

Damian Thomas

2017-02-10

## Topics

## 1 Exploratory Data Analysis

- Definition
- Motivating Example
- Data Analysis Process

## 2 Toolbox

- Statistical Functions
- Plotting Functions
- Data Transformation

# Exploratory Data Analysis<sup>1</sup>

*"In statistics, exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods."*

<sup>1</sup>Source:

*"In statistics, exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods."*

## Goals

- Suggest hypotheses about the causes of observed phenomena
- Assess assumptions on which statistical inference will be based
- Support the selection of appropriate statistical tools and techniques
- Provide a basis for further data collection through surveys or experiments

<sup>1</sup>Source:

- Informal—no defined model or assumptions
- Learn about the data, underlying structure
- Gather information to inform modelling choices
- Generate questions

- Formal-rigorous statistical methods
- Dependent on assumptions (random, normal, iid, linear, etc.)
- Model Specification (regressions, ANOVA)
- Parameter estimation & hypothesis testing

## Motivating Example

*"Anscombe's quartet comprises four data sets that have nearly identical simple descriptive statistics, yet appear very different when graphed. Each data set consists of eleven (x,y) points. They were constructed in 1973 by the statistician Francis Anscombe to demonstrate both the importance of graphing data before analyzing it and the effect of outliers on statistical properties."*<sup>2</sup>

<sup>2</sup>Source: [http://en.wikipedia.org/wiki/Anscombe's\\_quartet](http://en.wikipedia.org/wiki/Anscombe's_quartet)

```
> data("anscombe")
> head(anscombe)
```

	x1	x2	x3	x4	y1	y2	y3	y4
1	10	10	10	8	8.04	9.14	7.46	6.58
2	8	8	8	8	6.95	8.14	6.77	5.76
3	13	13	13	8	7.58	8.74	12.74	7.71
4	9	9	9	8	8.81	8.77	7.11	8.84
5	11	11	11	8	8.33	9.26	7.81	8.47
6	14	14	14	8	9.96	8.10	8.84	7.04



# Anscombe's Quartet

x1	y1
Min. : 4.0	Min. : 4.260
1st Qu.: 6.5	1st Qu.: 6.315
Median : 9.0	Median : 7.580
Mean : 9.0	Mean : 7.501
3rd Qu.:11.5	3rd Qu.: 8.570
Max. :14.0	Max. :10.840

```

          x1          y1
var: 11.000000  4.127269
sd:   3.316625  2.031568

```

```
correlation
0.8164205
```

# Anscombe's Quartet

x2	y2
Min. : 4.0	Min. :3.100
1st Qu.: 6.5	1st Qu.:6.695
Median : 9.0	Median :8.140
Mean : 9.0	Mean :7.501
3rd Qu.:11.5	3rd Qu.:8.950
Max. :14.0	Max. :9.260

```

              x2              y2
var:  11.000000  4.127629
sd:    3.316625  2.031657

```

```
correlation
0.8162365
```

# Anscombe's Quartet

x3	y3
Min. : 4.0	Min. : 5.39
1st Qu.: 6.5	1st Qu.: 6.25
Median : 9.0	Median : 7.11
Mean : 9.0	Mean : 7.50
3rd Qu.:11.5	3rd Qu.: 7.98
Max. :14.0	Max. :12.74

```

              x3              y3
var:  11.000000  4.122620
sd:    3.316625  2.030424

```

```
correlation
0.8162867
```

# Anscombe's Quartet

x4	y4
Min. : 8	Min. : 5.250
1st Qu.: 8	1st Qu.: 6.170
Median : 8	Median : 7.040
Mean : 9	Mean : 7.501
3rd Qu.: 8	3rd Qu.: 8.190
Max. :19	Max. :12.500

```

          x4          y4
var:  11.000000  4.123249
sd:    3.316625  2.030579

```

```
correlation
0.8165214
```

## Anscombe's Quartet<sup>4</sup>

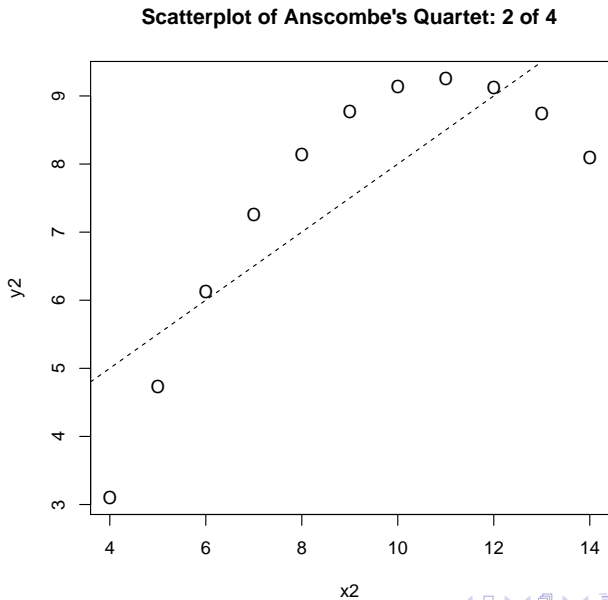
**Table:** Statistical Similarities (for all four data sets)

