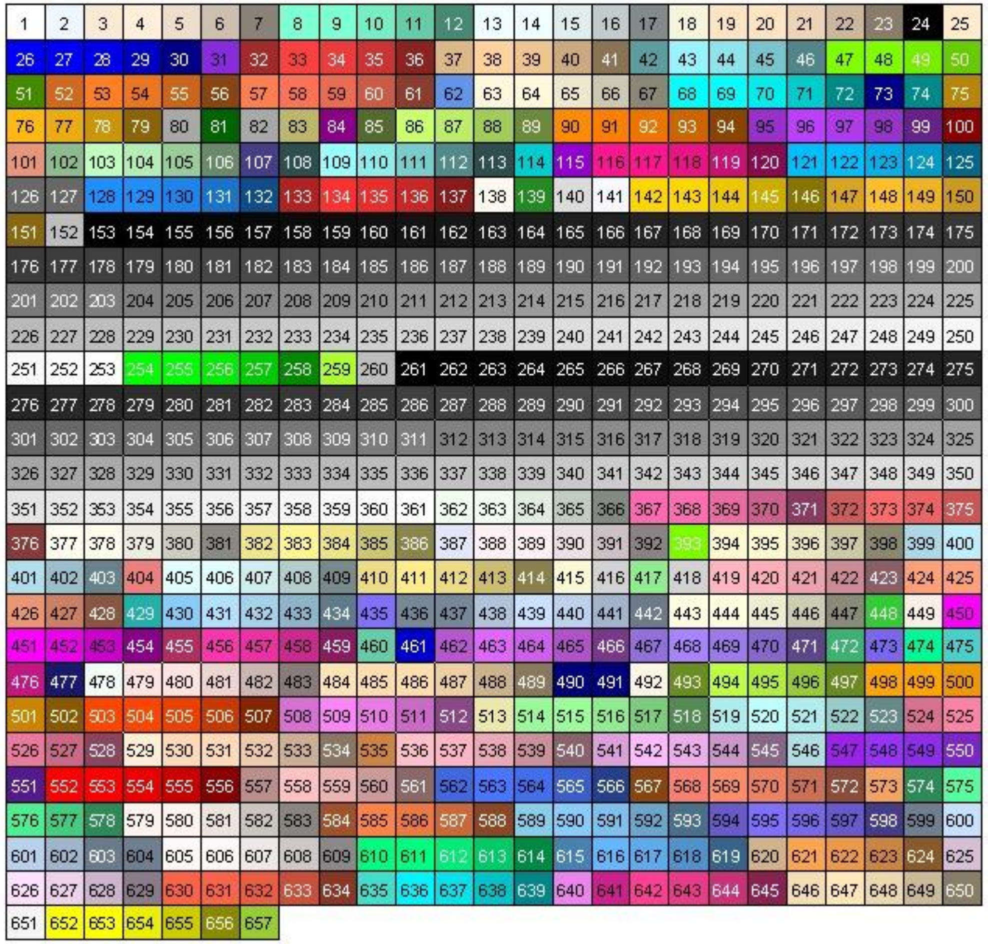
Color Parameters

- col = : Default plotting color
- col.axis = : Color for axis annotation
- col.lab = : Color for x and y labels
- col.main = : Color for main titles
- col.sub = : Color for sub-titles
- bg = : Plot background color
- fg = : Plot foreground color

Color Values

- Color value(s) can be specified as index, name, hexadecimal, or RGB.

R colors



Source: http://research.stowers-institute.org/efg/R/Color/Chart/index.htm

(cont.)

- Use colors() to show all color names

- Use the following commands to create a vector of *n* contiguous colors:
  - rainbow(n)
  - heat.colors(n)
  - terrain.colors(n)
  - topo.colors(n)
  - cm.colors(n)

pch = : Plot Character

- Either a single character or an integer code for one of a set of graphics symbols to be used as plotting symbols.

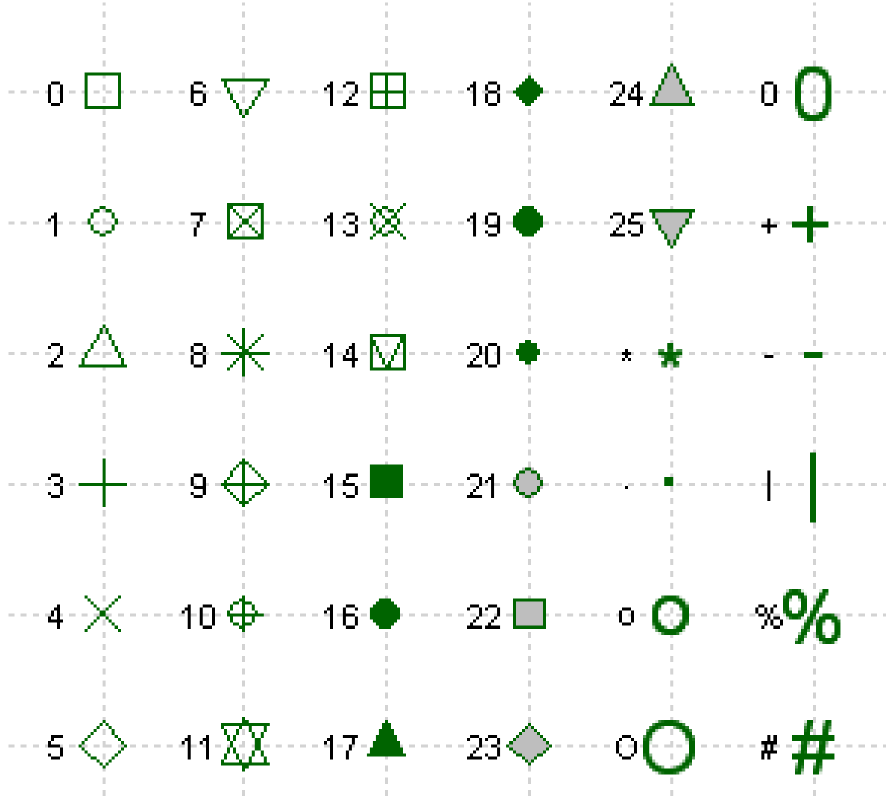


Image source: http://www.statmethods.net/advgraphs/parameters.html

Size Magnification for Text and Symbols

- cex = : A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default.

