Applied Psycholinguistics
Descriptive Statistics

Hyunah Ahn

Revised: September 17, 2020

Central Tendency

The 3M's

- Mean: The average of all values in a data set
  - R code for the function: mean()
- Median: The value in the middle of a data set when it is sorted from the smallest to the largest
  - R code for the function: median()
- Mode: The most frequent value in a data set
  - R code for the function: there is no built-in function for mode
  - For the particular data set we will be looking at below, all 48 values are unique. Because there is no repeated value, finding the mode (the most frequent value) in the data set is meaningless anyway. However, by looking at the histogram, one can see where values are most clustered.

```
growth<-read.csv("C:/Users/hyuna/OneDrive/Documents/01SNU/03GraduateSeminar/crawley/data/
    growth.csv") #load data

attach(growth) # Tell R that you will be working with the loaded data.
head(growth) # Take a look at the first six lines of your data. You can briefly see what columns
    there are and what kind of values there are (e.g., characters, numbers, etc.)
```

```
  supplement  diet     gain
1  supergain wheat 17.37125
2  supergain wheat 16.81489
3  supergain wheat 18.08184
4  supergain wheat 15.78175
5    control wheat 17.70656
6    control wheat 18.22717
```

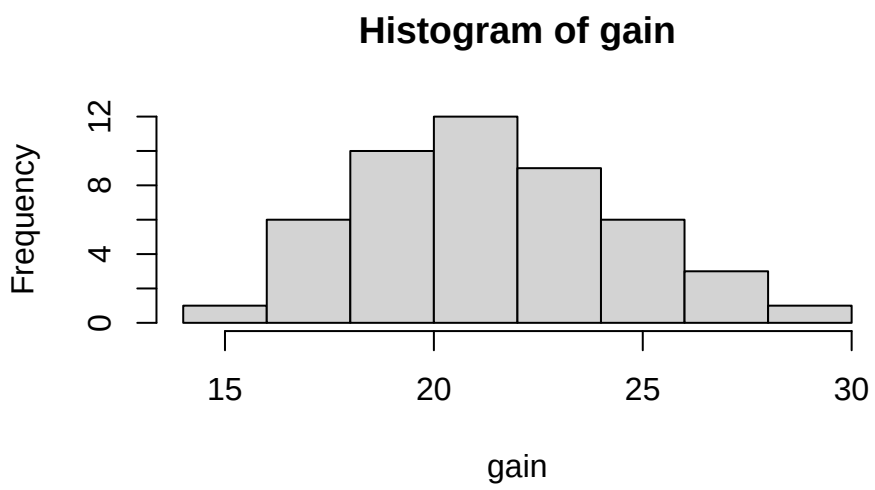
```
mean(gain) # Find the mean of the gain vector
```

```
[1] 21.39393
```

```
median(gain) # Find the central value of the gain vector
```

```
[1] 21.24808
```

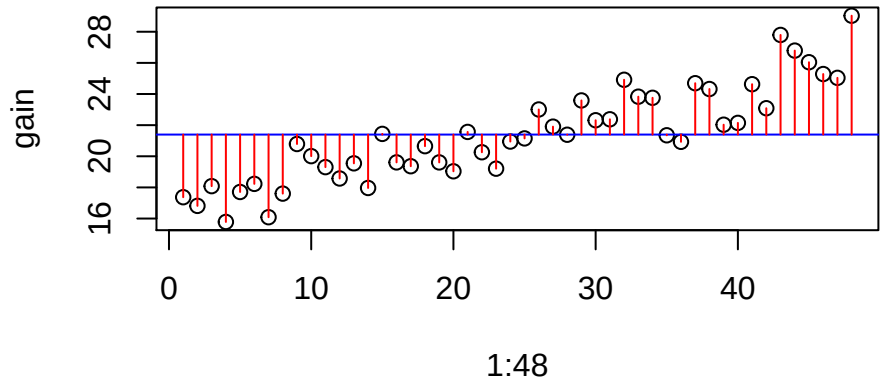
```
hist(gain) # Find how values are distributed
```



As you can see from the histogram above, most data points can be observed somewhere around 21. Also, the mean, the median, and the mode are quite close to one another. It means that the data are quite, if not perfectly, normally distributed. For many natural phenomena, it is believed that measurements are normally distributed. Height, birth weight, and IQs are a few examples that are normally distributed.