Property	Value	Accuracy
mean(x)	9	exact
var(x)	11	exact
mean(y)	7.50	to 2 decimal places
var(y)	4.125	plus/minus 0.003
cor(x, y)	0.816	to 3 decimal places
regression	$y = 3.00 + 0.500x$	2 and 3 decimals

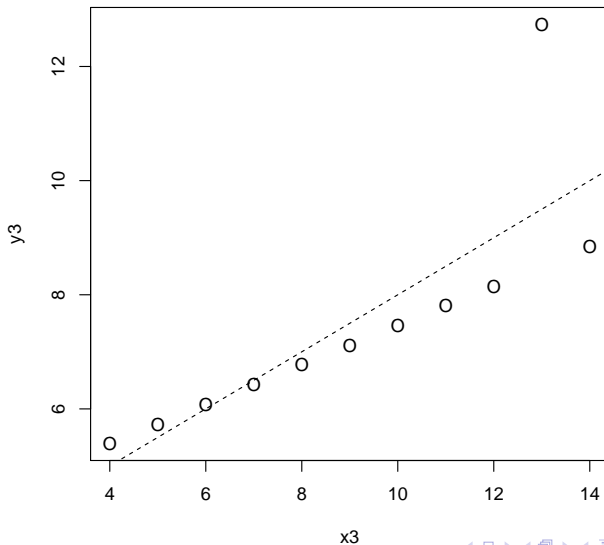
<sup>4</sup>Source: [https://en.wikipedia.org/wiki/Anscombe's\\_quartet](https://en.wikipedia.org/wiki/Anscombe's_quartet)

### Scatterplot of Anscombe's Quartet: 1 of 4





### Scatterplot of Anscombe's Quartet: 3 of 4



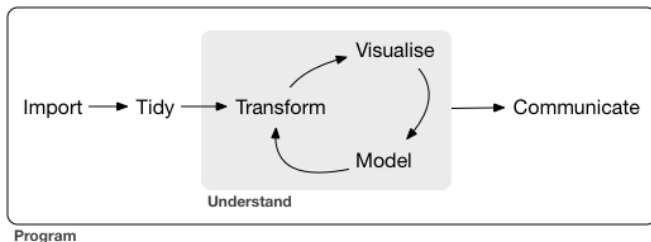


### Scatterplot of Anscombe's Quartet: 4 of 4





# The Data Analysis Process<sup>5</sup>



- Have a question in mind
- Write code to carry out each step
- Save the code so you can reproduce (and share) your work

<sup>5</sup>Source: R for Data Science <http://r4ds.had.co.nz/>

# Toolbox

- Statistical Summaries
  - Extremes: range, minimum, maximum
  - Location: median, mean
  - Spread: quartiles, variance, standard deviation
  - Shape: skew, modality
  - Interactions: tables, correlations
- Visualizations
  - Box plot
  - Scatter plot
  - Line plot
  - Bar plot
  - Histogram
- Transformations
  - Subset / Select
  - Create Variables
  - Aggregate
  - Merge

- `min()`, `max()`
- `mean()`, `median()`
- `sd()`, `var()`
- `quantile()`, `IQR()`
- `cov()`, `cor()`
- `summary()`
- `table()`
- etc., see `?stats` for more

[1] 6

[1] 6

[1] 1

[1] 6

[1] 1

[1] 2



[1] 6

[1] 1

[1] 2

The first unnamed argument of `mean()` is assumed to be an input vector, the rest are considered options. All of the unnamed arguments to `sum()` are assumed to be input vectors.

```
> ?mean
```

# Review using functions: named arguments

```
> # missing values are "contagious"
```

```
> 1 + 2 + 3 + NA
```

```
[1] NA
```

```
> sum(1, 2, 3, NA)
```

```
[1] NA
```

```
> mean(c(1, 2, 3, NA))
```

```
[1] NA
```