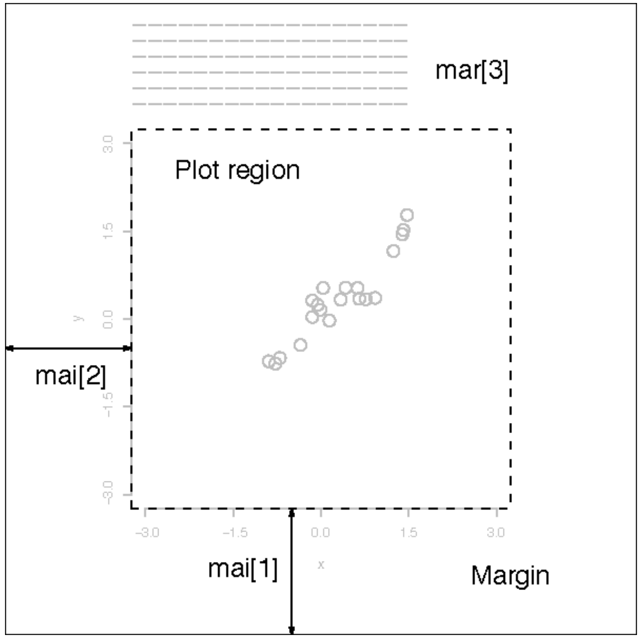
- cex.axis = : Magnification of axis annotation relative to cex
- cex.lab = : Magnification of x and y labels relative to cex
- cex.main = : Magnification of main titles relative to cex
- cex.sub = : Magnification of sub titles relative to cex

Font Parameters

- font = : Integer specifying font to use (1=plain, 2=bold, 3=italic, 4=bold italic, 5=symbol)
- font.axis = : Font for axis annotation
- font.lab = : Font for x and y labels
- font.main = : Font for main titles
- font.sub = : Font for sub-titles
- ps = : Font point size (text size = ps*cex)
- family = : Font family for drawing text

Margins: mar or mai

- mai = : A numerical vector of the form c(bottom, left, top, right) which gives the margin size specified in inches.
- mar = :A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot. The default is c(5, 4, 4, 2) + 0.1.



Visualize Data Using ggplot2 Package

ggplot2 Package

- Based on the Grammar of Graphics, a general scheme for data visualization

- A graph can be built from several components:
  - A dataset
  - A set of geoms(geometric objects)
  - A coordinate system
  - ...

- Two common types of usage
  - Use quick plotting qplot() function to create basic graphs
  - Use ggplot() and other functions to create ggplot2 graphs

For more detail, refer to http://docs.ggplot2.org/current/

Quick Plotting

- qplot() is the basic quick plotting function in the ggplot2 package.
- It is very similar to the base R plot() function.
- It's a convenient wrapper for creating a number of different types of plots using a consistent calling scheme.

qplot() Syntax

```
qplot(x, y = NULL, ..., data, facets = NULL, margins = FALSE, geom = "auto",
    xlim = c(NA, NA), ylim = c(NA, NA), log = "", main = NULL,
    xlab = deparse(substitute(x)), ylab = deparse(substitute(y)), asp = NA)
```

x, y, ...	Aesthetics passed into each layer
data	Data frame to use
facets	faceting formula to use.
margins	display marginal facets?
geom	Character vector specifying geom(s) to draw. Defaults to "point" if x and y are specified, and "histogram" if only x is specified. Other values include "smooth", "boxplot", "line", "density", "bar", and "jitter".
xlim, ylim	X and y axis limits
log	Which variables to log transform ("x", "y", or "xy")
main, xlab, ylab	Character vector (or expression) giving plot title, x axis label, and y axis label respectively.
asp	The y/x aspect ratio

R Code: ggplot2::qplot()

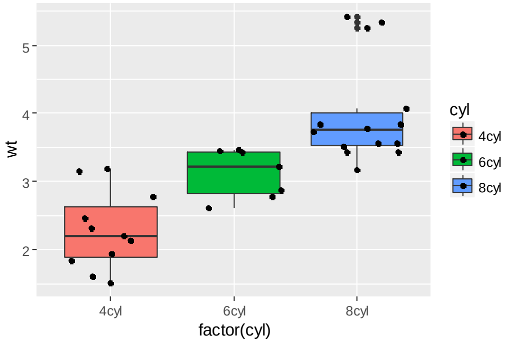
```
library(ggplot2)

# Use default geoms
qplot(mpg, data = mtcars)
qplot(mpg, wt, data = mtcars)
qplot(mpg, wt, data = mtcars, colour = cyl)
qplot(mpg, wt, data = mtcars, size = cyl)
qplot(mpg, wt, data = mtcars, facets = vs ~ am)

# Use different geoms
qplot(mpg, data = mtcars, geom = "density")
qplot(mpg, data = mtcars, geom = "dotplot")
qplot(cyl, mpg, data = mtcars, geom = "boxplot")
qplot(cyl, data = mtcars, geom = "bar")
qplot(mpg, wt, data = mtcars, geom = c("path","point"))
qplot(mpg, wt, data = mtcars, geom = "smooth")
qplot(mpg, wt, data = mtcars, geom=c("point", "smooth"))
qplot(mpg, wt, data = mtcars, geom=c("point", "smooth"), color = cyl)
qplot(mpg, wt, data = mtcars, geom=c("point", "smooth"), shape = cyl)

qplot(cyl, wt, data = mtcars, geom = c("boxplot", "point"))
qplot(factor(cyl), wt, data = mtcars, geom = c("boxplot", "jitter"))
qplot(factor(cyl), wt, data = mtcars, geom = c("boxplot", "jitter"), fill = cyl)
```

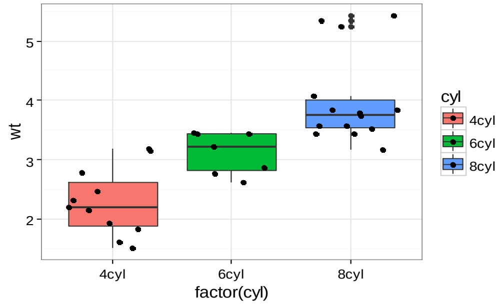
Customize ggplot2 Graphs

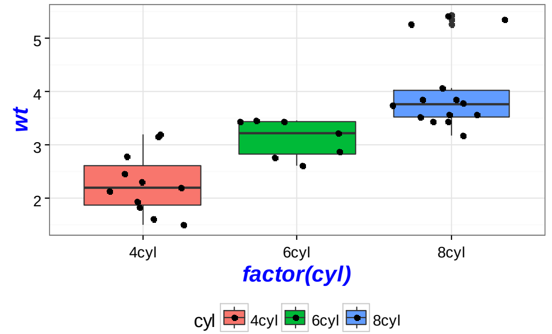
- We know that the display of base R graphs can be customized by using par().
- For ggplot2 graphs, we use a set of theme functions.
- We can use "+" to chain ggplot2 functions to draw the final graph.

R Code: Customize ggplot2 Graphs

```
## Use theme to customize display ##
p <- qplot(factor(cyl), wt, data = mtcars, geom =
c("boxplot", "jitter"), fill = cyl)
# Use print(p) or simply use "p" to show the
graph p
print(p)
p
# Use "+" to chain ggplot2 functions to draw
the final graph
p + theme_bw()

p + theme_bw() +
theme(axis.title=element_text(face="bold.italic",
size="14", color="blue"),
        legend.position="bottom")
```







Use ggplot() and Other Functions

- Syntax:
  ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())
  - Aesthetic mappings describe how variables in the data are mapped to visual properties (aesthetics) of geoms.

- We typically construct a plot incrementally, using the "+" operator to add layers to the existing ggplot object.

