

## Statistics in GeoGebra ([tecooper@northgeorgia.edu](mailto:tecooper@northgeorgia.edu))

Source: The GeoGebra Online Manual (<http://wiki.geogebra.org/en/Commands>)

### One Variable Descriptive

#### *Numerical*

##### **Mean[ <List of Numbers> ]**

Calculates the arithmetic mean of the list elements.

##### **MeanX[List of Points]**

Calculates the mean of the  $x$ -coordinates of the points in the list.

##### **MeanY[List of Points]**

Calculates the mean of the  $y$ -coordinates of the points in the list.

##### **Median[ <List of Numbers> ]**

Determines the median of the list elements.

##### **Mode[List of Numbers]**

Determines the mode(s) of the list elements.

##### **Max[List of Numbers]**

Determines the maximum of the list elements.

##### **Min[List of Numbers]**

Determines the minimum of the list elements.

##### **Percentile[<List of Numbers>, <Percent>]**

Let  $P$  equal the given *Percent*.

Returns the value that cuts off the first  $P$  percent of the number list when the list is sorted in ascending order. *Percent* must be a number in the interval  $0 < P \leq 1$ .

##### **Q1[List of Numbers]**

Determines the lower quartile of the list elements.

**Note:** The commands Quartile and Percentile use different rules and do not always return matching results. For example,

$$Q1[\{1,2,3,4\}] = 1.5$$

$$\text{Percentile}[\{1,2,3,4\}, 0.25] = 1.25$$

##### **Q3[List of Numbers]**

Determines the upper quartile of the list elements.

##### **SampleSD[ <List of Numbers> ]**

Returns sample standard deviation (uses  $n-1$ ) of given list of numbers.

There is also a SampleSDX and a SampleSDY.

##### **SD[ <List of Numbers> ]**

Calculates the standard deviation (uses  $n$ ) of the numbers in the list.

There is also an SDX and an SDY.

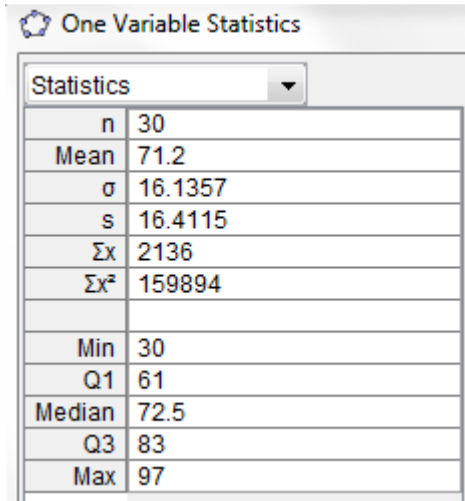
##### **SampleVariance[ <List of Numbers> ]**

Returns sample variance (uses  $n-1$ ) of given list of numbers.

##### **Variance[ <List of Numbers> ]**

Returns variance (uses  $n$ ) of given list of numbers.

From Icon available in Spreadsheet view:



Statistics	
n	30
Mean	71.2
$\sigma$	16.1357
s	16.4115
$\Sigma x$	2136
$\Sigma x^2$	159894
Min	30
Q1	61
Median	72.5
Q3	83
Max	97

## Charts and Graphs

### FrequencyTable[<List of Class Boundaries C>, <List of Raw Data L> ]

Returns a table (as text) whose first column contains intervals (classes) and second column contains the count of numbers in  $L$  which belong to the interval in the first column. All intervals except the highest interval are of the form  $[a, b)$ . The highest interval has the form  $[a, b]$ .

### DotPlot[ <List of Raw Data> ]

Returns a dot plot for the given list of numbers, as well as the list of the dot plot points. If a number  $n$  appears in the list of raw data  $k$  times, the returned list contains points  $(n, 1), (n, 2), \dots, (n, k)$ .

### StemPlot[ <List> ]

Returns a stem plot of the given list of numbers. Outliers are removed from the plot and listed separately.

An outlier is defined as a value outside the interval  $[ Q1 - 1.5 (Q3 - Q1) , Q3 + 1.5 (Q3 - Q1) ]$ .

### StemPlot[ <List>, <Adjustment -1|0|1> ]

Returns a stem plot of the given list of numbers.

If *Adjustment* = -1 the default stem unit is divided by 10

If *Adjustment* = 0 nothing is changed

If *Adjustment* = 1 the default stem unit is multiplied by 10

### BoxPlot[yOffset, yScale, List of Raw Data]

Creates a box plot using the given raw data and whose vertical position in the coordinate system is controlled by variable *yOffset* and whose height is influenced by factor *yScale*.