## Review using functions: named arguments

```
> # missing values are "contagious"
```

```
> 1 + 2 + 3 + NA
```

```
[1] NA
```

```
> sum(1, 2, 3, NA)
```

```
[1] NA
```

```
> mean(c(1, 2, 3, NA))
```

```
[1] NA
```

```
> # cure: exclude missing values
```

```
> sum(1, 2, 3, NA, na.rm = TRUE)
```

[1] 6

```
> mean(c(1, 2, 3, NA), na.rm = TRUE)
```

[1] 2

## Data: Motor Trend Car Road Tests

```
> data(mtcars) # load built-in data
```

```
> str(mtcars) # view structure
```

```
'data.frame':      32 obs. of  11 variables:
```

```
$ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2
```

```
$ cyl : num 6 6 4 6 8 6 8 4 4 6 ...
```

```
$ disp: num 160 160 108 258 360 ...
```

```
$ hp : num 110 110 93 110 175 105 245 62 95 123 ...
```

```
$ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3
```

```
$ wt : num  2.62 2.88 2.32 3.21 3.44 ...
```

```
$ qsec: num 16.5 17 18.6 19.4 17 ...
```

```
$ vs : num 0 0 1 1 0 1 0 1 1 1 ...
```

```
$ am : num 1 1 1 0 0 0 0 0 0 0 ...
```

```
$ gear: num 4 4 4 3 3 3 3 4 4 4 ...
```

```
$ carb: num 4 4 1 1 2 1 4 2 2 4 ...
```

A data frame with 32 observations on 11 variables.

[ ,11] carb Number of carburetors

```
> data(mtcars) # load built-in data
> str(mtcars)  # view structure
> head(mtcars)
> ?mtcars
```

[1] 10.4

[1] 20.09062

[1] 19.2

[1] 33.9

0%      25%      50%      75%      100%

10.400 15.425 19.200 22.800 33.900

25%      75%

15.425 22.800

```
> summary(mtcars$mpg)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
10.40	15.42	19.20	20.09	22.80	33.90



## Shortcut: the summary() function

```
> # Input a data frame
> summary(mtcars[, 1:3])
```

mpg		cyl		disp	
Min.	:10.40	Min.	:4.000	Min.	: 71.1
1st Qu.:	15.43	1st Qu.:	4.000	1st Qu.:	120.8
Median	:19.20	Median	:6.000	Median	:196.3
Mean	:20.09	Mean	:6.188	Mean	:230.7
3rd Qu.:	22.80	3rd Qu.:	8.000	3rd Qu.:	326.0
Max.	:33.90	Max.	:8.000	Max.	:472.0

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ 🔍 ↺

```
# input 1 vector
```

4	6	8
11	7	14

```
# input 1 vector
```

11 7 14

```
# input 2 vectors
```

0 1

4 3 8

6 4 3

8 12 2

4 6 8

11 7 14

0 1

4 3 8

6 4 3

8 12 2

```
+ Manual = mtcars$am) # set labels
```

Manual

Cylinders	0	1
-----------	---	---

4 3 8

6 4 3

$$8 \quad 12 \quad 2$$

	am	
cyl	0	1
4	3	8
6	4	3
8	12	2

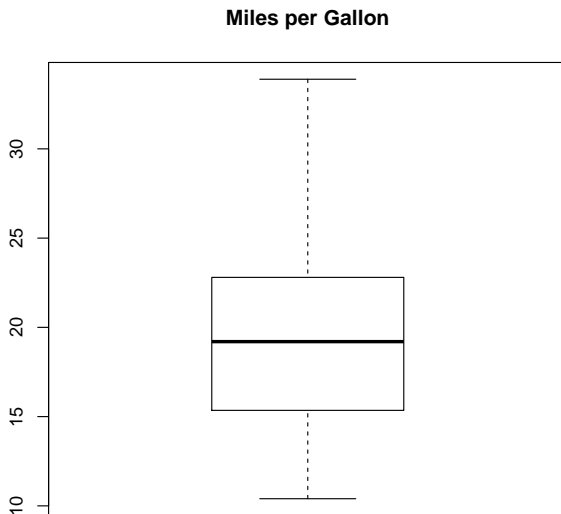
## R has several distinct plotting systems

- Base R functions
  - hist()
  - barplot()
  - boxplot()
  - plot()
- lattice package
- ggplot2 package

# Boxplot

```
> boxplot(mtcars$mpg,
+         main = "Miles per Gallon")
```





# Boxplot Interpretation

Compare:

```
> boxplot(mtcars$mpg)
```

- VS -

```
> summary(mtcars$mpg)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
10.40	15.42	19.20	20.09	22.80	33.90