R Code: Use ggplot() Function

```
# Construct basic plot
p1 <- ggplot(data = mtcars, aes(x = wt, y = mpg))
p2 <- ggplot(data = mtcars, aes(x = cyl, y = mpg))
# Plot points with default display style
p1 + geom_point()
p2 + geom_boxplot()
# Use different display
p1 + geom_point(size = 4)
p1 + geom_point(col = "blue")
p1 + geom_point(aes(col = cyl))
p1 + geom_point(col = "blue") + geom_line()
# Add smoothing line (by default using loess smoothing method)
p1 + geom_point(col = "blue") + geom_smooth()
# Add smoothing line using linear regression method
p1 + geom_point(col = "red") + geom_smooth(method = "lm")
p1 + geom_point(col = "red") + geom_smooth(method = "lm", se = FALSE)
p1 + geom_point(col = "red") + geom_smooth(method = "lm", se = FALSE) + theme_bw()

p1 + geom_point(aes(col = cyl)) +
  geom_smooth(method = "lm", se = FALSE) + theme_bw() +
  labs(title = "Relationship between Weight and MPG", x="Weight", y = "Miles per Gallon") +
  scale_color_discrete(name = "", labels = c("4 cylinders","6 cylinders", "8 cylinders"))

p1 + geom_point(aes(col = am)) +
  geom_smooth(method = "lm", se = FALSE) + theme_bw() +
  labs(title = "Relationship between Weight and MPG", x="Weight", y = "Miles per Gallon") +
  scale_color_discrete(name = "Tansmission")
```

Visualize Spatial Data

Spatial Data

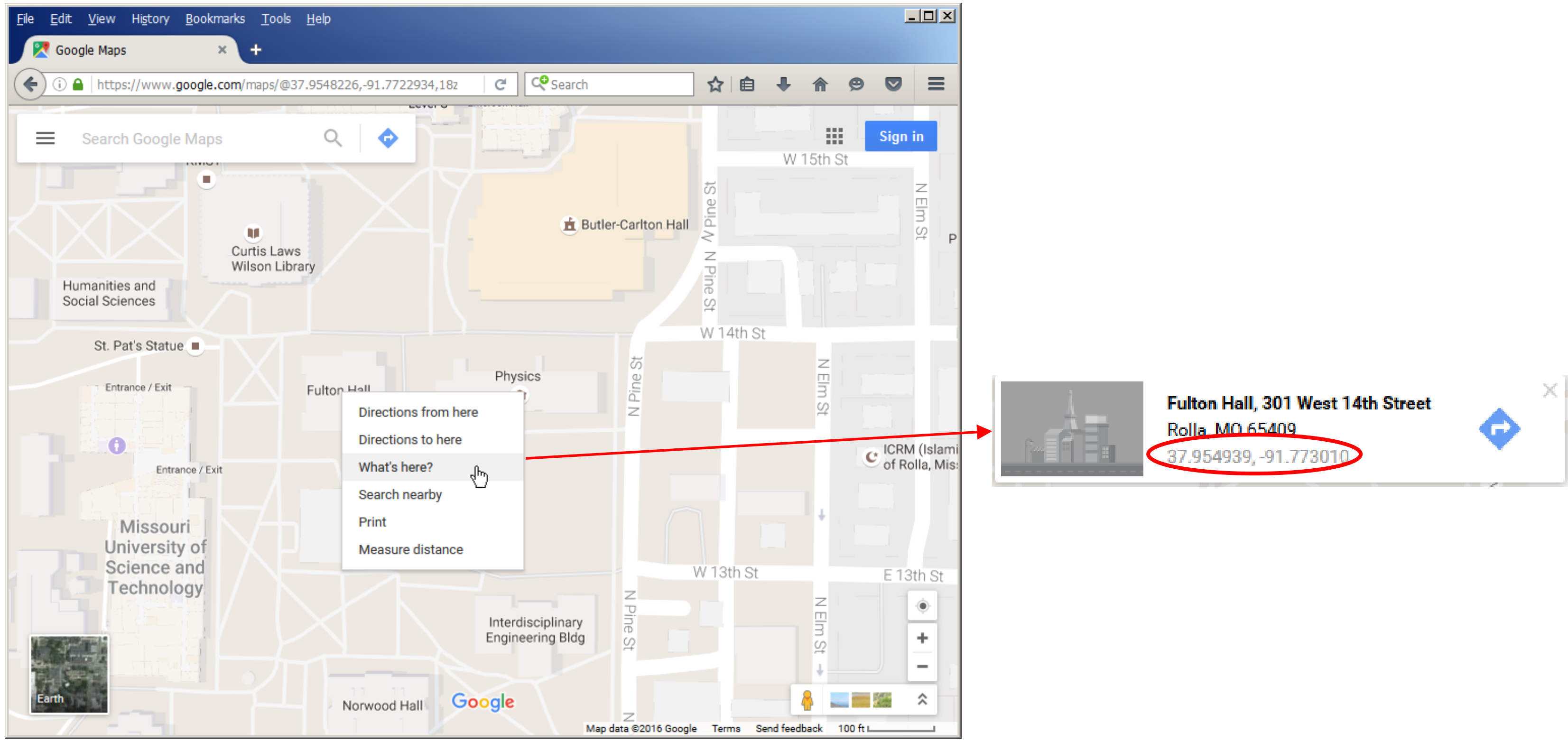
- Spatial data (a.k.a. geospatial data) represent the location, size, and shape of physical objects (such as cities, lake, mountains, buildings etc.) by numbers in a geographic coordinate system.

- Geographic information systems (GIS) can visualize and analyze the spatial data.

- R has many packages that provide GIS functions to analyze spatial data. For the detail, refer to https://cran.r-project.org/web/views/Spatial.html

Get the Coordinates of a Place through Google Maps

- On Google Maps, right-click the place or area.
- Select What's here?
- A card appears at the bottom of the screen with more info.



Understand Spatial Data

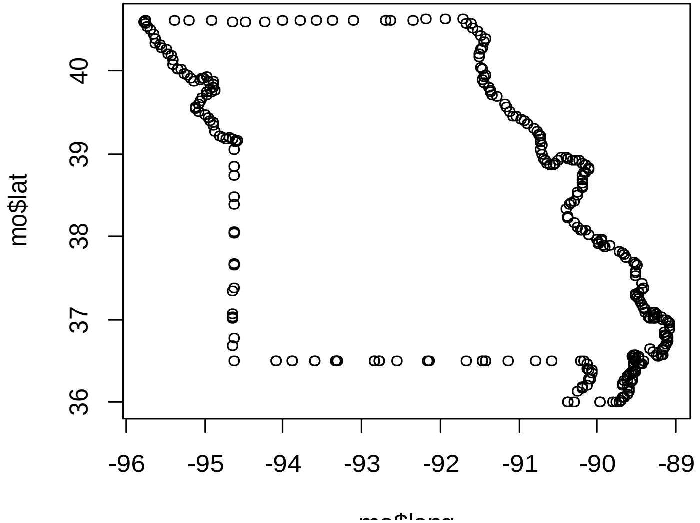
- US state boundaries map data in “maps” package
- Use ggplot2::map_data("state") to create a data frame containing map boundary data for all US states
- Check the Missouri state map boundary data

