Data Analytics
CS301
Basic Stats

Week 7: 28th Feb
Spring 2020
Oliver BONHAM-CARTER

Writing Functions

(You might need this later!)

```
functionName <- function(arg1, arg2, arg3=2, ...) {
  newVar <- sin(arg1) + sin(arg2) # do useful stuff
  newVar / arg3  # Return value }
```

```
functionName(2,3,1) # run function with inputs
```

- **functionName**: is the function's name
- **args**: arguments of the function, also called formals to import data into a function. No limit to the number for a function.
- **Return value**: The last line of the code is the value that will be returned by the function. It is not necessary that a function return anything

Example of Function

```
#Return the sum of squares:
sumOfSquares <- function(x,y) {
  x^2 + y^2
 }
#run sumOfSquares () with x=2 and y=4
sumOfSquares(2,4) # returns 20
```

Another Simple Example

```
# function to plot points on the canvas
redPlot <- function(x, y) {
        plot(x, y, col="red")
        }
# run the function
redPlot (2,4) # plot a red point
redPlot (c(2:10), c(2:10)) # a series of points
```

Another Simple Example

```
# determine points and color
colorPlot <- function(x, y, c) {
        plot(x, y, col=c)
        }
# run the function
colorPlot(x, y, "red") # plot a red point
colorPlot (c(2:10), c(2:10), "blue")
```

Yet, Another Example: Using An If-Else Statement

```
GimmeAtLeastFive <- function(inNum){
  if(inNum >= 5){
      print("That is at least five")
    }
  else{
      print("not enough")
  }
}
```

Basic Stats

- We will spend some time looking at different types of statistical tests so that they can be implemented in code.

Medians

Median

First, arrange the observations in an ascending order.

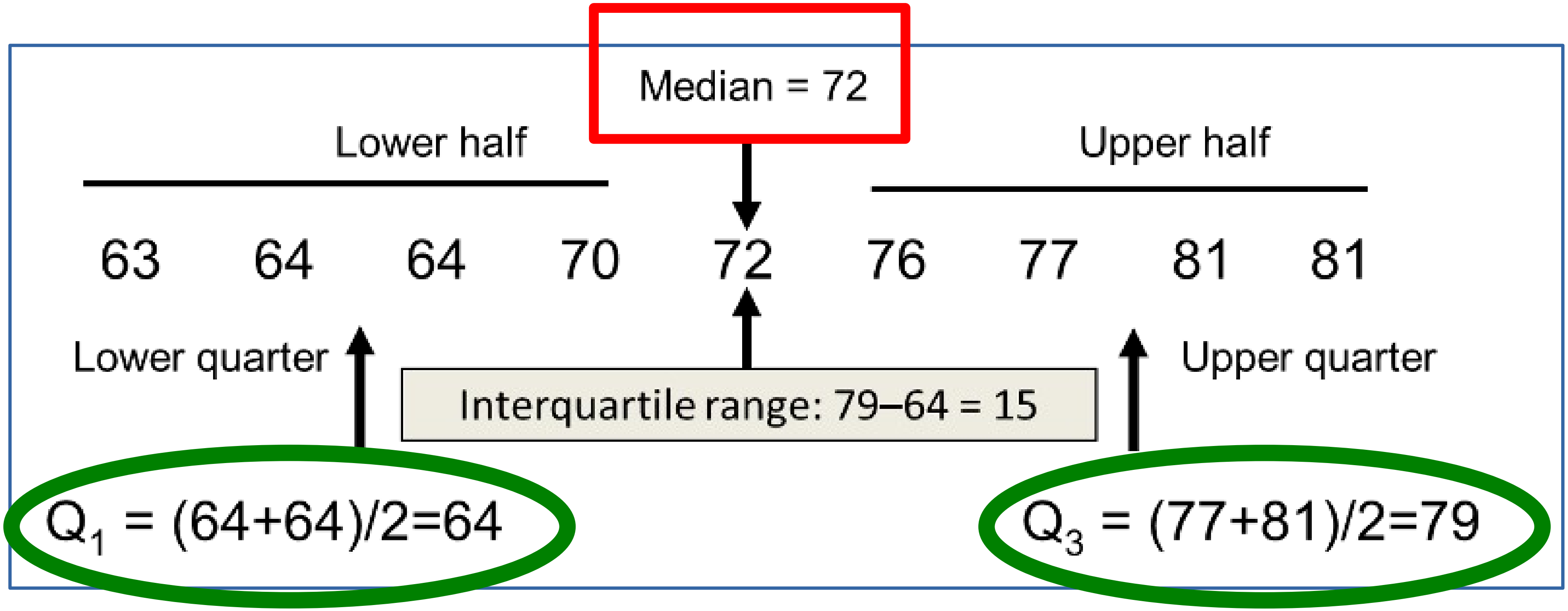
If the number of observations (n) is odd: the median is the value at position

$$\left(\frac{n+1}{2}\right)$$

If the number of observations (n) is even:

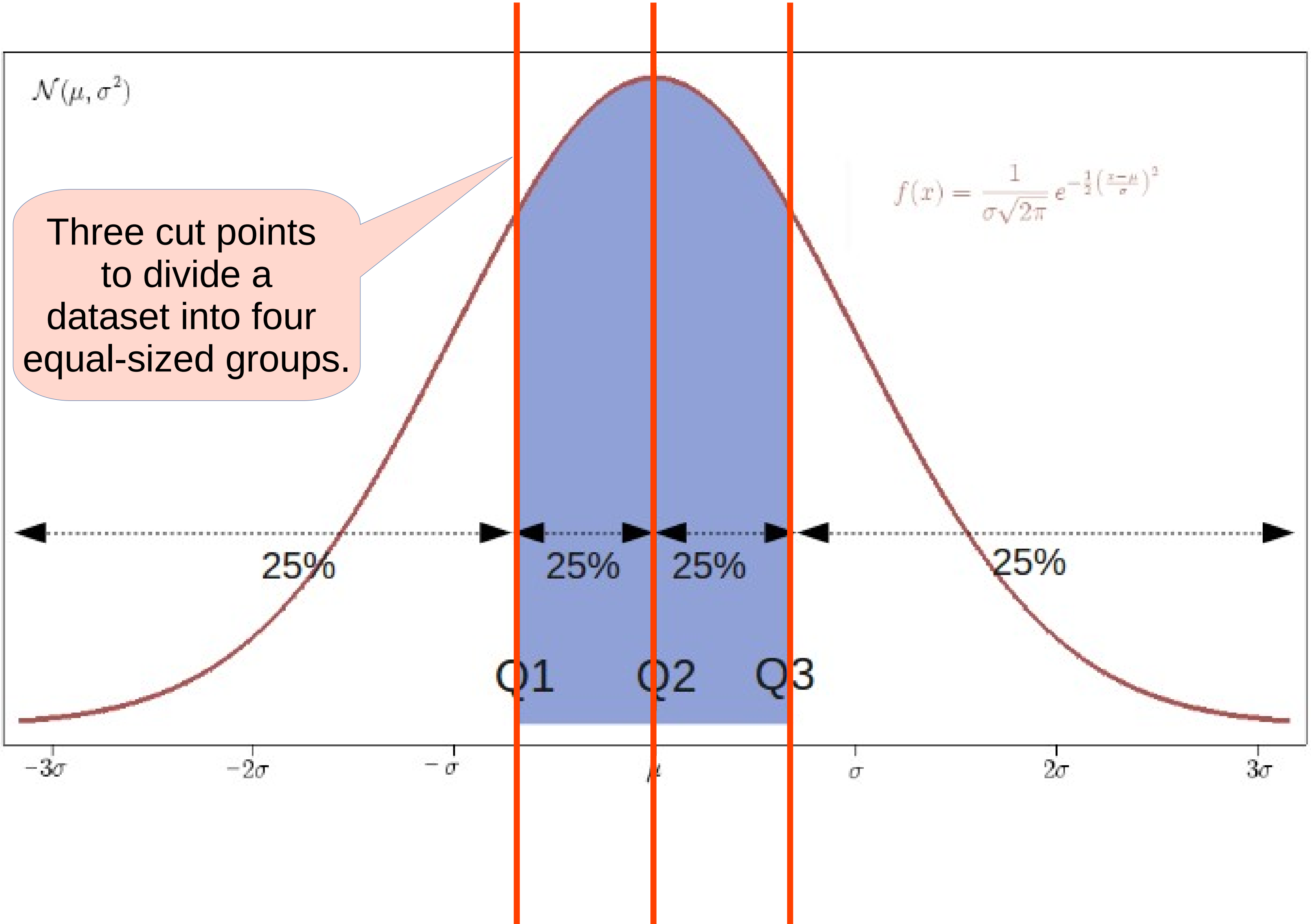
1. Find the value at position $\left(\frac{n}{2}\right)$
2. Find the value at position $\left(\frac{n+1}{2}\right)$
3. Find the average of the two values to get the median.