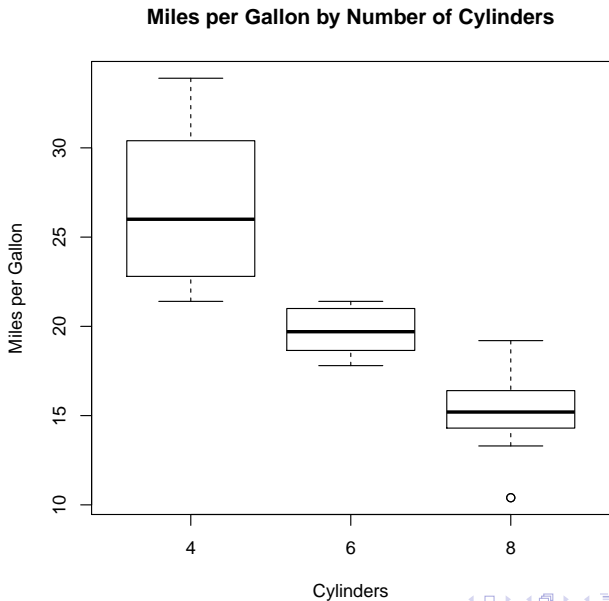
The boxplot function can also take a *formula*<sup>6</sup> as an argument

```
mpg ~ cyl "mpg conditional on cyl"
```

```
> boxplot(mpg ~ cyl,
+         data = mtcars,
+         main = "Miles per Gallon by Number of Cylinders",
+         xlab = "Cylinders",
+         ylab = "Miles per Gallon")
```

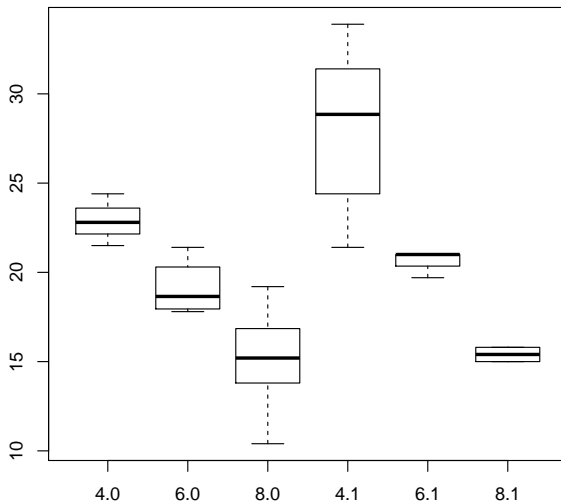
<sup>6</sup>an expression using vectors and the  $\sim$  operator. See `?formula` or `?~` for more

```
> boxplot(mpg ~ cyl,
+         data = mtcars,
+         main = "Miles per Gallon by Number of Cylinders",
+         xlab = "Cylinders",
+         ylab = "Miles per Gallon")
```



```
> # Expand the formula
> boxplot(mpg ~ cyl + am,
+         data = mtcars,
+         main = "MPG by Number of Cyliinders & Transmissio
```

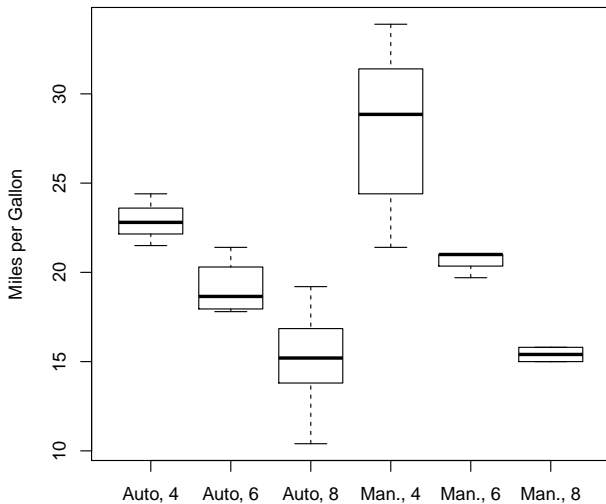
### MPG by Number of Cylinders & Transmission



```
> # Relabel the x axis
>
> xaxis <- c("Auto, 4", "Auto, 6", "Auto, 8",
+           "Man., 4", "Man., 6", "Man., 8")
> boxplot(mpg ~ cyl + am,
+         data = mtcars,
+         main = "MPG by Number of Cylinders & Transmission",
+         xlab = " ",
+         ylab = "Miles per Gallon",
+         names = xaxis)
```