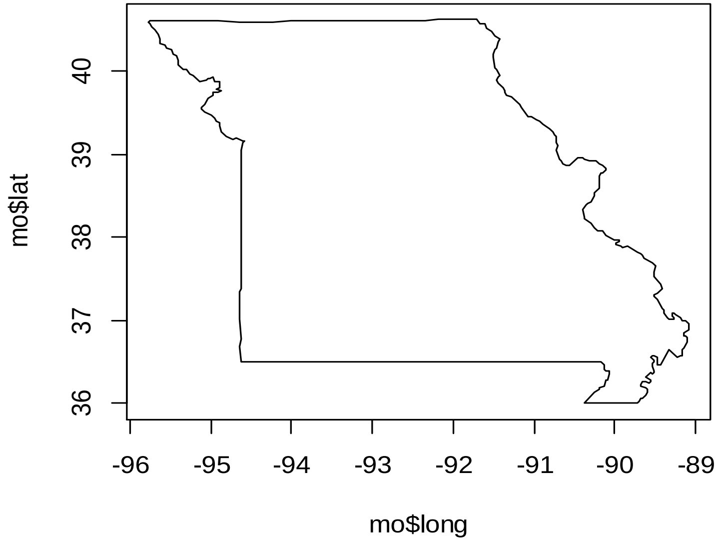
Longitude Latitude Sequence of coordinate

	long	lat	group	order	region	subregion
7804	-95.75271	40.61125	27	7804	missouri	NA
7805	-95.37456	40.60552	27	7805	missouri	NA
7806	-95.20267	40.60552	27	7806	missouri	NA
7807	-94.91619	40.59979	27	7807	missouri	NA
7808	-94.63544	40.59406	27	7808	missouri	NA
7809	-94.48074	40.59406	27	7809	missouri	NA
7810	-94.24010	40.59406	27	7810	missouri	NA
7811	-94.01665	40.59979	27	7811	missouri	NA

Use plot() to Draw Missouri State Boundary

```
# Create a data frame of map data for all states
us_states <- ggplot2::map_data("state")
head(us_states)
# Select Missouri data
mo <-
us_states[which(us_states$region=="missouri"),]
mo # long: longitude, lat: latitude

# Plot the shape of Missouri state
plot(mo$long,mo$lat)

# A better plot of the shape
plot(mo$long,mo$lat, type = "l")
```

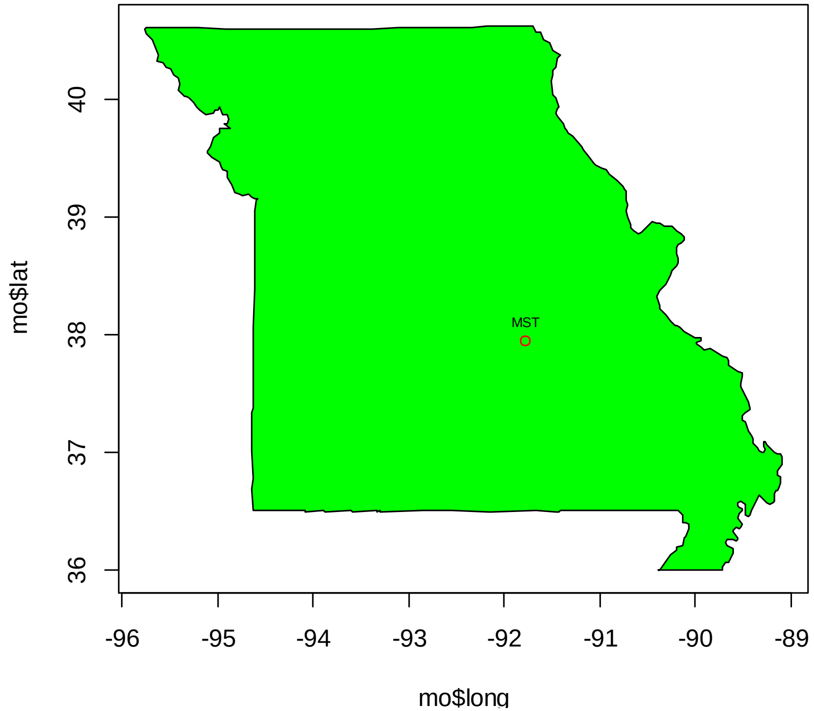




(cont.)

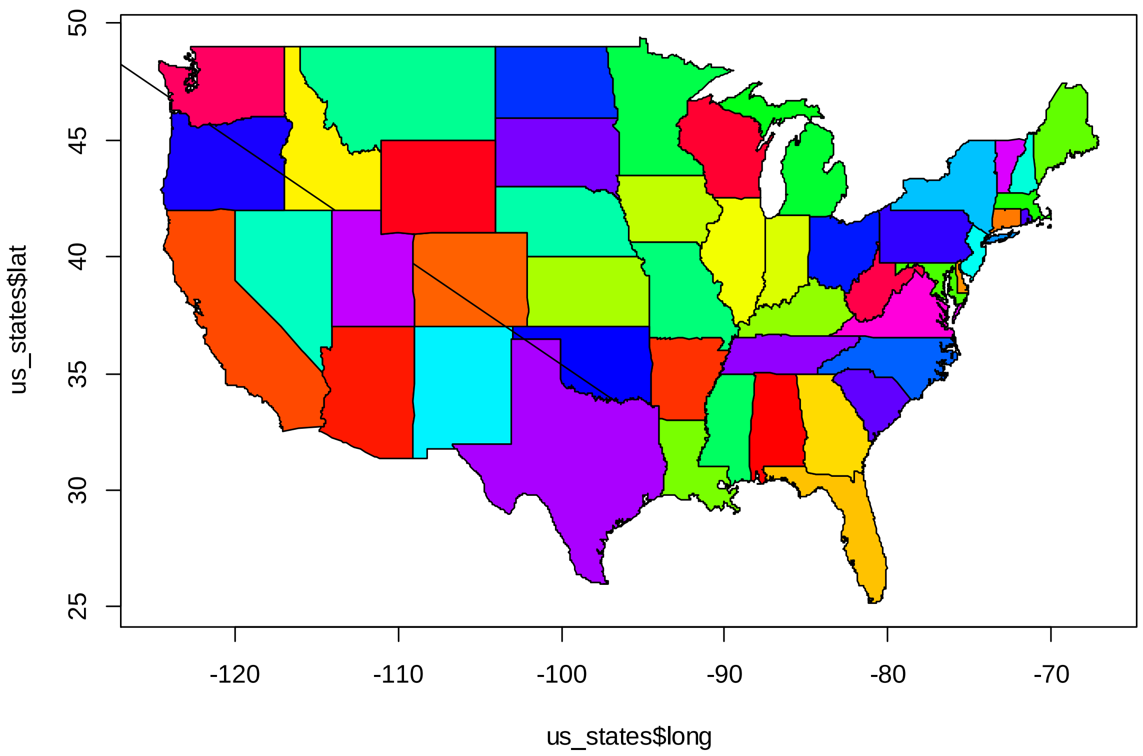
- Add more details

```
# Add more details
plot(mo$long,mo$lat, type = "n") # Draw a blank plot
polygon(mo$long,mo$lat, col = "green") # Add polygon
points(-91.774069, 37.954234, col = "red") # Add MST location
text(-91.774069, 37.954234,"MST", pos = 3, cex = 0.6) # Label the location
```