Medians



- What does Q1 and Q3 indicate?
  - Quantiles: allow us to determine placements in the set of numbers

Quantiles:
Quarters of Data



Quantiles:

Quarters of Data

```
# find the quantiles of the following set.
qnums <- c(3, 6, 7, 8, 8, 10, 13, 15, 16, 20)
summary(qnums)
```

```
> qnums <- c(3, 6, 7, 8, 8, 10, 13, 15, 16, 20)
> summary(qnums)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   3.00    7.25    9.00   10.60   14.50   20.00 
```

Finding Quantiles

- Finding 1st and 3rd quantiles is to determine the positions at the ¼ and ¾ marks, respectively.

Quartile	Calculation	Result
Zeroth quartile	Although not universally accepted, one can also speak of the zeroth quartile. This is the minimum value of the set, so the zeroth quartile in this example would be 3.	3
First Quantile	The rank of the first quartile is 10×(1/4) = 2.5, which rounds up to 3, meaning that 3 is the rank in the population (from least to greatest values) at which approximately 1/4 of the values are less than the value of the first quartile. The third value in the population is 7.	7
Second Quantile	The rank of the second quartile (same as the median) is 10×(2/4) = 5, which is an integer, while the number of values (10) is an even number, so the average of both the fifth and sixth values is taken—that is (8+10)/2 = 9, though any value from 8 through to 10 could be taken to be the median.	9
Third Quantile	The rank of the third quartile is 10×(3/4) = 7.5, which rounds up to 8. The eighth value in the population is 15.	15
Fourth quartile	Although not universally accepted, one can also speak of the fourth quartile. This is the maximum value of the set, so the fourth quartile in this example would be 20. Under the Nearest Rank definition of quantile, the rank of the fourth quartile is the rank of the biggest number, so the rank of the fourth quartile would be 10.	20

1st 2nd 3rd

Original Data: 3, 6, **7**, 8, **8, 10**, 13, **15**, 16, 20

Consider this... `summary()`

Choose "AirPassengers" having only one column.

```
View(AirPassengers)
# general meta data
summary(AirPassengers)
```

	AirPassengers
1	112
2	118
3	132
4	129
5	121
6	135
7	148
8	148

```
> summary(AirPassengers)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  104.0   180.0   265.5   280.3   360.5   622.0
```

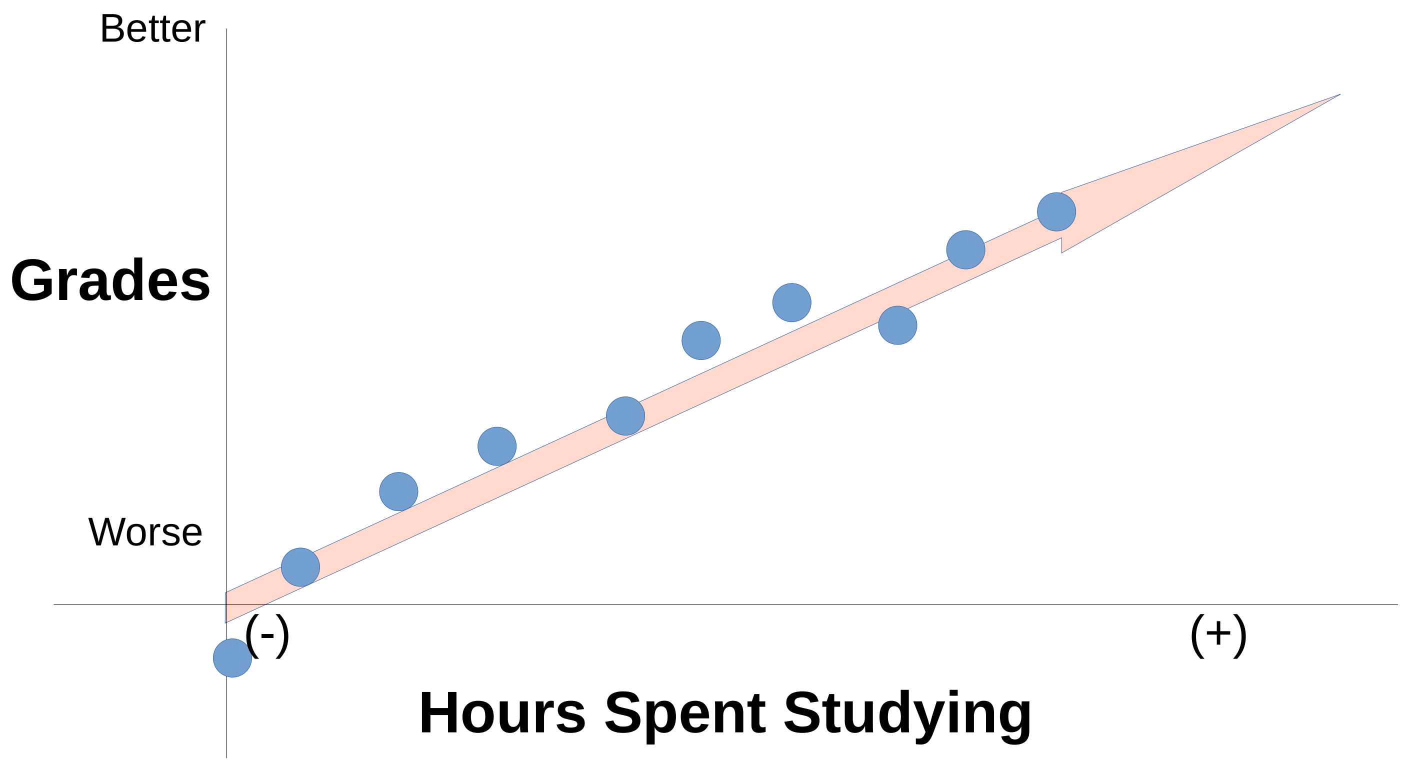
Consider this… `summary()`

- Min: Minimum value (lower bound)
- Max: Maximum value (upper bound)
- Mean: Average value across the set
- Median:
  - The middle number (if num of observations is odd)
  - The average of the middle pair (if num of observations is even)

```
> summary(AirPassengers)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  104.0   180.0   265.5   280.3   360.5   622.0
```