### MPG by Number of Cylinders & Transmission

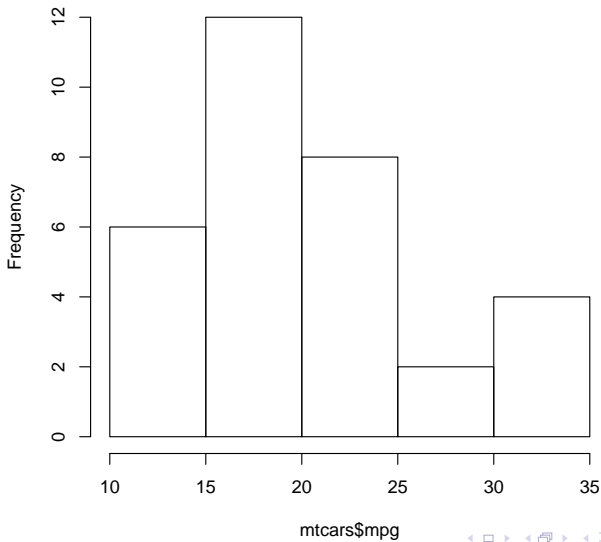


# Try it

```
> boxplot(mpg ~ cyl,  
+         data = mtcars,  
+         main = "Miles per Gallon by Number of Cylinders",  
+         xlab = "Cylinders",  
+         ylab = "Miles per Gallon")
```

0

```
> hist(mtcars$mpg)
```



```
> # Add options
> hist(mtcars$mpg,
+       breaks = 10,
+       main = "Histogram of Miles per Gallon",
+       xlab = "Miles per gallon",
+       col = "red")
```

A histogram showing the frequency of miles per gallon (mpg) for cars. The x-axis is labeled 'Miles per gallon' and ranges from 10 to 30. The y-axis is labeled 'Frequency' and ranges from 0 to 7. The histogram shows the frequency of cars in different mpg bins.

Miles per gallon bin	Frequency
10 - 12.5	2
12.5 - 15	1
15 - 17.5	7
17.5 - 20	3
20 - 22.5	5
22.5 - 25	5
25 - 27.5	2
27.5 - 30	2
30 - 32.5	1
32.5 - 35	0
35 - 37.5	2
37.5 - 40	2

```
> hist(mtcars$mpg,
+       breaks = 10,
+       main = "Histogram of Miles per Gallon",
+       xlab = "Miles per gallon",
+       col = "red")
```

# Bar Plot

Takes a named vector and plots it

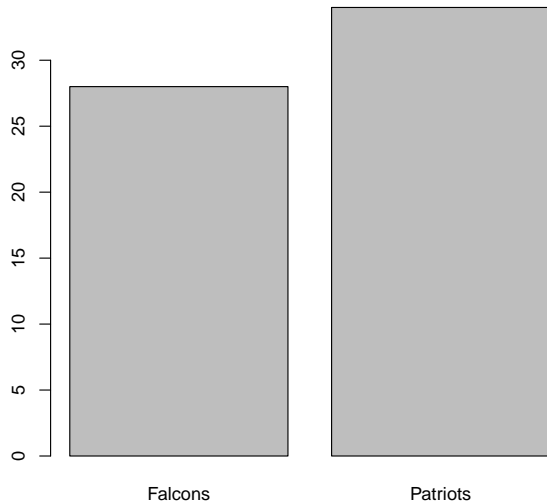
```
> scores <- c(Falcons = 28,  
+             Patriots = 34)
```



# Bar Plot

Takes a named vector and plots it

```
> scores <- c(Falcons = 28,  
+             Patriots = 34)  
  
> barplot(scores)
```



# Bar Plot<sup>7</sup>





Use the table function create named vector with counts

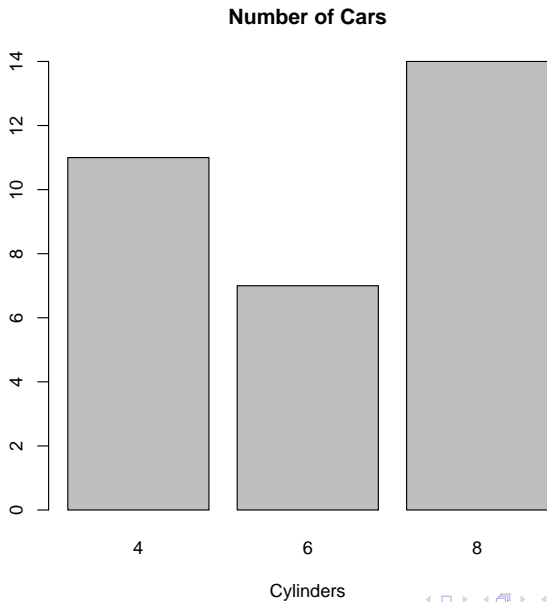
```
> counts <- table(mtcars$cyl)
> counts

 4  6  8
11  7 14

> barplot(counts,
+         main = "Number of Cars",
+         xlab = "Cylinders")
```