Draw Boundaries for All States

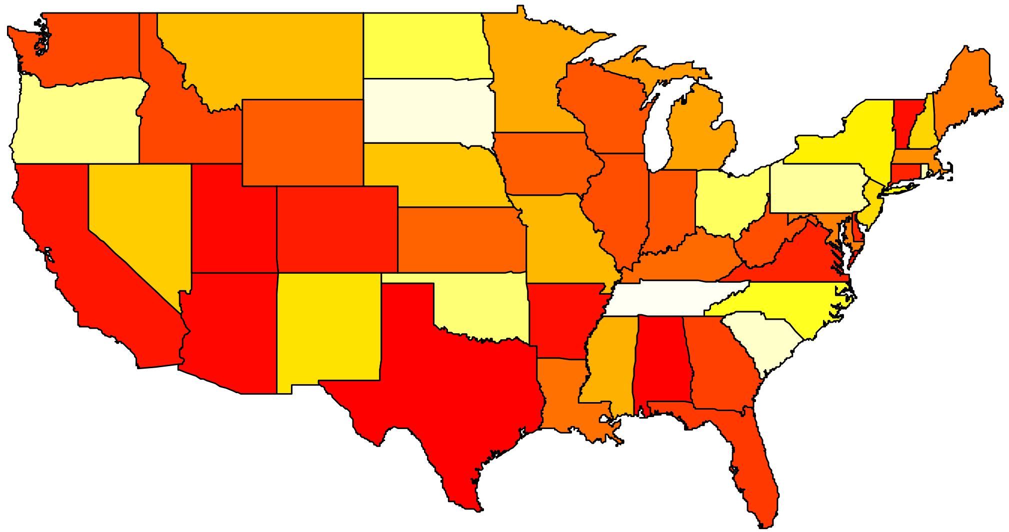
```
plot(us_states$long,us_states$lat, type = "n")
col_set = rainbow(63)
for (i in unique(us_states$group)){
  state <- us_states[which(us_states$group == i),]
  polygon(state$long,state$lat, col = col_set[i])
}
```



Use maps::map() function to draw

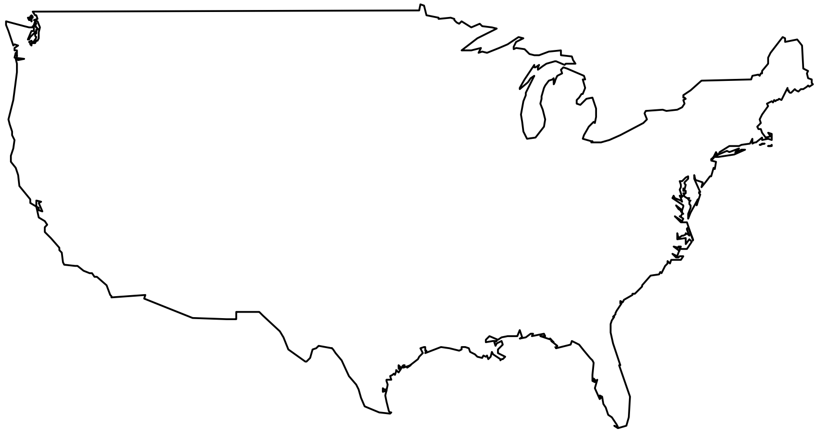
- A better way to draw map is to use functions in R packages

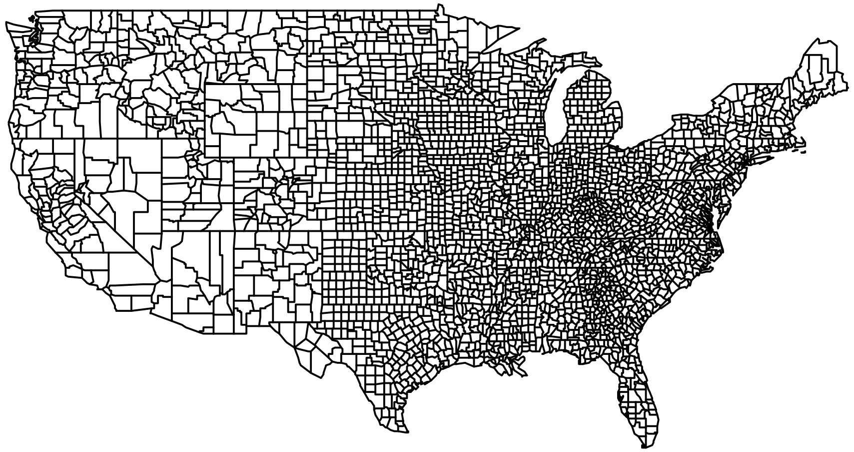
```
maps::map(database = "state",region = "missouri", fill = TRUE, col = "green")
maps::map(database = "state", fill = TRUE, col = heat.colors(49))
```



(cont.)

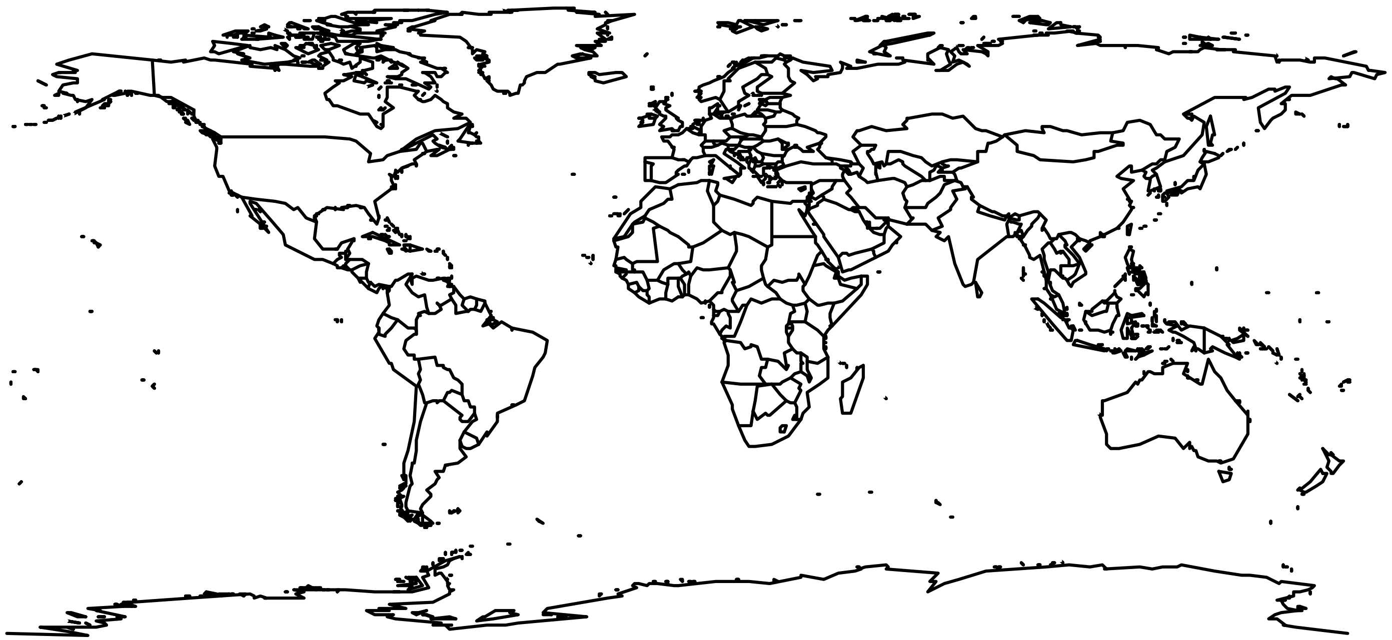
```
maps::map(database = "usa")
maps::map(database = "county")
```





