

Namespace LinqStatistics

Classes

[Bin](#)

A discrete count of items which fall into a given range

[EnumerableStats](#)

Static class with statistical extension methods for [IEnumerable<T>](#) Methods return NaN instead of throwing exceptions in cases that would result in divide by zero

[ItemCount<T>](#)

Represents the count of an item in a collection

Structs

[LeastSquares](#)

Represents the result of a LeastSquares calculation of the form $y = mX + b$

[Range<T>](#)

An ordered pair of values, representing a segment.

Enums

[BinningMode](#)

Controls how the range of the bins are determined

Class Bin

Namespace: [LinqStatistics](#)

Assembly: LinqStatistics.dll

A discrete count of items which fall into a given range

```
[ComVisible(false)]
public sealed class Bin : IItemCount<double>, IEquatable<ItemCount<double>>, IEquatable<Bin>
```

Inheritance

[object](#) ↗ ← [ItemCount<double](#) ↗ ← Bin

Implements

[IEquatable](#) ↗ <[ItemCount<double](#) ↗ >>, [IEquatable](#) ↗ <[Bin](#)>

Inherited Members

[ItemCount<double>.RepresentativeValue](#), [ItemCount<double>.Count](#),
[ItemCount<double>.Equals\(ItemCount<double>\)](#), [object.Equals\(object, object\)](#) ↗, [object.GetType\(\)](#) ↗,
[object.ReferenceEquals\(object, object\)](#) ↗

Constructors

Bin(double, double, double, int, bool)

ctor

```
public Bin(double v, double min, double max, int count, bool maxInclusive = false)
```

Parameters

v [double](#) ↗

Representative value for the Bin

min [double](#) ↗

The minimum value of the Range

`max` [double](#)

The maximum value of the range

`count` [int](#)

The number of items in the Bin

`maxInclusive` [bool](#)

Should Max be included in the Range or excluded - default is excluded

Properties

MaxInclusive

Determines whether Max should be included or excluded in the range

```
public bool MaxInclusive { get; }
```

Property Value

[bool](#)

A discrete count of items which fall into a given range

Range

The range

```
public Range<double> Range { get; }
```

Property Value

[Range<double>](#)

A discrete count of items which fall into a given range

Methods

Contains(double)

Determines if a value is contained with the segment

```
public bool Contains(double item)
```

Parameters

item [double](#)

The item to check

Returns

[bool](#)

True if item is contained in the range - taking into account MaxInclusive

Equals(Bin)

[Equals\(T\)](#)

```
public bool Equals(Bin other)
```

Parameters

other [Bin](#)

Returns

[bool](#)

Equals(object)

[Equals\(object\)](#)

```
public override bool Equals(object obj)
```

Parameters

obj [object](#)

Returns

[bool](#)

GetHashCode()

[GetHashCode\(\)](#).

```
public override int GetHashCode()
```

Returns

[int](#)

ToString()

[ToString\(\)](#).

```
public override string ToString()
```

Returns

[string](#)

Operators

operator ==(Bin, Bin)

```
public static bool operator ==(Bin lhs, Bin rhs)
```

Parameters

[lhs Bin](#)

[rhs Bin](#)

Returns

[bool](#)

operator !=(Bin, Bin)

```
public static bool operator !=(Bin lhs, Bin rhs)
```

Parameters

[lhs Bin](#)

[rhs Bin](#)

Returns

[bool](#)

Enum BinningMode

Namespace: [LinqStatistics](#)

Assembly: LinqStatistics.dll

Controls how the range of the bins are determined

```
public enum BinningMode
```

Fields

ExpandRange = 2

The total range will be expanded such that the min is less than the sequence min and max is greater than the sequence max. The number of bins will be equal to $(\text{max} - \text{min}) / (\text{binCount} - 1)$ in essence adding an extra bin and shrinking the bin size.

MaxValueInclusive = 1

The minimum will be the sequence min and the maximum equal to sequence max. The last bin will be inclusive instead of exclusive.

Unbounded = 0

The minimum will be equal to the sequence min and the maximum equal to infinity.

Class EnumerableStats

Namespace: [LinqStatistics](#)

Assembly: LinqStatistics.dll

Static class with statistical extension methods for [IEnumerable<T>](#) Methods return NaN instead of throwing exceptions in cases that would result in divide by zero

```
public static class EnumerableStats
```

Inheritance

[object](#) ← EnumerableStats

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

AllValues<T>(IEnumerable<T?>)

Returns all non-null items in a sequence

```
public static IEnumerable<T> AllValues<T>(this IEnumerable<T?> source) where T : struct
```

Parameters

source [IEnumerable](#)<T?>

The Sequence

Returns

[IEnumerable](#)<T>

All non-null elements in the sequence

Type Parameters

T

The type of the sequence

AllValues<T>(IEnumerable<Tuple<T?, T?>>)

Returns all elements in a sequence of Tuples where the Tuple's are not null

```
public static IEnumerable<Tuple<T, T>> AllValues<T>(this IEnumerable<Tuple<T?, T?>> source)
where T : struct
```

Parameters

source [IEnumerable<Tuple<T?, T?>>](#)

The sequence

Returns

[IEnumerable<Tuple<T, T>>](#)

All Tuples in the sequence with non-null items

Type Parameters

T

The type of the Tuple's Items

AverageNaN(IEnumerable<double>)

Computes the sample Average of a sequence of double values.

```
public static double AverageNaN(this IEnumerable<double> source)
```

Parameters

source [IEnumerable<double>](#)

A sequence of double values to calculate the Average of.

Returns

double

The Average of the sequence of values.

AverageNaN(IEnumerable<int>)

Computes the sample Average of a sequence of int values.

```
public static double AverageNaN(this IEnumerable<int> source)
```

Parameters

source IEnumerable<int>

A sequence of int values to calculate the Average of.

Returns

double

The Average of the sequence of values.

AverageNaN(IEnumerable<long>)

Computes the sample Average of a sequence of long values.

```
public static double AverageNaN(this IEnumerable<long> source)
```

Parameters

source IEnumerable<long>

A sequence of long values to calculate the Average of.

Returns

double

The Average of the sequence of values.

AverageNaN(IEnumerable<double?>)

Computes the sample Average of a sequence of nullable double values.

```
public static double? AverageNaN(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable<double?>](#)

A sequence of nullable double values to calculate the Average of.

Returns

[double?](#)

The Average of the sequence of values, or null if the source sequence is empty or contains only values that are null.

AverageNaN(IEnumerable<int?>)

Computes the sample Average of a sequence of nullable int values.

```
public static double? AverageNaN(this IEnumerable<int?> source)
```

Parameters

source [IEnumerable<int?>](#)

A sequence of nullable int values to calculate the Average of.

Returns

[double?](#)

The Average of the sequence of values, or null if the source sequence is empty or contains only values that are null.

AverageNaN(IEnumerable<long?>)

Computes the sample Average of a sequence of nullable long values.

```
public static double? AverageNaN(this IEnumerable<long?> source)
```

Parameters

source [IEnumerable<long?>](#)

A sequence of nullable long values to calculate the Average of.

Returns

[double?](#)

The Average of the sequence of values, or null if the source sequence is empty or contains only values that are null.

AverageNaN(IEnumerable<float?>)

Computes the sample Average of a sequence of nullable float values.

```
public static float? AverageNaN(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

A sequence of nullable float values to calculate the Average of.

Returns

[float?](#)

The Average of the sequence of values, or null if the source sequence is empty or contains only values that are null.

AverageNaN(IEnumerable<float>)

Computes the sample Average of a sequence of float values.

```
public static float AverageNaN(this IEnumerable<float> source)
```

Parameters

source [IEnumerable](#)<[float](#)>

A sequence of float values to calculate the Average of.

Returns

[float](#)

The Average of the sequence of values.

AverageNaN<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the sample Average of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double AverageNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<[TSource](#)>

The sequence of elements.

selector [Func](#)<[TSource](#), [double](#)>

A transform function to apply to each element.

Returns

[double](#)

The Average of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

AverageNaN<TSource>(IEnumarable<TSource>, Func<TSource, int>)

Computes the sample Average of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double AverageNaN<TSource>(this IEnumarable<TSource> source, Func<TSource, int> selector)
```

Parameters

source [IEnumarable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [int](#)>

A transform function to apply to each element.

Returns

[double](#)

The Average of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

AverageNaN<TSource>(IEnumarable<TSource>, Func<TSource, long>)

Computes the sample Average of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double AverageNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, long>

A transform function to apply to each element.

Returns

[double](#)

The Average of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

AverageNaN<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the sample Average of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? AverageNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector `Func<TSource, double?>`

A transform function to apply to each element.

Returns

`double?`

The Average of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

AverageNaN<TSource>(IEnumerable<TSource>, Func<TSource, int?>)

Computes the sample Average of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? AverageNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, int?> selector)
```

Parameters

source `IEnumerable<TSource>`

The sequence of elements.

selector `Func<TSource, int?>`

A transform function to apply to each element.

Returns

`double?`

The Average of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

AverageNaN<TSource>(IEnumarable<TSource>, Func<TSource, long?>)

Computes the sample Average of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? AverageNaN<TSource>(this IEnumarable<TSource> source, Func<TSource, long?> selector)
```

Parameters

source [IEnumarable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, long?>

A transform function to apply to each element.

Returns

[double](#)?

The Average of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

AverageNaN<TSource>(IEnumarable<TSource>, Func<TSource, float?>)

Computes the sample Average of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? AverageNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, float?>

A transform function to apply to each element.

Returns

[float](#)?

The Average of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

AverageNaN<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the sample Average of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float AverageNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [float](#)>

A transform function to apply to each element.

Returns

[float](#)

The Average of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

BinCountRice<T>(IEnumerable<T>)

http://en.wikipedia.org/wiki/Histogram#Number_of_bins_and_width

```
public static int BinCountRice<T>(this IEnumerable<T> source)
```

Parameters

source [IEnumerable](#)<T>

The sequence of elements.

Returns

[int](#)

The number of bins to use to create a histogram.

Type Parameters

T

The type of elements in the sequence

BinCountSquareRoot<T>(IEnumerable<T>)

http://en.wikipedia.org/wiki/Histogram#Number_of_bins_and_width

```
public static int BinCountSquareRoot<T>(this IEnumerable<T> source)
```

Parameters

source [IEnumerable](#)<T>

The sequence of elements.

Returns

[int](#)

The number of bins to use to create a histogram.

Type Parameters

T

The type of elements in the sequence

BinCountSturges<T>(IEnumerable<T>)

http://en.wikipedia.org/wiki/Histogram#Number_of_bins_and_width

```
public static int BinCountSturges<T>(this IEnumerable<T> source)
```

Parameters

source [IEnumerable](#)<T>

The sequence of elements.

Returns

[int](#)

The number of bins to use to create a histogram.

Type Parameters

T

The type of elements in the sequence

CountEach<T>(IEnumerable<T>)

Counts each unique element in a sequence

```
public static IEnumerable<ItemCount<T>> CountEach<T>(this IEnumerable<T> source)
```

Parameters

source [IEnumerable](#)<T>

The sequence to count

Returns

[IEnumerable](#)<[ItemCount](#)<T>>

The count of each unique element

Type Parameters

T

The type of the sequence

CountEach<T>(IEnumerable<T>, IEqualityComparer<T>)

Counts each unique element in a sequence

```
public static IEnumerable<ItemCount<T>> CountEach<T>(this IEnumerable<T> source,  
IEqualityComparer<T> comparer)
```

Parameters

source [IEnumerable<T>](#)

The sequence to count

comparer [IEqualityComparer<T>](#)

Comparer used to determine equality between elements

Returns

[IEnumerable<ItemCount<T>>](#)

The count of each unique element

Type Parameters

T

The type of the sequence

CountEach<TSource, TResult>(IEnumerable<TSource>, Func<TSource, TResult>)

Counts each unique element in a sequence

```
public static IEnumerable<ItemCount<TResult>> CountEach<TSource, TResult>(this  
    IEnumerable<TSource> source, Func<TSource, TResult> selector)
```

Parameters

source [IEnumerable<TSource>](#)

A sequence of values to calculate the unique count of.

selector [Func<TSource, TResult>](#)

A transform function to apply to each element.

Returns

[IEnumerable<ItemCount<TResult>>](#)

The count of each unique element

Type Parameters

TSource

The type of the elements of source.

TResult

The type of the element selected

CountEach<TSource, TResult>(IEnumerable<TSource>, Func<TSource, TResult>, IEqualityComparer<TResult>)

Counts each unique element in a sequence

```
public static IEnumerable<ItemCount<TResult>> CountEach<TSource, TResult>
(this IEnumerable<TSource> source, Func<TSource, TResult> selector,
IEqualityComparer<TResult> comparer)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the unique count of.

selector [Func](#)<TSource, TResult>

A transform function to apply to each element.

comparer [IEqualityComparer](#)<TResult>

Comparer used to determine equality between elements

Returns

[IEnumerable](#)<[ItemCount](#)<TResult>>

The count of each unique element

Type Parameters

TSource

The type of the elements of source.

TResult

The type of the element selected

Covariance(IEnumerable<decimal>, IEnumerable<decimal>)

Computes the Covariance of two sequences of decimal values.

```
public static decimal Covariance(this IEnumerable<decimal> source,  
IEnumerable<decimal> other)
```

Parameters

source [IEnumerable<decimal>](#)

The first sequence of decimal values to calculate the Covariance of.

other [IEnumerable<decimal>](#)

The second sequence of decimal values to calculate the Covariance of.

Returns

[decimal](#)

The Covariance of the two sequence of values.

Remarks

$$cov(X, Y) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 (y_i - \bar{y})^2$$

Covariance(IEnumerable<double>, IEnumerable<double>)

Computes the Covariance of two sequences of double values.

```
public static double Covariance(this IEnumerable<double> source, IEnumerable<double> other)
```

Parameters

source [IEnumerable<double>](#)

The first sequence of double values to calculate the Covariance of.

other [IEnumerable<double>](#)

The second sequence of double values to calculate the Covariance of.

Returns

[double](#)

The Covariance of the two sequence of values.

Remarks

$$cov(X, Y) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 (y_i - \bar{y})^2$$

Covariance(IEnumerable<int>, IEnumerable<int>)

Computes the Covariance of two sequences of int values.

```
public static double Covariance(this IEnumerable<int> source, IEnumerable<int> other)
```

Parameters

source [IEnumerable<int>](#)

The first sequence of int values to calculate the Covariance of.

other [IEnumerable<int>](#)

The second sequence of int values to calculate the Covariance of.

Returns

double

The Covariance of the two sequence of values.

Remarks

$$cov(X, Y) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 (y_i - \bar{y})^2$$

Covariance(IEnumerable<long>, IEnumerable<long>)

Computes the Covariance of two sequences of long values.

```
public static double Covariance(this IEnumerable<long> source, IEnumerable<long> other)
```

Parameters

source IEnumerable<long>

The first sequence of long values to calculate the Covariance of.

other IEnumerable<long>

The second sequence of long values to calculate the Covariance of.

Returns

double

The Covariance of the two sequence of values.

Remarks

$$cov(X, Y) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 (y_i - \bar{y})^2$$

Covariance(IEnumerable<decimal?>, IEnumerable<decimal?>)

Computes the Covariance of two sequences of nullable decimal values.

```
public static decimal? Covariance(this IEnumerable<decimal?> source, IEnumerable<decimal?> other)
```

Parameters

source [IEnumerable<decimal?>](#)

The first sequence of nullable decimal values to calculate the Covariance of.

other [IEnumerable<decimal?>](#)

The second sequence of nullable decimal values to calculate the Covariance of.

Returns

[decimal?](#)

The Covariance of the two sequence of values.

Remarks

$$cov(X, Y) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 (y_i - \bar{y})^2$$

Covariance(IEnumerable<double?>, IEnumerable<double?>)

Computes the Covariance of two sequences of nullable double values.

```
public static double? Covariance(this IEnumerable<double?> source, IEnumerable<double?> other)
```

Parameters

source [IEnumerable<double?>](#)

The first sequence of nullable double values to calculate the Covariance of.

other [IEnumerable<double?>](#)

The second sequence of nullable double values to calculate the Covariance of.

Returns

double?

The Covariance of the two sequence of values.

Remarks

$$cov(X, Y) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 (y_i - \bar{y})^2$$

Covariance(IEnumerable<int?>, IEnumerable<int?>)

Computes the Covariance of two sequences of nullable int values.

```
public static double? Covariance(this IEnumerable<int?> source, IEnumerable<int?> other)
```

Parameters

source IEnumerable<int?>

The first sequence of nullable int values to calculate the Covariance of.

other IEnumerable<int?>

The second sequence of nullable int values to calculate the Covariance of.

Returns

double?

The Covariance of the two sequence of values.

Remarks

$$cov(X, Y) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 (y_i - \bar{y})^2$$

Covariance(IEnumerable<long?>, IEnumerable<long?>)

Computes the Covariance of two sequences of nullable long values.

```
public static double? Covariance(this IEnumerable<long?> source, IEnumerable<long?> other)
```

Parameters

source [IEnumerable<long?>](#)

The first sequence of nullable long values to calculate the Covariance of.

other [IEnumerable<long?>](#)

The second sequence of nullable long values to calculate the Covariance of.

Returns

[double?](#)

The Covariance of the two sequence of values.

Remarks

$$cov(X, Y) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 (y_i - \bar{y})^2$$

Covariance(IEnumerable<float?>, IEnumerable<float?>)

Computes the Covariance of two sequences of nullable float values.

```
public static float? Covariance(this IEnumerable<float?> source, IEnumerable<float?> other)
```

Parameters

source [IEnumerable<float?>](#)

The first sequence of nullable float values to calculate the Covariance of.

other [IEnumerable<float?>](#)

The second sequence of nullable float values to calculate the Covariance of.

Returns

[float](#)?

The Covariance of the two sequence of values.

Remarks

$$cov(X, Y) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 (y_i - \bar{y})^2$$

Covariance(IEnumerable<float>, IEnumerable<float>)

Computes the Covariance of two sequences of float values.

```
public static float Covariance(this IEnumerable<float> source, IEnumerable<float> other)
```

Parameters

source [IEnumerable](#)<[float](#)>

The first sequence of float values to calculate the Covariance of.

other [IEnumerable](#)<[float](#)>

The second sequence of float values to calculate the Covariance of.

Returns

[float](#)

The Covariance of the two sequence of values.

Remarks

$$cov(X, Y) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 (y_i - \bar{y})^2$$

CovarianceNaN(IEnumerable<double>, IEnumerable<double>)

Computes the Covariance of two sequences of double values.

```
public static double CovarianceNaN(this IEnumerable<double> source,  
IEnumerable<double> other)
```

Parameters

source [IEnumerable<double>](#)

The first sequence of double values to calculate the Covariance of.

other [IEnumerable<double>](#)

The second sequence of double values to calculate the Covariance of.

Returns

[double](#)

The Covariance of the two sequence of values.

CovarianceNaN(IEnumerable<int>, IEnumerable<int>)

Computes the Covariance of two sequences of int values.

```
public static double CovarianceNaN(this IEnumerable<int> source, IEnumerable<int> other)
```

Parameters

source [IEnumerable<int>](#)

The first sequence of int values to calculate the Covariance of.

other [IEnumerable<int>](#)

The second sequence of int values to calculate the Covariance of.

Returns

[double](#)

The Covariance of the two sequence of values.

CovarianceNaN(IEnumerable<long>, IEnumerable<long>)

Computes the Covariance of two sequences of long values.

```
public static double CovarianceNaN(this IEnumerable<long> source, IEnumerable<long> other)
```

Parameters

source [IEnumerable<long>](#)

The first sequence of long values to calculate the Covariance of.

other [IEnumerable<long>](#)

The second sequence of long values to calculate the Covariance of.

Returns

[double](#)

The Covariance of the two sequence of values.

CovarianceNaN(IEnumerable<double?>, IEnumerable<double?>)

Computes the Covariance of two sequences of nullable double values.

```
public static double? CovarianceNaN(this IEnumerable<double?> source, IEnumerable<double?> other)
```

Parameters

source [IEnumerable<double?>](#)

The first sequence of nullable double values to calculate the Covariance of.

other [IEnumerable<double?>](#)

The second sequence of nullable double values to calculate the Covariance of.

Returns

[double](#)?

The Covariance of the two sequence of values.

CovarianceNaN(IEnumerable<int?>, IEnumerable<int?>)

Computes the Covariance of two sequences of nullable int values.

```
public static double? CovarianceNaN(this IEnumerable<int?> source, IEnumerable<int?> other)
```

Parameters

source [IEnumerable](#)<int?>

The first sequence of nullable int values to calculate the Covariance of.

other [IEnumerable](#)<int?>

The second sequence of nullable int values to calculate the Covariance of.

Returns

[double](#)?

The Covariance of the two sequence of values.