Plotting a graph to see how scattered (or clustered) data points are

```
plot(1:48, gain) # This line creates a graph that has 1 from 48 on the x-axis and all 48 values in
    the gain vector of the growth file on the y-axis. Each data point is marked as a dot.
abline(h=mean(gain), col="blue") #abline means a line that connects point A and point B on a plot.
    h means a horizontal line and the horizontal line will be drawn on the mean value of all
    values of "gain." The color will be blue.
for (i in 1:48) {lines(c(i, i), c(mean(gain), gain[i]), col = "red")} # This last command will
    create lines that connect each data point to the horizontal line created. It can show the
    variance (or the degree to which data are scattered).
```



Variance

As you have seen in the plot above, each data point is either close or far from the mean value of the entire data set. To see how scattered/clustered the data are to the center, you can calculate "variance."

Calculating variance in R

```
# 01 Let's first print all values from the gain vector in data set 'Growth.'
round(gain, 2) # This prints the vector of gain only up to two places below zero. You can just
    print the vector of gain as is, but then, you will get 5 places below zero, which clutters
    your screen too much. But if you're curious, just type 'gain' in the console and see what
    happens.
```

```
 [1] 17.37 16.81 18.08 15.78 17.71 18.23 16.09 17.60 20.79 20.01 19.30 18.57
[13] 19.55 17.96 21.44 19.61 19.36 20.65 19.61 19.04 21.56 20.26 19.20 20.95
[25] 21.15 23.01 21.90 21.38 23.59 22.32 22.38 24.91 23.83 23.76 21.35 20.93
[37] 24.69 24.32 22.03 22.14 24.63 23.09 27.79 26.79 26.04 25.28 25.04 29.03
```

```
# 02 Then, let's calculate the distance between each data point from the mean of gain

distance ← gain − mean(gain) # This creates a vector composed of all the distances between data
    points and the mean.

round(distance,2) # This prints out all the values in the distance vector rounding them to two
    places below zero.
```

```
 [1] -4.02 -4.58 -3.31 -5.61 -3.69 -3.17 -5.31 -3.79 -0.61 -1.39 -2.09 -2.82
[13] -1.85 -3.43  0.04 -1.79 -2.03 -0.74 -1.79 -2.36  0.17 -1.13 -2.19 -0.45
[25] -0.25  1.61  0.51 -0.01  2.19  0.92  0.98  3.52  2.44  2.36 -0.05 -0.46
[37]  3.30  2.93  0.63  0.75  3.24  1.69  6.40  5.39  4.65  3.89  3.64  7.64
```

```
# 03 Now, let's try and add up all the distances.

sumOfDistance ← sum(distance) # This adds up all the values in the distance vector.

sumOfDistance # This prints out the sum.
```

```
[1] 7.460699e-14
```

```
# 04 And if we divide the sum by the number of values in gain (48) to see how scattered/clustered
    data points are on average.

numval←length(gain) # This counts the number of data points in the gain vector.

numval # This prints out the count.
```

```
[1] 48
```

```
average.distance ← sumOfDistance / numval # This calculates the average distance.

average.distance # This prints out the average distance.
```

```
[1] 1.554312e-15
```

```
# 05 Oops, it tells us that the data points in the gain vector are not scattered from the central
    value at all. That is wrong! The positive and negative values canceled each other.

# 06 Now, let's go back to # 02 and make the values positive by squaring them.

sq.distance ← distance^2

round(sq.distance,2) # This prints out all the squared values.
```

```
 [1] 16.18 20.97 10.97 31.50 13.60 10.03 28.17 14.38  0.37  1.92  4.37  7.96
[13]  3.41 11.76  0.00  3.19  4.13  0.55  3.20  5.56  0.03  1.29  4.79  0.20
[25]  0.06  2.60  0.26  0.00  4.81  0.85  0.97 12.39  5.93  5.58  0.00  0.22
[37] 10.89  8.56  0.40  0.56 10.47  2.87 40.97 29.10 21.61 15.13 13.29 58.30
```

```
# 07 We can add up the squared values now.

sumOfSquares ← sum(sq.distance)
```

```r
# 08 Then, we should divide the sum of squares above not by the length of the gain vector but the
    length of the gain vector minus one.

degreeOfreedom ← length(gain)-1

variance ← sumOfSquares / degreeOfreedom

variance # This prints out the calculated variance.
```

```
[1] 9.454215
```

```r
# 09 As the last step, to check if our calculation is correct, use the built-in variance
    calculation function within R and see if we have the same values.

var(gain)
```

```
[1] 9.454215
```

```r
# It looks like we have calculated the variance of the "gain" vector in the "growth" data set
    correctly!
```

Calculating variance manually

1. In the data folder, find 'worms.csv,' save the file as an .xlsx file somewhere else.
2. Once you're done with saving a copy of the data, calculate the following for worm density:
    - Mean:
    - Median:
    - Mode:
    - Variance:
      For variance, you need to calculate (1) the distance between each data point and the mean of the data set and (2) the degree of freedom.
3. once you're done with calculating the values above manually, use the

   `var()`

   function in R to check if your calculation was correct.