(cont.)

```r
maps::map(database = "world")
```



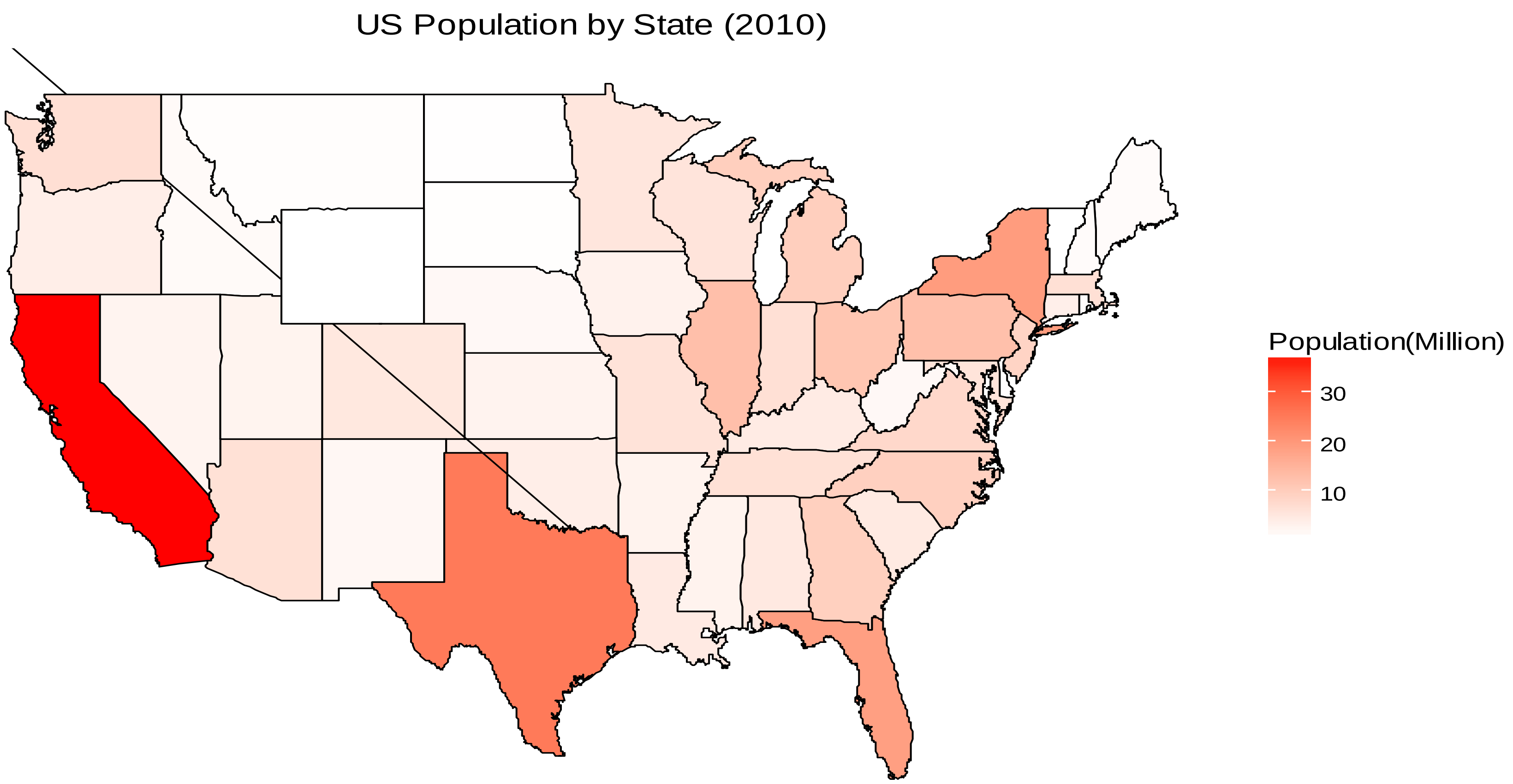
Application Case: Visualize US Population on Maps

- Data Source:
  - The original dataset of US population by state was collected from U.S. Census Bureau (https://www.census.gov/popest/data/state/totals/2015/tables/NST-EST2015-01.csv)
  - The following CSV file simplifies the data by only including 2010 data.
    US_Population_2010.csv

- Task:
  - To visualize the population by state on the map of US states

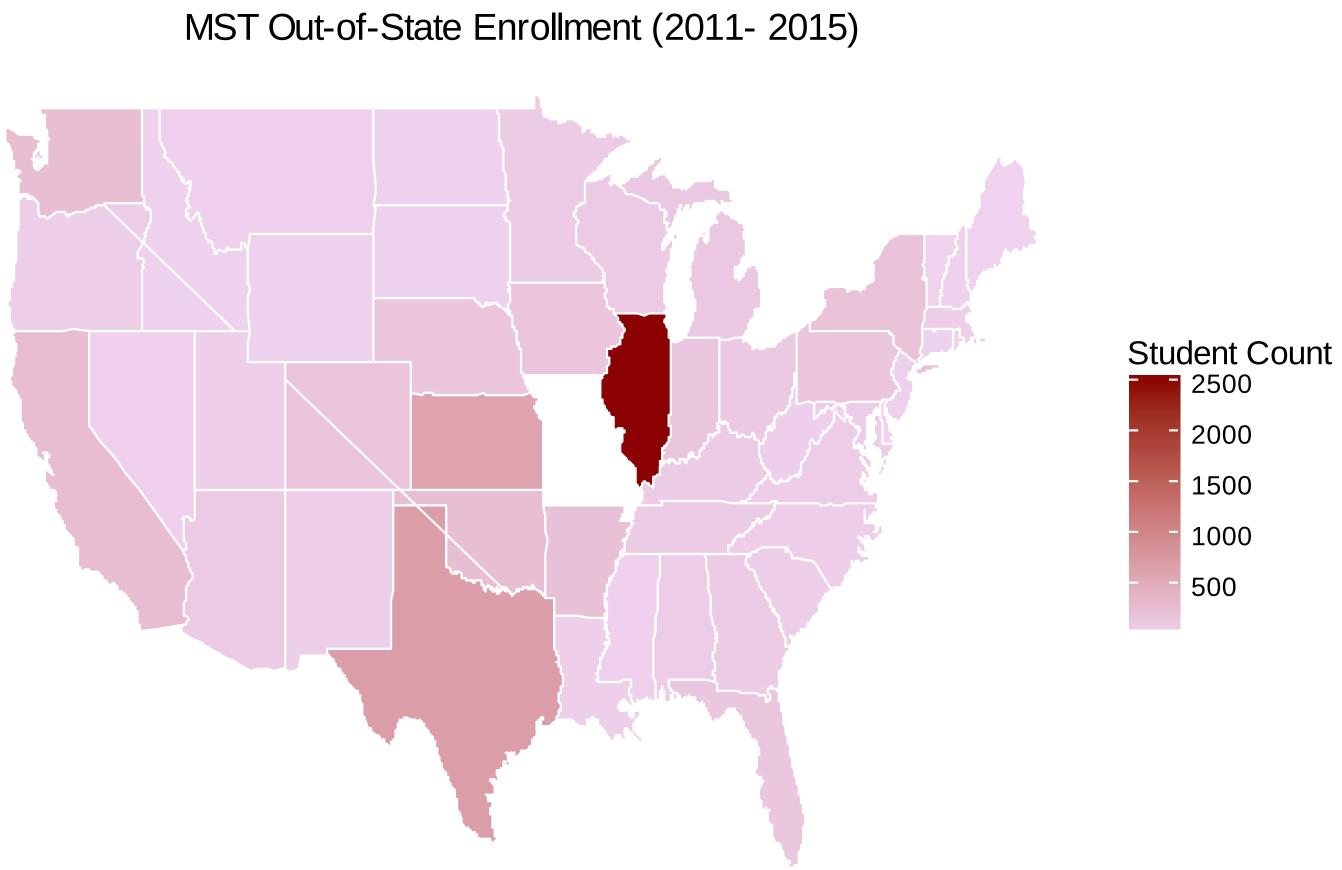
R Code: Visualize US Population on Maps

- Refer to R_Population_on_Map.R



Homework #9 (due Oct 27 11:59 PM)