CovarianceNaN(IEnumerable<long?>, IEnumerable<long?>)

Computes the Covariance of two sequences of nullable long values.

```
public static double? CovarianceNaN(this IEnumerable<long?> source, IEnumerable<long?> other)
```

Parameters

source [IEnumerable](#)<long?>

The first sequence of nullable long values to calculate the Covariance of.

other [IEnumerable](#)<long?>

The second sequence of nullable long values to calculate the Covariance of.

Returns

[double](#)?

The Covariance of the two sequence of values.

CovarianceNaN(IEnumerable<float?>, IEnumerable<float?>)

Computes the Covariance of two sequences of nullable float values.

```
public static float? CovarianceNaN(this IEnumerable<float?> source, IEnumerable<float?> other)
```

Parameters

source [IEnumerable](#)<[float](#)?>

The first sequence of nullable float values to calculate the Covariance of.

other [IEnumerable](#)<[float](#)?>

The second sequence of nullable float values to calculate the Covariance of.

Returns

[float](#)?

The Covariance of the two sequence of values.

CovarianceNaN(IEnumerable<float>, IEnumerable<float>)

Computes the Covariance of two sequences of float values.

```
public static float CovarianceNaN(this IEnumerable<float> source, IEnumerable<float> other)
```

Parameters

source [IEnumerable](#)<[float](#)>

The first sequence of float values to calculate the Covariance of.

other [IEnumerable<float>](#)

The second sequence of float values to calculate the Covariance of.

Returns

[float](#)

The Covariance of the two sequence of values.

Histogram(IEnumerable<decimal>, int, BinningMode)

Computes the Histogram of a sequence of decimal values.

```
public static IEnumerable<Bin> Histogram(this IEnumerable<decimal> source, int binCount,  
BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable<decimal>](#)

A sequence of decimal values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable<Bin>](#)

The Histogram of the sequence of decimal values.

Histogram(IEnumerable<double>, int, BinningMode)

Computes the Histogram of a sequence of double values.

```
public static IEnumerable<Bin> Histogram(this IEnumerable<double> source, int binCount,  
BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable<double>](#)

A sequence of double values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable<Bin>](#)

The Histogram of the sequence of double values.

Histogram(IEnumerable<Int128>, int, BinningMode)

Computes the Histogram of a sequence of Int128 values.

```
public static IEnumerable<Bin> Histogram(this IEnumerable<Int128> source, int binCount,  
BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable<Int128>](#)

A sequence of Int128 values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

`mode` [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable](#)<[Bin](#)>

The Histogram of the sequence of Int128 values.

Histogram(IEnumerable<int>, int, BinningMode)

Computes the Histogram of a sequence of int values.

```
public static IEnumerable<Bin> Histogram(this IEnumerable<int> source, int binCount,  
BinningMode mode = BinningMode.Unbounded)
```

Parameters

`source` [IEnumerable](#)<[int](#)>

A sequence of int values to calculate the Histogram of.

`binCount` [int](#)

The number of bins into which to segregate the data.

`mode` [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable](#)<[Bin](#)>

The Histogram of the sequence of int values.

Histogram(IEnumerable<long>, int, BinningMode)

Computes the Histogram of a sequence of long values.

```
public static IEnumerable<Bin> Histogram(this IEnumerable<long> source, int binCount,  
BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable<long>](#)

A sequence of long values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable<Bin>](#)

The Histogram of the sequence of long values.

Histogram(IEnumerable<decimal?>, int, BinningMode)

Computes the Histogram of a sequence of nullable decimal values.

```
public static IEnumerable<Bin> Histogram(this IEnumerable<decimal?> source, int binCount,  
BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable<decimal?>](#)

A sequence of nullable decimal values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable<Bin>](#)

The Histogram of the sequence of nullable decimal values.

Histogram(IEnumerable<double?>, int, BinningMode)

Computes the Histogram of a sequence of nullable double values.

```
public static IEnumerable<Bin> Histogram(this IEnumerable<double?> source, int binCount,  
BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable<double?>](#)

A sequence of nullable double values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable<Bin>](#)

The Histogram of the sequence of nullable double values.

Histogram(IEnumerable<Int128?>, int, BinningMode)

Computes the Histogram of a sequence of nullable Int128 values.

```
public static IEnumerable<Bin> Histogram(this IEnumerable<Int128?> source, int binCount,
```

```
    BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable<Int128?>](#)

A sequence of nullable Int128 values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable<Bin>](#)

The Histogram of the sequence of nullable Int128 values.

Histogram(IEnumerable<int?>, int, BinningMode)

Computes the Histogram of a sequence of nullable int values.

```
public static IEnumerable<Bin> Histogram(this IEnumerable<int?> source, int binCount,
    BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable<int?>](#)

A sequence of nullable int values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable<Bin>](#)

The Histogram of the sequence of nullable int values.

Histogram(IEnumerable<long?>, int, BinningMode)

Computes the Histogram of a sequence of nullable long values.

```
public static IEnumerable<Bin> Histogram(this IEnumerable<long?> source, int binCount,  
BinningMode mode = BinningMode.Unbounded)
```

Parameters

[source](#) [IEnumerable<long?>](#)

A sequence of nullable long values to calculate the Histogram of.

[binCount](#) [int](#)

The number of bins into which to segregate the data.

[mode](#) [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable<Bin>](#)

The Histogram of the sequence of nullable long values.

Histogram(IEnumerable<float?>, int, BinningMode)

Computes the Histogram of a sequence of nullable float values.

```
public static IEnumerable<Bin> Histogram(this IEnumerable<float?> source, int binCount,  
BinningMode mode = BinningMode.Unbounded)
```

Parameters

`source` [IEnumerable<float?>](#)

A sequence of nullable float values to calculate the Histogram of.

`binCount` [int](#)

The number of bins into which to segregate the data.

`mode` [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable<Bin>](#)

The Histogram of the sequence of nullable float values.

Histogram(IEnumerable<float>, int, BinningMode)

Computes the Histogram of a sequence of float values.

```
public static IEnumerable<Bin> Histogram(this IEnumerable<float> source, int binCount,  
BinningMode mode = BinningMode.Unbounded)
```

Parameters

`source` [IEnumerable<float>](#)

A sequence of float values to calculate the Histogram of.

`binCount` [int](#)

The number of bins into which to segregate the data.

`mode` [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable<Bin>](#)

The Histogram of the sequence of float values.

Histogram<TSource>(IEnumerable<TSource>, int, Func<TSource, decimal>, BinningMode)

Computes the Histogram of a sequence of decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static IEnumerable<Bin> Histogram<TSource>(this IEnumerable<TSource> source, int  
binCount, Func<TSource, decimal> selector, BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable<TSource>](#)

A sequence of values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

selector [Func<TSource, decimal>](#)

A transform function to apply to each element.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable<Bin>](#)

The Histogram of the sequence of decimal values.

Type Parameters

TSource

The type of the elements of source.

Histogram<TSource>(IEnumerable<TSource>, int, Func<TSource, double>, BinningMode)

Computes the Histogram of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static IEnumerable<Bin> Histogram<TSource>(this IEnumerable<TSource> source, int binCount, Func<TSource, double> selector, BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

selector [Func](#)<TSource, double>

A transform function to apply to each element.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable](#)<Bin>

The Histogram of the sequence of double values.

Type Parameters

TSource

The type of the elements of source.

Histogram<TSource>(IEnumerable<TSource>, int, Func<TSource, Int128>, BinningMode)

Computes the Histogram of a sequence of Int128 values that are obtained by invoking a transform function on each element of the input sequence.

```
public static IEnumerable<Bin> Histogram<TSource>(this IEnumerable<TSource> source, int binCount, Func<TSource, Int128> selector, BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

selector [Func](#)<TSource, [Int128](#)>

A transform function to apply to each element.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable](#)<Bin>

The Histogram of the sequence of Int128 values.

Type Parameters

TSource

The type of the elements of source.

Histogram<TSource>(IEnumerable<TSource>, int, Func<TSource, int>, BinningMode)

Computes the Histogram of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static IEnumerable<Bin> Histogram<TSource>(this IEnumerable<TSource> source, int binCount, Func<TSource, int> selector, BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

selector [Func](#)<TSource, [int](#)>

A transform function to apply to each element.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable](#)<Bin>

The Histogram of the sequence of int values.

Type Parameters

TSource

The type of the elements of source.

Histogram<TSource>(IEnumerable<TSource>, int, Func<TSource, long>, BinningMode)

Computes the Histogram of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static IEnumerable<Bin> Histogram<TSource>(this IEnumerable<TSource> source, int binCount, Func<TSource, long> selector, BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

selector [Func](#)<TSource, [long](#)>

A transform function to apply to each element.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable](#)<[Bin](#)>

The Histogram of the sequence of long values.

Type Parameters

TSource

The type of the elements of source.

Histogram<TSource>(IEnumerable<TSource>, int, Func<TSource, decimal?>, BinningMode)

Computes the Histogram of a sequence of nullable decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static IEnumerable<Bin> Histogram<TSource>(this IEnumerable<TSource> source, int  
binCount, Func<TSource, decimal?> selector, BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

selector [Func](#)<TSource, [decimal](#)?>

A transform function to apply to each element.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable](#)<Bin>

The Histogram of the sequence of nullable decimal values.

Type Parameters

TSource

The type of the elements of source.

Histogram<TSource>(IEnumerable<TSource>, int, Func<TSource, double?>, BinningMode)

Computes the Histogram of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static IEnumerable<Bin> Histogram<TSource>(this IEnumerable<TSource> source, int  
binCount, Func<TSource, double?> selector, BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

selector [Func<TSource, double?>](#)

A transform function to apply to each element.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable<Bin>](#)

The Histogram of the sequence of nullable double values.

Type Parameters

TSource

The type of the elements of source.

Histogram<TSource>(IEnumerable<TSource>, int, Func<TSource, Int128?>, BinningMode)

Computes the Histogram of a sequence of nullable Int128 values that are obtained by invoking a transform function on each element of the input sequence.

```
public static IEnumerable<Bin> Histogram<TSource>(this IEnumerable<TSource> source, int  
binCount, Func<TSource, Int128?> selector, BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable<TSource>](#)

A sequence of values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

selector [Func<TSource, Int128?>](#)

A transform function to apply to each element.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable](#)<Bin>

The Histogram of the sequence of nullable Int128 values.

Type Parameters

TSource

The type of the elements of source.

Histogram<TSource>(IEnumerable<TSource>, int, Func<TSource, int?>, BinningMode)

Computes the Histogram of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static IEnumerable<Bin> Histogram<TSource>(this IEnumerable<TSource> source, int  
binCount, Func<TSource, int?> selector, BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

selector [Func](#)<TSource, [int](#)?>

A transform function to apply to each element.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable](#)<Bin>

The Histogram of the sequence of nullable int values.

Type Parameters

TSource

The type of the elements of source.

Histogram<TSource>(IEnumerable<TSource>, int, Func<TSource, long?>, BinningMode)

Computes the Histogram of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static IEnumerable<Bin> Histogram<TSource>(this IEnumerable<TSource> source, int  
binCount, Func<TSource, long?> selector, BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

selector [Func](#)<TSource, long?>

A transform function to apply to each element.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable<Bin>](#)

The Histogram of the sequence of nullable long values.

Type Parameters

TSource

The type of the elements of source.

Histogram<TSource>(IEnumerable<TSource>, int, Func<TSource, float?>, BinningMode)

Computes the Histogram of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static IEnumerable<Bin> Histogram<TSource>(this IEnumerable<TSource> source, int binCount, Func<TSource, float?> selector, BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable<TSource>](#)

A sequence of values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

selector [Func<TSource, float?>](#)

A transform function to apply to each element.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable<Bin>](#)

The Histogram of the sequence of nullable float values.

Type Parameters

TSource

The type of the elements of source.

Histogram<TSource>(IEnumerable<TSource>, int, Func<TSource, float>, BinningMode)

Computes the Histogram of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static IEnumerable<Bin> Histogram<TSource>(this IEnumerable<TSource> source, int binCount, Func<TSource, float> selector, BinningMode mode = BinningMode.Unbounded)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Histogram of.

binCount [int](#)

The number of bins into which to segregate the data.

selector [Func](#)<TSource, float>

A transform function to apply to each element.

mode [BinningMode](#)

The method used to determine the range of each bin

Returns

[IEnumerable](#)<Bin>

The Histogram of the sequence of float values.

Type Parameters

TSource

The type of the elements of source.

Kurtosis(IEnumerable<decimal>)

Computes the sample Kurtosis of a sequence of decimal values

```
public static decimal Kurtosis(this IEnumerable<decimal> source)
```

Parameters

source [IEnumerable<decimal>](#)

A sequence of decimal values to calculate the Kurtosis of.

Returns

[decimal](#)

The Kurtosis of the sequence of values.

Remarks

$$K = \frac{N(N+1)(N-1)\sum_{i=1}^N(x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N(x_i - \bar{x})^2)^2}$$

Kurtosis(IEnumerable<double>)

Computes the sample Kurtosis of a sequence of double values

```
public static double Kurtosis(this IEnumerable<double> source)
```

Parameters

source [IEnumerable<double>](#)

A sequence of double values to calculate the Kurtosis of.

Returns

double

The Kurtosis of the sequence of values.

Remarks

$$K = \frac{N(N+1)(N-1) \sum_{i=1}^N (x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N (x_i - \bar{x})^2)^2}$$

Kurtosis(IEnumerable<int>)

Computes the sample Kurtosis of a sequence of int values

```
public static double Kurtosis(this IEnumerable<int> source)
```

Parameters

source IEnumerable<int>

A sequence of int values to calculate the Kurtosis of.

Returns

double

The Kurtosis of the sequence of values.

Remarks

$$K = \frac{N(N+1)(N-1) \sum_{i=1}^N (x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N (x_i - \bar{x})^2)^2}$$

Kurtosis(IEnumerable<long>)

Computes the sample Kurtosis of a sequence of long values

```
public static double Kurtosis(this IEnumerable<long> source)
```

Parameters

source [IEnumerable<long>](#)

A sequence of long values to calculate the Kurtosis of.

Returns

[double](#)

The Kurtosis of the sequence of values.

Remarks

$$K = \frac{N(N+1)(N-1)\sum_{i=1}^N(x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N(x_i - \bar{x})^2)^2}$$

Kurtosis(IEnumerable<decimal?>)

Computes the sample Kurtosis of a sequence of nullable decimal values

```
public static decimal? Kurtosis(this IEnumerable<decimal?> source)
```

Parameters

source [IEnumerable<decimal?>](#)

A sequence of nullable decimal values to calculate the Kurtosis of.

Returns

[decimal?](#)

The Kurtosis of the sequence of values.

Remarks

$$K = \frac{N(N+1)(N-1)\sum_{i=1}^N(x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N(x_i - \bar{x})^2)^2}$$

Kurtosis(IEnumerable<double?>)

Computes the sample Kurtosis of a sequence of nullable double values

```
public static double? Kurtosis(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable](#)<[double](#)?>

A sequence of nullable double values to calculate the Kurtosis of.

Returns

[double](#)?
The Kurtosis of the sequence of values.

Remarks

$$K = \frac{N(N+1)(N-1)\sum_{i=1}^N(x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N(x_i - \bar{x})^2)^2}$$

Kurtosis(IEnumerable<int?>)

Computes the sample Kurtosis of a sequence of nullable int values

```
public static double? Kurtosis(this IEnumerable<int?> source)
```

Parameters

source [IEnumerable](#)<[int](#)?>

A sequence of nullable int values to calculate the Kurtosis of.

Returns

[double](#)?
The Kurtosis of the sequence of values.

Remarks

$$K = \frac{N(N+1)(N-1)\sum_{i=1}^N(x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N(x_i - \bar{x})^2)^2}$$

Kurtosis(IEnumerable<long?>)

Computes the sample Kurtosis of a sequence of nullable long values

```
public static double? Kurtosis(this IEnumerable<long?> source)
```

Parameters

source [IEnumerable](#)<[long](#)?>

A sequence of nullable long values to calculate the Kurtosis of.

Returns

[double](#)?
The Kurtosis of the sequence of values.

Remarks

$$K = \frac{N(N+1)(N-1)\sum_{i=1}^N(x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N(x_i - \bar{x})^2)^2}$$

Kurtosis(IEnumerable<float?>)

Computes the sample Kurtosis of a sequence of nullable float values

```
public static float? Kurtosis(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable](#)<[float](#)?>

A sequence of nullable float values to calculate the Kurtosis of.

Returns

float?

The Kurtosis of the sequence of values.

Remarks

$$K = \frac{N(N+1)(N-1) \sum_{i=1}^N (x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N (x_i - \bar{x})^2)^2}$$

Kurtosis(IEnumerable<float>)

Computes the sample Kurtosis of a sequence of float values

```
public static float Kurtosis(this IEnumerable<float> source)
```

Parameters

source IEnumerable<float>

A sequence of float values to calculate the Kurtosis of.

Returns

float

The Kurtosis of the sequence of values.

Remarks

$$K = \frac{N(N+1)(N-1) \sum_{i=1}^N (x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N (x_i - \bar{x})^2)^2}$$

KurtosisNaN(IEnumerable<double>)

Computes the sample Kurtosis of a sequence of double values

```
public static double KurtosisNaN(this IEnumerable<double> source)
```

Parameters

source `IEnumerable<double>`

A sequence of double values to calculate the Kurtosis of.

Returns

`double`

The Kurtosis of the sequence of values.

KurtosisNaN(IEnumerable<int>)

Computes the sample Kurtosis of a sequence of int values

`public static double KurtosisNaN(this IEnumerable<int> source)`

Parameters

source `IEnumerable<int>`

A sequence of int values to calculate the Kurtosis of.

Returns

`double`

The Kurtosis of the sequence of values.

KurtosisNaN(IEnumerable<long>)

Computes the sample Kurtosis of a sequence of long values

`public static double KurtosisNaN(this IEnumerable<long> source)`

Parameters

source `IEnumerable<long>`

A sequence of long values to calculate the Kurtosis of.

Returns

double ↗

The Kurtosis of the sequence of values.

KurtosisNaN(IEnumerable<double?>)

Computes the sample Kurtosis of a sequence of nullable double values

```
public static double? KurtosisNaN(this IEnumerable<double?> source)
```

Parameters

source IEnumerable ↗<double ↗?>

A sequence of nullable double values to calculate the Kurtosis of.

Returns

double ↗?

The Kurtosis of the sequence of values.

KurtosisNaN(IEnumerable<int?>)

Computes the sample Kurtosis of a sequence of nullable int values

```
public static double? KurtosisNaN(this IEnumerable<int?> source)
```

Parameters

source IEnumerable ↗<int ↗?>

A sequence of nullable int values to calculate the Kurtosis of.

Returns

double ↗?

The Kurtosis of the sequence of values.

KurtosisNaN(IEnumerable<long?>)

Computes the sample Kurtosis of a sequence of nullable long values

```
public static double? KurtosisNaN(this IEnumerable<long?> source)
```

Parameters

source [IEnumerable<long?>](#)

A sequence of nullable long values to calculate the Kurtosis of.

Returns

[double?](#)

The Kurtosis of the sequence of values.

KurtosisNaN(IEnumerable<float?>)

Computes the sample Kurtosis of a sequence of nullable float values

```
public static float? KurtosisNaN(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

A sequence of nullable float values to calculate the Kurtosis of.

Returns

[float?](#)

The Kurtosis of the sequence of values.

KurtosisNaN(IEnumerable<float>)

Computes the sample Kurtosis of a sequence of float values

```
public static float KurtosisNaN(this IEnumerable<float> source)
```

Parameters

source [IEnumerable<float>](#)

A sequence of float values to calculate the Kurtosis of.

Returns

[float](#)

The Kurtosis of the sequence of values.

KurtosisNaN<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the sample Kurtosis of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double KurtosisNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable<TSource>](#)

A sequence of values that are used to calculate a Kurtosis

selector [Func<TSource, double>](#)

A transform function to apply to each element.

Returns

[double](#)

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

KurtosisNaN<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the sample Kurtosis of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double KurtosisNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, int>

A transform function to apply to each element.

Returns

[double](#)

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

KurtosisNaN<TSource>(IEnumarable<TSource>, Func<TSource, long>)

Computes the sample Kurtosis of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double KurtosisNaN<TSource>(this IEnumarable<TSource> source, Func<TSource, long> selector)
```

Parameters

source [IEnumarable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, long>

A transform function to apply to each element.