Correlation

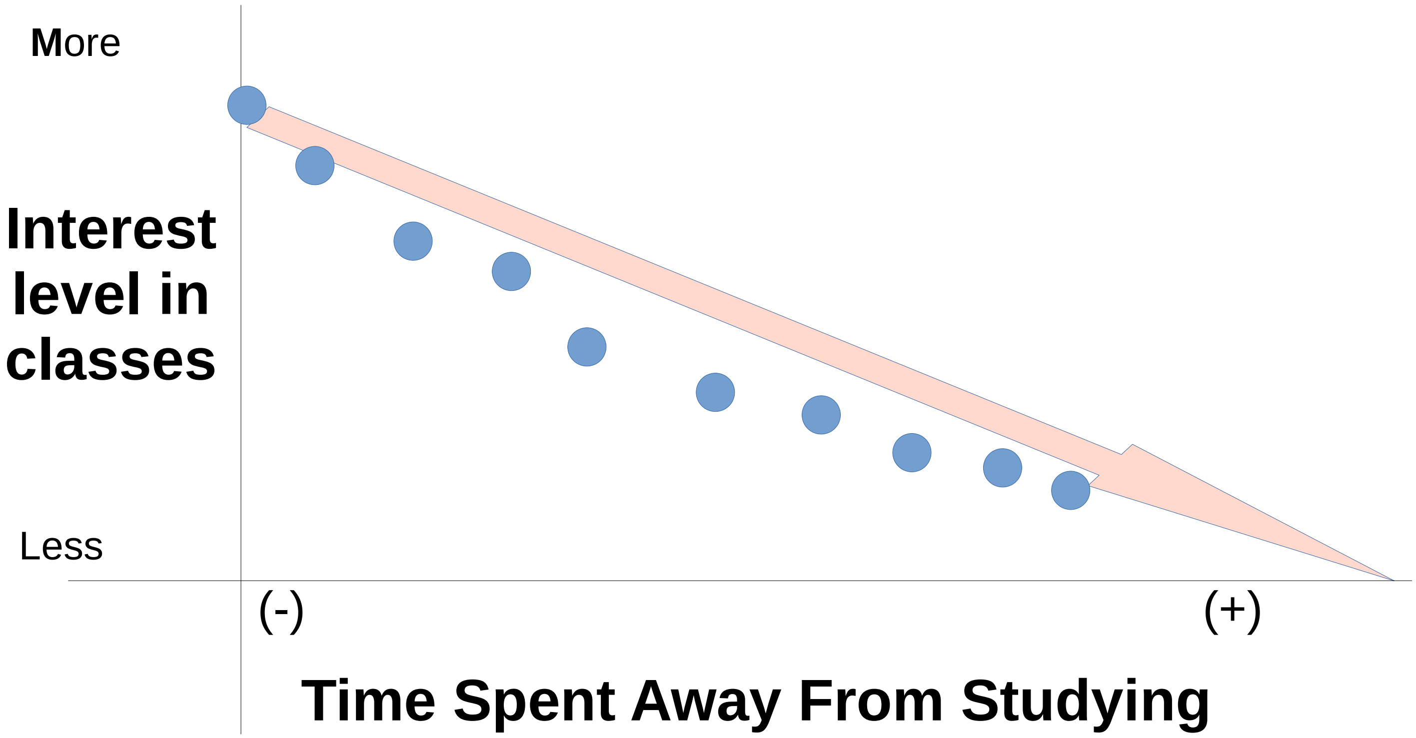
- **Positive correlation** exists when two variables move in the same direction.



Points lie close to a straight line that has a **positive** gradient.

Correlation

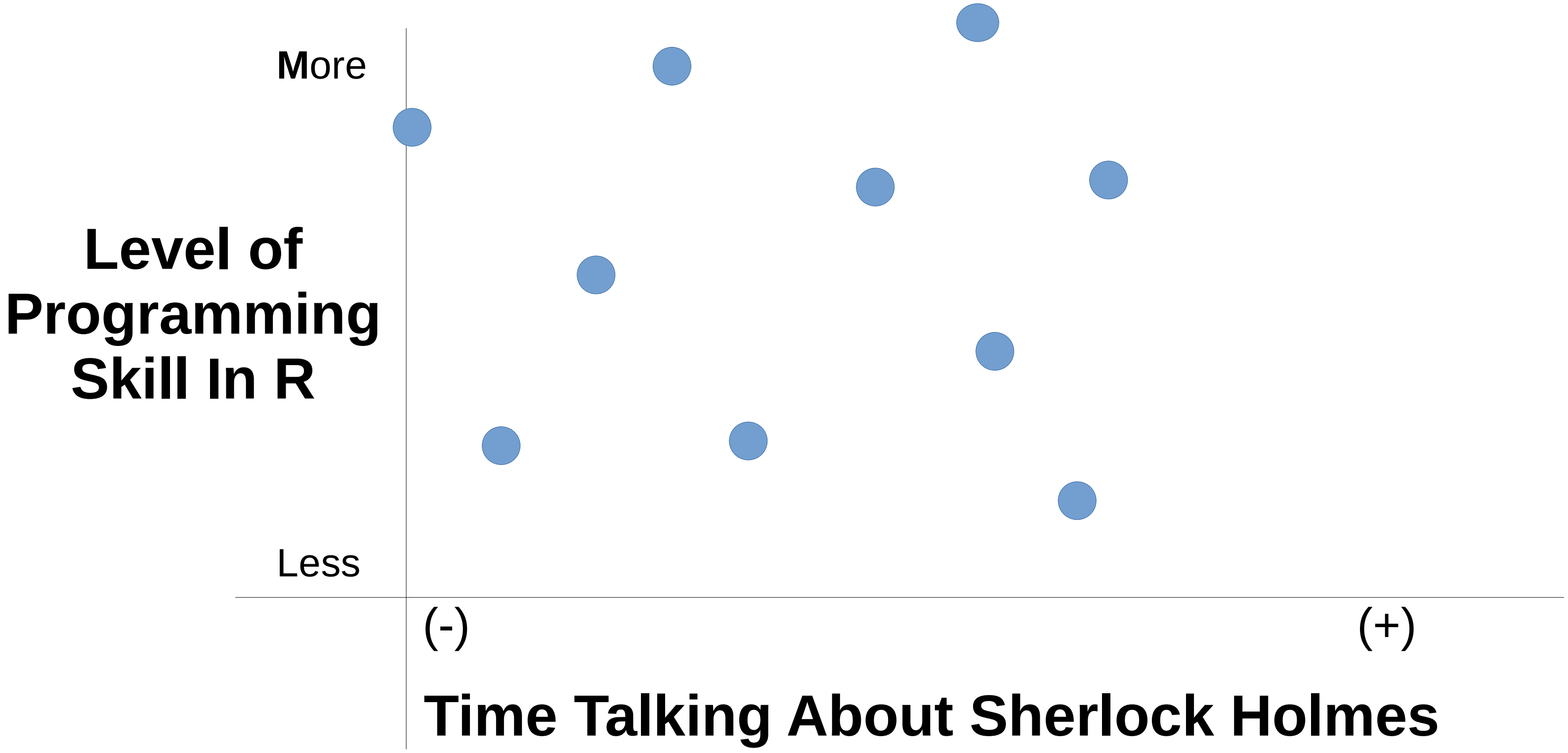
- **Negative correlation** exists when two variables move in the opposite directions.



Points lie close to a straight line that has a **negative** gradient.

Correlation

- **No correlation** exists when two variables are independent of each other.