The meaning of variance

So, basically, variance is calculated by first subtracting each data point from the mean (or the other way around, you only need to be consistent in the direction), squaring the values, adding them up, and dividing the sum of squares by the degree of freedom. Do you have any questions at this point?

I knew you would have the question!!!! You see, don't you also feel bothered that we have squared the distances from each data point to the mean in the name of making them positive? We can just use absolute values of the distances! Why should we square them? Now wouldn't we think that the extent to which data points are scattered is greater than it actually is? Variance is still a mathematically important concept for analysis but we will now turn to Standard Deviation, a.k.a. the square root of variance, $\sqrt{variance}$, to see more easily and clearly how scattered/clustered data are.

Standard Deviation

As it says above, the standard deviation is an indicator to show the degree to which data are scattered in general. When y is the name of the data set, n is the number of data points in the set $(x_1, x_2, x_3, ...x_n)$, we can...

Calculating Standard Deviation

Step 1

Calculate the mean (μ)!

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}$$

Step 2

Subtract each data point from the mean (or the other way around; just be consistent!)

$$\mu - x_1, \mu - x_2, \mu - x_3, ..., \mu - x_n$$

Step 3

Square all the values from the subtraction.

$$(\mu - x_1)^2, (\mu - x_2)^2, (\mu - x_3)^2, ..., (\mu - x_n)^2$$

Step 4

Sum all of them (= add them all up)!

$$\sum_{i=1}^{n} (\mu - x_i)^2$$

And now it's called "Sum of Squares"

Step 5

Get the degree of freedom by subtracting 1 from the number of data points in the set.

$$n - 1$$

Step 6

Divide the sum of squares by the degree of freedom.

$$\frac{\sum_{i=1}^{n} (\mu - x_i)^2}{n-1}$$

What? This is just the variance! The same thing we did above!

Step 7

Now get the square root of the variance.

$$\sqrt{\frac{\sum_{i=1}^{n} (\mu - x_i)^2}{n-1}}$$

We call this square root of variance 'Standard Deviation' and it is commonly marked as σ or *SD*.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (\mu - x_i)^2}{n-1}}$$

Calculating Standard Deviation Manually

4 Calculate the standard deviation of the worm density from the worms.csv file.

5 If you can't manually calculate the square root of the variance obtained from the earlier variance calculation, you can use the

```
sd()
```

function from R.

Exercises

Calculating the variance by hand

The table below presents values from Garden B in the data file 'gardens.csv.'

1. How many observations (n) are there in the data set?
2. Find the mean value of the raw data values.
3. Find the discrepancies between the mean and the raw values.
4. Square the discrepancies.
5. Add up all the discrepancies to find the sum of squares.
6. What is the formula for calculating the variance?
7. What is the variance of the data set?
8. What is the formula for calculating the standard deviation? (Hint: If you can't calculate the square root of a number by hand, you can use the `sqrt()` function in the R console window.)
9. What is the standard deviation?
10. You can use R to calculate the variance and standard deviation. Compare if your calculation is the same as the R output.

	Raw data	$x_i - \mu$	$(x_i - \mu)^2$
1	5		
2	5		
3	6		
4	7		
5	4		
6	4		
7	3		
8	5		
9	6		
10	5		
Sum			
Average			

Variance:

Standard Deviation:

The meaning of variance and standard deviation

In the previous section, we studied how we can briefly take a peak at the central tendency and variance of a data set. Variance and standard deviation, in a nutshell, are indicators of how data are scattered or clustered around (above and below) the mean of the data. But standard deviation can give us a much more systematic way to discuss and interpret data.

Rolling a dice twice to get a score

Let's imagine we're rolling a dice twice at a time and the sum of the two numbers becomes our scores. The smallest score we can get is 2 and the biggest 12. There is only one way of getting the score of 2 but two ways to get 3, three ways to get 4, and so on and so forth. Table 1 shows all the combinations of the first roll and the second roll and the sum of the two numbers from the two rolls.