Returns

[double](#)

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

KurtosisNaN<TSource>(IEnumarable<TSource>, Func<TSource, double?>)

Computes the sample Kurtosis of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? KurtosisNaN<TSource>(this IEnumarable<TSource> source, Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, [double](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

KurtosisNaN<TSource>(IEnumarable<TSource>, Func<TSource, int?>)

Computes the sample Kurtosis of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? KurtosisNaN<TSource>(this IEnumarable<TSource> source, Func<TSource, int?> selector)
```

Parameters

source [IEnumarable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, [int](#)?>

A transform function to apply to each element.

Returns

double?

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

KurtosisNaN<TSource>(IEnumarable<TSource>, Func<TSource, long?>)

Computes the sample Kurtosis of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? KurtosisNaN<TSource>(this IEnumarable<TSource> source, Func<TSource, long?> selector)
```

Parameters

source IEnumerable<TSource>

A sequence of values that are used to calculate a Kurtosis

selector Func<TSource, long?>

A transform function to apply to each element.

Returns

double?

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

KurtosisNaN<TSource>(IEnumarable<TSource>, Func<TSource, float?>)

Computes the sample Kurtosis of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? KurtosisNaN<TSource>(this IEnumarable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumarable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, [float](#)?>

A transform function to apply to each element.

Returns

[float](#)?

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

KurtosisNaN<TSource>(IEnumarable<TSource>, Func<TSource, float>)

Computes the sample Kurtosis of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float KurtosisNaN<TSource>(this IEnumarable<TSource> source, Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, [float](#)>

A transform function to apply to each element.

Returns

[float](#)

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Kurtosis<TSource>(IEnumerable<TSource>, Func<TSource, decimal>)

Computes the sample Kurtosis of a sequence of decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal Kurtosis<TSource>(this IEnumerable<TSource> source, Func<TSource, decimal> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, [decimal](#)>

A transform function to apply to each element.

Returns

[decimal](#)

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$K = \frac{N(N+1)(N-1)\sum_{i=1}^N(x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N(x_i - \bar{x})^2)^2}$$

Kurtosis<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the sample Kurtosis of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Kurtosis<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, [double](#)>

A transform function to apply to each element.

Returns

[double](#)

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$K = \frac{N(N+1)(N-1) \sum_{i=1}^N (x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N (x_i - \bar{x})^2)^2}$$

Kurtosis<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the sample Kurtosis of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Kurtosis<TSource>(this IEnumerable<TSource> source, Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, [int](#)>

A transform function to apply to each element.

Returns

[double](#)

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$K = \frac{N(N+1)(N-1)\sum_{i=1}^N(x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N(x_i - \bar{x})^2)^2}$$

Kurtosis<TSource>(IEnumerable<TSource>, Func<TSource, long>)

Computes the sample Kurtosis of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Kurtosis<TSource>(this IEnumerable<TSource> source, Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, long>

A transform function to apply to each element.

Returns

[double](#)

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$K = \frac{N(N+1)(N-1)\sum_{i=1}^N(x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N(x_i - \bar{x})^2)^2}$$

Kurtosis<TSource>(IEnumerable<TSource>, Func<TSource, decimal?>)

Computes the sample Kurtosis of a sequence of nullable decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal? Kurtosis<TSource>(this IEnumerable<TSource> source, Func<TSource, decimal?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, decimal?>

A transform function to apply to each element.

Returns

[decimal](#)?

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$K = \frac{N(N+1)(N-1)\sum_{i=1}^N(x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N(x_i - \bar{x})^2)^2}$$

Kurtosis<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the sample Kurtosis of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Kurtosis<TSource>(this IEnumerable<TSource> source, Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, double?>

A transform function to apply to each element.

Returns

[double](#)?

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$K = \frac{N(N + 1)(N - 1) \sum_{i=1}^N (x_i - \bar{x})^4}{(N - 2)(N - 3)(\sum_{i=1}^N (x_i - \bar{x})^2)^2}$$

Kurtosis<TSource>(IEnumerable<TSource>, Func<TSource, int?>)

Computes the sample Kurtosis of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Kurtosis<TSource>(this IEnumerable<TSource> source, Func<TSource, int?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, [int](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$K = \frac{N(N + 1)(N - 1) \sum_{i=1}^N (x_i - \bar{x})^4}{(N - 2)(N - 3) (\sum_{i=1}^N (x_i - \bar{x})^2)^2}$$

Kurtosis<TSource>(IEnumerable<TSource>, Func<TSource, long?>)

Computes the sample Kurtosis of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Kurtosis<TSource>(this IEnumerable<TSource> source, Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, [long](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$K = \frac{N(N+1)(N-1) \sum_{i=1}^N (x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N (x_i - \bar{x})^2)^2}$$

Kurtosis<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the sample Kurtosis of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? Kurtosis<TSource>(this IEnumerable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, [float](#)?>

A transform function to apply to each element.

Returns

[float](#)?

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$K = \frac{N(N+1)(N-1) \sum_{i=1}^N (x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N (x_i - \bar{x})^2)^2}$$

Kurtosis<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the sample Kurtosis of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float Kurtosis<TSource>(this IEnumerable<TSource> source, Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Kurtosis

selector [Func](#)<TSource, float>

A transform function to apply to each element.

Returns

[float](#)

The Kurtosis of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$K = \frac{N(N+1)(N-1) \sum_{i=1}^N (x_i - \bar{x})^4}{(N-2)(N-3)(\sum_{i=1}^N (x_i - \bar{x})^2)^2}$$

LeastSquares(IEnumerable<Tuple<decimal, decimal>>)

Computes the LeastSquares of a sequence of Tuple{decimal, decimal} values.

```
public static LeastSquares LeastSquares(this IEnumerable<Tuple<decimal, decimal>> source)
```

Parameters

source [IEnumerable<Tuple<decimal, decimal>>](#)

A sequence of Tuple{decimal, decimal} values to calculate the LeastSquares of.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

LeastSquares(IEnumerable<Tuple<double, double>>)

Computes the LeastSquares of a sequence of Tuple{double, double} values.

```
public static LeastSquares LeastSquares(this IEnumerable<Tuple<double, double>> source)
```

Parameters

source [IEnumerable<Tuple<double, double>>](#)

A sequence of Tuple{double, double} values to calculate the LeastSquares of.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

LeastSquares(IEnumerable<Tuple<Int128, Int128>>)

Computes the LeastSquares of a sequence of Tuple{Int128, Int128} values.

```
public static LeastSquares LeastSquares(this IEnumerable<Tuple<Int128, Int128>> source)
```

Parameters

source [IEnumerable<Tuple<Int128, Int128>>](#)

A sequence of Tuple{Int128, Int128} values to calculate the LeastSquares of.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

LeastSquares(IEnumerable<Tuple<int, int>>)

Computes the LeastSquares of a sequence of Tuple{int, int} values.

```
public static LeastSquares LeastSquares(this IEnumerable<Tuple<int, int>> source)
```

Parameters

source [IEnumerable<Tuple<int, int>>](#)

A sequence of Tuple{int, int} values to calculate the LeastSquares of.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

LeastSquares(IEnumerable<Tuple<long, long>>)

Computes the LeastSquares of a sequence of Tuple{long, long} values.

```
public static LeastSquares LeastSquares(this IEnumerable<Tuple<long, long>> source)
```

Parameters

source [IEnumerable<Tuple<long, long>>](#)

A sequence of Tuple{long, long} values to calculate the LeastSquares of.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

LeastSquares(IEnumerable<Tuple<decimal?, decimal?>>)

Computes the LeastSquares of a sequence of Tuple{decimal?, decimal?} values.

```
public static LeastSquares? LeastSquares(this IEnumerable<Tuple<decimal?, decimal?>> source)
```

Parameters

source [IEnumerable<Tuple<decimal?, decimal?>>](#)

A sequence of Tuple{decimal?, decimal?} values to calculate the LeastSquares of.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values, or null if the source sequence is empty or contains only values that are null.

LeastSquares(IEnumerable<Tuple<double?, double?>>)

Computes the LeastSquares of a sequence of Tuple{double?, double?} values.

```
public static LeastSquares? LeastSquares(this IEnumerable<Tuple<double?, double?>> source)
```

Parameters

source `IEnumerable<Tuple<double?, double?>>`

A sequence of Tuple{double?, double?} values to calculate the LeastSquares of.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values, or null if the source sequence is empty or contains only values that are null.

LeastSquares(IEnumerable<Tuple<Int128?, Int128?>>)

Computes the LeastSquares of a sequence of Tuple{Int128?, Int128?} values.

```
public static LeastSquares? LeastSquares(this IEnumerable<Tuple<Int128?, Int128?>> source)
```

Parameters

source `IEnumerable<Tuple<Int128?, Int128?>>`

A sequence of Tuple{Int128?, Int128?} values to calculate the LeastSquares of.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values, or null if the source sequence is empty or contains only values that are null.

LeastSquares(IEnumerable<Tuple<int?, int?>>)

Computes the LeastSquares of a sequence of Tuple{int?, int?} values.

```
public static LeastSquares? LeastSquares(this IEnumerable<Tuple<int?, int?>> source)
```

Parameters

source [IEnumerable<Tuple<int?, int?>>](#)

A sequence of Tuple{int?, int?} values to calculate the LeastSquares of.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values, or null if the source sequence is empty or contains only values that are null.

LeastSquares(IEnumerable<Tuple<long?, long?>>)

Computes the LeastSquares of a sequence of Tuple{long?, long?} values.

```
public static LeastSquares? LeastSquares(this IEnumerable<Tuple<long?, long?>> source)
```

Parameters

source [IEnumerable<Tuple<long?, long?>>](#)

A sequence of Tuple{long?, long?} values to calculate the LeastSquares of.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values, or null if the source sequence is empty or contains only values that are null.

LeastSquares(IEnumerable<Tuple<float?, float?>>)

Computes the LeastSquares of a sequence of Tuple{float?, float?} values.

```
public static LeastSquares? LeastSquares(this IEnumerable<Tuple<float?, float?>> source)
```

Parameters

source [IEnumerable<Tuple<float?, float?>>](#)

A sequence of Tuple{float?, float?} values to calculate the LeastSquares of.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values, or null if the source sequence is empty or contains only values that are null.

LeastSquares(IEnumerable<Tuple<float, float>>)

Computes the LeastSquares of a sequence of Tuple{float, float} values.

```
public static LeastSquares LeastSquares(this IEnumerable<Tuple<float, float>> source)
```

Parameters

source [IEnumerable<Tuple<float, float>>](#)

A sequence of Tuple{float, float} values to calculate the LeastSquares of.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

LeastSquaresNaN(IEnumerable<Tuple<decimal, decimal>>)

Computes the LeastSquares of a sequence of Tuple{decimal, decimal} values.

```
public static LeastSquares LeastSquaresNaN(this IEnumerable<Tuple<decimal, decimal>> source)
```

Parameters

source `IEnumerable<Tuple<decimal, decimal>>`

A sequence of Tuple{decimal, decimal} values to calculate the LeastSquares of.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

LeastSquaresNaN(IEnumerable<Tuple<double, double>>)

Computes the LeastSquares of a sequence of Tuple{double, double} values.

```
public static LeastSquares LeastSquaresNaN(this IEnumerable<Tuple<double, double>> source)
```

Parameters

source `IEnumerable<Tuple<double, double>>`

A sequence of Tuple{double, double} values to calculate the LeastSquares of.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

LeastSquaresNaN(IEnumerable<Tuple<int, int>>)

Computes the LeastSquares of a sequence of Tuple{int, int} values.

```
public static LeastSquares LeastSquaresNaN(this IEnumerable<Tuple<int, int>> source)
```

Parameters

source `IEnumerable<Tuple<int, int>>`

A sequence of Tuple{int, int} values to calculate the LeastSquares of.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

LeastSquaresNaN(IEnumerable<Tuple<long, long>>)

Computes the LeastSquares of a sequence of Tuple{long, long} values.

```
public static LeastSquares LeastSquaresNaN(this IEnumerable<Tuple<long, long>> source)
```

Parameters

`source` [IEnumerable<Tuple<long, long>>](#)

A sequence of Tuple{long, long} values to calculate the LeastSquares of.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

LeastSquaresNaN(IEnumerable<Tuple<decimal?, decimal?>>)

Computes the LeastSquares of a sequence of Tuple{decimal?, decimal?} values.

```
public static LeastSquares? LeastSquaresNaN(this IEnumerable<Tuple<decimal?, decimal?>> source)
```

Parameters

`source` [IEnumerable<Tuple<decimal?, decimal?>>](#)

A sequence of Tuple{decimal?, decimal?} values to calculate the LeastSquares of.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values, or null if the source sequence is empty or contains only values that are null.

LeastSquaresNaN(IEnumerable<Tuple<double?, double?>>)

Computes the LeastSquares of a sequence of Tuple{double?, double?} values.

```
public static LeastSquares? LeastSquaresNaN(this IEnumerable<Tuple<double?, double?>>
    >> source)
```

Parameters

source [IEnumerable<Tuple<double?, double?>>](#)

A sequence of Tuple{double?, double?} values to calculate the LeastSquares of.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values, or null if the source sequence is empty or contains only values that are null.

LeastSquaresNaN(IEnumerable<Tuple<int?, int?>>)

Computes the LeastSquares of a sequence of Tuple{int?, int?} values.

```
public static LeastSquares? LeastSquaresNaN(this IEnumerable<Tuple<int?, int?>> source)
```

Parameters

source [IEnumerable<Tuple<int?, int?>>](#)

A sequence of Tuple{int?, int?} values to calculate the LeastSquares of.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values, or null if the source sequence is empty or contains only values that are null.

LeastSquaresNaN(IEnumerable<Tuple<long?, long?>>)

Computes the LeastSquares of a sequence of Tuple{long?, long?} values.

```
public static LeastSquares? LeastSquaresNaN(this IEnumerable<Tuple<long?, long?>> source)
```

Parameters

source [IEnumerable<Tuple<long?, long?>>](#)

A sequence of Tuple{long?, long?} values to calculate the LeastSquares of.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values, or null if the source sequence is empty or contains only values that are null.

LeastSquaresNaN(IEnumerable<Tuple<float?, float?>>)

Computes the LeastSquares of a sequence of Tuple{float?, float?} values.

```
public static LeastSquares? LeastSquaresNaN(this IEnumerable<Tuple<float?, float?>> source)
```

Parameters

source [IEnumerable<Tuple<float?, float?>>](#)

A sequence of Tuple{float?, float?} values to calculate the LeastSquares of.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values, or null if the source sequence is empty or contains only values that are null.

LeastSquaresNaN(IEnumerable<Tuple<float, float>>)

Computes the LeastSquares of a sequence of Tuple{float, float} values.

```
public static LeastSquares LeastSquaresNaN(this IEnumerable<Tuple<float, float>> source)
```

Parameters

source [IEnumerable](#)<Tuple<float, float>>

A sequence of Tuple{float, float} values to calculate the LeastSquares of.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

LeastSquaresNaN<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<decimal, decimal>>)

Computes the LeastSquares of a sequence of Tuple{decimal, decimal} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares LeastSquaresNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, Tuple<decimal, decimal>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector [Func](#)<TSource, Tuple<decimal, decimal>>

A transform function to apply to each element.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquaresNaN<TSource>(I`Numerable`<TSource>, Func<TSource, Tuple<double, double>>)

Computes the LeastSquares of a sequence of Tuple{double, double} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares LeastSquaresNaN<TSource>(this INumerable<TSource> source,  
Func<TSource, Tuple<double, double>> selector)
```

Parameters

source [I`Numerable`<TSource>](#)

A sequence of values to calculate the LeastSquares of.

selector [Func<TSource, Tuple<double, double>>](#)

A transform function to apply to each element.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquaresNaN<TSource>(IEnumarable<TSource>, Func<TSource, Tuple<int, int>>)

Computes the LeastSquares of a sequence of Tuple{int, int} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares LeastSquaresNaN<TSource>(this IEnumarable<TSource> source,  
Func<TSource, Tuple<int, int>> selector)
```

Parameters

source IEnumarable<TSource>

A sequence of values to calculate the LeastSquares of.

selector Func<TSource, Tuple<int, int>>

A transform function to apply to each element.

Returns

LeastSquares

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquaresNaN<TSource>(IEnumarable<TSource>, Func<TSource, Tuple<long, long>>)

Computes the LeastSquares of a sequence of Tuple{long, long} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares LeastSquaresNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<long, long>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector [Func](#)<TSource, [Tuple](#)<long, long>>

A transform function to apply to each element.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquaresNaN<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<decimal?, decimal?>>)

Computes the LeastSquares of a sequence of Tuple{decimal?, decimal?} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares? LeastSquaresNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<decimal?, decimal?>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector `Func<TSource, Tuple<decimal?, decimal?>>`

A transform function to apply to each element.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquaresNaN<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<double?, double?>>)

Computes the LeastSquares of a sequence of Tuple{double?, double?} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares? LeastSquaresNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, Tuple<double?, double?>> selector)
```

Parameters

source `IEnumerable<TSource>`

A sequence of values to calculate the LeastSquares of.

selector `Func<TSource, Tuple<double?, double?>>`

A transform function to apply to each element.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquaresNaN<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<int?, int?>>)

Computes the LeastSquares of a sequence of Tuple{int?, int?} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares? LeastSquaresNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<int?, int?>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector [Func](#)<TSource, [Tuple](#)<[int](#)?, [int](#)?>>

A transform function to apply to each element.

Returns

[LeastSquares](#)?

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquaresNaN<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<long?, long?>>)

Computes the LeastSquares of a sequence of Tuple{long?, long?} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares? LeastSquaresNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<long?, long?>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector [Func](#)<TSource, [Tuple](#)<[long](#)?>, [long](#)?>>

A transform function to apply to each element.

Returns

[LeastSquares](#)?

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquaresNaN<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<float?, float?>>)

Computes the LeastSquares of a sequence of Tuple{float?, float?} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares? LeastSquaresNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<float?, float?>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector `Func<TSource, Tuple<float?, float?>>`

A transform function to apply to each element.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquaresNaN<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<float, float>>)

Computes the LeastSquares of a sequence of Tuple{float, float} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares LeastSquaresNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, Tuple<float, float>> selector)
```

Parameters

source `IEnumerable<TSource>`

A sequence of values to calculate the LeastSquares of.

selector `Func<TSource, Tuple<float, float>>`

A transform function to apply to each element.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquares<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<decimal, decimal>>)

Computes the LeastSquares of a sequence of Tuple{decimal, decimal} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares LeastSquares<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<decimal, decimal>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector [Func](#)<TSource, [Tuple](#)<[decimal](#), [decimal](#)>>

A transform function to apply to each element.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquares<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<double, double>>)

Computes the LeastSquares of a sequence of Tuple{double, double} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares LeastSquares<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<double, double>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector [Func](#)<TSource, [Tuple](#)<double, double>>

A transform function to apply to each element.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquares<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<Int128, Int128>>)

Computes the LeastSquares of a sequence of Tuple{Int128, Int128} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares LeastSquares<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<Int128, Int128>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector `Func<TSource, Tuple<Int128, Int128>>`

A transform function to apply to each element.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquares<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<int, int>>)

Computes the LeastSquares of a sequence of Tuple{int, int} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares LeastSquares<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<int, int>> selector)
```

Parameters

source `IEnumerable<TSource>`

A sequence of values to calculate the LeastSquares of.

selector `Func<TSource, Tuple<int, int>>`

A transform function to apply to each element.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquares<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<long, long>>)

Computes the LeastSquares of a sequence of Tuple{long, long} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares<TSource> LeastSquares<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<long, long>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector [Func](#)<TSource, [Tuple](#)<long, long>>

A transform function to apply to each element.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquares<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<decimal?, decimal?>>)

Computes the LeastSquares of a sequence of Tuple{decimal?, decimal?} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares? LeastSquares<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<decimal?, decimal?>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector [Func](#)<TSource, [Tuple](#)<decimal?, decimal?>>

A transform function to apply to each element.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquares<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<double?, double?>>)

Computes the LeastSquares of a sequence of Tuple{double?, double?} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares? LeastSquares<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<double?, double?>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector `Func<TSource, Tuple<double?, double?>>`

A transform function to apply to each element.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquares<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<Int128?, Int128?>>)

Computes the LeastSquares of a sequence of Tuple{Int128?, Int128?} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares? LeastSquares<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<Int128?, Int128?>> selector)
```

Parameters

source `IEnumerable<TSource>`

A sequence of values to calculate the LeastSquares of.

selector `Func<TSource, Tuple<Int128?, Int128?>>`

A transform function to apply to each element.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquares<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<int?, int?>>)

Computes the LeastSquares of a sequence of Tuple{int?, int?} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares? LeastSquares<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<int?, int?>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector [Func](#)<TSource, [Tuple](#)<[int](#)?, [int](#)?>>

A transform function to apply to each element.

Returns

[LeastSquares](#)?

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquares<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<long?, long?>>)

Computes the LeastSquares of a sequence of Tuple{long?, long?} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares? LeastSquares<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<long?, long?>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector [Func](#)<TSource, [Tuple](#)<[long](#)?>, [long](#)?>>

A transform function to apply to each element.