### BoxPlot[yOffset, yScale, Start Value a, Q1, Median, Q3, End Value b]

Creates a box plot for the given statistical data in interval  $[a, b]$ .

### Histogram[<List of Class Boundaries>, <List of Heights>]

Creates a histogram with bars of the given heights. The class boundaries determine the width and position of each bar of the histogram.

**Histogram[<List of Class Boundaries>, <List of Raw Data>, <Boolean Use Density>, <Density Scale Factor>(optional)]**

Creates a histogram using the raw data. The class boundaries determine the width and position of each bar of the histogram and are used to determine how many data elements lie in each class (includes left end points). Bar height is determined as follows

- If *Use Density = true*, height = (Density Scale Factor) \* (class frequency) / (class width)
- If *Use Density = false*, height = class frequency

By default, Use Density = true and Density Scale Factor = 1. This creates a histogram with total area = n, the number of data values.

For a normalized histogram, use the density factor 1/n.

**Histogram[ <Boolean Cumulative>, <List of Class Boundaries>, <List of Raw Data>, <Boolean Use Density> , <Density Scale Factor> (optional) ]**

If Cumulative is true this creates a histogram where each bar height equals the frequency of the class plus the sum of all previous frequencies.

**HistogramRight[ <List of Class Boundaries B>, <List of Heights H> ]**

Same as Histogram[B, H]

**HistogramRight[ <List of Class Boundaries B>, <List of Raw Data D>, <Boolean Use Density> , <Density Scale Factor f> (optional) ]**

Same as Histogram[B, D, Use Density, f], except that if a datum is equal to the right border of a class, it is counted in this class and not in the next one.

**HistogramRight[ <Boolean Cumulative>, <List of Class Boundaries B>, <List of Raw Data D>, <Boolean Use Density> , <Density Scale Factor f> (optional) ]**

Same as Histogram[Cumulative, B, D, Use Density, f], except that if a datum is equal to the right border of a class, it is counted in this class and not in the next one.

## Regression

**CorrelationCoefficient[List of x-Coordinates, List of y-Coordinates]**

Calculates the product moment correlation coefficient using the given *x*- and *y*-coordinates.

**CorrelationCoefficient[List of Points]**

Calculates the product moment correlation coefficient using the coordinates of the given points.

**FitLine[List of Points]**

Calculates the *y* on *x* regression line of the points.

**FitLineX[List of Points]**

Calculates the *x* on *y* regression line of the points.

**Fit[ <List of Points>,<List of Functions> ]**

Calculates a linear combination of functions to the points in the list.

**Example:** With  $L=\{A,B,C,\dots\}$ ,  $f(x)=1$ ,  $g(x)=x$ ,  $h(x)=e^x$ ,  $F=\{f,g,h\}$  the command Fit[L,F] calculates a function of the form  $a + b x + c e^x$  to the points in the list.

**FitExp[ <List of Points> ]**

Calculates the exponential regression curve.

**Example:** FitExp[{(0, 1), (2, 4)}] gives  $e^{0.69x}$ .

**Note:** Euler's number *e* can be obtained by pressing ALT + e.

**FitLog[ <List of Points> ]**

Calculates the logarithmic regression curve.

**Example:** FitLog[{(e,1), (e^2, 4)}] gives  $3 \ln(x) - 2$ .

**FitPoly**[ <List of Points>, <Degree n of Polynomial> ]

Calculates the regression polynomial of degree  $n$ .

**Example:** FitPoly[{{(-1, -1), (0, 1), (1, 1), (2, 5)}, 3}] gives  $x^3 - x^2 + 1$ .

**FitPow**[ <List of Points> ]

Calculates the regression curve in the form  $a x^b$ .

**Example:** FitPow[{{(1, 1), (3, 2), (7, 4)}]} creates the regression curve  $0.97 x^{0.71}$ .

**FitSin**[ <List of Points> ]

Calculates the regression curve in the form  $a + b \sin(cx + d)$ .

**Example:** FitSin[{{(1, 1), (2, 2), (3, 1), (4, 0), (5, 1), (6, 2)}]} gives  $1 + \sin(1.5708x - 1.5708)$ .

**FitLogistic**[List of Points]

Calculates the regression curve in the form  $a/(1+b e^{-kx})$ .