Table 1: All possible scores of rolling a dice twice

	1	**2**	**3**	**4**	**5**	**6**
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

Actually rolling a dice twice

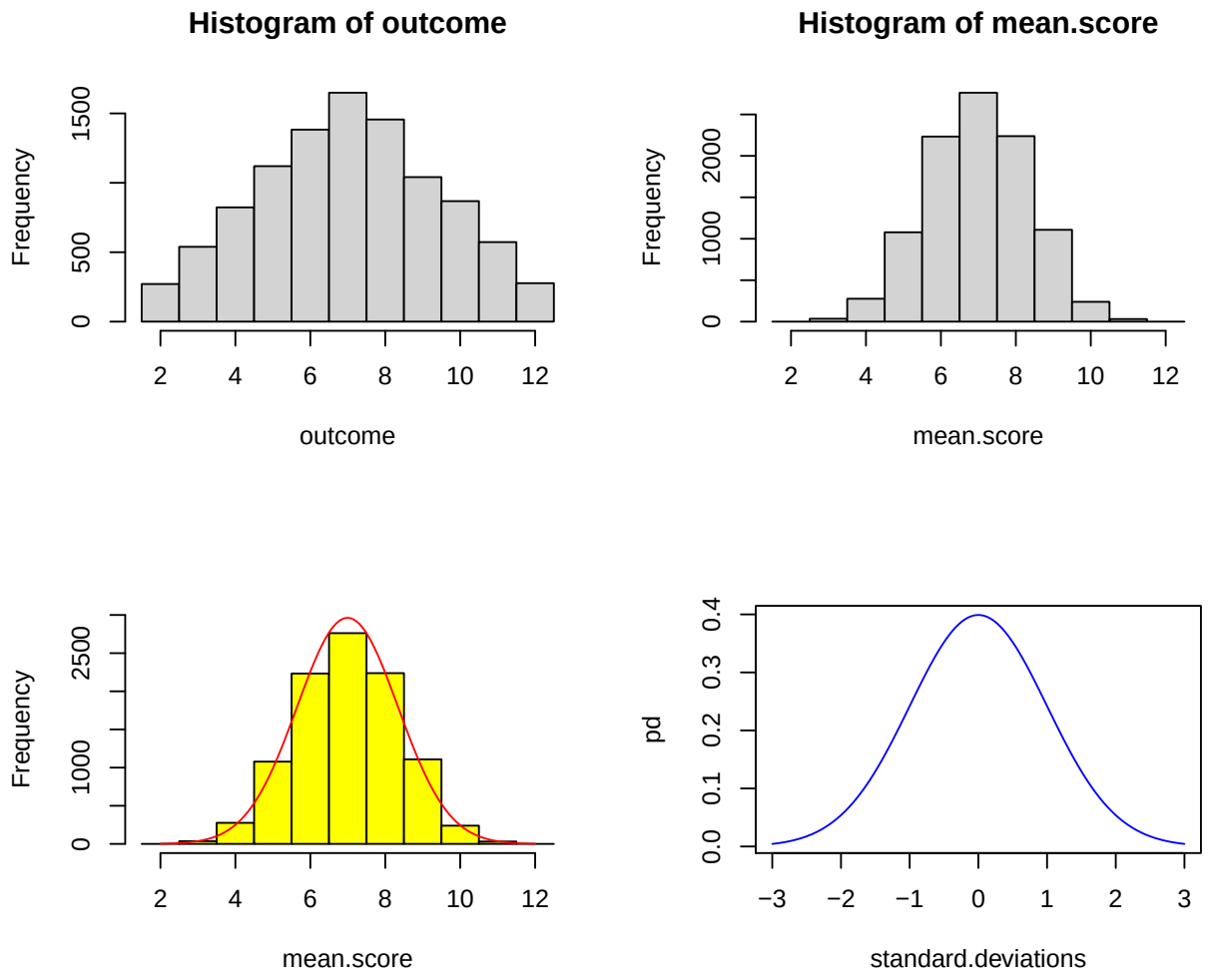
Of course, when we actually roll a dice, we will not get all the combinations exactly once. You might get the 3-3 combination several times and you might never get the 6-6 combination at all. This will be the case if you play the game (of rolling a dice twice to get a score) just a few times between a few friends. But what if you play the game 10000 times? The following codes will create four different plots.