Returns

[LeastSquares](#)?

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquares<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<float?, float?>>)

Computes the LeastSquares of a sequence of Tuple{float?, float?} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares? LeastSquares<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<float?, float?>> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the LeastSquares of.

selector `Func<TSource, Tuple<float?, float?>>`

A transform function to apply to each element.

Returns

[LeastSquares?](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

LeastSquares<TSource>(IEnumerable<TSource>, Func<TSource, Tuple<float, float>>)

Computes the LeastSquares of a sequence of Tuple{float, float} values that are obtained by invoking a transform function on each element of the input sequence.

```
public static LeastSquares LeastSquares<TSource>(this IEnumerable<TSource> source,  
Func<TSource, Tuple<float, float>> selector)
```

Parameters

source `IEnumerable<TSource>`

A sequence of values to calculate the LeastSquares of.

selector `Func<TSource, Tuple<float, float>>`

A transform function to apply to each element.

Returns

[LeastSquares](#)

The LeastSquares of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Median(IEnumerable<decimal>)

Computes the Median of a sequence of decimal values.

```
public static decimal Median(this IEnumerable<decimal> source)
```

Parameters

source [IEnumerable<decimal>](#)

A sequence of decimal values to calculate the Median of.

Returns

[decimal](#)

The Median of the sequence of values.

Median(IEnumerable<double>)

Computes the Median of a sequence of double values.

```
public static double Median(this IEnumerable<double> source)
```

Parameters

source [IEnumerable<double>](#)

A sequence of double values to calculate the Median of.

Returns

[double](#)

The Median of the sequence of values.

Median(IEnumerable<int>)

Computes the Median of a sequence of int values.

```
public static double Median(this IEnumerable<int> source)
```

Parameters

source [IEnumerable<int>](#)

A sequence of int values to calculate the Median of.

Returns

[double](#)

The Median of the sequence of values.

Median(IEnumerable<long>)

Computes the Median of a sequence of long values.

```
public static double Median(this IEnumerable<long> source)
```

Parameters

source [IEnumerable<long>](#)

A sequence of long values to calculate the Median of.

Returns

[double](#)

The Median of the sequence of values.

Median(IEnumerable<decimal?>)

Computes the Median of a sequence of nullable decimal values, or null if the source sequence is empty or contains only values that are null.

```
public static decimal? Median(this IEnumerable<decimal?> source)
```

Parameters

source [IEnumerable](#)<[decimal](#)?>

A sequence of nullable decimal values to calculate the Median of.

Returns

[decimal](#)?>

The Median of the sequence of values.

Median(IEnumerable<double?>)

Computes the Median of a sequence of nullable double values, or null if the source sequence is empty or contains only values that are null.

```
public static double? Median(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable](#)<[double](#)?>

A sequence of nullable double values to calculate the Median of.

Returns

[double](#)?>

The Median of the sequence of values.

Median(IEnumerable<int?>)

Computes the Median of a sequence of mullable int values, or null if the source sequence is empty or contains only values that are null.

```
public static double? Median(this IEnumerable<int?> source)
```

Parameters

source [IEnumerable](#)<[int](#)?>

A sequence of nullable int values to calculate the Median of.

Returns

[double](#)?
The Median of the sequence of values.

Median(IEnumerable<long?>)

Computes the Median of a sequence of mullable long values, or null if the source sequence is empty or contains only values that are null.

```
public static double? Median(this IEnumerable<long?> source)
```

Parameters

source [IEnumerable](#)<[long](#)?>

A sequence of nullable long values to calculate the Median of.

Returns

[double](#)?
The Median of the sequence of values.

Median(IEnumerable<float?>)

Computes the Median of a sequence of mullable float values, or null if the source sequence is empty or contains only values that are null.

```
public static float? Median(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

A sequence of nullable float values to calculate the Median of.

Returns

[float?](#)

The Median of the sequence of values.

Median(IEnumerable<float>)

Computes the Median of a sequence of float values.

```
public static float Median(this IEnumerable<float> source)
```

Parameters

source [IEnumerable<float>](#)

A sequence of float values to calculate the Median of.

Returns

[float](#)

The Median of the sequence of values.

Median<TSource>(IEnumerable<TSource>, Func<TSource, decimal>)

Computes the Median of a sequence of decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal Median<TSource>(this IEnumerable<TSource> source, Func<TSource, decimal> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Median of.

selector [Func](#)<TSource, decimal>

A transform function to apply to each element.

Returns

[decimal](#)

The Median of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Median<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the Median of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Median<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Median of.

selector [Func](#)<TSource, [double](#)>

A transform function to apply to each element.

Returns

[double](#)

The Median of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Median<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the Median of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Median<TSource>(this IEnumerable<TSource> source, Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Median of.

selector [Func](#)<TSource, [int](#)>

A transform function to apply to each element.

Returns

[double](#)

The Median of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Median<TSource>(IEnumerable<TSource>, Func<TSource, long>)

Computes the Median of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Median<TSource>(this IEnumerable<TSource> source, Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Median of.

selector [Func](#)<TSource, long>

A transform function to apply to each element.

Returns

[double](#)

The Median of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Median<TSource>(IEnumerable<TSource>, Func<TSource, decimal?>)

Computes the Median of a sequence of nullable decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal? Median<TSource>(this IEnumerable<TSource> source, Func<TSource, decimal?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Median of.

selector [Func](#)<TSource, decimal?>

A transform function to apply to each element.

Returns

[decimal](#)?

The Median of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Median<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the Median of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Median<TSource>(this IEnumerable<TSource> source, Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Median of.

selector `Func<TSource, double?>`

A transform function to apply to each element.

Returns

`double?`

The Median of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Median<TSource>(IEnumerable<TSource>, Func<TSource, int?>)

Computes the Median of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Median<TSource>(this IEnumerable<TSource> source, Func<TSource, int?> selector)
```

Parameters

source `IEnumerable<TSource>`

A sequence of values to calculate the Median of.

selector `Func<TSource, int?>`

A transform function to apply to each element.

Returns

`double?`

The Median of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Median<TSource>(IEnumerable<TSource>, Func<TSource, long?>)

Computes the Median of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Median<TSource>(this IEnumerable<TSource> source, Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Median of.

selector [Func](#)<TSource, long?>

A transform function to apply to each element.

Returns

[double](#)?

The Median of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Median<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the Median of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? Median<TSource>(this IEnumerable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Median of.

selector [Func](#)<TSource, float?>

A transform function to apply to each element.

Returns

[float](#)?

The Median of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Median<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the Median of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float Median<TSource>(this IEnumerable<TSource> source, Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Median of.

selector [Func](#)<TSource, [float](#)>

A transform function to apply to each element.

Returns

[float](#)

The Median of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

MinMax(IEnumerable<decimal>)

Computes the MinMax of a sequence of decimal values.

```
public static Range<decimal> MinMax(this IEnumerable<decimal> source)
```

Parameters

source [IEnumerable](#)<[decimal](#)>

The sequence of elements.

Returns

[Range](#)<[decimal](#)>

The MinMax.

MinMax(IEnumerable<double>)

Computes the MinMax of a sequence of double values.

```
public static Range<double> MinMax(this IEnumerable<double> source)
```

Parameters

source [IEnumerable<double>](#)

The sequence of elements.

Returns

[Range<double>](#)

The MinMax.

MinMax(IEnumerable<Int128>)

Computes the MinMax of a sequence of Int128 values.

```
public static Range<Int128> MinMax(this IEnumerable<Int128> source)
```

Parameters

source [IEnumerable<Int128>](#)

The sequence of elements.

Returns

[Range<Int128>](#)

The MinMax.

MinMax(IEnumerable<int>)

Computes the MinMax of a sequence of int values.

```
public static Range<int> MinMax(this IEnumerable<int> source)
```

Parameters

`source` [IEnumerable<int>](#)

The sequence of elements.

Returns

[Range<int>](#)

The MinMax.

MinMax(IEnumerable<long>)

Computes the MinMax of a sequence of long values.

```
public static Range<long> MinMax(this IEnumerable<long> source)
```

Parameters

`source` [IEnumerable<long>](#)

The sequence of elements.

Returns

[Range<long>](#)

The MinMax.

MinMax(IEnumerable<decimal?>)

Computes the MinMax of a sequence of nullable decimal values.

```
public static Range<decimal>? MinMax(this IEnumerable<decimal?> source)
```

Parameters

`source` [IEnumerable<decimal?>](#)

The sequence of elements.

Returns

[Range<decimal>?](#)

The MinMax.

MinMax(IEnumerable<double?>)

Computes the MinMax of a sequence of nullable double values.

```
public static Range<double>? MinMax(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable<double?>](#)

The sequence of elements.

Returns

[Range<double>?](#)

The MinMax.

MinMax(IEnumerable<Int128?>)

Computes the MinMax of a sequence of nullable Int128 values.

```
public static Range<Int128>? MinMax(this IEnumerable<Int128?> source)
```

Parameters

source [IEnumerable<Int128?>](#)

The sequence of elements.

Returns

[Range<Int128>?](#)

The MinMax.

MinMax(IEnumerable<int?>)

Computes the MinMax of a sequence of nullable int values.

```
public static Range<int>? MinMax(this IEnumerable<int?> source)
```

Parameters

source [IEnumerable<int?>](#)

The sequence of elements.

Returns

[Range<int>?](#)

The MinMax.

MinMax(IEnumerable<long?>)

Computes the MinMax of a sequence of nullable long values.

```
public static Range<long>? MinMax(this IEnumerable<long?> source)
```

Parameters

source [IEnumerable<long?>](#)

The sequence of elements.

Returns

[Range<long>?](#)

The MinMax.

MinMax(IEnumerable<float?>)

Computes the MinMax of a sequence of nullable float values.

```
public static Range<float>? MinMax(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

The sequence of elements.

Returns

[Range<float?>?](#)

The MinMax.

MinMax(IEnumerable<float>)

Computes the MinMax of a sequence of float values.

```
public static Range<float> MinMax(this IEnumerable<float> source)
```

Parameters

source [IEnumerable<float>](#)

The sequence of elements.

Returns

[Range<float?>](#)

The MinMax.

MinMaxNaN(IEnumerable<double>)

Computes the MinMax of a sequence of double values.

```
public static Range<double> MinMaxNaN(this IEnumerable<double> source)
```

Parameters

source [IEnumerable<double>](#)

The sequence of elements.

Returns

[Range<double>](#)

The MinMax.

MinMaxNaN(IEnumerable<double?>)

Computes the MinMax of a sequence of nullable double values.

```
public static Range<double>? MinMaxNaN(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable<double?>](#)

The sequence of elements.

Returns

[Range<double>?](#)

The MinMax.

MinMaxNaN(IEnumerable<float?>)

Computes the MinMax of a sequence of nullable float values.

```
public static Range<float>? MinMaxNaN(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

The sequence of elements.

Returns

[Range<float>?](#)

The MinMax.

MinMaxNaN(IEnumerable<float>)

Computes the MinMax of a sequence of float values.

```
public static Range<float> MinMaxNaN(this IEnumerable<float> source)
```

Parameters

source [IEnumerable<float>](#)

The sequence of elements.

Returns

[Range<float>](#)

The MinMax.

MinMaxNaN<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the MinMax of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<double> MinMaxNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

`source` [IEnumerable](#)<TSource>

The sequence of elements.

`selector` [Func](#)<TSource, [double](#)>

A transform function to apply to each element.

Returns

[Range](#)<[double](#)>

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMaxNaN<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the MinMax of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<double>? MinMaxNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, double?> selector)
```

Parameters

`source` [IEnumerable](#)<TSource>

The sequence of elements.

`selector` [Func](#)<TSource, [double](#)?>

A transform function to apply to each element.

Returns

[Range<double>?](#)

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMaxNaN<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the MinMax of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<float>? MinMaxNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, float?>

A transform function to apply to each element.

Returns

[Range<float>?](#)

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMaxNaN<TSource>(IEnumarable<TSource>, Func<TSource, float>)

Computes the MinMax of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<float> MinMaxNaN<TSource>(this IEnumarable<TSource> source,  
Func<TSource, float> selector)
```

Parameters

source [IEnumarable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, float>

A transform function to apply to each element.

Returns

[Range](#)<float>

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMax<TSource>(IEnumarable<TSource>, Func<TSource, decimal>)

Computes the MinMax of a sequence of decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<decimal> MinMax<TSource>(this IEnumarable<TSource> source, Func<TSource,  
decimal> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [decimal](#)>

A transform function to apply to each element.

Returns

[Range](#)<[decimal](#)>

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMax<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the MinMax of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<double> MinMax<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)>

A transform function to apply to each element.

Returns

[Range<double>](#)

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMax<TSource>(IEnumerable<TSource>, Func<TSource, Int128>)

Computes the MinMax of a sequence of Int128 values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<Int128> MinMax<TSource>(this IEnumerable<TSource> source, Func<TSource, Int128> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, Int128>

A transform function to apply to each element.

Returns

[Range<Int128>](#)

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMax<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the MinMax of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<int> MinMax<TSource>(this IEnumerable<TSource> source, Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [int](#)>

A transform function to apply to each element.

Returns

[Range](#)<[int](#)>

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMax<TSource>(IEnumerable<TSource>, Func<TSource, long>)

Computes the MinMax of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<long> MinMax<TSource>(this IEnumerable<TSource> source, Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [long](#)>

A transform function to apply to each element.

Returns

[Range](#)<[long](#)>

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMax<TSource>(IEnumerable<TSource>, Func<TSource, decimal?>)

Computes the MinMax of a sequence of nullable decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<decimal>? MinMax<TSource>(this IEnumerable<TSource> source,  
Func<TSource, decimal?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [decimal](#)?>

A transform function to apply to each element.

Returns

[Range<decimal>?](#)

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMax<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the MinMax of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<double>? MinMax<TSource>(this IEnumerable<TSource> source, Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)?>

A transform function to apply to each element.

Returns

[Range<double>?](#)

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMax<TSource>(IEnumerable<TSource>, Func<TSource, Int128?>)

Computes the MinMax of a sequence of nullable Int128 values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<Int128>? MinMax<TSource>(this IEnumerable<TSource> source, Func<TSource, Int128?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [Int128](#)?>

A transform function to apply to each element.

Returns

[Range](#)<[Int128](#)>?

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMax<TSource>(IEnumerable<TSource>, Func<TSource, int?>)

Computes the MinMax of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<int>? MinMax<TSource>(this IEnumerable<TSource> source, Func<TSource, int?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [int](#)?>

A transform function to apply to each element.

Returns

[Range](#)<[int](#)>?

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMax<TSource>(IEnumerable<TSource>, Func<TSource, long?>)

Computes the MinMax of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<long>? MinMax<TSource>(this IEnumerable<TSource> source, Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [long](#)?>

A transform function to apply to each element.

Returns

[Range<long>?](#)

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMax<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the MinMax of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<float>? MinMax<TSource>(this IEnumerable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, float?>

A transform function to apply to each element.

Returns

[Range<float>?](#)

The MinMax.

Type Parameters

TSource

The type of the elements of source.

MinMax<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the MinMax of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Range<float> MinMax<TSource>(this IEnumerable<TSource> source, Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, float>

A transform function to apply to each element.

Returns

[Range](#)<float>

The MinMax.

Type Parameters

TSource

The type of the elements of source.

Mode<T>(IEnumerable<T?>)

Calculates the mode of a sequences of elements

```
public static T? Mode<T>(this IEnumerable<T?> source) where T : struct
```

Parameters

source [IEnumerable](#)<T?>

A sequence of values to calculate the Mode of.

Returns

T?

The mode of a sequence of values

Type Parameters

T

The type of the elements in the source sequence

Mode<T>(IEnumerable<T>)

Calculates the Mode of a sequences of elements

```
public static T? Mode<T>(this IEnumerable<T> source) where T : struct
```

Parameters

source [IEnumerable](#)<T>

A sequence of values to calculate the Mode of.

Returns

T?

The Mode of a sequence of values

Type Parameters

T

The type of the elements in the source sequence

Mode<TSource, TMode>(IEnumerable<TSource>,

Func<TSource, TMode?>)

Computes the Mode of a sequence of values that are obtained by invoking a transform function on each element of the input sequence.

```
public static TMode? Mode<TSource, TMode>(this IEnumerable<TSource> source, Func<TSource, TMode?> selector) where TMode : struct
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Mode of.

selector [Func](#)<TSource, TMode?>

A transform function to apply to each element.

Returns

TMode?

The Mode of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

TMode

The type of the Mode

Mode<TSource, TMode>(IEnumerable<TSource>, Func<TSource, TMode>)

Computes the Mode of a sequence of values that are obtained by invoking a transform function on each element of the input sequence.

```
public static TMode? Mode<TSource, TMode>(this IEnumerable<TSource> source, Func<TSource, TMode> selector) where TMode : struct
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the Mode of.

selector [Func](#)<TSource, TMode>

A transform function to apply to each element.

Returns

TMode?

The Mode of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

TMode

The type of the Mode

Modes<T>(IEnumerable<T?>)

Calculates the modes of a sequences of elements

```
public static IEnumerable<T> Modes<T>(this IEnumerable<T?> source) where T : struct
```

Parameters

source [IEnumerable](#)<T?>

A sequence of values to calculate the Modes of.

Returns

[IEnumerable](#)<T>

The modes of a sequence of values

Type Parameters

T

The type of the elements in the source sequence

Modes<T>(IEnumerable<T>)

Calculates the modes of a sequences of nullable elements

```
public static IEnumerable<T> Modes<T>(this IEnumerable<T> source) where T : struct
```

Parameters

source [IEnumerable](#)<T>

A sequence of values to calculate the Modes of.

Returns

[IEnumerable](#)<T>

The modes of a sequence of values

Type Parameters

T

The type of the elements in the source sequence

Pearson(IEnumerable<decimal>, IEnumerable<decimal>)

Computes the Pearson of two sequences of decimal values.

```
public static decimal Pearson(this IEnumerable<decimal> source, IEnumerable<decimal> other)
```

Parameters

source [IEnumerable<decimal>](#)

The first sequence of decimal values to calculate the Pearson of.

other [IEnumerable<decimal>](#)

The second sequence of decimal values to calculate the Pearson of.

Returns

[decimal](#)

The Pearson value of two sequences.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson(IEnumerable<double>, IEnumerable<double>)

Computes the Pearson of two sequences of double values.

```
public static double Pearson(this IEnumerable<double> source, IEnumerable<double> other)
```

Parameters

source [IEnumerable<double>](#)

The first sequence of double values to calculate the Pearson of.

other [IEnumerable<double>](#)

The second sequence of double values to calculate the Pearson of.

Returns

double

The Pearson value of two sequences.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson(IEnumerable<int>, IEnumerable<int>)

Computes the Pearson of two sequences of int values.

```
public static double Pearson(this IEnumerable<int> source, IEnumerable<int> other)
```

Parameters

source IEnumerable<int>

The first sequence of int values to calculate the Pearson of.

other IEnumerable<int>

The second sequence of int values to calculate the Pearson of.

Returns

double

The Pearson value of two sequences.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson(IEnumerable<long>, IEnumerable<long>)

Computes the Pearson of two sequences of long values.

```
public static double Pearson(this IEnumerable<long> source, IEnumerable<long> other)
```

Parameters

source [IEnumerable<long>](#)

The first sequence of long values to calculate the Pearson of.

other [IEnumerable<long>](#)

The second sequence of long values to calculate the Pearson of.

Returns

[double](#)

The Pearson value of two sequences.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson(IEnumerable<decimal?>, IEnumerable<decimal?>)

Computes the Pearson of two sequences of nullable decimal values.

```
public static decimal? Pearson(this IEnumerable<decimal?> source, IEnumerable<decimal?> other)
```

Parameters

source [IEnumerable<decimal?>](#)

The first sequence of nullable decimal values to calculate the Pearson of.

other [IEnumerable<decimal?>](#)

The second sequence of nullable decimal values to calculate the Pearson of.

Returns

[decimal](#)?

The Pearson value of two sequences.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson(IEnumerable<double?>, IEnumerable<double?>)

Computes the Pearson of two sequences of nullable double values.

```
public static double? Pearson(this IEnumerable<double?> source, IEnumerable<double?> other)
```

Parameters

source [IEnumerable](#)<[double](#)?>

The first sequence of nullable double values to calculate the Pearson of.

other [IEnumerable](#)<[double](#)?>

The second sequence of nullable double values to calculate the Pearson of.

Returns

[double](#)?

The Pearson value of two sequences.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson(IEnumerable<int?>, IEnumerable<int?>)

Computes the Pearson of two sequences of nullable int values.