- Plot MST out-of-state enrollment on US map



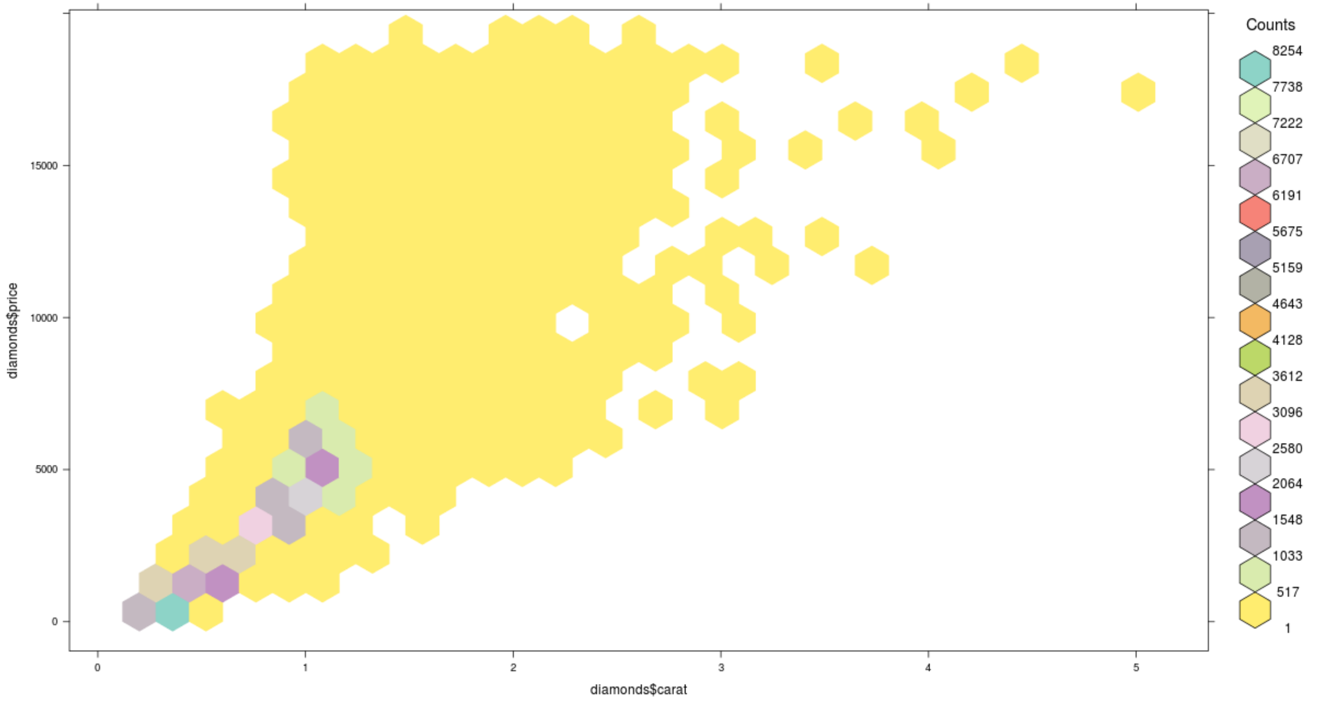
Some Advanced Visualization Methods

Some Advanced Visualization

- Hexagonal Binning
- Mosaic Plot
- Heat Map

Hexagonal Binning

- A form of bivariate histogram useful for visualizing the structure in datasets with large N.
  - The xy plane over the set (range(x), range(y)) is tessellated by a regular grid of hexagons;
  - The number of points falling in each hexagon are counted and stored in a data structure;
  - The hexagons with count > 0 are plotted using a color ramp or varying the radius of the hexagon in proportion to the counts.

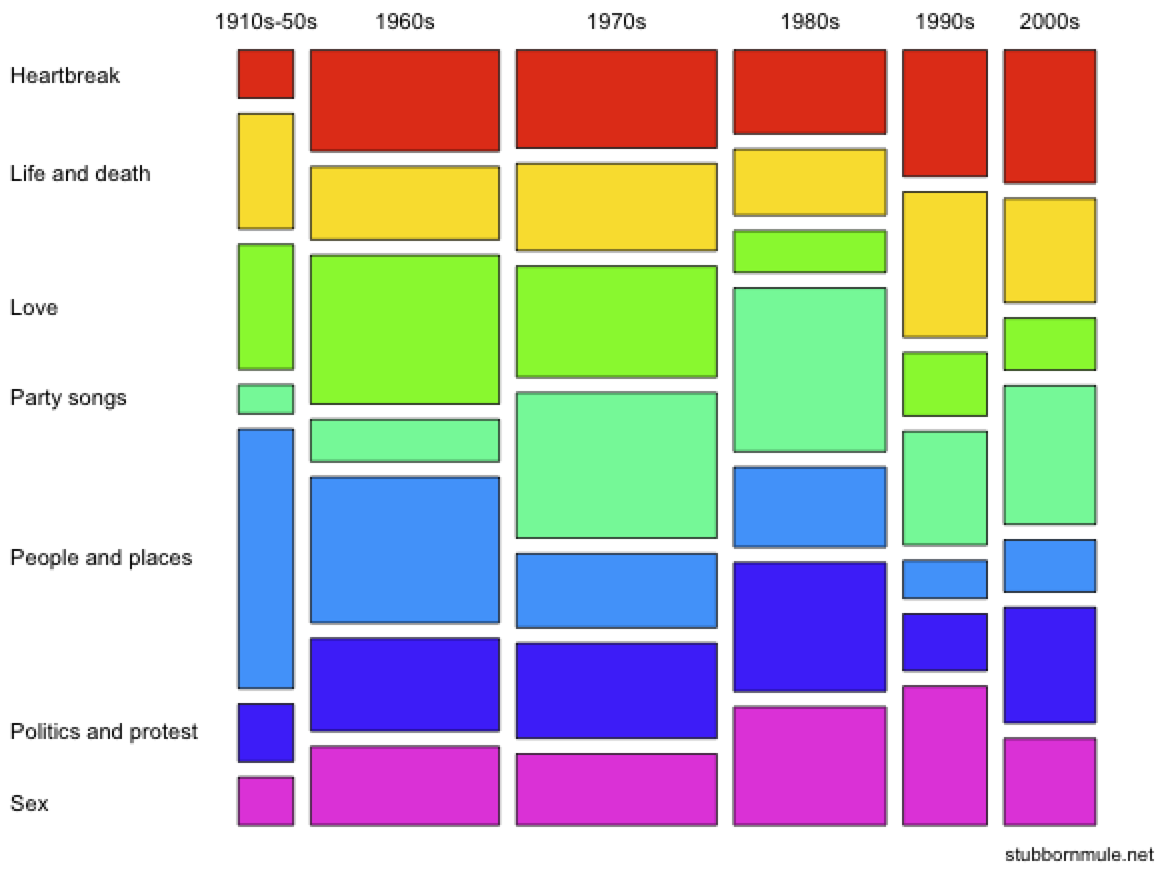


Source: https://cran.r-project.org/web/packages/hexbin/vignettes/hexagon_binning.pdf