1. The first plot will random-sample a number (from Table 1) 10000 times and will plot it with the scores on the x-axis and the frequency of the scores on the y-axis. As you can see from the 'histogram of outcome' plot, the graph does not really show a bell curve. It is more of a triangle shape.

2. For a more normally distributed pattern of data, let's create another vector named '`mean.score`.' You can first random-sample the '`game`' vector 3 times and average out the three numbers. Repeat this process 10000 times and fill the '`mean.score`' vector with the 10000 numbers. As you can see, the second histogram of '`mean.score`' shows a normally distributed pattern, the so-called bell curve.

```
par(mfrow=c(2,2)) # for 2 x 2 plot arrangement.

# The first plot

score<-2:12 # A sequence of integers from 2 to 12
ways<-c(1,2,3,4,5,6,5,4,3,2,1) # Number of ways to get each score
game<-rep(score, ways) # Repeat each score by the number of ways.
outcome<-numeric(10000) # A numeric vector of length 100 named 'outcome'
for(i in 1:10000)outcome[i]<-sample(game, 1) # Fill 'outcome' by random-sampling from 'game'
    10000 times.
hist(outcome, breaks=(1.5:12.5)) # Not a bell curve.

# The second plot

mean.score<-numeric(10000) # A numeric vector named 'mean.score' of length 10000
for(i in 1:10000)mean.score[i]<-mean(sample(game, 3)) # Sample 3 numbers from 'game' and
    average it out, repeat it 10000 times.
```

```
hist(mean.score, breaks = (1.5:12.5)) # Normally distributed

# The third plot: histogram + density plot
xv←seq(2, 12, 0.1)
yv←10000*dnorm(xv, mean(mean.score), sd(mean.score))
hist(mean.score, breaks=(1.5:12.5), ylim = c(0, 3000), col = "yellow", main ="")
lines(xv, yv, col ="red")

# The normal distribution

standard.deviations←seq(−3, 3, 0.01)
pd←dnorm(standard.deviations)
plot(standard.deviations, pd, type = "l", col = "blue")
```




```
mean(mean.score) # The mean value will change every time you create a vector of mean scores.
```

```
[1] 6.9934
```

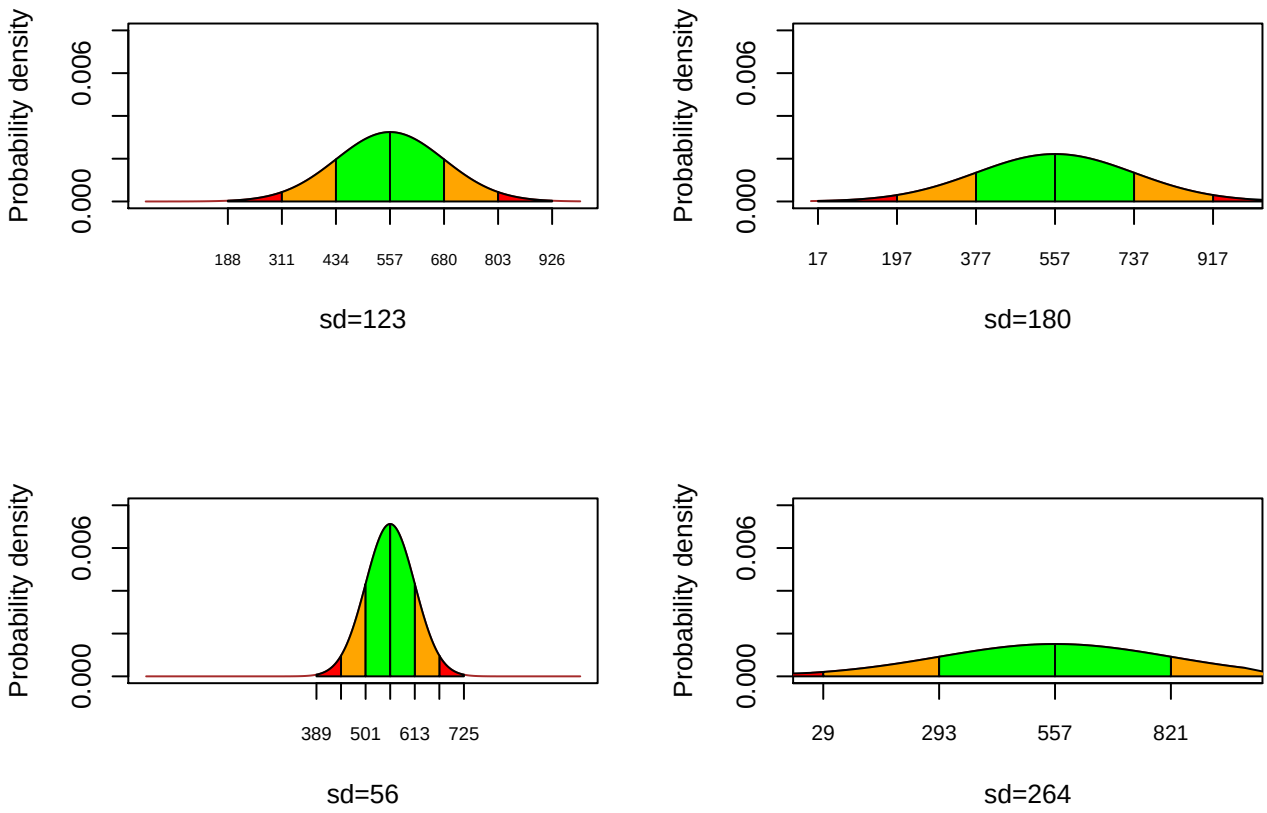
```
sd(mean.score) # The SD will also change every time you create a vector of mean scores.
```

```
[1] 1.346124
```