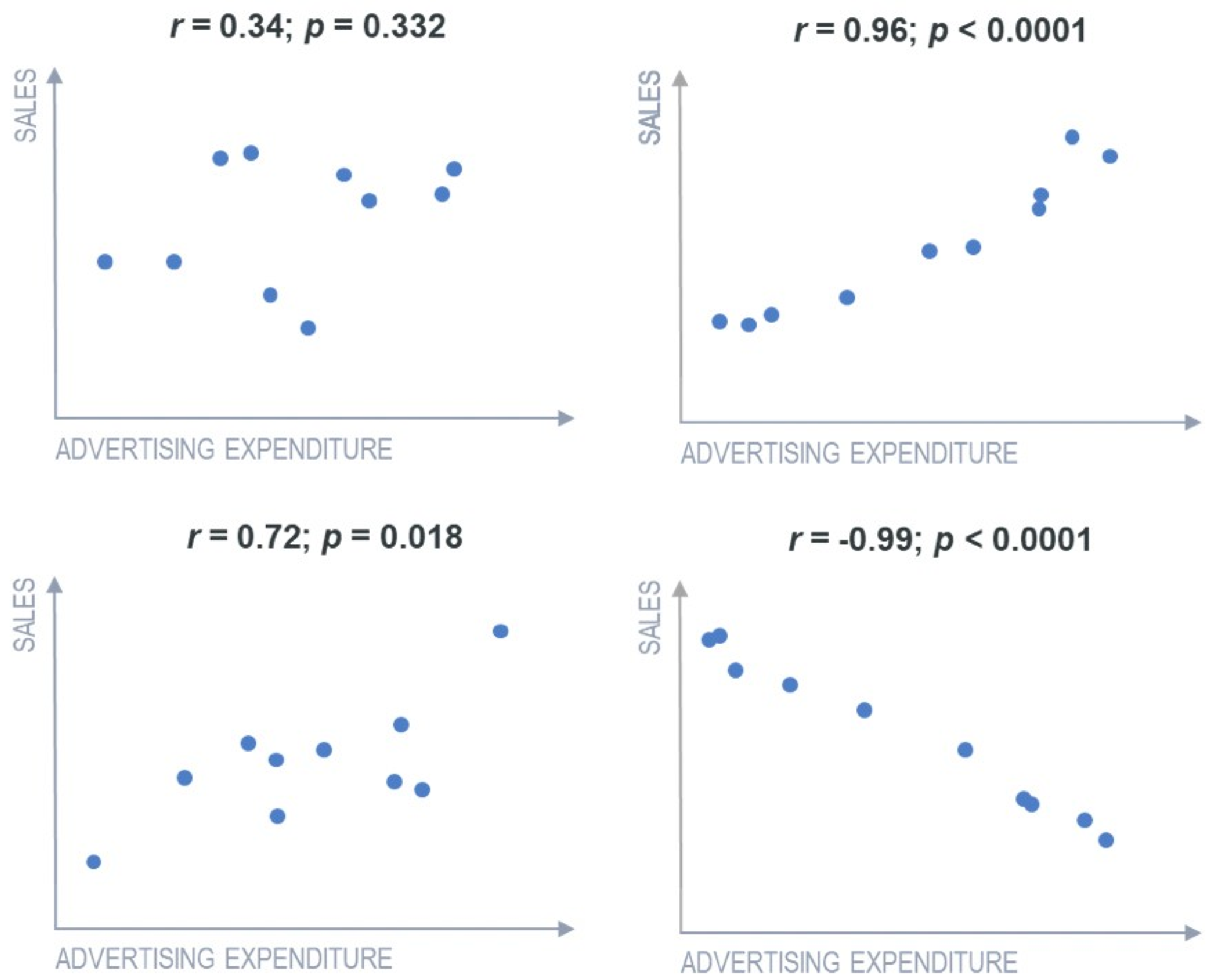


No pattern exists in the layout of points. :-(

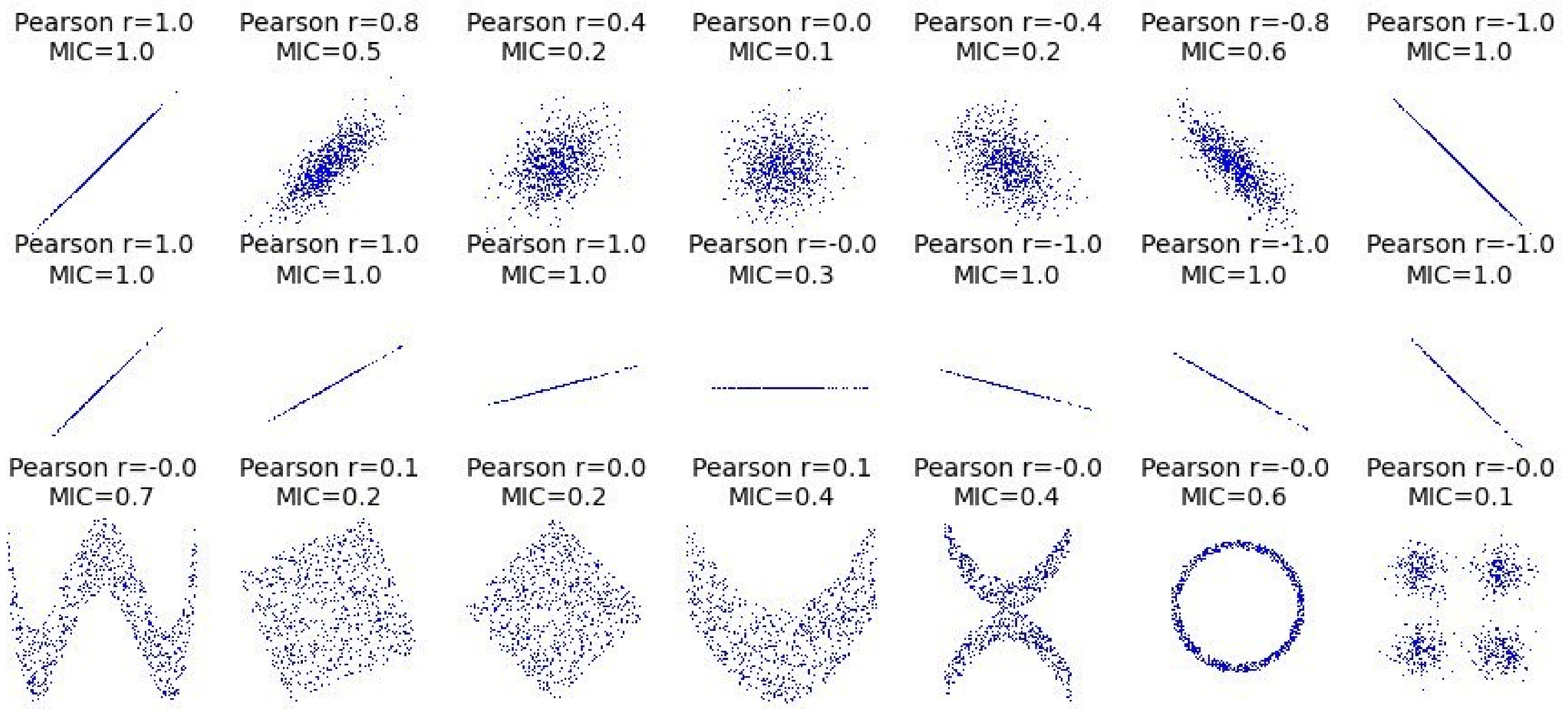
Measurements of Correlation (By the Numbers)

- A correlation of 1 indicates a perfect positive correlation.
- A correlation of -1 indicates a perfect negative correlation.
- A correlation of 0 indicates that there is no relationship between the different variables.
- Values between -1 and 1 denote the strength of the correlation, as shown in the example below.