```
public static double? Pearson(this IEnumerable<int?> source, IEnumerable<int?> other)
```

Parameters

source [IEnumerable<int?>](#)

The first sequence of nullable int values to calculate the Pearson of.

other [IEnumerable<int?>](#)

The second sequence of nullable int values to calculate the Pearson of.

Returns

[double?](#)

The Pearson value of two sequences.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson(IEnumerable<long?>, IEnumerable<long?>)

Computes the Pearson of two sequences of nullable long values.

```
public static double? Pearson(this IEnumerable<long?> source, IEnumerable<long?> other)
```

Parameters

source [IEnumerable<long?>](#)

The first sequence of nullable long values to calculate the Pearson of.

other [IEnumerable<long?>](#)

The second sequence of nullable long values to calculate the Pearson of.

Returns

[double](#)?

The Pearson value of two sequences.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson(IEnumerable<float?>, IEnumerable<float?>)

Computes the Pearson of two sequences of nullable float values.

```
public static float? Pearson(this IEnumerable<float?> source, IEnumerable<float?> other)
```

Parameters

source [IEnumerable](#)<[float](#)?>

The first sequence of nullable float values to calculate the Pearson of.

other [IEnumerable](#)<[float](#)?>

The second sequence of nullable float values to calculate the Pearson of.

Returns

[float](#)?

The Pearson value of two sequences.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson(IEnumerable<float>, IEnumerable<float>)

Computes the Pearson of two sequences of float values.

```
public static float Pearson(this IEnumerable<float> source, IEnumerable<float> other)
```

Parameters

source [IEnumerable<float>](#)

The first sequence of float values to calculate the Pearson of.

other [IEnumerable<float>](#)

The second sequence of float values to calculate the Pearson of.

Returns

[float](#)

The Pearson value of two sequences.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson(IEnumerable<Tuple<decimal, decimal>>)

Computes the Pearson of the Item values of a sequence of Tuple{decimal, decimal} values.

```
public static decimal Pearson(this IEnumerable<Tuple<decimal, decimal>> source)
```

Parameters

source [IEnumerable<Tuple<decimal, decimal>>](#)

The type of the Tuple's Items.

Returns

[decimal](#)

The Pearson value.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson(IEnumerable<Tuple<double, double>>)

Computes the Pearson of the Item values of a sequence of Tuple{double, double} values.

```
public static double Pearson(this IEnumerable<Tuple<double, double>> source)
```

Parameters

source [IEnumerable](#)<[Tuple](#)<[double](#), [double](#)>>

The type of the Tuple's Items.

Returns

[double](#)

The Pearson value.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson(IEnumerable<Tuple<int, int>>)

Computes the Pearson of the Item values of a sequence of Tuple{int, int} values.

```
public static double Pearson(this IEnumerable<Tuple<int, int>> source)
```

Parameters

source [IEnumerable](#)<[Tuple](#)<[int](#), [int](#)>>

The type of the Tuple's Items.

Returns

double

The Pearson value.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson(IEnumerable<Tuple<long, long>>)

Computes the Pearson of the Item values of a sequence of Tuple{long, long} values.

```
public static double Pearson(this IEnumerable<Tuple<long, long>> source)
```

Parameters

source IEnumerable<Tuple<long, long>>

The type of the Tuple's Items.

Returns

double

The Pearson value.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson(IEnumerable<Tuple<float, float>>)

Computes the Pearson of the Item values of a sequence of Tuple{float, float} values.

```
public static float Pearson(this IEnumerable<Tuple<float, float>> source)
```

Parameters

source [IEnumerable<Tuple<float, float>>](#)

The type of the Tuple's Items.

Returns

[float](#)

The Pearson value.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

PearsonNaN(IEnumerable<double>, IEnumerable<double>)

Computes the Pearson of two sequences of double values.

```
public static double PearsonNaN(this IEnumerable<double> source, IEnumerable<double> other)
```

Parameters

source [IEnumerable<double>](#)

The first sequence of double values to calculate the Pearson of.

other [IEnumerable<double>](#)

The second sequence of double values to calculate the Pearson of.

Returns

[double](#)

The Pearson value of two sequences.

PearsonNaN(IEnumerable<int>, IEnumerable<int>)

Computes the Pearson of two sequences of int values.

```
public static double PearsonNaN(this IEnumerable<int> source, IEnumerable<int> other)
```

Parameters

source [IEnumerable<int>](#)

The first sequence of int values to calculate the Pearson of.

other [IEnumerable<int>](#)

The second sequence of int values to calculate the Pearson of.

Returns

[double](#)

The Pearson value of two sequences.

PearsonNaN(IEnumerable<long>, IEnumerable<long>)

Computes the Pearson of two sequences of long values.

```
public static double PearsonNaN(this IEnumerable<long> source, IEnumerable<long> other)
```

Parameters

source [IEnumerable<long>](#)

The first sequence of long values to calculate the Pearson of.

other [IEnumerable<long>](#)

The second sequence of long values to calculate the Pearson of.

Returns

[double](#)

The Pearson value of two sequences.

PearsonNaN(IEnumerable<double?>, IEnumerable<double?>)

Computes the Pearson of two sequences of nullable double values.

```
public static double? PearsonNaN(this IEnumerable<double?> source, IEnumerable<double?> other)
```

Parameters

source [IEnumerable<double?>](#)

The first sequence of nullable double values to calculate the Pearson of.

other [IEnumerable<double?>](#)

The second sequence of nullable double values to calculate the Pearson of.

Returns

[double?](#)

The Pearson value of two sequences.

PearsonNaN(IEnumerable<int?>, IEnumerable<int?>)

Computes the Pearson of two sequences of nullable int values.

```
public static double? PearsonNaN(this IEnumerable<int?> source, IEnumerable<int?> other)
```

Parameters

source [IEnumerable<int?>](#)

The first sequence of nullable int values to calculate the Pearson of.

other [IEnumerable<int?>](#)

The second sequence of nullable int values to calculate the Pearson of.

Returns

double?

The Pearson value of two sequences.

PearsonNaN(IEnumerable<long?>, IEnumerable<long?>)

Computes the Pearson of two sequences of nullable long values.

```
public static double? PearsonNaN(this IEnumerable<long?> source, IEnumerable<long?> other)
```

Parameters

source IEnumerable<long?>

The first sequence of nullable long values to calculate the Pearson of.

other IEnumerable<long?>

The second sequence of nullable long values to calculate the Pearson of.

Returns

double?

The Pearson value of two sequences.

PearsonNaN(IEnumerable<float?>, IEnumerable<float?>)

Computes the Pearson of two sequences of nullable float values.

```
public static float? PearsonNaN(this IEnumerable<float?> source, IEnumerable<float?> other)
```

Parameters

source IEnumerable<float?>

The first sequence of nullable float values to calculate the Pearson of.

other IEnumerable<float?>

The second sequence of nullable float values to calculate the Pearson of.

Returns

[float](#)?

The Pearson value of two sequences.

PearsonNaN(IEnumerable<float>, IEnumerable<float>)

Computes the Pearson of two sequences of float values.

```
public static float PearsonNaN(this IEnumerable<float> source, IEnumerable<float> other)
```

Parameters

source [IEnumerable](#)<[float](#)>

The first sequence of float values to calculate the Pearson of.

other [IEnumerable](#)<[float](#)>

The second sequence of float values to calculate the Pearson of.

Returns

[float](#)

The Pearson value of two sequences.

PearsonNaN(IEnumerable<Tuple<double, double>>)

Computes the Pearson of the Item values of a sequence of Tuple{double, double} values.

```
public static double PearsonNaN(this IEnumerable<Tuple<double, double>> source)
```

Parameters

source [IEnumerable](#)<[Tuple](#)<[double](#), [double](#)>>

The type of the Tuple's Items.

Returns

[double](#)

The Pearson value.

PearsonNaN(IEnumerable<Tuple<int, int>>)

Computes the Pearson of the Item values of a sequence of Tuple{int, int} values.

```
public static double PearsonNaN(this IEnumerable<Tuple<int, int>> source)
```

Parameters

source [IEnumerable](#)<Tuple<int, int>>

The type of the Tuple's Items.

Returns

[double](#)

The Pearson value.

PearsonNaN(IEnumerable<Tuple<long, long>>)

Computes the Pearson of the Item values of a sequence of Tuple{long, long} values.

```
public static double PearsonNaN(this IEnumerable<Tuple<long, long>> source)
```

Parameters

source [IEnumerable](#)<Tuple<long, long>>

The type of the Tuple's Items.

Returns

[double](#)

The Pearson value.

PearsonNaN(IEnumerable<Tuple<float, float>>)

Computes the Pearson of the Item values of a sequence of Tuple{float, float} values.

```
public static float PearsonNaN(this IEnumerable<Tuple<float, float>> source)
```

Parameters

source [IEnumerable](#)<[Tuple](#)<float, float>>

The type of the Tuple's Items.

Returns

[float](#)

The Pearson value.

PearsonNaN<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource, double>)

Computes the Pearson of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double PearsonNaN<TSource>(this IEnumerable<TSource> source,  
IEnumerable<TSource> other, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of double values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of double values to calculate the Pearson of.

selector `Func<TSource, double>`

A transform function to apply to each element.

Returns

`double`

The Pearson of the sequence of values.

Type Parameters

`TSource`

The type of the elements of source.

`PearsonNaN<TSource>(IEnumerable<TSource>,
IEnumerable<TSource>, Func<TSource, int>)`

Computes the Pearson of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double PearsonNaN<TSource>(this IEnumerable<TSource> source,  
IEnumerable<TSource> other, Func<TSource, int> selector)
```

Parameters

source `IEnumerable<TSource>`

The first sequence of int values to calculate the Pearson of.

other `IEnumerable<TSource>`

The second sequence of int values to calculate the Pearson of.

selector `Func<TSource, int>`

A transform function to apply to each element.

Returns

[double](#)

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

PearsonNaN<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource, long>)

Computes the Pearson of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double PearsonNaN<TSource>(this IEnumerable<TSource> source,  
    IEnumerable<TSource> other, Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of long values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of long values to calculate the Pearson of.

selector [Func](#)<TSource, long>

A transform function to apply to each element.

Returns

[double](#)

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

PearsonNaN<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource, double?>)

Computes the Pearson of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? PearsonNaN<TSource>(this IEnumerable<TSource> source,  
IEnumerable<TSource> other, Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of double values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of double values to calculate the Pearson of.

selector [Func](#)<TSource, double?>

A transform function to apply to each element.

Returns

[double](#)?

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

PearsonNaN<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource, int?>)

Computes the Pearson of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? PearsonNaN<TSource>(this IEnumerable<TSource> source,  
IEnumerable<TSource> other, Func<TSource, int?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of int values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of int values to calculate the Pearson of.

selector [Func](#)<TSource, int?>

A transform function to apply to each element.

Returns

[double](#)?

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

**PearsonNaN<TSource>(IEnumerable<TSource>,
IEnumerable<TSource>, Func<TSource, long?>)**

Computes the Pearson of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? PearsonNaN<TSource>(this IEnumerable<TSource> source,  
IEnumerable<TSource> other, Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of long values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of long values to calculate the Pearson of.

selector [Func](#)<TSource, [long](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

**PearsonNaN<TSource>(IEnumerable<TSource>,
IEnumerable<TSource>, Func<TSource, float?>)**

Computes the Pearson of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? PearsonNaN<TSource>(this IEnumerable<TSource> source,  
IEnumerable<TSource> other, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of float values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of float values to calculate the Pearson of.

selector `Func<TSource, float>`

A transform function to apply to each element.

Returns

`float?`

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

**PearsonNaN<TSource>(IEnumerable<TSource>,
IEnumerable<TSource>, Func<TSource, float>)**

Computes the Pearson of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float PearsonNaN<TSource>(this IEnumerable<TSource> source,  
IEnumerable<TSource> other, Func<TSource, float> selector)
```

Parameters

source `IEnumerable<TSource>`

The first sequence of float values to calculate the Pearson of.

other `IEnumerable<TSource>`

The second sequence of float values to calculate the Pearson of.

selector `Func<TSource, float>`

A transform function to apply to each element.

Returns

[float](#)

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Pearson<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource, decimal>)

Computes the Pearson of a sequence of nullable decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal Pearson<TSource>(this IEnumerable<TSource> source,  
IEnumerable<TSource> other, Func<TSource, decimal> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of decimal values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of decimal values to calculate the Pearson of.

selector [Func](#)<TSource, decimal>

A transform function to apply to each element.

Returns

[decimal](#)

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource, double>)

Computes the Pearson of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Pearson<TSource>(this IEnumerable<TSource> source, IEnumerable<TSource> other, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of double values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of double values to calculate the Pearson of.

selector [Func](#)<TSource, double>

A transform function to apply to each element.

Returns

[double](#)

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource, int>)

Computes the Pearson of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Pearson<TSource>(this IEnumerable<TSource> source, IEnumerable<TSource> other, Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of int values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of int values to calculate the Pearson of.

selector [Func](#)<TSource, int>

A transform function to apply to each element.

Returns

[double](#)

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource, long>)

Computes the Pearson of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Pearson<TSource>(this IEnumerable<TSource> source, IEnumerable<TSource> other, Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of long values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of long values to calculate the Pearson of.

selector [Func](#)<TSource, long>

A transform function to apply to each element.

Returns

[double](#)

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource, decimal?>)

Computes the Pearson of a sequence of nullable decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal? Pearson<TSource>(this IEnumerable<TSource> source,  
IEnumerable<TSource> other, Func<TSource, decimal?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of decimal values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of decimal values to calculate the Pearson of.

selector [Func](#)<TSource, decimal?>

A transform function to apply to each element.

Returns

[decimal](#)?

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource, double?>)

Computes the Pearson of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Pearson<TSource>(this IEnumerable<TSource> source,  
IEnumerable<TSource> other, Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of double values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of double values to calculate the Pearson of.

selector [Func](#)<TSource, double?>

A transform function to apply to each element.

Returns

[double](#)?

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource, int?>)

Computes the Pearson of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Pearson<TSource>(this IEnumerable<TSource> source,  
IEnumerable<TSource> other, Func<TSource, int?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of int values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of int values to calculate the Pearson of.

selector [Func](#)<TSource, int?>

A transform function to apply to each element.

Returns

[double](#)?

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource, long?>)

Computes the Pearson of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Pearson<TSource>(this IEnumerable<TSource> source,  
IEnumerable<TSource> other, Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of long values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of long values to calculate the Pearson of.

selector [Func](#)<TSource, long?>

A transform function to apply to each element.

Returns

[double](#)?

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource, float?>)

Computes the Pearson of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? Pearson<TSource>(this IEnumerable<TSource> source, IEnumerable<TSource> other, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of float values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of float values to calculate the Pearson of.

selector [Func](#)<TSource, float?>

A transform function to apply to each element.

Returns

[float](#)?

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Pearson<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, Func<TSource, float>)

Computes the Pearson of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float Pearson<TSource>(this IEnumerable<TSource> source, IEnumerable<TSource> other, Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The first sequence of float values to calculate the Pearson of.

other [IEnumerable](#)<TSource>

The second sequence of float values to calculate the Pearson of.

selector [Func](#)<TSource, float>

A transform function to apply to each element.

Returns

[float](#)

The Pearson of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

Range(IEnumerable<decimal>)

Computes the Range of a sequence of decimal values.

```
public static decimal Range(this IEnumerable<decimal> source)
```

Parameters

source [IEnumerable](#)<[decimal](#)>

The sequence of elements.

Returns

[decimal](#)

The Range.

Range(IEnumerable<double>)

Computes the Range of a sequence of double values.

```
public static double Range(this IEnumerable<double> source)
```

Parameters

source [IEnumerable](#)<[double](#)>

The sequence of elements.

Returns

[double](#)

The Range.

Range(IEnumerable<Int128>)

Computes the Range of a sequence of Int128 values.

```
public static Int128 Range(this IEnumerable<Int128> source)
```

Parameters

source [IEnumerable](#)<[Int128](#)>

The sequence of elements.

Returns

[Int128](#)

The Range.

Range(IEnumerable<int>)

Computes the Range of a sequence of int values.

```
public static int Range(this IEnumerable<int> source)
```

Parameters

source [IEnumerable](#)<[int](#)>

The sequence of elements.

Returns

[int](#)

The Range.

Range(IEnumerable<long>)

Computes the Range of a sequence of long values.

```
public static long Range(this IEnumerable<long> source)
```

Parameters

`source` [IEnumerable<long>](#)

The sequence of elements.

Returns

[long](#)

The Range.

Range(IEnumerable<decimal?>)

Computes the Range of a sequence of nullable decimal values.

```
public static decimal? Range(this IEnumerable<decimal?> source)
```

Parameters

`source` [IEnumerable<decimal?>](#)

The sequence of elements.

Returns

[decimal?](#)

The Range.

Range(IEnumerable<double?>)

Computes the Range of a sequence of nullable double values.

```
public static double? Range(this IEnumerable<double?> source)
```

Parameters

`source` [IEnumerable<double?>](#)

The sequence of elements.

Returns

[double](#)?

The Range.

Range(IEnumerable<Int128?>)

Computes the Range of a sequence of nullable Int128 values.

```
public static Int128? Range(this IEnumerable<Int128?> source)
```

Parameters

source [IEnumerable](#)<[Int128](#)?>

The sequence of elements.

Returns

[Int128](#)?

The Range.

Range(IEnumerable<int?>)

Computes the Range of a sequence of nullable int values.

```
public static int? Range(this IEnumerable<int?> source)
```

Parameters

source [IEnumerable](#)<[int](#)?>

The sequence of elements.

Returns

[int](#)?

The Range.

Range(IEnumerable<long?>)

Computes the Range of a sequence of nullable long values.

```
public static long? Range(this IEnumerable<long?> source)
```

Parameters

source [IEnumerable](#)<[long](#)?>

The sequence of elements.

Returns

[long](#)?

The Range.

Range(IEnumerable<float?>)

Computes the Range of a sequence of nullable float values.

```
public static float? Range(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable](#)<[float](#)?>

The sequence of elements.

Returns

[float](#)?

The Range.

Range(IEnumerable<float>)

Computes the Range of a sequence of float values.

```
public static float Range(this IEnumerable<float> source)
```

Parameters

source [IEnumerable<float>](#)

The sequence of elements.

Returns

[float](#)

The Range.

RangeNaN(IEnumerable<double>)

Computes the Range of a sequence of double values.

```
public static double RangeNaN(this IEnumerable<double> source)
```

Parameters

source [IEnumerable<double>](#)

The sequence of elements.

Returns

[double](#)

The Range.

RangeNaN(IEnumerable<double?>)

Computes the Range of a sequence of nullable double values.

```
public static double? RangeNaN(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable<double?>](#)

The sequence of elements.

Returns

[double?](#)

The Range.

RangeNaN(IEnumerable<float?>)

Computes the Range of a sequence of nullable float values.

```
public static float? RangeNaN(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

The sequence of elements.

Returns

[float?](#)

The Range.

RangeNaN(IEnumerable<float>)

Computes the Range of a sequence of float values.

```
public static float RangeNaN(this IEnumerable<float> source)
```

Parameters

source [IEnumerable](#)<[float](#)>

The sequence of elements.

Returns

[float](#)

The Range.

RangeNaN<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the Range of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double RangeNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)>

A transform function to apply to each element.

Returns

[double](#)

The Range.

Type Parameters

TSource

The type of the elements of source.

RangeNaN<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the Range of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? RangeNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Range.

Type Parameters

TSource

The type of the elements of source.

RangeNaN<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the Range of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? RangeNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [float](#)?>

A transform function to apply to each element.

Returns

[float](#)?

The Range.

Type Parameters

TSource

The type of the elements of source.

RangeNaN<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the Range of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float RangeNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [float](#)>

A transform function to apply to each element.

Returns

[float](#)

The Range.

Type Parameters

TSource

The type of the elements of source.

Range<TSource>(IEnumerable<TSource>, Func<TSource, decimal>)

Computes the Range of a sequence of decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal Range<TSource>(this IEnumerable<TSource> source, Func<TSource, decimal> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [decimal](#)>

A transform function to apply to each element.

Returns

[decimal](#)

The Range.

Type Parameters

TSource

The type of the elements of source.

Range<TSource>(IEnumarable<TSource>, Func<TSource, double>)

Computes the Range of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Range<TSource>(this IEnumarable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumarable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)>

A transform function to apply to each element.

Returns

[double](#)

The Range.

Type Parameters

TSource

The type of the elements of source.

Range<TSource>(IEnumarable<TSource>, Func<TSource, Int128>)

Computes the Range of a sequence of Int128 values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Int128 Range<TSource>(this IEnumarable<TSource> source, Func<TSource, Int128> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [Int128](#)>

A transform function to apply to each element.

Returns

[Int128](#)

The Range.

Type Parameters

TSource

The type of the elements of source.

Range<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the Range of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static int Range<TSource>(this IEnumerable<TSource> source, Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [int](#)>

A transform function to apply to each element.

Returns

[int](#)

The Range.