3. To show that the averaged out scores are normally distributed, we can add a probability density line on top, as in the third graph.
4. We can also show only the normal distribution of the values.
5. But what does it really mean that data are normally distributed?

Normal Distribution

- When data are normally distributed, 68% of the data fall into the range of one standard deviation (σ) below and above the mean, 95% of the data fall into the range of two σs below and above the mean, and 99.7% of the data fall into $\pm 3\sigma$s.
- To see what it means more easily, let's take a look at the graphs below.
- The four graphs below have the exact same mean of 557 but their standard deviations differ from 123 to 180 to 56 to 264.
- So what do these four graphs teach us? Even four different schools say that their students' English test scores are the same, the students' ability might greatly differ from school to school.



Z distribution

Let's say you're working for the HR department of a company and needs to screen applicants based on their English test scores. Applicants submit their best test scores but unfortunately, raw scores might not always give you a very clear idea of how high or low the applicant's English proficiency is. More often than not, people use percentiles to estimate a test-taker's ability. Standard deviations can come in handy when we try to estimate a test-taker's percentile if we can assume that data are normally distributed. Likewise, in normal distribution, knowing the probability density of a data point will always allow us to estimate a test-taker's score.

In the first graph above, the standard deviation is 123 points. When we normalize data, we set the mean value (557 in this case) to be zero and one standard deviation becomes z-score of 1. Therefore, a test taker who scored

434 points in the first group is described to be -1 *SD* away from the mean.

Let's say that these four graphs show the distributions of different tests. In that case, saying "I scored 700 on a test" will not be very informative. Scoring 700 in the first plot means the person is above 1 *SD* but the same score is lower than 1 *SD* in the second and fourth graphs, and close to 3 *SD* in the third graph. We can actually calculate the z-score of 700 in the four different cases.

The formula for calculating the z-score is..

$z = \frac{x_i - \mu}{\sigma}$

Then, the z-score for 700 in the first graph is...

$z = \frac{700-557}{123} = 1.16$

And in the second graph, it will be...

$z = \frac{700-557}{180} = 0.79$

And in the third graph, it will be...

$z = \frac{700-557}{56} = 2.55$

Finally, in the last graph, it will be...

$z = \frac{700-557}{264} = 0.54$

1. In which test do you think the score of 700 will result in the highest z-score?
2. Can we estimate the person's ranking by z-score?
3. If we know a person's percentile rank, can we estimate the person's score?

Using a Normal Distribution Table

In the first graph, the score of 700 was converted to the z-score of 1.16. When you know the z-score, you can use the z distribution (normal distribution) table to find out the person's position among all test-takers. Open the z-distribution table and find the row of 1.1 in the first column and the column of .06 on the first row. The value in the intersection of the row and the column is the cumulative percentile of the z-score of 1.16. The table says the percentile is 0.8770. It means the person's score is higher than that of 87.7% of all the test takers in the exam.

It's awesome that you can infer the peron's ranking using the z-score. However, carrying the z-score table will not be that awesome. Thanks to our awesome R, we no longer need to carry distribution tables (Yes, tableS! There are more distribution tables coming along in the following weeks!).