Mosaic Plot

- Mosaic plot is a graphical method for visualizing data from two or more qualitative variables

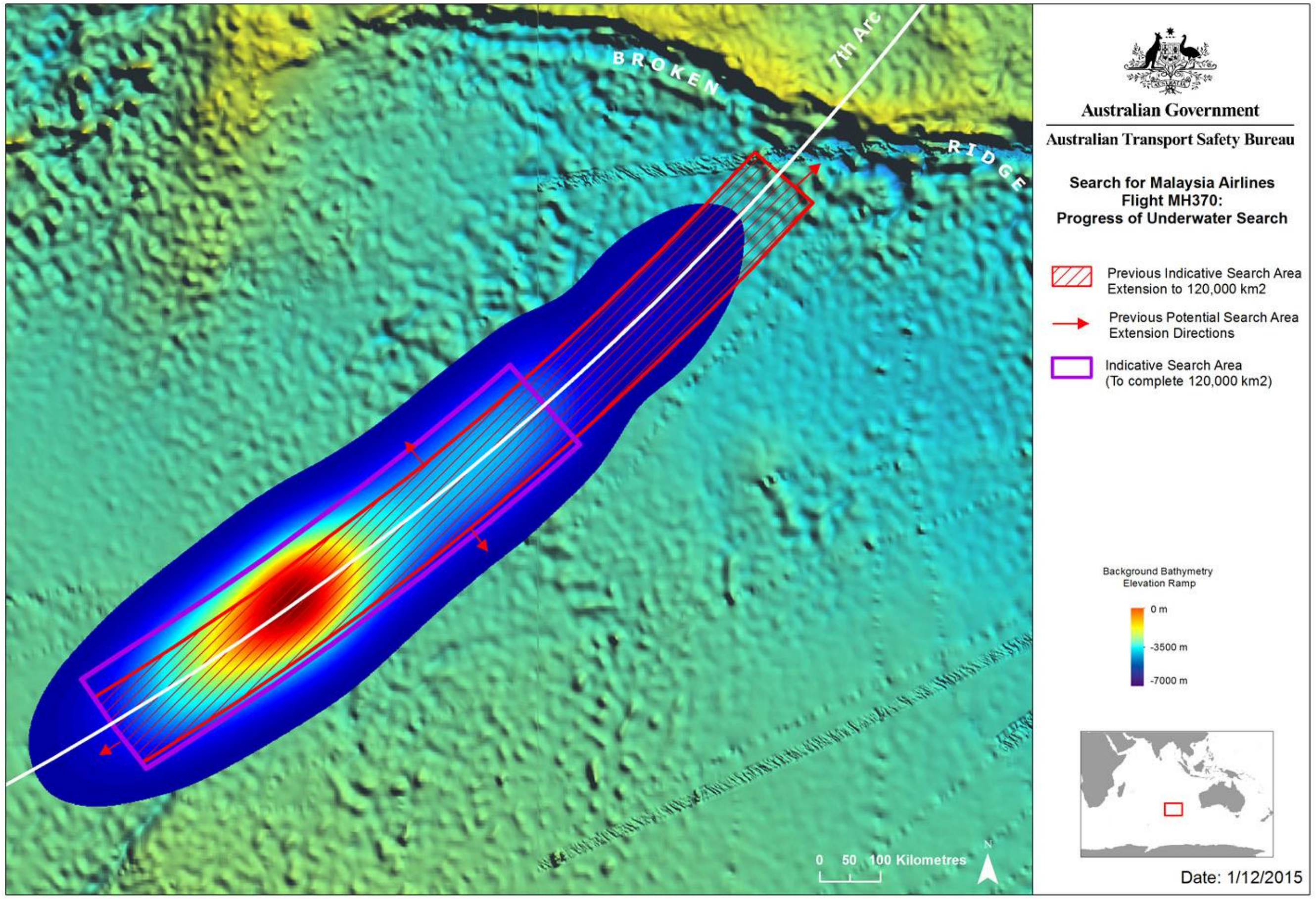
Mosaic plot showing cross-sectional distribution through time of different musical themes in the Guardian's list of "1000 songs to hear before you die".



Source: https://en.wikipedia.org/wiki/Mosaic_plot

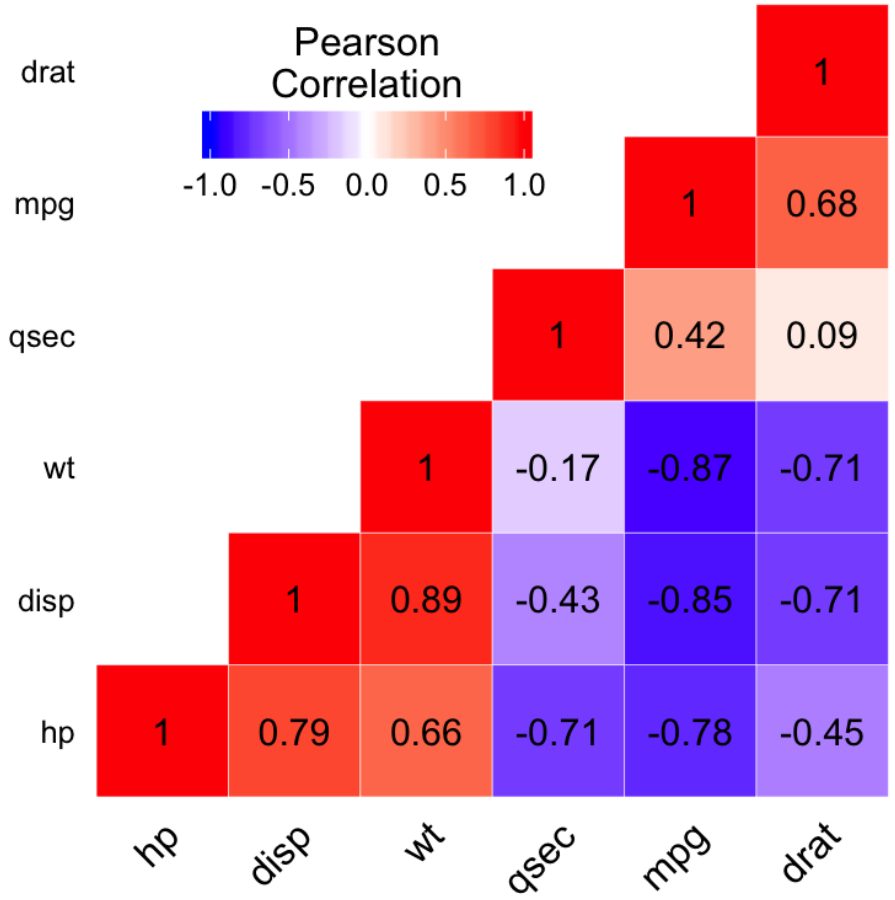
Heat Map: Using Color to Represent a 3rd Dimension

- The probable location of missing Malaysia Airlines Flight 370



Heat Map: Using Color to Represent a 3rd Dimension

- A Correlation Heat Map



Q & A