Type Parameters

TSource

The type of the elements of source.

Range<TSource>(IEnumerable<TSource>, Func<TSource, long>)

Computes the Range of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static long Range<TSource>(this IEnumerable<TSource> source, Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, long>

A transform function to apply to each element.

Returns

[long](#)

The Range.

Type Parameters

TSource

The type of the elements of source.

Range<TSource>(IEnumable<TSource>, Func<TSource, decimal?>)

Computes the Range of a sequence of nullable decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal? Range<TSource>(this IEnumable<TSource> source, Func<TSource, decimal?> selector)
```

Parameters

source [IEnumable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, decimal?>

A transform function to apply to each element.

Returns

[decimal](#)?

The Range.

Type Parameters

TSource

The type of the elements of source.

Range<TSource>(IEnumable<TSource>, Func<TSource, double?>)

Computes the Range of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Range<TSource>(this IEnumable<TSource> source, Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Range.

Type Parameters

TSource

The type of the elements of source.

Range<TSource>(IEnumarable<TSource>, Func<TSource, Int128?>)

Computes the Range of a sequence of nullable Int128 values that are obtained by invoking a transform function on each element of the input sequence.

```
public static Int128? Range<TSource>(this IEnumarable<TSource> source, Func<TSource, Int128?> selector)
```

Parameters

source [IEnumarable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [Int128](#)?>

A transform function to apply to each element.

Returns

[Int128](#)?

The Range.

Type Parameters

TSource

The type of the elements of source.

Range<TSource>(IEnumerable<TSource>, Func<TSource, int?>)

Computes the Range of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static int? Range<TSource>(this IEnumerable<TSource> source, Func<TSource, int?
> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [int](#)?>

A transform function to apply to each element.

Returns

[int](#)?

The Range.

Type Parameters

TSource

The type of the elements of source.

Range<TSource>(IEnumable<TSource>, Func<TSource, long?>)

Computes the Range of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static long? Range<TSource>(this IEnumable<TSource> source, Func<TSource, long?> selector)
```

Parameters

source [IEnumable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, long?>

A transform function to apply to each element.

Returns

[long](#)?

The Range.

Type Parameters

TSource

The type of the elements of source.

Range<TSource>(IEnumable<TSource>, Func<TSource, float?>)

Computes the Range of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? Range<TSource>(this IEnumable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [float](#)?>

A transform function to apply to each element.

Returns

[float](#)?

The Range.

Type Parameters

TSource

The type of the elements of source.

Range<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the Range of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float Range<TSource>(this IEnumerable<TSource> source, Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [float](#)>

A transform function to apply to each element.

Returns

[float](#)

The Range.

Type Parameters

TSource

The type of the elements of source.

RootMeanSquare(IEnumerable<decimal>)

The RootMeanSquare of the sequence of decimal values.

```
public static decimal RootMeanSquare(this IEnumerable<decimal> source)
```

Parameters

source [IEnumerable](#)<[decimal](#)>

A sequence of decimal values to calculate the RootMeanSquare of.

Returns

[decimal](#)

The RootMeanSquare.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare(IEnumerable<double>)

The RootMeanSquare of the sequence of double values.

```
public static double RootMeanSquare(this IEnumerable<double> source)
```

Parameters

`source` [IEnumerable<double>](#)

A sequence of double values to calculate the RootMeanSquare of.

Returns

[double](#)

The RootMeanSquare.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare(IEnumerable<int>)

The RootMeanSquare of the sequence of int values.

```
public static double RootMeanSquare(this IEnumerable<int> source)
```

Parameters

`source` [IEnumerable<int>](#)

A sequence of int values to calculate the RootMeanSquare of.

Returns

[double](#)

The RootMeanSquare.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare(IEnumerable<long>)

The RootMeanSquare of the sequence of long values.

```
public static double RootMeanSquare(this IEnumerable<long> source)
```

Parameters

source [IEnumerable<long>](#)

A sequence of long values to calculate the RootMeanSquare of.

Returns

[double](#)

The RootMeanSquare.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare(IEnumerable<decimal?>)

The RootMeanSquare of the sequence of nullable decimal values, or null if the source sequence is empty or contains only values that are null.

```
public static decimal? RootMeanSquare(this IEnumerable<decimal?> source)
```

Parameters

source [IEnumerable<decimal?>](#)

A sequence of decimal values to calculate the RootMeanSquare of.

Returns

[decimal](#)?

The RootMeanSquare.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare(IEnumerable<double?>)

The RootMeanSquare of the sequence of nullable double values, or null if the source sequence is empty or contains only values that are null.

```
public static double? RootMeanSquare(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable<double?>](#)

A sequence of double values to calculate the RootMeanSquare of.

Returns

[double?](#)

The RootMeanSquare.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare(IEnumerable<int?>)

The RootMeanSquare of the sequence of nullable int values, or null if the source sequence is empty or contains only values that are null.

```
public static double? RootMeanSquare(this IEnumerable<int?> source)
```

Parameters

`source` [IEnumerable<int?>](#)

A sequence of int values to calculate the RootMeanSquare of.

Returns

[double?](#)

The RootMeanSquare.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare(IEnumerable<long?>)

The RootMeanSquare of the sequence of nullable long values, or null if the source sequence is empty or contains only values that are null.

```
public static double? RootMeanSquare(this IEnumerable<long?> source)
```

Parameters

`source` [IEnumerable<long?>](#)

A sequence of long values to calculate the RootMeanSquare of.

Returns

[double?](#)

The RootMeanSquare.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare(IEnumerable<float?>)

The RootMeanSquare of the sequence of nullable float values, or null if the source sequence is empty or contains only values that are null.

```
public static float? RootMeanSquare(this IEnumerable<float?> source)
```

Parameters

source `IEnumerable<float?>`

A sequence of float values to calculate the RootMeanSquare of.

Returns

`float?`

The RootMeanSquare.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare(IEnumerable<float>)

The RootMeanSquare of the sequence of float values.

```
public static float RootMeanSquare(this IEnumerable<float> source)
```

Parameters

source `IEnumerable<float>`

A sequence of float values to calculate the RootMeanSquare of.

Returns

`float?`

The RootMeanSquare.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquareNaN(IEnumerable<double>)

The RootMeanSquare of the sequence of double values.

```
public static double RootMeanSquareNaN(this IEnumerable<double> source)
```

Parameters

source [IEnumerable<double>](#)

A sequence of double values to calculate the RootMeanSquare of.

Returns

[double](#)

The RootMeanSquare.

RootMeanSquareNaN(IEnumerable<int>)

The RootMeanSquare of the sequence of int values.

```
public static double RootMeanSquareNaN(this IEnumerable<int> source)
```

Parameters

source [IEnumerable<int>](#)

A sequence of int values to calculate the RootMeanSquare of.

Returns

double

The RootMeanSquare.

RootMeanSquareNaN(IEnumerable<long>)

The RootMeanSquare of the sequence of long values.

```
public static double RootMeanSquareNaN(this IEnumerable<long> source)
```

Parameters

source IEnumerable<long>

A sequence of long values to calculate the RootMeanSquare of.

Returns

double

The RootMeanSquare.

RootMeanSquareNaN(IEnumerable<double?>)

The RootMeanSquare of the sequence of nullable double values, or null if the source sequence is empty or contains only values that are null.

```
public static double? RootMeanSquareNaN(this IEnumerable<double?> source)
```

Parameters

source IEnumerable<double?>

A sequence of double values to calculate the RootMeanSquare of.

Returns

double?

The RootMeanSquare.

RootMeanSquareNaN(IEnumerable<int?>)

The RootMeanSquare of the sequence of nullable int values, or null if the source sequence is empty or contains only values that are null.

```
public static double? RootMeanSquareNaN(this IEnumerable<int?> source)
```

Parameters

`source` [IEnumerable<int?>](#)

A sequence of int values to calculate the RootMeanSquare of.

Returns

[double?](#)

The RootMeanSquare.

RootMeanSquareNaN(IEnumerable<long?>)

The RootMeanSquare of the sequence of nullable long values, or null if the source sequence is empty or contains only values that are null.

```
public static double? RootMeanSquareNaN(this IEnumerable<long?> source)
```

Parameters

`source` [IEnumerable<long?>](#)

A sequence of long values to calculate the RootMeanSquare of.

Returns

[double?](#)

The RootMeanSquare.

RootMeanSquareNaN(IEnumerable<float?>)

The RootMeanSquare of the sequence of nullable float values, or null if the source sequence is empty or contains only values that are null.

```
public static float? RootMeanSquareNaN(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

A sequence of float values to calculate the RootMeanSquare of.

Returns

[float?](#)

The RootMeanSquare.

RootMeanSquareNaN(IEnumerable<float>)

The RootMeanSquare of the sequence of float values.

```
public static float RootMeanSquareNaN(this IEnumerable<float> source)
```

Parameters

source [IEnumerable<float>](#)

A sequence of float values to calculate the RootMeanSquare of.

Returns

[float](#)

The RootMeanSquare.

RootMeanSquareNaN<TSource>(IEnumerable<TSource>,

Func<TSource, double>()

Computes the RootMeanSquare of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double RootMeanSquareNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, double>

A transform function to apply to each element.

Returns

[double](#)

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

RootMeanSquareNaN<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the RootMeanSquare of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double RootMeanSquareNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, [int](#)>

A transform function to apply to each element.

Returns

[double](#)

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

RootMeanSquareNaN<TSource>(IEnumarable<TSource>, Func<TSource, long>)

Computes the RootMeanSquare of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double RootMeanSquareNaN<TSource>(this IEnumarable<TSource> source,  
Func<TSource, long> selector)
```

Parameters

source [IEnumarable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, [long](#)>

A transform function to apply to each element.

Returns

[double](#)

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

RootMeanSquareNaN<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the RootMeanSquare of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? RootMeanSquareNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, [double](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

RootMeanSquareNaN<TSource>(IEnumerable<TSource>, Func<TSource, int?>)

Computes the RootMeanSquare of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? RootMeanSquareNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, int?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, [int](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

RootMeanSquareNaN<TSource>(IEnumerable<TSource>, Func<TSource, long?>)

Computes the RootMeanSquare of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? RootMeanSquareNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, [long](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

RootMeanSquareNaN<TSource>(IEnumarable<TSource>, Func<TSource, float?>)

Computes the RootMeanSquare of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? RootMeanSquareNaN<TSource>(this IEnumarable<TSource> source,  
Func<TSource, float?> selector)
```

Parameters

source [IEnumarable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, [float](#)?>

A transform function to apply to each element.

Returns

[float](#)?

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

RootMeanSquareNaN<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the RootMeanSquare of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float RootMeanSquareNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, float>

A transform function to apply to each element.

Returns

[float](#)

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

RootMeanSquare<TSource>(IEnumerable<TSource>, Func<TSource, decimal>)

Computes the RootMeanSquare of a sequence of decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal RootMeanSquare<TSource>(this IEnumerable<TSource> source,  
Func<TSource, decimal> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, [decimal](#)>

A transform function to apply to each element.

Returns

[decimal](#)

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the RootMeanSquare of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double RootMeanSquare<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, double>

A transform function to apply to each element.

Returns

[double](#)

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the RootMeanSquare of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double RootMeanSquare<TSource>(this IEnumerable<TSource> source, Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, [int](#)>

A transform function to apply to each element.

Returns

[double](#)

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare<TSource>(IEnumerable<TSource>, Func<TSource, long>)

Computes the RootMeanSquare of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double RootMeanSquare<TSource>(this IEnumerable<TSource> source, Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, [long](#)>

A transform function to apply to each element.

Returns

[double](#)

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare<TSource>(IEnumerable<TSource>, Func<TSource, decimal?>)

Computes the RootMeanSquare of a sequence of nullable decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal? RootMeanSquare<TSource>(this IEnumerable<TSource> source,  
Func<TSource, decimal?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, [decimal](#)?>

A transform function to apply to each element.

Returns

[decimal](#)?

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the RootMeanSquare of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? RootMeanSquare<TSource>(this IEnumerable<TSource> source,  
Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, [double](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare<TSource>(IEnumerable<TSource>, Func<TSource, int?>)

Computes the RootMeanSquare of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? RootMeanSquare<TSource>(this IEnumerable<TSource> source,  
Func<TSource, int?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, int?>

A transform function to apply to each element.

Returns

[double](#)?

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare<TSource>(IEnumerable<TSource>, Func<TSource, long?>)

Computes the RootMeanSquare of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? RootMeanSquare<TSource>(this IEnumerable<TSource> source,  
Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, long?>

A transform function to apply to each element.

Returns

[double](#)?

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the RootMeanSquare of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? RootMeanSquare<TSource>(this IEnumerable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, [float](#)?>

A transform function to apply to each element.

Returns

[float](#)?

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

RootMeanSquare<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the RootMeanSquare of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float RootMeanSquare<TSource>(this IEnumerable<TSource> source, Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values to calculate the RootMeanSquare of.

selector [Func](#)<TSource, float>

A transform function to apply to each element.

Returns

[float](#)

The RootMeanSquare.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$rms = \sqrt{\frac{\sum_{i=1}^N x_i^2}{N}}$$

Skewness(IEnumerable<decimal>)

Computes the sample Skewness of a sequence of decimal values

```
public static decimal Skewness(this IEnumerable<decimal> source)
```

Parameters

source [IEnumerable](#)<decimal>

A sequence of decimal values to calculate the Skewness of.

Returns

[decimal](#)

The Skewness of the sequence of values.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness(IEnumerable<double>)

Computes the sample Skewness of a sequence of double values

```
public static double Skewness(this IEnumerable<double> source)
```

Parameters

source [IEnumerable](#)<[double](#)>

A sequence of double values to calculate the Skewness of.

Returns

[double](#)

The Skewness of the sequence of values.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness(IEnumerable<int>)

Computes the sample Skewness of a sequence of int values

```
public static double Skewness(this IEnumerable<int> source)
```

Parameters

source [IEnumerable<int>](#)

A sequence of int values to calculate the Skewness of.

Returns

[double](#)

The Skewness of the sequence of values.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness(IEnumerable<long>)

Computes the sample Skewness of a sequence of long values

```
public static double Skewness(this IEnumerable<long> source)
```

Parameters

source [IEnumerable<long>](#)

A sequence of long values to calculate the Skewness of.

Returns

[double](#)

The Skewness of the sequence of values.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness(IEnumerable<decimal?>)

Computes the sample Skewness of a sequence of nullable decimal values

```
public static decimal? Skewness(this IEnumerable<decimal?> source)
```

Parameters

source [IEnumerable](#)<[decimal](#)?>

A sequence of nullable decimal values to calculate the Skewness of.

Returns

[decimal](#)?

The Skewness of the sequence of values.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness(IEnumerable<double?>)

Computes the sample Skewness of a sequence of nullable double values

```
public static double? Skewness(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable](#)<[double](#)?>

A sequence of nullable double values to calculate the Skewness of.

Returns

double?

The Skewness of the sequence of values.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness(IEnumerable<int?>)

Computes the sample Skewness of a sequence of nullable int values

```
public static double? Skewness(this IEnumerable<int?> source)
```

Parameters

source IEnumerable<int?>

A sequence of nullable int values to calculate the Skewness of.

Returns

double?

The Skewness of the sequence of values.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness(IEnumerable<long?>)

Computes the sample Skewness of a sequence of nullable long values

```
public static double? Skewness(this IEnumerable<long?> source)
```

Parameters

source [IEnumerable<long?>](#)

A sequence of nullable long values to calculate the Skewness of.

Returns

[double?](#)

The Skewness of the sequence of values.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness(IEnumerable<float?>)

Computes the sample Skewness of a sequence of nullable float values

```
public static float? Skewness(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

A sequence of nullable float values to calculate the Skewness of.

Returns

[float?](#)

The Skewness of the sequence of values.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness(IEnumerable<float>)

Computes the sample Skewness of a sequence of float values

```
public static float Skewness(this IEnumerable<float> source)
```

Parameters

source [IEnumerable<float>](#)

A sequence of float values to calculate the Skewness of.

Returns

[float](#)

The Skewness of the sequence of values.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

SkewnessNaN(IEnumerable<double>)

Computes the sample Skewness of a sequence of double values

```
public static double SkewnessNaN(this IEnumerable<double> source)
```

Parameters

source [IEnumerable<double>](#)

A sequence of double values to calculate the Skewness of.

Returns

[double](#)

The Skewness of the sequence of values.

SkewnessNaN(IEnumerable<int>)

Computes the sample Skewness of a sequence of int values

```
public static double SkewnessNaN(this IEnumerable<int> source)
```

Parameters

`source` [IEnumerable<int>](#)

A sequence of int values to calculate the Skewness of.

Returns

[double](#)

The Skewness of the sequence of values.

SkewnessNaN(IEnumerable<long>)

Computes the sample Skewness of a sequence of long values

```
public static double SkewnessNaN(this IEnumerable<long> source)
```

Parameters

`source` [IEnumerable<long>](#)

A sequence of long values to calculate the Skewness of.

Returns

[double](#)

The Skewness of the sequence of values.

SkewnessNaN(IEnumerable<double?>)

Computes the sample Skewness of a sequence of nullable double values

```
public static double? SkewnessNaN(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable<double?>](#)

A sequence of nullable double values to calculate the Skewness of.

Returns

[double?](#)

The Skewness of the sequence of values.

SkewnessNaN(IEnumerable<int?>)

Computes the sample Skewness of a sequence of nullable int values

```
public static double? SkewnessNaN(this IEnumerable<int?> source)
```

Parameters

source [IEnumerable<int?>](#)

A sequence of nullable int values to calculate the Skewness of.

Returns

[double?](#)

The Skewness of the sequence of values.

SkewnessNaN(IEnumerable<long?>)

Computes the sample Skewness of a sequence of nullable long values

```
public static double? SkewnessNaN(this IEnumerable<long?> source)
```

Parameters

`source IEnumerable<long?>`

A sequence of nullable long values to calculate the Skewness of.

Returns

`double?`

The Skewness of the sequence of values.

SkewnessNaN(IEnumerable<float?>)

Computes the sample Skewness of a sequence of nullable float values

```
public static float? SkewnessNaN(this IEnumerable<float?> source)
```

Parameters

`source IEnumerable<float?>`

A sequence of nullable float values to calculate the Skewness of.

Returns

`float?`

The Skewness of the sequence of values.

SkewnessNaN(IEnumerable<float>)

Computes the sample Skewness of a sequence of float values

```
public static float SkewnessNaN(this IEnumerable<float> source)
```

Parameters

`source IEnumerable<float>`

A sequence of float values to calculate the Skewness of.

Returns

[float](#)

The Skewness of the sequence of values.

SkewnessNaN<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the sample Skewness of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double SkewnessNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, [double](#)>

A transform function to apply to each element.

Returns

[double](#)

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

SkewnessNaN<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the sample Skewness of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double SkewnessNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, [int](#)>

A transform function to apply to each element.

Returns

[double](#)

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

SkewnessNaN<TSource>(IEnumerable<TSource>, Func<TSource, long>)

Computes the sample Skewness of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double SkewnessNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, [long](#)>

A transform function to apply to each element.

Returns

[double](#)

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

SkewnessNaN<TSource>(IEnumarable<TSource>, Func<TSource, double?>)

Computes the sample Skewness of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? SkewnessNaN<TSource>(this IEnumarable<TSource> source, Func<TSource, double?> selector)
```

Parameters

source [IEnumarable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, [double](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

SkewnessNaN<TSource>(IEnumerable<TSource>, Func<TSource, int?>)

Computes the sample Skewness of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? SkewnessNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, int?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, [int](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

SkewnessNaN<TSource>(IEnumerable<TSource>, Func<TSource, long?>)

Computes the sample Skewness of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? SkewnessNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, long?>

A transform function to apply to each element.

Returns

[double](#)?

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

SkewnessNaN<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the sample Skewness of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? SkewnessNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, [float](#)?>

A transform function to apply to each element.

Returns

[float](#)?