---

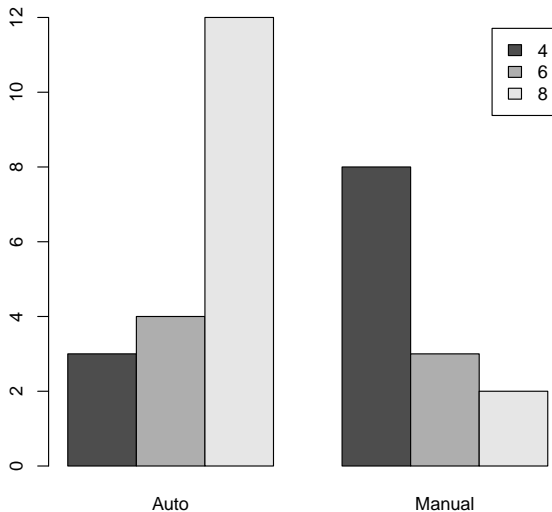
<sup>7</sup>Adapted from: <http://www.statmethods.net/graphs/bar.html>    



Use the table function to create a two-way frequency table, and plotting options to group bars

```
> counts <- table(mtcars$cyl, mtcars$am)
> colnames(counts) <- c("Auto", "Manual")
> barplot(counts,
+         main = "Number of Cars by Trasmission and Cylinders",
+         xlab = "Trasmission",
+         beside = TRUE,
+         legend = rownames(counts))
>
```

### Number of Cars by Trasmission and Cylinders



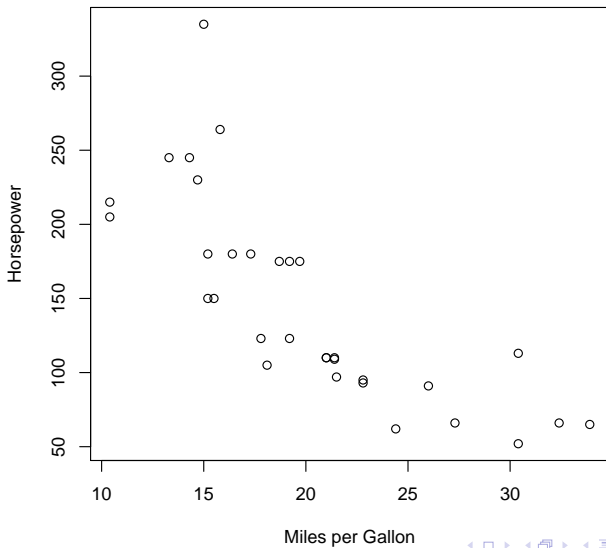
Trasmission

```
> # Tabulate
> counts <- table(mtcars$cyl, mtcars$am)
> colnames(counts) <- c("Auto", "Manual")
> # Plot
> barplot(counts,
+         main = "Number of Cars by Trasmission and Cylinders",
+         xlab = "Trasmission",
+         beside = TRUE,
+         legend = rownames(counts))
```

# Scatterplot

```
> plot(mtcars$mpg,  
+      mtcars$hp,  
+      xlab = "Miles per Gallon",  
+      ylab = "Horsepower")
```