Measurements of Correlation



Measurements of Correlation

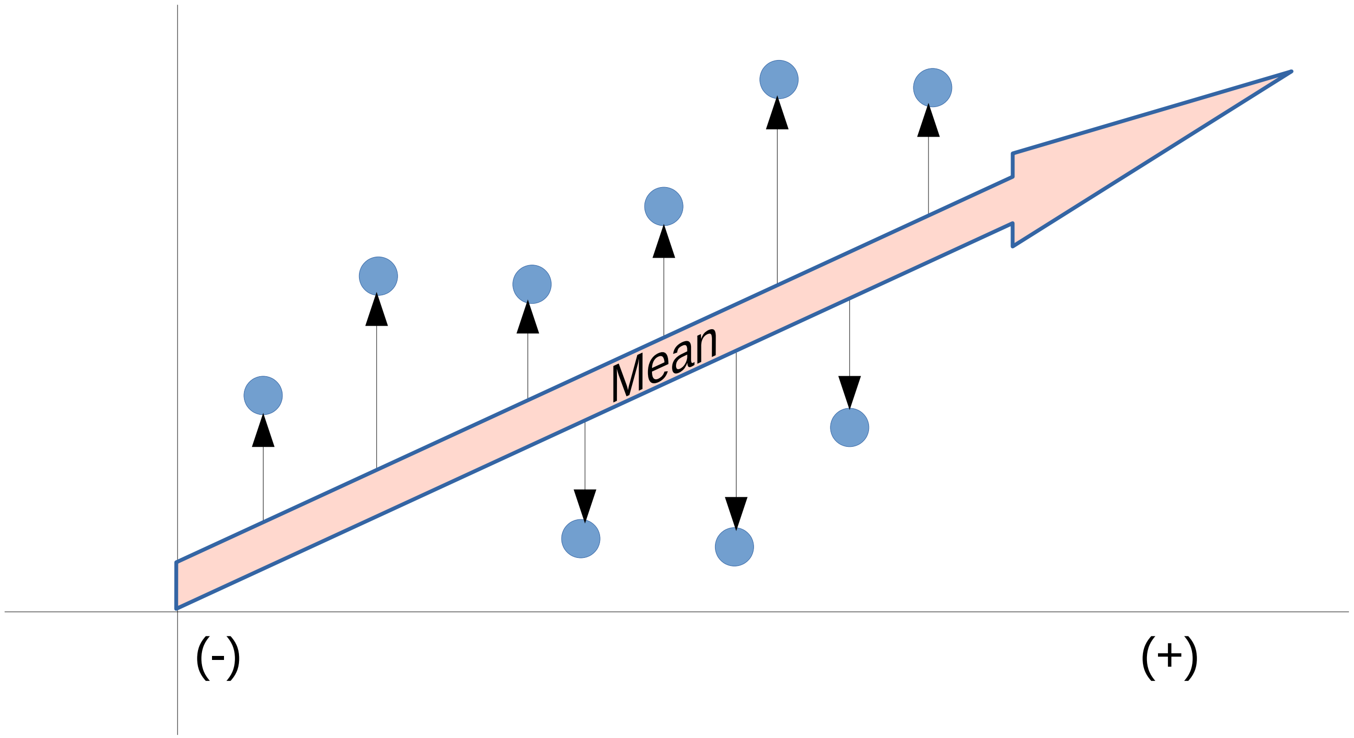


Examples Taken From Online Help

```
? cor # online help
## Two simple vectors
cor(1:10, 2:11) # == 1
cor(1:10, -2:-11) # == -1

## matrix
cor(longley)

## Correlation Matrix of Multivariate sample:
(Cl <- cor(longley))
## Graphical Correlation Matrix:
symnum(Cl) # highly correlated

## Pearson’s r
symnum(clP <- cor(longley, method = "pearson")) #default

## Spearman's rho,  Kendall's tau and
# Data is of non-bivariate normal distribution
symnum(clS <- cor(longley, method = "spearman"))
symnum(clK <- cor(longley, method = "kendall"))


## How much do they differ?
i <- lower.tri(Cl)
cor(cbind(P = Cl[i], S = clS[i], K = clK[i]))
```

Variance

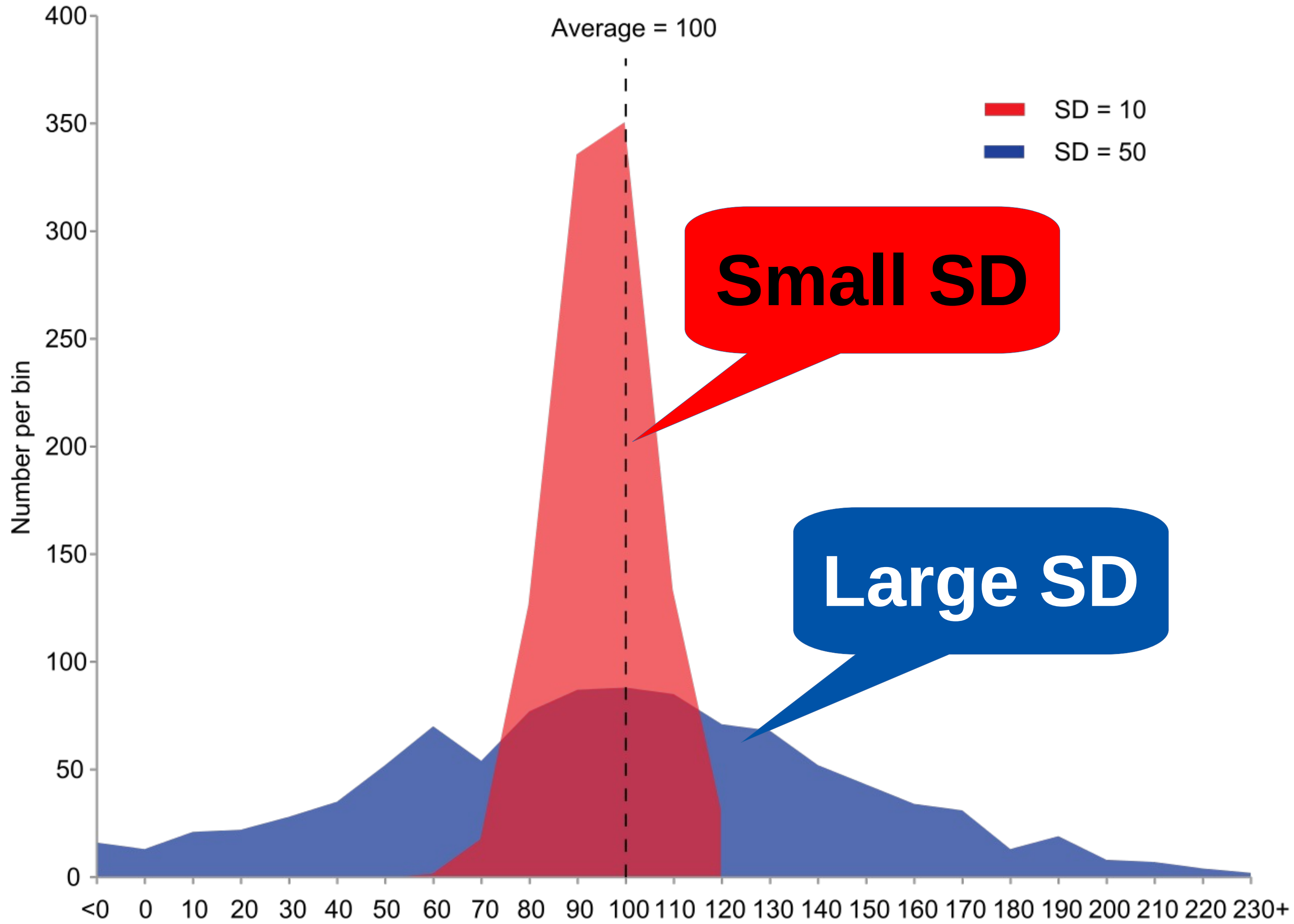
- The average of the squared differences from the mean of a dataset.
- The difference between our results and the expectation




```
myData <- c( 9, 2, 5, 4, 12, 7, 8, 11, 9, 3, 7, 4, 12, 5, 4, 10, 9, 6, 9, 4)
var(myData) # Variance is 9.368421

# Standard Deviation
sqrt(var(myData)) # Standard deviation is 3.060788
```