We can use the following functions to quickly find the percentile of a z-score.

```
pnorm(z-score)
```

Also, if you'd like to quickly find out a person's score by percentile, the following code will be of great help.

```
qnorm(percentile, mean, sd)
```

If the mean score is 557 and the standard deviation is 56, what will a person on the 98% on top score? Try the following code in R.

```
qnorm(0.98, 557, 56)
```

What score did you get?

How normal is normal distribution?

We have learned that normal distribution is shown as the bell-curve of a density plot. The so-called bell-curvedness, however, is not always perfectly symmetrical and we'd like to quantify the extent to which the data deviate from the normal bell curve. To do that, we use the two measurements called skewness and kurtosis.

Skewness

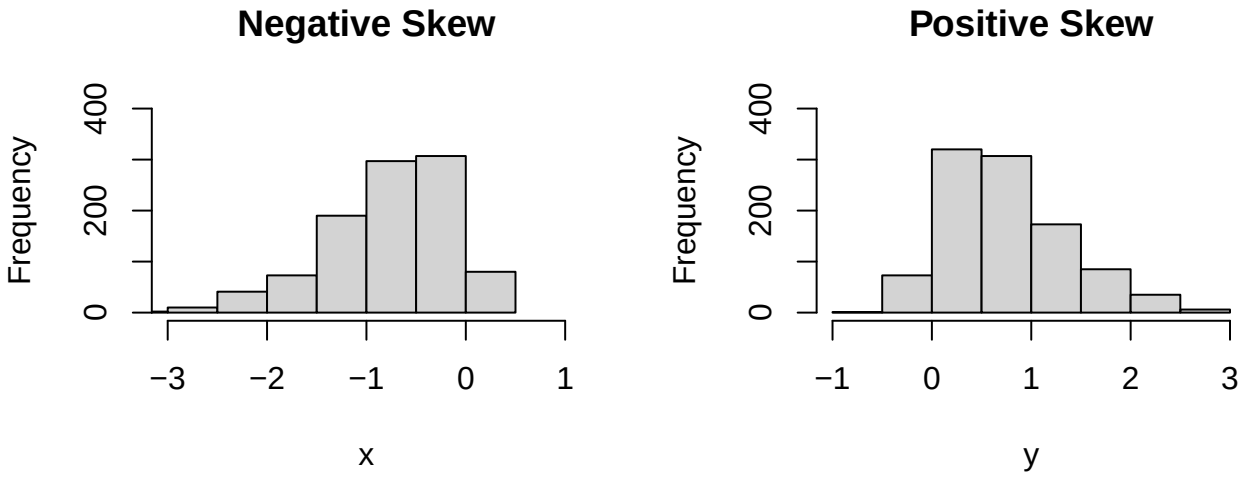
Skewness, the indicator of how data are centered left or right to the median, can be calculated by dividing the third moment around the mean (the sum of cubes ($\sum_{i=1}^{n}(x_n - \mu)$) divided by the sample size) by the cubed $\sigma = \sqrt{s^2}$

$$M_3 = \frac{\sum_{i=1}^{n}(x_n - \mu)^3}{n}$$

$$\sigma^3$$

$$skew = \gamma_1 = \frac{M_3}{\sigma^3} = \frac{\sum_{i=1}^{n}(x_n - \mu)^3/n}{\sigma^3}$$

You can see negatively skewed data on the lefthand side and positively skewed data on the righthand side. When we say data are negatively skewed, it means the data have a longer tail to the left of the median. When data are positively skewed, it means the longer tail is to the right of the median.



The rule of thumb in interpreting the skewness value is to check if its absolute value is greater than 1.

$|\gamma_1| \leq 0.5$; data are normally distributed.

$0.5 \leq |\gamma_1| \leq 1$; data are slightly skewed.