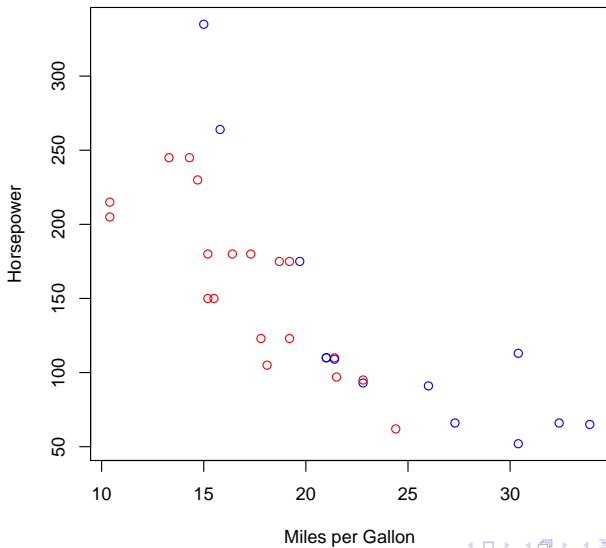




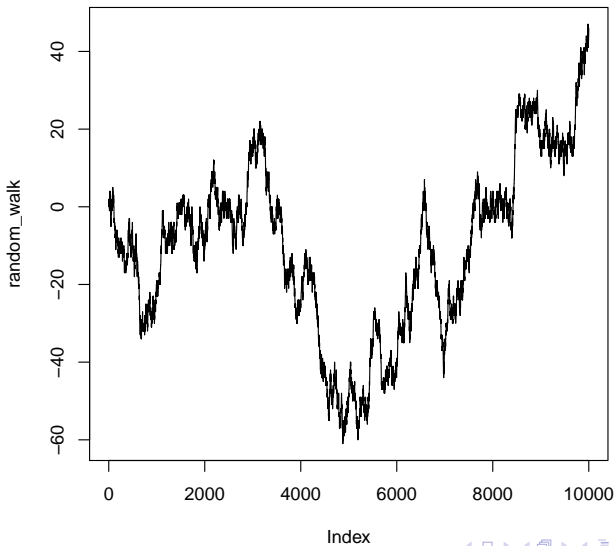
9

```
> plot(mtcars$mpg,
+      mtcars$hp,
+      xlab = "Miles per Gallon",
+      ylab = "Horsepower")
```





◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻



- subset rows
- subset columns
- make new variables
- aggregate rows
- merge data sets

```
> big_block <- subset(mtcars, disp > 328)
```



[1] 8 11



## Subset Columns: `subset()`

```
> size_metrics <- subset(mtcars,
+                          select = c("disp", "cyl", "wt"))
> dim(mtcars)
[1] 32 11
> dim(size_metrics)
[1] 32  3
```

- Use subset operators to isolate the column,
- Use the assignment operator to give it a new value.
- If the target variable doesn't exist yet, it will be added.

```
> mtcars$power_wt_ratio <- mtcars$hp / mtcars$wt
```

- Use subset operators to isolate the column,
- Use the assignment operator to give it a new value.
- If the target variable doesn't exist yet, it will be added.

```
> mtcars$power_wt_ratio <- mtcars$hp / mtcars$wt
```

```
> dim(mtcars)
```

[1] 32 12

```
> head(mtcars[, c("hp", "wt", "power_wt_ratio")])
```

	hp	wt	power_wt_ratio
Mazda RX4	110	2.620	41.98473
Mazda RX4 Wag	110	2.875	38.26087
Datsun 710	93	2.320	40.08621
Hornet 4 Drive	110	3.215	34.21462
Hornet Sportabout	175	3.440	50.87209
Valiant	105	3.460	30.34682

Three styles to choose from. All of the subset operators work for this purpose.

```
> mtcars$power_wt_ratio <- mtcars$hp / mtcars$wt
> mtcars[["power_wt_ratio"]] <- mtcars$hp / mtcars$wt
> mtcars[, "power_wt_ratio"] <- mtcars$hp / mtcars$wt
```

## Try it

Compute horsepower to weight ratio

```
> mtcars$power_wt_ratio <- mtcars$hp / mtcars$wt
> head(mtcars[, c("hp", "wt", "power_wt_ratio")])
```

## Make New Variables: examples

How might we go about creating variables to identify:

- fast cars (low 0-to-60 times)
- heavy cars (above average weight)
- domestic vs import (based on make & model name)



Use the `[i, j]` format to subset rows. Only the observations specified by the `i` expression are affected. Repeat as necessary

```
> # quantile threshold values
```

```
> q20 <- quantile(mtcars$qsec, probs = .20)
```

```
> q80 <- quantile(mtcars$qsec, probs = .80)
```

## Make New Variables: Assigning to a subset

Use the `[i, j]` format to subset rows. Only the observations specified by the `i` expression are affected. Repeat as necessary

```
> # quantile threshold values
```

```
> q20 <- quantile(mtcars$qsec, probs = .20)
```

```
> q80 <- quantile(mtcars$qsec, probs = .80)
```

```
> # pick rows for fast cars
```

```
> # write to the new variable "quickness"
```

```
> mtcars[mtcars$qsec <= q20, "quickness"] <- "fast"
```

```
> # quantile threshold values
```

```
> q20 <- quantile(mtcars$qsec, probs = .20)
```

```
> q80 <- quantile(mtcars$qsec, probs = .80)
```

```
> # pick rows for fast cars
```

```
> # write to the new variable "quickness"
```

```
> mtcars[mtcars$qsec <= q20, "quickness"] <- "fast"
```

```
> # only fast cars affected
```

```
> head(subset(mtcars, select = "quickness"))
```

quickness

Mazda RX4 fast

Mazda RX4 Wag &lt;NA&gt;