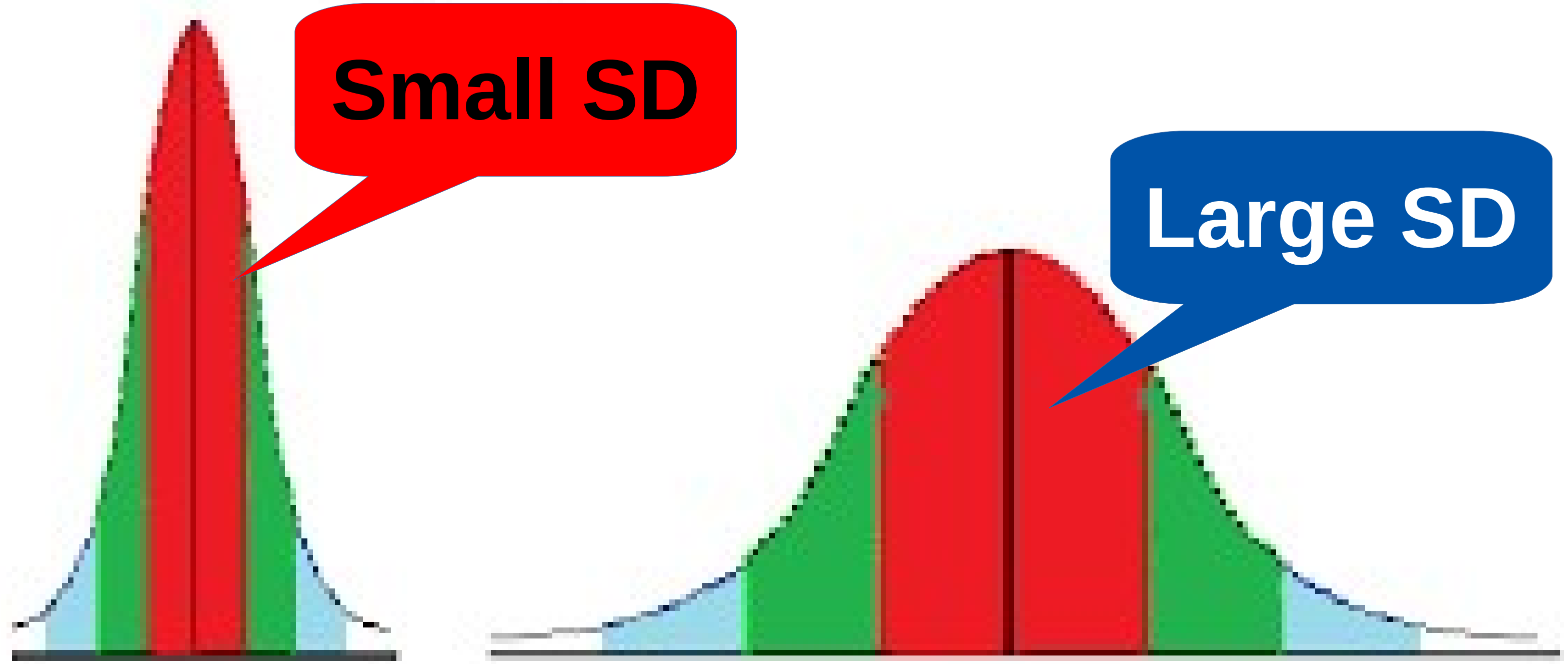
Standard Deviation

- A quantity calculated to indicate the extent of deviation for a group as a whole.
- An idea of the *spread* of data from the mean

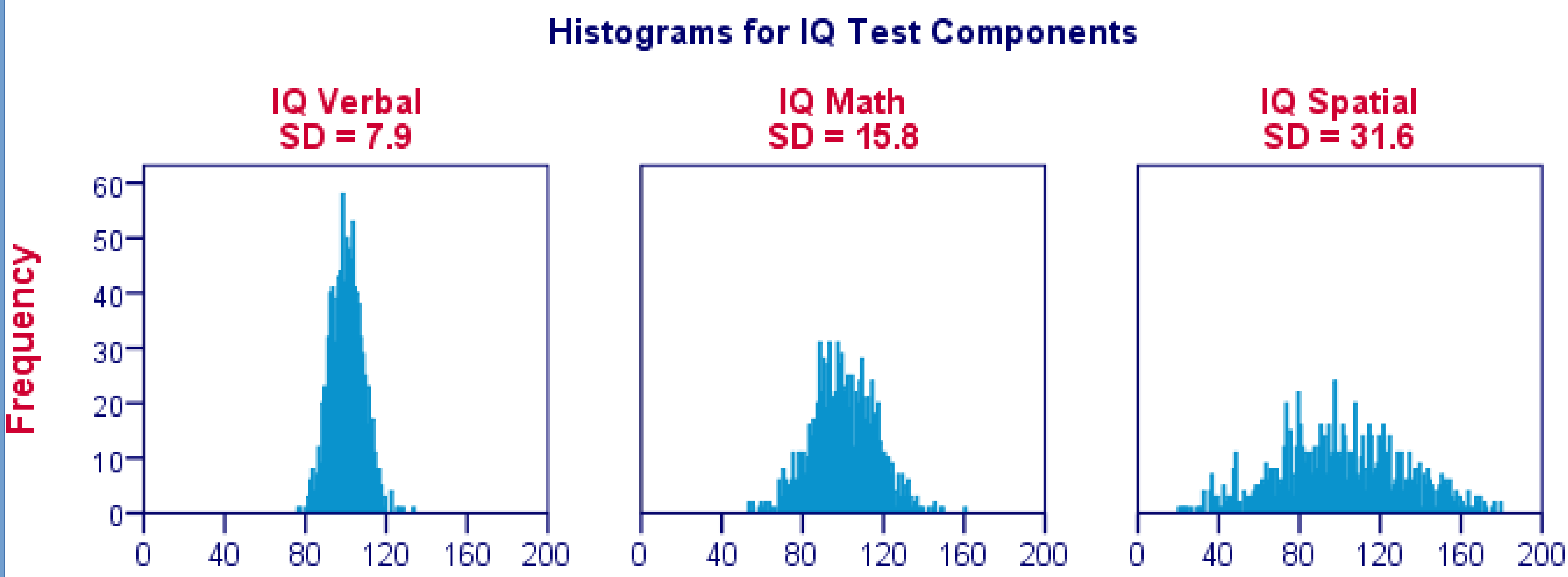


Standard Deviation

- Equal to the square root of variance(X)
  - sqrt(Var(X))
- A measure that is used to quantify the amount of variation or dispersion of a set of data values.
- A **low standard deviation** indicates that the data points tend to be **close to the mean** (also called the expected value) of the set
- A **high standard deviation** indicates that the data points are **spread out** over a wider range of values.