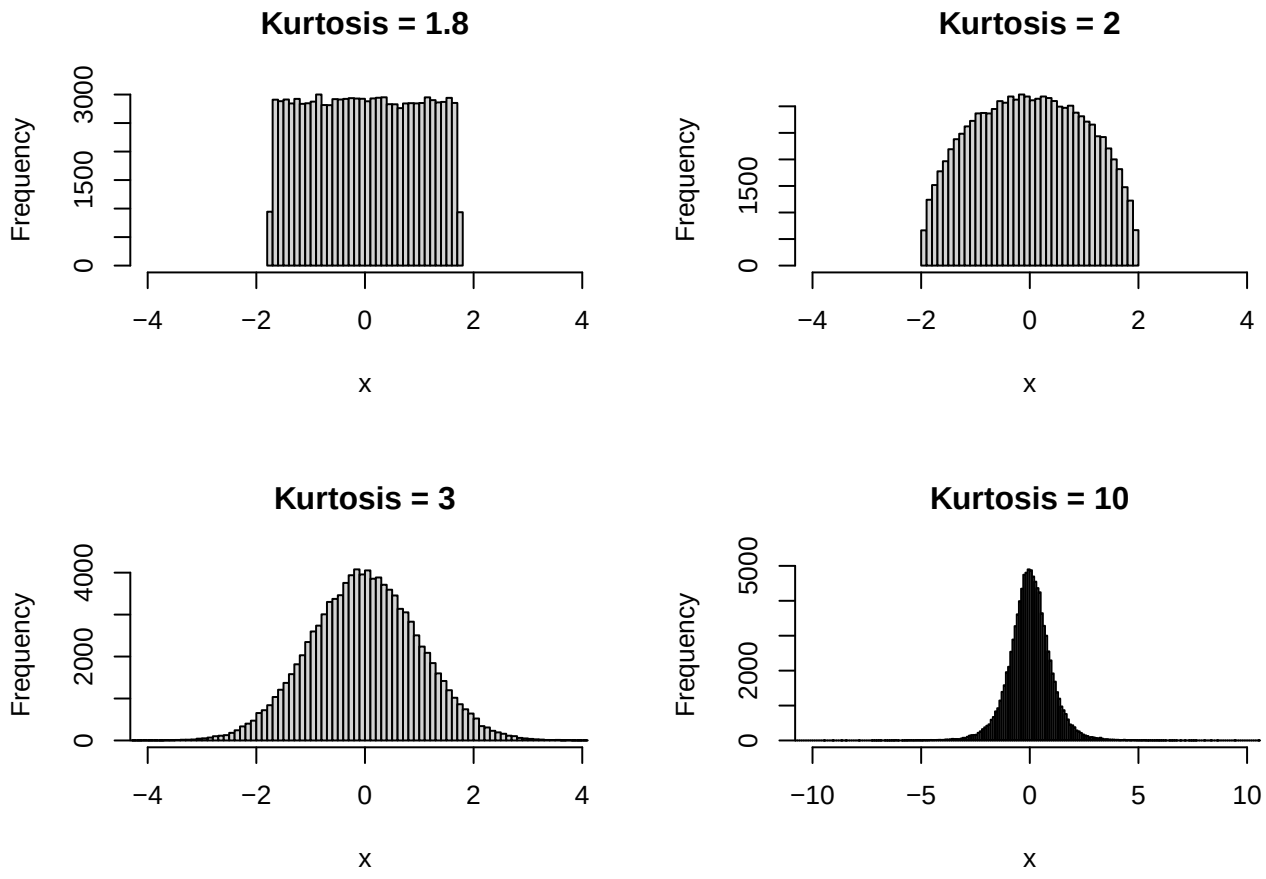
$|\gamma_1| \geq 1$; data are highly skewed.

Are you worried that you have to memorize just another formula to calculate skewness? Well, no worries. There are many functions to check skewness (and kurtosis) in several R packages. For now, I'll introduce just two.

```
library(fBasics)
basicStats(data)

library(moments)
skewness(data)
kurtosis(data)
```

Kurtosis

Kurtosis, the indicator of the extent to which the top of the curve is pointy or flat. When the top of the distribution curve is flat, the distribution is platykurtic. If the top of the distribution curve is pointy, the distribution is leptokurtic.



Continuing from Skewness, kurtosis also involves the discussion of 'moments.' The fourth moment around the mean (M_4) is calcuated with the difference between the mean and each data point raised to the fourth power and with the sums of fourth power ($\sum(x-\mu)^4$) divided by the sample size. And if you divide the fourth moment by the square of the variance ($s^4 = (s^2)^2$).

$$M_4 = \frac{\sum_{i=1}^{n}(x_n-\mu)^4}{n}$$

$$s^4 = (\sqrt{s^2})^4$$

$$kurtosis = \gamma_2 = \frac{M_4}{s^4} = \frac{\sum_{i=1}^{n}(x_n-\mu)^4/n}{(s^2)^2}$$

As you see from the plots above, normal distribution has the kurtosis value of 3. But depending on which package you use, the package might give you a normalized value. You can visually check the normality and add 3 if it has a very flat top and the kurtosis value is smaller than 3. The greater the kurtosis value, the curve has a pointy top and it reaches the normal bell curve around 3.