Datsun 710 <NA>

Hornet 4 Drive &lt;NA&gt;

Hornet Sportabout (NA)

```
> # pick rows for moderate cars
> # write to the existing variable "quickness"
> i <- q20 < mtcars$qsec & mtcars$qsec <= q80
> mtcars[i, "quickness"] <- "normal"
```

## Assigning to a subset of rows

Repeat for remaining subsets

```
> # pick rows for moderate cars
> # write to the existing variable "quickness"
> i <- q20 < mtcars$qsec & mtcars$qsec <= q80
> mtcars[i, "quickness"] <- "normal"

> # only moderate cars are affected
> head(subset(mtcars, select = "quickness"))
```

	quickness
Mazda RX4	fast
Mazda RX4 Wag	normal
Datsun 710	normal
Hornet 4 Drive	<NA>
Hornet Sportabout	normal
Valiant	<NA>

## Make New Variables: Assigning to a subset

Repeat for last subset

```
> mtcars[mtcars$qsec > q80, "quickness"] <- "slow"
> head(subset(mtcars, select = "quickness"))
```

	quickness
Mazda RX4	fast
Mazda RX4 Wag	normal
Datsun 710	normal
Hornet 4 Drive	slow
Hornet Sportabout	normal
Valiant	slow

### Alternate approach: `ifelse()` function

```
> mean_wt <- mean(mtcars$wt)
> mtcars$weight_class <- ifelse(mtcars$wt <= mean_wt,
+                               "light", # true cases
+                               "heavy") # false cases
> # result assigned conditionally
> head(subset(mtcars, select = c("wt", "weight_class")))
```

	wt	weight_class
Mazda RX4	2.620	light
Mazda RX4 Wag	2.875	light
Datsun 710	2.320	light
Hornet 4 Drive	3.215	light
Hornet Sportabout	3.440	heavy
Valiant	3.460	heavy



`aggregate()`: Splits the data into subsets, computes summary statistics for each, and returns the result<sup>8</sup>

```
> aggregate(mtcars$mpg,           # data
+           by = list(mtcars$cyl), # grouping variables
+           mean)                  # function
```

	Group.1	x
1	4	26.66364
2	6	19.74286
3	8	15.10000

<sup>8</sup>R documentation: `?aggregate()`

```
> df <- aggregate(mtcars$mpg,
+                 by = list(mtcars$cyl, mtcars$am),
+                 mean)
> names(df) <- c("cyl", "am", "avg_mpg")
> df
```

	cyl	am	avg_mpg
1	4	0	22.90000
2	6	0	19.12500
3	8	0	15.05000
4	4	1	28.07500
5	6	1	20.56667
6	8	1	15.40000

# Merging

## Combine data frames based on shared values

```
> # Sample Data
```

x1 x2

2 b 2

3 c 3

x1      x3

2 b FALSE

```
3  d  TRUE
```

100

```
> df <- merge(A, B, by = "x1")
```

```
> print(df)
```

	x1	x2	x3
1	a	1	TRUE
2	b	2	FALSE

100

```
> df <- merge(A, B, by = "x1", all = TRUE)
```

```
> print(df)
```

	x1	x2	x3
1	a	1	TRUE
2	b	2	FALSE
3	c	3	NA
4	d	NA	TRUE



# Merging

All values of  $x_1$  in  $A$

```
> df <- merge(A, B, by = "x1", all.x = TRUE)
```

# Merging

All values of  $x_1$  in  $A$

```
> df <- merge(A, B, by = "x1", all.x = TRUE)
```

```
> print(df)
```

	x1	x2	x3
1	a	1	TRUE
2	b	2	FALSE
3	c	3	NA

\_\_\_\_\_

TABLE 1. *Continued*

# Merging

All values of  $x_1$  in  $B$

```
> df <- merge(A, B, by = "x1", all.y = TRUE)
```

```
> print(df)
```

	x1	x2	x3
1	a	1	TRUE
2	b	2	FALSE
3	d	NA	TRUE

```
> A$x3 <- NA           # add x3 column to A
> B$x2 <- NA           # add X2 column to B
> df <- rbind(A, B)
```

```
> A$x3 <- NA          # add x3 column to A
> B$x2 <- NA          # add X2 column to B
> df <- rbind(A, B)

> print(df)
```

	x1	x2	x3
1	a	1	NA
2	b	2	NA
3	c	3	NA
4	a	NA	TRUE
5	b	NA	FALSE
6	d	NA	TRUE

`read.table()` family of functions: read raw data saved in delimited text files, and return a data frame object.

```
> ?read.table()
```

```
> ?read.csv()
```

```
> ?read.delim()
```