Small SD
Large SD



Histograms for IQ Test Components
IQ Verbal
SD = 7.9
IQ Math
SD = 15.8
IQ Spatial
SD = 31.6
Frequency
0
10
20
30
40
50
60
0
40
80
120
160
200

Putting Things Together: Find Some Basic Stats

```
library(dplyr) # and load tidyverse too!
data_people <- tibble::tribble(
  ~EyeColour, ~Height, ~Weight, ~Age,
  "Blue",          1.8, 110L, 18L,
  "Brown",         1.9, 150L, 34L,
  "Blue",          1.7, 207L, 28L,
  "Brown",         1.9, 170L, 21L,
  "Blue",          1.9, 164L, 29L,
  "Brown",         1.9, 183L, 31L,
  "Brown",         1.9, 175L, 20L,
  "Blue",          1.9, 202L, 27L
)
```

Calculate the Body Mass Index (BMI)

```
# Find the average BMI of people with blue eyes using piping

# Note: BMI = (height / (weight * weight))

data_people %>% select(EyeColour, Height, Weight) %>%
filter(EyeColour=="Blue") %>% mutate(BMI = Weight / Height^2)
%>% summary(averageBMI == mean(BMI))
```

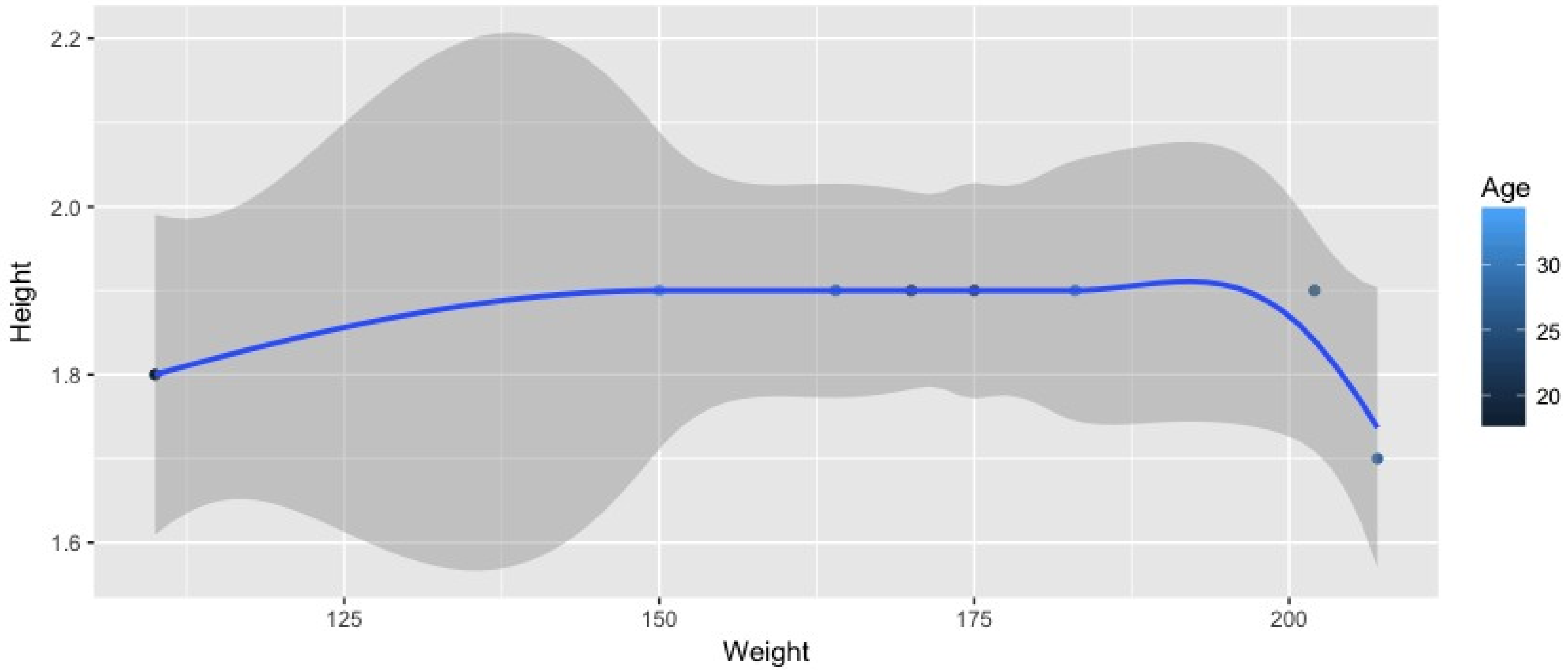
```
  EyeColour             Height          Weight           BMI
 Length:4           Min.   :1.700   Min.   :110.0   Min.   :33.95
 Class :character   1st Qu.:1.775   1st Qu.:150.5   1st Qu.:42.56
 Mode  :character   Median :1.850   Median :183.0   Median :50.69
                    Mean   :1.825   Mean   :170.8   Mean   :51.74
                    3rd Qu.:1.900   3rd Qu.:203.2   3rd Qu.:59.87
                    Max.   :1.900   Max.   :207.0   Max.   :71.63
```

The actual data

```
data_people %>% select(EyeColour, Height, Weight) %>%
filter(EyeColour=="Blue") %>% mutate(BMI = Weight / Height^2)
```

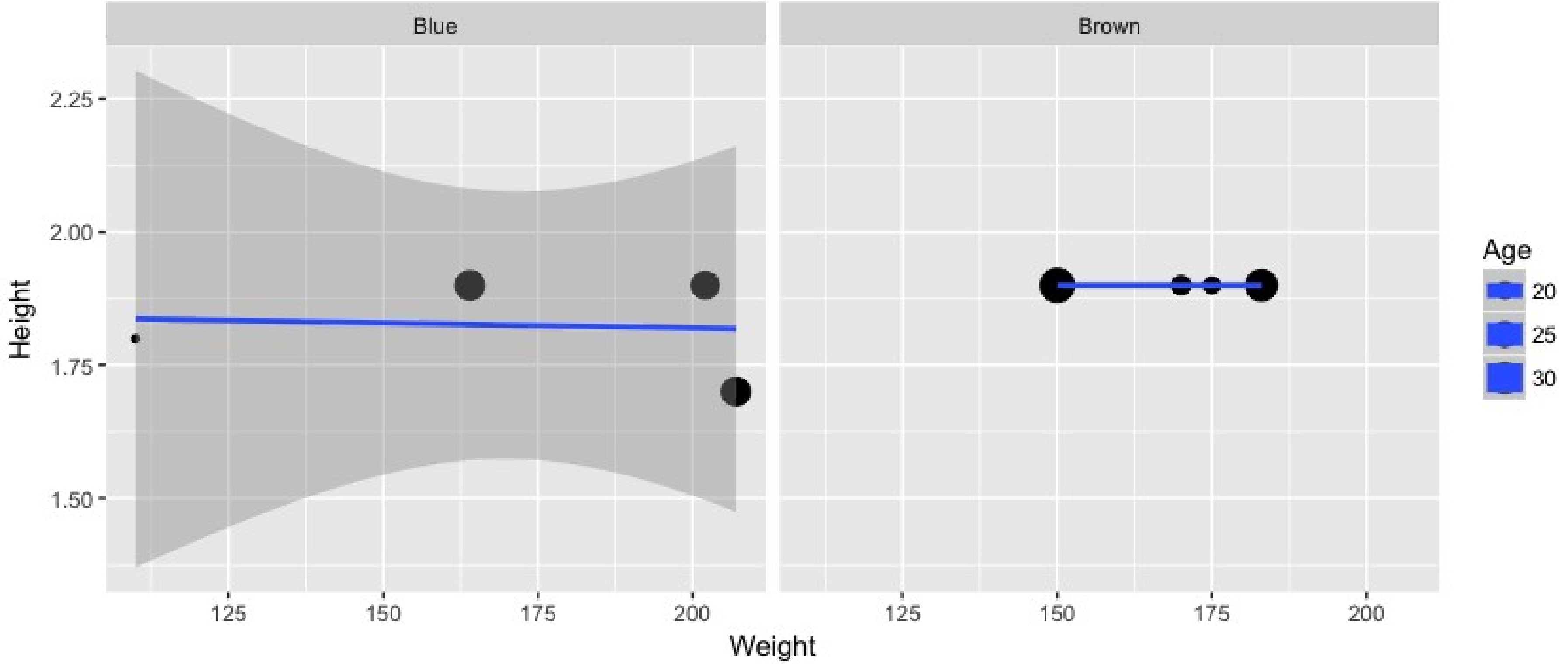
ggplot()

```
data_people %>% filter(Height, Weight) %>%
ggplot(aes(x = Weight, y = Height, col = Age))
+ geom_point() + geom_smooth()

# Try playing with the settings!!
```