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

SkewnessNaN<TSource>(IEnumarable<TSource>, Func<TSource, float>)

Computes the sample Skewness of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float SkewnessNaN<TSource>(this IEnumarable<TSource> source, Func<TSource, float> selector)
```

Parameters

source [IEnumarable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, [float](#)>

A transform function to apply to each element.

Returns

[float](#)

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Skewness<TSource>(IEnumerable<TSource>, Func<TSource, decimal>)

Computes the sample Skewness of a sequence of decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal Skewness<TSource>(this IEnumerable<TSource> source, Func<TSource, decimal> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, [decimal](#)>

A transform function to apply to each element.

Returns

[decimal](#)

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the sample Skewness of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Skewness<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, [double](#)>

A transform function to apply to each element.

Returns

[double](#)

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the sample Skewness of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Skewness<TSource>(this IEnumerable<TSource> source, Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, [int](#)>

A transform function to apply to each element.

Returns

[double](#)

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness<TSource>(IEnumerable<TSource>, Func<TSource, long>)

Computes the sample Skewness of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Skewness<TSource>(this IEnumerable<TSource> source, Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, long>

A transform function to apply to each element.

Returns

[double](#)

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness<TSource>(IEnumerable<TSource>, Func<TSource, decimal?>)

Computes the sample Skewness of a sequence of nullable decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal? Skewness<TSource>(this IEnumerable<TSource> source, Func<TSource, decimal?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, [decimal](#)?>

A transform function to apply to each element.

Returns

[decimal](#)?

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the sample Skewness of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Skewness<TSource>(this IEnumerable<TSource> source, Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, [double](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness<TSource>(IEnumerable<TSource>, Func<TSource, int?>)

Computes the sample Skewness of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Skewness<TSource>(this IEnumerable<TSource> source, Func<TSource, int?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, [int](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness<TSource>(IEnumerable<TSource>, Func<TSource, long?>)

Computes the sample Skewness of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Skewness<TSource>(this IEnumerable<TSource> source, Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, long?>

A transform function to apply to each element.

Returns

[double](#)?

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the sample Skewness of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? Skewness<TSource>(this IEnumerable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, [float](#)?>

A transform function to apply to each element.

Returns

[float](#)?

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

Skewness<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the sample Skewness of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float Skewness<TSource>(this IEnumerable<TSource> source, Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

A sequence of values that are used to calculate a Skewness

selector [Func](#)<TSource, float>

A transform function to apply to each element.

Returns

[float](#)

The Skewness of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$S = \frac{N\sqrt{N-1}}{N-2} \frac{\sum_{i=1}^N (x_i - \bar{x})^3}{(\sum_{i=1}^N (x_i - \bar{x})^2)^{3/2}}$$

StandardDeviation(IEnumerable<decimal>)

Computes the sample StandardDeviation of a sequence of decimal values.

```
public static decimal StandardDeviation(this IEnumerable<decimal> source)
```

Parameters

source [IEnumerable<decimal>](#)

A sequence of decimal values to calculate the StandardDeviation of.

Returns

[decimal](#)

The StandardDeviation of the sequence of values.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation(IEnumerable<double>)

Computes the sample StandardDeviation of a sequence of double values.

```
public static double StandardDeviation(this IEnumerable<double> source)
```

Parameters

source [IEnumerable<double>](#)

A sequence of double values to calculate the StandardDeviation of.

Returns

[double](#)

The StandardDeviation of the sequence of values.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation(IEnumerable<int>)

Computes the sample StandardDeviation of a sequence of int values.

```
public static double StandardDeviation(this IEnumerable<int> source)
```

Parameters

source [IEnumerable<int>](#)

A sequence of int values to calculate the StandardDeviation of.

Returns

[double](#)

The StandardDeviation of the sequence of values.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation(IEnumerable<long>)

Computes the sample StandardDeviation of a sequence of long values.

```
public static double StandardDeviation(this IEnumerable<long> source)
```

Parameters

source [IEnumerable<long>](#)

A sequence of long values to calculate the StandardDeviation of.

Returns

[double](#)

The StandardDeviation of the sequence of values.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation(IEnumerable<decimal?>)

Computes the sample StandardDeviation of a sequence of nullable decimal values.

```
public static decimal? StandardDeviation(this IEnumerable<decimal?> source)
```

Parameters

source [IEnumerable](#)<[decimal](#)?>

A sequence of nullable decimal values to calculate the StandardDeviation of.

Returns

[decimal](#)?

The StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation(IEnumerable<double?>)

Computes the sample StandardDeviation of a sequence of nullable double values.

```
public static double? StandardDeviation(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable<double?>](#)

A sequence of nullable double values to calculate the StandardDeviation of.

Returns

[double?](#)

The StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation(IEnumerable<int?>)

Computes the sample StandardDeviation of a sequence of nullable int values.

```
public static double? StandardDeviation(this IEnumerable<int?> source)
```

Parameters

source [IEnumerable<int?>](#)

A sequence of nullable int values to calculate the StandardDeviation of.

Returns

[double?](#)

The StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation(IEnumerable<long?>)

Computes the sample StandardDeviation of a sequence of nullable long values.

```
public static double? StandardDeviation(this IEnumerable<long?> source)
```

Parameters

source [IEnumerable<long?>](#)

A sequence of nullable long values to calculate the StandardDeviation of.

Returns

[double?](#)

The StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation(IEnumerable<float?>)

Computes the sample StandardDeviation of a sequence of nullable float values.

```
public static float? StandardDeviation(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

A sequence of nullable float values to calculate the StandardDeviation of.

Returns

[float](#)?

The StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation(IEnumerable<float>)

Computes the sample StandardDeviation of a sequence of float values.

```
public static float StandardDeviation(this IEnumerable<float> source)
```

Parameters

source [IEnumerable](#)<[float](#)>

A sequence of float values to calculate the StandardDeviation of.

Returns

[float](#)?

The StandardDeviation of the sequence of values.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviationNaN(IEnumerable<double>)

Computes the sample StandardDeviation of a sequence of double values.

```
public static double StandardDeviationNaN(this IEnumerable<double> source)
```

Parameters

source [IEnumerable](#)<[double](#)>

A sequence of double values to calculate the StandardDeviation of.

Returns

[double](#)

The StandardDeviation of the sequence of values.

StandardDeviationNaN(IEnumerable<int>)

Computes the sample StandardDeviation of a sequence of int values.

```
public static double StandardDeviationNaN(this IEnumerable<int> source)
```

Parameters

source [IEnumerable](#)<[int](#)>

A sequence of int values to calculate the StandardDeviation of.

Returns

[double](#)

The StandardDeviation of the sequence of values.

StandardDeviationNaN(IEnumerable<long>)

Computes the sample StandardDeviation of a sequence of long values.

```
public static double StandardDeviationNaN(this IEnumerable<long> source)
```

Parameters

source [IEnumerable<long>](#)

A sequence of long values to calculate the StandardDeviation of.

Returns

[double](#)

The StandardDeviation of the sequence of values.

StandardDeviationNaN(IEnumerable<double?>)

Computes the sample StandardDeviation of a sequence of nullable double values.

```
public static double? StandardDeviationNaN(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable<double?>](#)

A sequence of nullable double values to calculate the StandardDeviation of.

Returns

[double](#)?

The StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

StandardDeviationNaN(IEnumerable<int?>)

Computes the sample StandardDeviation of a sequence of nullable int values.

```
public static double? StandardDeviationNaN(this IEnumerable<int?> source)
```

Parameters

`source` `IEnumerable<int?>`

A sequence of nullable int values to calculate the StandardDeviation of.

Returns

`double?`

The StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

StandardDeviationNaN(IEnumerable<long?>)

Computes the sample StandardDeviation of a sequence of nullable long values.

```
public static double? StandardDeviationNaN(this IEnumerable<long?> source)
```

Parameters

`source` `IEnumerable<long?>`

A sequence of nullable long values to calculate the StandardDeviation of.

Returns

`double?`

The StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

StandardDeviationNaN(IEnumerable<float?>)

Computes the sample StandardDeviation of a sequence of nullable float values.

```
public static float? StandardDeviationNaN(this IEnumerable<float?> source)
```

Parameters

`source` `IEnumerable<float?>`

A sequence of nullable float values to calculate the StandardDeviation of.

Returns

`float?`

The StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

StandardDeviationNaN(IEnumerable<float>)

Computes the sample StandardDeviation of a sequence of float values.

```
public static float StandardDeviationNaN(this IEnumerable<float> source)
```

Parameters

`source` `IEnumerable<float>`

A sequence of float values to calculate the StandardDeviation of.

Returns

`float`

The StandardDeviation of the sequence of values.

StandardDeviationNaN<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the sample StandardDeviation of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double StandardDeviationNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)>

A transform function to apply to each element.

Returns

[double](#)

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

StandardDeviationNaN<TSource>(IQueryable<TSource>, Func<TSource, int>)

Computes the sample StandardDeviation of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double StandardDeviationNaN<TSource>(this IQueryable<TSource> source,  
Func<TSource, int> selector)
```

Parameters

source [IQueryable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [int](#)>

A transform function to apply to each element.

Returns

[double](#)

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

StandardDeviationNaN<TSource>(IEnumerable<TSource>, Func<TSource, long>)

Computes the sample StandardDeviation of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double StandardDeviationNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, long>

A transform function to apply to each element.

Returns

[double](#)

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

StandardDeviationNaN<TSource>(IEnumarable<TSource>, Func<TSource, double?>)

Computes the sample StandardDeviation of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? StandardDeviationNaN<TSource>(this IEnumarable<TSource> source,  
Func<TSource, double?> selector)
```

Parameters

source [IEnumarable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

StandardDeviationNaN<TSource>(IEnumarable<TSource>, Func<TSource, int?>)

Computes the sample StandardDeviation of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? StandardDeviationNaN<TSource>(this IEnumarable<TSource> source,  
Func<TSource, int?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [int](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

StandardDeviationNaN<TSource>(IEnumarable<TSource>, Func<TSource, long?>)

Computes the sample StandardDeviation of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? StandardDeviationNaN<TSource>(this IEnumarable<TSource> source,  
Func<TSource, long?> selector)
```

Parameters

source [IEnumarable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [long](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

StandardDeviationNaN<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the sample StandardDeviation of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? StandardDeviationNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [float](#)?>

A transform function to apply to each element.

Returns

[float](#)?

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

StandardDeviationNaN<TSource>(IEnumable<TSource>, Func<TSource, float>)

Computes the sample StandardDeviation of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float StandardDeviationNaN<TSource>(this IEnumable<TSource> source,  
Func<TSource, float> selector)
```

Parameters

source [IEnumable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, float>

A transform function to apply to each element.

Returns

[float](#)

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

StandardDeviationP(IEnumable<decimal>)

Computes the population StandardDeviation of a sequence of decimal values.

```
public static decimal StandardDeviationP(this IEnumable<decimal> source)
```

Parameters

source [IEnumable](#)<decimal>

A sequence of decimal values to calculate the population StandardDeviation of.

Returns

[decimal](#)

The population StandardDeviation of the sequence of values.

Remarks

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

StandardDeviationP(IEnumerable<double>)

Computes the population StandardDeviation of a sequence of double values.

```
public static double StandardDeviationP(this IEnumerable<double> source)
```

Parameters

source [IEnumerable](#)<[double](#)>

A sequence of double values to calculate the population StandardDeviation of.

Returns

[double](#)

The population StandardDeviation of the sequence of values.

Remarks

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

StandardDeviationP(IEnumerable<int>)

Computes the population StandardDeviation of a sequence of int values.

```
public static double StandardDeviationP(this IEnumerable<int> source)
```

Parameters

source [IEnumerable<int>](#)

A sequence of int values to calculate the population StandardDeviation of.

Returns

[double](#)

The population StandardDeviation of the sequence of values.

Remarks

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

StandardDeviationP(IEnumerable<long>)

Computes the population StandardDeviation of a sequence of long values.

```
public static double StandardDeviationP(this IEnumerable<long> source)
```

Parameters

source [IEnumerable<long>](#)

A sequence of long values to calculate the population StandardDeviation of.

Returns

[double](#)

The population StandardDeviation of the sequence of values.

Remarks

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

StandardDeviationP(IEnumerable<decimal?>)

Computes the population StandardDeviation of a sequence of nullable decimal values.

```
public static decimal? StandardDeviationP(this IEnumerable<decimal?> source)
```

Parameters

source [IEnumerable<decimal?>](#)

A sequence of nullable decimal values to calculate the population StandardDeviation of.

Returns

[decimal?](#)

The population StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

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

StandardDeviationP(IEnumerable<double?>)

Computes the population StandardDeviation of a sequence of nullable double values.

```
public static double? StandardDeviationP(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable<double?>](#)

A sequence of nullable double values to calculate the population StandardDeviation of.

Returns

[double](#)?

The population StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

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

StandardDeviationP(IEnumerable<int?>)

Computes the population StandardDeviation of a sequence of nullable int values.

```
public static double? StandardDeviationP(this IEnumerable<int?> source)
```

Parameters

source [IEnumerable](#)<int?>

A sequence of nullable int values to calculate the population StandardDeviation of.

Returns

[double](#)?

The population StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

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

StandardDeviationP(IEnumerable<long?>)

Computes the population StandardDeviation of a sequence of nullable long values.

```
public static double? StandardDeviationP(this IEnumerable<long?> source)
```

Parameters

source [IEnumerable<long?>](#)

A sequence of nullable long values to calculate the population StandardDeviation of.

Returns

[double?](#)

The population StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

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

StandardDeviationP(IEnumerable<float?>)

Computes the population StandardDeviation of a sequence of nullable float values.

```
public static float? StandardDeviationP(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

A sequence of nullable float values to calculate the population StandardDeviation of.

Returns

[float?](#)

The population StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

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

StandardDeviationP(IEnumerable<float>)

Computes the population StandardDeviation of a sequence of float values.

```
public static float StandardDeviationP(this IEnumerable<float> source)
```

Parameters

source [IEnumerable<float>](#)

A sequence of float values to calculate the population StandardDeviation of.

Returns

[float](#)

The population StandardDeviation of the sequence of values.

Remarks

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

StandardDeviationPNaN(IEnumerable<double>)

Computes the population StandardDeviation of a sequence of double values.

```
public static double StandardDeviationPNaN(this IEnumerable<double> source)
```

Parameters

source [IEnumerable<double>](#)

A sequence of double values to calculate the population StandardDeviation of.

Returns

[double](#)

The population StandardDeviation of the sequence of values.

StandardDeviationPNaN(IEnumerable<int>)

Computes the population StandardDeviation of a sequence of int values.

```
public static double StandardDeviationPNaN(this IEnumerable<int> source)
```

Parameters

source [IEnumerable](#)<int>

A sequence of int values to calculate the population StandardDeviation of.

Returns

[double](#)

The population StandardDeviation of the sequence of values.

StandardDeviationPNaN(IEnumerable<long>)

Computes the population StandardDeviation of a sequence of long values.

```
public static double StandardDeviationPNaN(this IEnumerable<long> source)
```

Parameters

source [IEnumerable](#)<long>

A sequence of long values to calculate the population StandardDeviation of.

Returns

[double](#)

The population StandardDeviation of the sequence of values.

StandardDeviationPNaN(IEnumerable<double?>)

Computes the population StandardDeviation of a sequence of nullable double values.

```
public static double? StandardDeviationPNaN(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable](#)<[double](#)?>

A sequence of nullable double values to calculate the population StandardDeviation of.

Returns

[double](#)?

The population StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

StandardDeviationPNaN(IEnumerable<int?>)

Computes the population StandardDeviation of a sequence of nullable int values.

```
public static double? StandardDeviationPNaN(this IEnumerable<int?> source)
```

Parameters

source [IEnumerable](#)<[int](#)?>

A sequence of nullable int values to calculate the population StandardDeviation of.

Returns

[double](#)?

The population StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

StandardDeviationPNaN(IEnumerable<long?>)

Computes the population StandardDeviation of a sequence of nullable long values.

```
public static double? StandardDeviationPNaN(this IEnumerable<long?> source)
```

Parameters

source [IEnumerable<long?>](#)

A sequence of nullable long values to calculate the population StandardDeviation of.

Returns

[double?](#)

The population StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

StandardDeviationPNaN(IEnumerable<float?>)

Computes the population StandardDeviation of a sequence of nullable float values.

```
public static float? StandardDeviationPNaN(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

A sequence of nullable float values to calculate the population StandardDeviation of.

Returns

[float?](#)

The population StandardDeviation of the sequence of values, or null if the source sequence is empty or contains only values that are null.

StandardDeviationPNaN(IEnumerable<float>)

Computes the population StandardDeviation of a sequence of float values.

```
public static float StandardDeviationPNaN(this IEnumerable<float> source)
```

Parameters

source [IEnumerable<float>](#)

A sequence of float values to calculate the population StandardDeviation of.

Returns

[float](#)

The population StandardDeviation of the sequence of values.

StandardDeviationPNaN<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the population StandardDeviation of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double StandardDeviationPNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable<TSource>](#)

The sequence of elements.

selector [Func<TSource, double>](#)

A transform function to apply to each element.

Returns

[double](#)

The population StandardDeviation of the sequence of values.

Type Parameters

[TSource](#)

The type of the elements of source.

StandardDeviationPNaN<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the population StandardDeviation of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double StandardDeviationPNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, int> selector)
```

Parameters

[source](#) [IEnumerable](#)<TSource>

The sequence of elements.

[selector](#) [Func](#)<TSource, int>

A transform function to apply to each element.

Returns

[double](#)

The population StandardDeviation of the sequence of values.

Type Parameters

[TSource](#)

The type of the elements of source.

StandardDeviationPNaN<TSource>(IEnumarable<TSource>, Func<TSource, long>)

Computes the population StandardDeviation of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double StandardDeviationPNaN<TSource>(this IEnumarable<TSource> source,  
Func<TSource, long> selector)
```

Parameters

source [IEnumarable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, long>

A transform function to apply to each element.

Returns

[double](#)

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

StandardDeviationPNaN<TSource>(IEnumarable<TSource>, Func<TSource, double?>)

Computes the population StandardDeviation of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? StandardDeviationPNaN<TSource>(this IEnumarable<TSource> source,  
Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

StandardDeviationPNaN<TSource>(IEnumarable<TSource>, Func<TSource, int?>)

Computes the population StandardDeviation of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? StandardDeviationPNaN<TSource>(this IEnumarable<TSource> source,  
Func<TSource, int?> selector)
```

Parameters

source [IEnumarable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [int](#)?>

A transform function to apply to each element.

Returns

double?

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

StandardDeviationPNaN<TSource>(IEnumerable<TSource>, Func<TSource, long?>)

Computes the population StandardDeviation of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? StandardDeviationPNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, long?> selector)
```

Parameters

source IEnumerable<TSource>

The sequence of elements.

selector Func<TSource, long?>

A transform function to apply to each element.

Returns

double?

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

StandardDeviationPNaN<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the population StandardDeviation of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? StandardDeviationPNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [float](#)?>

A transform function to apply to each element.

Returns

[float](#)?

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

StandardDeviationPNaN<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the population StandardDeviation of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float StandardDeviationPNaN<TSource>(this IEnumerable<TSource> source,  
Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [float](#)>

A transform function to apply to each element.

Returns

[float](#)

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

StandardDeviationP<TSource>(IEnumerable<TSource>, Func<TSource, decimal>)

Computes the population StandardDeviation of a sequence of decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal StandardDeviationP<TSource>(this IEnumerable<TSource> source,  
Func<TSource, decimal> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [decimal](#)>

A transform function to apply to each element.

Returns

[decimal](#)

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

StandardDeviationP<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the population StandardDeviation of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double StandardDeviationP<TSource>(this IEnumerable<TSource> source,  
Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)>

A transform function to apply to each element.

Returns

[double](#)

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

StandardDeviationP<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the population StandardDeviation of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double StandardDeviationP<TSource>(this IEnumerable<TSource> source,  
Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, int>

A transform function to apply to each element.

Returns

[double](#)

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

StandardDeviationP<TSource>(IEnumerable<TSource>, Func<TSource, long>)

Computes the population StandardDeviation of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double StandardDeviationP<TSource>(this IEnumerable<TSource> source,  
Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, long>

A transform function to apply to each element.

Returns

[double](#)

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

StandardDeviationP<TSource>(IEnumerable<TSource>, Func<TSource, decimal?>)

Computes the population StandardDeviation of a sequence of nullable decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal? StandardDeviationP<TSource>(this IEnumerable<TSource> source,  
Func<TSource, decimal?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [decimal](#)?>

A transform function to apply to each element.

Returns

[decimal](#)?

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

StandardDeviationP<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the population StandardDeviation of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? StandardDeviationP<TSource>(this IEnumerable<TSource> source,  
Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

StandardDeviationP<TSource>(IEnumerable<TSource>, Func<TSource, int?>)

Computes the population StandardDeviation of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? StandardDeviationP<TSource>(this IEnumerable<TSource> source,  
Func<TSource, int?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, int?>

A transform function to apply to each element.

Returns

[double](#)?

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

StandardDeviationP<TSource>(IEnumerable<TSource>, Func<TSource, long?>)

Computes the population StandardDeviation of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? StandardDeviationP<TSource>(this IEnumerable<TSource> source,  
Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [long](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

StandardDeviationP<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the population StandardDeviation of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? StandardDeviationP<TSource>(this IEnumerable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector `Func<TSource, float>`

A transform function to apply to each element.