**Note:** The first and last data points should be fairly close to the curve. The list should have at least 3 points, preferably more.

## One Variable Tests

From Icon: Z Test of a Mean, T Test of a Mean, Z Estimate of a Mean, T Estimate of a Mean

**TTest**[<List of Sample Data>,<Hypothesized Mean>,<Tail>]

Performs a one sample T test of a population mean using the given list of sample data. *Hypothesized Mean* is the population mean assumed in the null hypothesis. *Tail* has possible values "<", ">", "≠".

These specify the alternative hypothesis as follows.

"<" = population mean < *Hypothesized Mean*

">" = population mean > *Hypothesized Mean*

"≠" = population mean ≠ *Hypothesized Mean*

Results are returned in list form as {Probability value, T test statistic}.

**TTest**[<Sample Mean>,<Sample Standard Deviation>,<Sample Size>,<Hypothesized Mean>,<Tail>]

Performs a one sample T test of a population mean using the given sample statistics. *Hypothesized Mean* and *Tail* are defined as above. Results are returned in list form as {Probability value, T test statistic}.

**Normal**[ <Mean  $\mu$ >, <Standard Deviation  $\sigma$ >, <Variable Value  $v$ > ]

Calculates the function  $\Phi((x - \mu) / \sigma)$  at  $v$  where  $\Phi$  is the cumulative distribution function for  $N(0,1)$ .

**Note:** Returns the probability for a given x-coordinate's value (or area under the normal distribution curve to the left of the given x-coordinate).

**TDistribution**[ <Degrees of Freedom  $d$ >, <Variable Value  $v$ > ]

Calculates the value of cumulative distribution function of t-distribution at  $v$ , i.e. the probability  $P(X \leq v)$  where  $X$  is a random variable with t-distribution with  $d$  degrees of freedom.

**Note:** Returns the probability for a given x-coordinate's value (or area under the t-distribution curve to the left of the given x-coordinate).

**InverseNormal**[Mean  $\mu$ , Standard Deviation  $\sigma$ , Probability  $P$ ]

Calculates the function  $\Phi^{-1}(P) * \sigma + \mu$  where  $\Phi^{-1}$  is the inverse of the cumulative distribution function  $\Phi$  for  $N(0,1)$ .

**Note:** Returns the x-coordinate with the given probability to the left under the normal distribution curve.

**InverseTDistribution[ <Degrees of Freedom d>, <Probability p> ]**

Computes the inverse of cumulative distribution function of t-distribution at  $p$ , where the t-distribution has  $d$  degrees of freedom. In other words, finds  $r$  such that  $P(X \leq r) = p$ , where  $X$  is random variable with t-distribution. Probability  $p$  must be from  $[0,1]$ .

**Two Variable Test**

**ANOVA[<List>, <List>, ...]**

Performs a one-way [ANOVA](#) test on the given lists of numbers.

Results are returned in list form as {P value, F test statistic}.

**TTestPaired[<List of Sample Data 1>,<List of Sample Data 2>,<Tail>]**

Performs a paired T test using the given lists of paired sample [data](#). *Tail* has possible values "<", ">" , "&ne;" that determine the following alternative hypotheses:

"<" =  $\mu < 0$

">" =  $\mu > 0$

"&ne;" =  $\mu \neq 0$

(  $\mu$  is the mean paired difference of the population)

Results are returned in list form as {Probability value, T test statistic}.

**TTest2[<List of Sample Data 1>,<List of Sample Data 2>,<Tail>,<Boolean Pooled>]**

Performs a T test of the difference between two population means using the given lists of sample [data](#). *Tail* has possible values "<", ">" , "&ne;" that determine the following alternative hypotheses:

"<" = difference in population means  $< 0$

">" = difference in population means  $> 0$

"&ne;" = difference in population means  $\neq 0$

If *Pooled* = true, then population variances are assumed equal and sample standard deviations are combined in calculation.

If *Pooled* = false, then population variances are not assumed equal and sample standard deviations are not combined.

Results are returned in list form as {Probability value, T test statistic}.

**TTest2[<Sample Mean 1 >,<Sample Standard Deviation 1>, <Sample Size 1>, <Sample Mean 2 >,<Sample Standard Deviation 2 >, <Sample Size 2>, <Tail>,<Boolean Pooled>]**

Performs a T test of the difference between two population means using the given sample statistics. *Tail* and *Pooled* are defined as above.