More With ggplot()!

```
data_people %>% filter(Height, Weight) %>%
ggplot(aes(x = Weight, y = Height, size = Age, col =
Age)) + geom_point() + geom_smooth(method = lm) +
facet_wrap(~EyeColour)

#Note: geom_smooth applies a linear model
```



Basic Stats:
Working With *p*-values

- Suppose: We are the producers of two kinds of drinks: green and purple. Each drink comes in a bottle and we would like to know whether the green and the purple drinks are filled to the same levels.
- We randomly select 9 bottles from our entire set of 100000 bottles

Comparing Populations

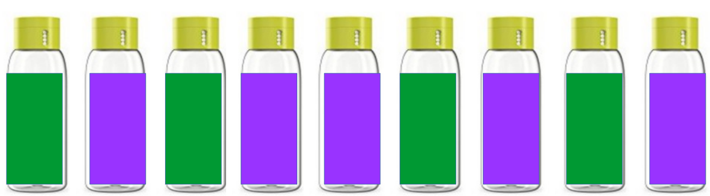
- By inspection,
  - **Purple bottles seem a little under-filled**
  - **Green bottles seem a little over-filled**
- Can we use a statistical test to conclude whether the whole batch is under- or over-filled?

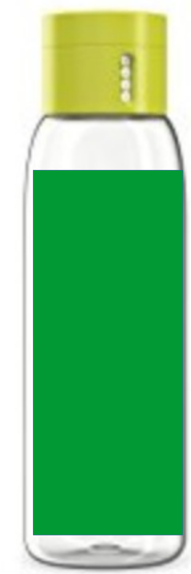
Hypothesis Testing

- We want to know: **Is there a statistically significant difference between the two groups in terms of the average extent to which the bottles are filled?**
  - **Null hypothesis (Ho)**: The bottles are filled at the same levels.
  - **Alternative hypothesis (Ha)**: There is a difference between the levels of drink in the bottles.
- Remember: we have a sample of ***only nine bottles from the super set of 100000 bottles***.
- Statistics is used to extrapolate from the small set to the larger set.

Is Our Sample Telling the Truth?

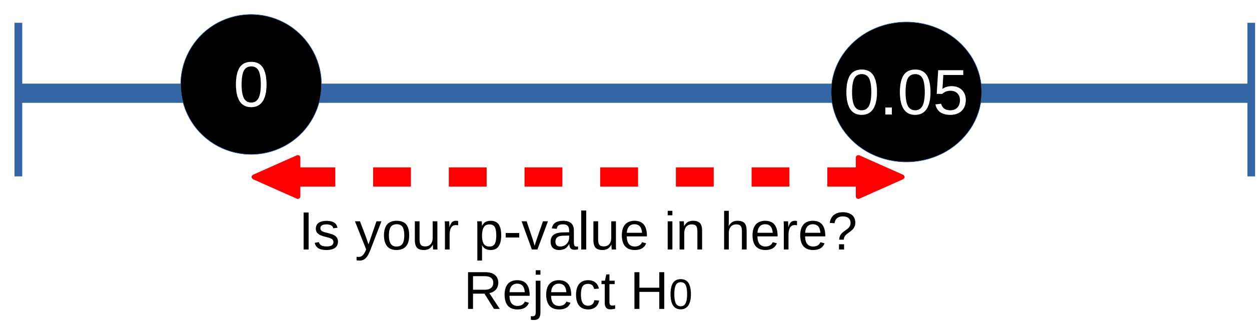
- We admit that our sample-selection may not necessarily represent our larger stock of bottles:
- The sample we selected for the test may still show that the green and purple bottles have been filled differently (*by accident*).





Use *p*-Values

- The p-Value says that we are sure that our sample size that we randomly selected is a good representation of our larger super set.
- Use a 95 confidence interval range: Our selected bottles fit within 95 percent of the entire set, meaning, a good representation of the entire set of 100000 bottles.
- **Reject the Null Hypothesis (H_0) when $p < 0.05$** (when *p* is close to zero)
- Rejecting H_0 means that something *non-random* is happening.



Basic Stats: Run a T-Test

```
data_drinks <- tibble::tribble(
  ~Observation, ~Colour, ~percentFull,
  1,"Green", 70,
  2,"Purple",30,
  3,"Green",50,
  4,"Purple",20,
  5,"Purple",15,
  6,"Green",90,
  7,"Purple",40,
  8,"Green",60,
  9,"Purple",15)
```

Basic Stats: T-Tests

```
data_drinks <- data_drinks %>%
    select(Colour, percentFull) #lose obs. num
#Run the t-test: a comparison of means.
t.test(data = data_drinks, percentFull ~ Colour)
```

 # Check the p-value:

- **If p-val =< alpha = 0.05: reject H_0.**
- **If p-val > alpha = 0.05: do not reject H_0.**

- **What do we conclude about our *data_drinks*?**

Automate Your Analysis With a Function

```
myOut <- t.test(data = data_drinks, percentFull ~ Colour)

myOut$p.value

rejectOrWhat <- function(pValue){

  if(pValue >= 0.05){

    print("Accept Null Hypothesis: nothing happening")

  }

  else{

    print("Reject Null Hypotheis: something is going 
on...")

  }}

rejectOrWhat(myOut$p.value)
```

```
#If p-val =< alpha = 0.05: reject H0.

#If p-val > alpha = 0.05: do not reject H0.
```

R's Built-In Data
Our Built in Data

R studio (R statistics) has plenty of included data-sets for practicing t-tests work.

```
# find sets
data()
```

```
Data sets in package 'datasets':

AirPassengers              Monthly Airline Passenger Numbers 1949-1960
BJsales                    Sales Data with Leading Indicator
BJsales.lead (BJsales)     Sales Data with Leading Indicator
BOD                        Biochemical Oxygen Demand
CO2                        Carbon Dioxide Uptake in Grass Plants
ChickWeight                Weight versus age of chicks on different diets
DNase                      Elisa assay of DNase
EuStockMarkets             Daily Closing Prices of Major European Stock Indices,
                           1991-1998
Formaldehyde               Determination of Formaldehyde
HairEyeColor               Hair and Eye Color of Statistics Students
Harman23.cor               Harman Example 2.3
```