Returns

`float?`

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

StandardDeviationP<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the population StandardDeviation of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float StandardDeviationP<TSource>(this IEnumerable<TSource> source,  
Func<TSource, float> selector)
```

Parameters

source `IEnumerable<TSource>`

The sequence of elements.

selector `Func<TSource, float>`

A transform function to apply to each element.

Returns

[float](#)

The population StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

StandardDeviation<TSource>(IEnumerable<TSource>, Func<TSource, decimal>)

Computes the sample StandardDeviation of a sequence of decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal StandardDeviation<TSource>(this IEnumerable<TSource> source,  
Func<TSource, decimal> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [decimal](#)>

A transform function to apply to each element.

Returns

[decimal](#)

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the sample StandardDeviation of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double StandardDeviation<TSource>(this IEnumerable<TSource> source,  
Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)>

A transform function to apply to each element.

Returns

[double](#)

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the sample StandardDeviation of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double StandardDeviation<TSource>(this IEnumerable<TSource> source,  
Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, int>

A transform function to apply to each element.

Returns

[double](#)

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation<TSource>(IEnumerable<TSource>, Func<TSource, long>)

Computes the sample StandardDeviation of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double StandardDeviation<TSource>(this IEnumerable<TSource> source,  
Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, long>

A transform function to apply to each element.

Returns

[double](#)

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation<TSource>(IEnumerable<TSource>, Func<TSource, decimal?>)

Computes the sample StandardDeviation of a sequence of nullable decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal? StandardDeviation<TSource>(this IEnumerable<TSource> source,  
Func<TSource, decimal?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, decimal?>

A transform function to apply to each element.

Returns

[decimal](#)?

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the sample StandardDeviation of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? StandardDeviation<TSource>(this IEnumerable<TSource> source,  
Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, double?>

A transform function to apply to each element.

Returns

[double](#)?

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation<TSource>(IEnumerable<TSource>, Func<TSource, int?>)

Computes the sample StandardDeviation of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? StandardDeviation<TSource>(this IEnumerable<TSource> source,  
Func<TSource, int?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [int](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation<TSource>(IEnumerable<TSource>, Func<TSource, long?>)

Computes the sample StandardDeviation of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? StandardDeviation<TSource>(this IEnumerable<TSource> source,  
Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector `Func<TSource, long?>`

A transform function to apply to each element.

Returns

`double?`

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the sample StandardDeviation of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? StandardDeviation<TSource>(this IEnumerable<TSource> source,  
Func<TSource, float?> selector)
```

Parameters

source `IEnumerable<TSource>`

The sequence of elements.

selector `Func<TSource, float?>`

A transform function to apply to each element.

Returns

[float](#)?

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

StandardDeviation<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the sample StandardDeviation of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float StandardDeviation<TSource>(this IEnumerable<TSource> source,  
Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, float>

A transform function to apply to each element.

Returns

[float](#)

The StandardDeviation of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

Sum(IEnumerable<BigInteger?>)

Computes the Sum of a sequence of nullable BigInteger values.

```
public static BigInteger? Sum(this IEnumerable<BigInteger?> source)
```

Parameters

source [IEnumerable](#)<[BigInteger](#)?>

A sequence of nullable BigInteger values to calculate the Sum of.

Returns

[BigInteger](#)?

The Sum of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Sum(IEnumerable<BigInteger>)

Computes the Sum of a sequence of BigInteger values.

```
public static BigInteger Sum(this IEnumerable<BigInteger> source)
```

Parameters

source [IEnumerable](#)<[BigInteger](#)>

A sequence of BigInteger values to calculate the Sum of.

Returns

[BigInteger](#)

The Sum of the sequence of values.

SumNaN<TSource>(IEnumerable<TSource>, Func<TSource, BigInteger?>)

Computes the Sum of a sequence of nullable BigInteger values that are obtained by invoking a transform function on each element of the input sequence.

```
public static BigInteger? SumNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, BigInteger?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, BigInteger?>

A transform function to apply to each element.

Returns

[BigInteger](#)?

The Sum of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Sum<TSource>(IEnumerable<TSource>, Func<TSource, BigInteger>)

Computes the Sum of a sequence of BigInteger values that are obtained by invoking a transform function on each element of the input sequence.

```
public static BigInteger Sum<TSource>(this IEnumerable<TSource> source, Func<TSource, BigInteger> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [BigInteger](#)>

A transform function to apply to each element.

Returns

[BigInteger](#)

The Sum of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Variance(IEnumerable<decimal>)

Computes the sample Variance of a sequence of decimal values.

```
public static decimal Variance(this IEnumerable<decimal> source)
```

Parameters

source [IEnumerable](#)<[decimal](#)>

A sequence of decimal values to calculate the Variance of.

Returns

[decimal](#)

The Variance of the sequence of values.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance(IEnumerable<double>)

Computes the sample Variance of a sequence of double values.

```
public static double Variance(this IEnumerable<double> source)
```

Parameters

source [IEnumerable](#)<[double](#)>

A sequence of double values to calculate the Variance of.

Returns

[double](#)

The Variance of the sequence of values.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance(IEnumerable<int>)

Computes the sample Variance of a sequence of int values.

```
public static double Variance(this IEnumerable<int> source)
```

Parameters

source [IEnumerable<int>](#)

A sequence of int values to calculate the Variance of.

Returns

[double](#)

The Variance of the sequence of values.

Remarks

$$s^2 = \frac{1}{N - 1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance(IEnumerable<long>)

Computes the sample Variance of a sequence of long values.

```
public static double Variance(this IEnumerable<long> source)
```

Parameters

source [IEnumerable<long>](#)

A sequence of long values to calculate the Variance of.

Returns

[double](#)

The Variance of the sequence of values.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance(IEnumerable<decimal?>)

Computes the sample Variance of a sequence of nullable decimal values.

```
public static decimal? Variance(this IEnumerable<decimal?> source)
```

Parameters

source [IEnumerable<decimal?>](#)

A sequence of nullable decimal values to calculate the Variance of.

Returns

[decimal?](#)

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance(IEnumerable<double?>)

Computes the sample Variance of a sequence of nullable double values.

```
public static double? Variance(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable<double?>](#)

A sequence of nullable double values to calculate the Variance of.

Returns

double?

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance(IEnumerable<int?>)

Computes the sample Variance of a sequence of nullable int values.

```
public static double? Variance(this IEnumerable<int?> source)
```

Parameters

source IEnumerable<int?>

A sequence of nullable int values to calculate the Variance of.

Returns

double?

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance(IEnumerable<long?>)

Computes the sample Variance of a sequence of nullable long values.

```
public static double? Variance(this IEnumerable<long?> source)
```

Parameters

source [IEnumerable<long?>](#)

A sequence of nullable long values to calculate the Variance of.

Returns

[double?](#)

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance(IEnumerable<float?>)

Computes the sample Variance of a sequence of nullable float values.

```
public static float? Variance(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

A sequence of nullable float values to calculate the Variance of.

Returns

[float?](#)

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance(IEnumerable<float>)

Computes the sample Variance of a sequence of float values.

```
public static float Variance(this IEnumerable<float> source)
```

Parameters

source [IEnumerable<float>](#)

A sequence of float values to calculate the Variance of.

Returns

[float](#)

The Variance of the sequence of values.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

VarianceNaN(IEnumerable<double>)

Computes the sample Variance of a sequence of double values.

```
public static double VarianceNaN(this IEnumerable<double> source)
```

Parameters

source [IEnumerable<double>](#)

A sequence of double values to calculate the Variance of.

Returns

double

The Variance of the sequence of values.

VarianceNaN(IEnumerable<int>)

Computes the sample Variance of a sequence of int values.

```
public static double VarianceNaN(this IEnumerable<int> source)
```

Parameters

source IEnumerable<int>

A sequence of int values to calculate the Variance of.

Returns

double

The Variance of the sequence of values.

VarianceNaN(IEnumerable<long>)

Computes the sample Variance of a sequence of long values.

```
public static double VarianceNaN(this IEnumerable<long> source)
```

Parameters

source IEnumerable<long>

A sequence of long values to calculate the Variance of.

Returns

double

The Variance of the sequence of values.

VarianceNaN(IEnumerable<double?>)

Computes the sample Variance of a sequence of nullable double values.

```
public static double? VarianceNaN(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable<double?>](#)

A sequence of nullable double values to calculate the Variance of.

Returns

[double?](#)

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

VarianceNaN(IEnumerable<int?>)

Computes the sample Variance of a sequence of nullable int values.

```
public static double? VarianceNaN(this IEnumerable<int?> source)
```

Parameters

source [IEnumerable<int?>](#)

A sequence of nullable int values to calculate the Variance of.

Returns

[double?](#)

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

VarianceNaN(IEnumerable<long?>)

Computes the sample Variance of a sequence of nullable long values.

```
public static double? VarianceNaN(this IEnumerable<long?> source)
```

Parameters

source [IEnumerable<long?>](#)

A sequence of nullable long values to calculate the Variance of.

Returns

[double?](#)

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

VarianceNaN(IEnumerable<float?>)

Computes the sample Variance of a sequence of nullable float values.

```
public static float? VarianceNaN(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

A sequence of nullable float values to calculate the Variance of.

Returns

[float?](#)

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

VarianceNaN(IEnumerable<float>)

Computes the sample Variance of a sequence of float values.

```
public static float VarianceNaN(this IEnumerable<float> source)
```

Parameters

source [IEnumerable](#)<[float](#)>

A sequence of float values to calculate the Variance of.

Returns

[float](#)

The Variance of the sequence of values.

VarianceNaN<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the sample Variance of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double VarianceNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<[TSource](#)>

The sequence of elements.

selector [Func](#)<[TSource](#), [double](#)>

A transform function to apply to each element.

Returns

[double](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VarianceNaN<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the sample Variance of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double VarianceNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, int>

A transform function to apply to each element.

Returns

[double](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VarianceNaN<TSource>(IEnumerable<TSource>, Func<TSource, long>)

Computes the sample Variance of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double VarianceNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, long>

A transform function to apply to each element.

Returns

[double](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VarianceNaN<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the sample Variance of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? VarianceNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector `Func<TSource, double?>`

A transform function to apply to each element.

Returns

`double?`

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VarianceNaN<TSource>(IEnumerable<TSource>, Func<TSource, int?>)

Computes the sample Variance of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? VarianceNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, int?> selector)
```

Parameters

source `IEnumerable<TSource>`

The sequence of elements.

selector `Func<TSource, int?>`

A transform function to apply to each element.

Returns

`double?`

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VarianceNaN<TSource>(IEnumerable<TSource>, Func<TSource, long?>)

Computes the sample Variance of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? VarianceNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, long?>

A transform function to apply to each element.

Returns

[double](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VarianceNaN<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the sample Variance of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? VarianceNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, float?>

A transform function to apply to each element.

Returns

[float](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VarianceNaN<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the sample Variance of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float VarianceNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [float](#)>

A transform function to apply to each element.

Returns

[float](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VarianceP(IEnumerable<decimal>)

Computes the population Variance of a sequence of decimal values.

```
public static decimal VarianceP(this IEnumerable<decimal> source)
```

Parameters

source [IEnumerable](#)<[decimal](#)>

A sequence of decimal values to calculate the population Variance of.

Returns

[decimal](#)

The Variance of the sequence of values.

Remarks

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

VarianceP(IEnumerable<double>)

Computes the population Variance of a sequence of double values.

```
public static double VarianceP(this IEnumerable<double> source)
```

Parameters

source [IEnumerable<double>](#)

A sequence of double values to calculate the population Variance of.

Returns

[double](#)

The Variance of the sequence of values.

Remarks

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

VarianceP(IEnumerable<int>)

Computes the population Variance of a sequence of int values.

```
public static double VarianceP(this IEnumerable<int> source)
```

Parameters

source [IEnumerable<int>](#)

A sequence of int values to calculate the population Variance of.

Returns

[double](#)

The Variance of the sequence of values.

Remarks

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

VarianceP(IEnumerable<long>)

Computes the population Variance of a sequence of long values.

```
public static double VarianceP(this IEnumerable<long> source)
```

Parameters

source [IEnumerable<long>](#)

A sequence of long values to calculate the population Variance of.

Returns

[double](#)

The Variance of the sequence of values.

Remarks

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

VarianceP(IEnumerable<decimal?>)

Computes the population Variance of a sequence of nullable decimal values.

```
public static decimal? VarianceP(this IEnumerable<decimal?> source)
```

Parameters

source [IEnumerable<decimal?>](#)

A sequence of nullable decimal values to calculate the population Variance of.

Returns

[decimal](#)?

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

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

VarianceP(IEnumerable<double?>)

Computes the population Variance of a sequence of nullable double values.

```
public static double? VarianceP(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable](#)<[double](#)?>

A sequence of nullable double values to calculate the population Variance of.

Returns

[double](#)?

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

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

VarianceP(IEnumerable<int?>)

Computes the population Variance of a sequence of nullable int values.

```
public static double? VarianceP(this IEnumerable<int?> source)
```

Parameters

source [IEnumerable<int?>](#)

A sequence of nullable int values to calculate the population Variance of.

Returns

[double?](#)

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

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

VarianceP(IEnumerable<long?>)

Computes the population Variance of a sequence of nullable long values.

```
public static double? VarianceP(this IEnumerable<long?> source)
```

Parameters

source [IEnumerable<long?>](#)

A sequence of nullable long values to calculate the population Variance of.

Returns

[double?](#)

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

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

VarianceP(IEnumerable<float?>)

Computes the population Variance of a sequence of nullable float values.

```
public static float? VarianceP(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

A sequence of nullable float values to calculate the population Variance of.

Returns

[float?](#)

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

Remarks

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

VarianceP(IEnumerable<float>)

Computes the population Variance of a sequence of float values.

```
public static float VarianceP(this IEnumerable<float> source)
```

Parameters

source [IEnumerable<float>](#)

A sequence of float values to calculate the population Variance of.

Returns

[float](#)

The Variance of the sequence of values.

Remarks

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

VariancePNaN(IEnumerable<double>)

Computes the population Variance of a sequence of double values.

```
public static double VariancePNaN(this IEnumerable<double> source)
```

Parameters

[source](#) [IEnumerable](#)<[double](#)>

A sequence of double values to calculate the population Variance of.

Returns

[double](#)

The Variance of the sequence of values.

VariancePNaN(IEnumerable<int>)

Computes the population Variance of a sequence of int values.

```
public static double VariancePNaN(this IEnumerable<int> source)
```

Parameters

[source](#) [IEnumerable](#)<[int](#)>

A sequence of int values to calculate the population Variance of.

Returns

[double](#)

The Variance of the sequence of values.

VariancePNaN(IEnumerable<long>)

Computes the population Variance of a sequence of long values.

```
public static double VariancePNaN(this IEnumerable<long> source)
```

Parameters

source [IEnumerable](#)<long>

A sequence of long values to calculate the population Variance of.

Returns

[double](#)

The Variance of the sequence of values.

VariancePNaN(IEnumerable<double?>)

Computes the population Variance of a sequence of nullable double values.

```
public static double? VariancePNaN(this IEnumerable<double?> source)
```

Parameters

source [IEnumerable](#)<double?>

A sequence of nullable double values to calculate the population Variance of.

Returns

[double](#)?

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

VariancePNaN(IEnumerable<int?>)

Computes the population Variance of a sequence of nullable int values.

```
public static double? VariancePNaN(this IEnumerable<int?> source)
```

Parameters

source [IEnumerable](#)<int?>

A sequence of nullable int values to calculate the population Variance of.

Returns

[double](#)?

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

VariancePNaN(IEnumerable<long?>)

Computes the population Variance of a sequence of nullable long values.

```
public static double? VariancePNaN(this IEnumerable<long?> source)
```

Parameters

source [IEnumerable](#)<long?>

A sequence of nullable long values to calculate the population Variance of.

Returns

[double](#)?

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

VariancePNaN(IEnumerable<float?>)

Computes the population Variance of a sequence of nullable float values.

```
public static float? VariancePNaN(this IEnumerable<float?> source)
```

Parameters

source [IEnumerable<float?>](#)

A sequence of nullable float values to calculate the population Variance of.

Returns

[float?](#)

The Variance of the sequence of values, or null if the source sequence is empty or contains only values that are null.

VariancePNaN(IEnumerable<float>)

Computes the population Variance of a sequence of float values.

```
public static float VariancePNaN(this IEnumerable<float> source)
```

Parameters

source [IEnumerable<float>](#)

A sequence of float values to calculate the population Variance of.

Returns

[float?](#)

The Variance of the sequence of values.

VariancePNaN<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the population Variance of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double VariancePNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)>

A transform function to apply to each element.

Returns

[double](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VariancePNaN<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the population Variance of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double VariancePNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [int](#)>

A transform function to apply to each element.

Returns

[double](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VariancePNaN<TSource>(IEnumerable<TSource>, Func<TSource, long>)

Computes the population Variance of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double VariancePNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [long](#)>

A transform function to apply to each element.

Returns

double

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VariancePNaN<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the population Variance of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? VariancePNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, double?> selector)
```

Parameters

source IEnumerable<TSource>

The sequence of elements.

selector Func<TSource, double?>

A transform function to apply to each element.

Returns

double?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VariancePNaN<TSource>(IEnumarable<TSource>, Func<TSource, int?>)

Computes the population Variance of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? VariancePNaN<TSource>(this IEnumarable<TSource> source, Func<TSource, int?> selector)
```

Parameters

source [IEnumarable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [int](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VariancePNaN<TSource>(IEnumarable<TSource>, Func<TSource, long?>)

Computes the population Variance of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? VariancePNaN<TSource>(this IEnumarable<TSource> source, Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [long](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VariancePNaN<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the population Variance of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? VariancePNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [float](#)?>

A transform function to apply to each element.

Returns

[float](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VariancePNaN<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the population Variance of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float VariancePNaN<TSource>(this IEnumerable<TSource> source, Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, float>

A transform function to apply to each element.

Returns

[float](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

VarianceP<TSource>(IEnumerable<TSource>, Func<TSource, decimal>)

Computes the population Variance of a sequence of decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal VarianceP<TSource>(this IEnumerable<TSource> source, Func<TSource, decimal> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [decimal](#)>

A transform function to apply to each element.

Returns

[decimal](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

VarianceP<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the population Variance of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double VarianceP<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, double>

A transform function to apply to each element.

Returns

[double](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

VarianceP<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the population Variance of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double VarianceP<TSource>(this IEnumerable<TSource> source, Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [int](#)>

A transform function to apply to each element.

Returns

[double](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

VarianceP<TSource>(IEnumerable<TSource>, Func<TSource, long>)

Computes the population Variance of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double VarianceP<TSource>(this IEnumerable<TSource> source, Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [long](#)>

A transform function to apply to each element.

Returns

[double](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

VarianceP<TSource>(IEnumerable<TSource>, Func<TSource, decimal?>)

Computes the population Variance of a sequence of nullable decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal? VarianceP<TSource>(this IEnumerable<TSource> source, Func<TSource, decimal?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [decimal](#)?>

A transform function to apply to each element.

Returns

[decimal](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

VarianceP<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the population Variance of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? VarianceP<TSource>(this IEnumerable<TSource> source, Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

VarianceP<TSource>(IEnumerable<TSource>, Func<TSource, int?>)

Computes the population Variance of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? VarianceP<TSource>(this IEnumerable<TSource> source, Func<TSource, int?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, int?>

A transform function to apply to each element.

Returns

[double](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

VarianceP<TSource>(IEnumerable<TSource>, Func<TSource, long?>)

Computes the population Variance of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? VarianceP<TSource>(this IEnumerable<TSource> source, Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, long?>

A transform function to apply to each element.

Returns

[double](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

VarianceP<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the population Variance of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? VarianceP<TSource>(this IEnumerable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [float](#)?>

A transform function to apply to each element.

Returns

[float](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

VarianceP<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the population Variance of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float VarianceP<TSource>(this IEnumerable<TSource> source, Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, float>

A transform function to apply to each element.

Returns

[float](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

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

Variance<TSource>(IEnumerable<TSource>, Func<TSource, decimal>)

Computes the sample Variance of a sequence of decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal Variance<TSource>(this IEnumerable<TSource> source, Func<TSource, decimal> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [decimal](#)>

A transform function to apply to each element.

Returns

[decimal](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance<TSource>(IEnumerable<TSource>, Func<TSource, double>)

Computes the sample Variance of a sequence of double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Variance<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)>

A transform function to apply to each element.

Returns

[double](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance<TSource>(IEnumerable<TSource>, Func<TSource, int>)

Computes the sample Variance of a sequence of int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Variance<TSource>(this IEnumerable<TSource> source, Func<TSource, int> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, int>

A transform function to apply to each element.

Returns

[double](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance<TSource>(IEnumerable<TSource>, Func<TSource, long>)

Computes the sample Variance of a sequence of long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double Variance<TSource>(this IEnumerable<TSource> source, Func<TSource, long> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, long>

A transform function to apply to each element.

Returns

[double](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance<TSource>(IEnumerable<TSource>, Func<TSource, decimal?>)

Computes the sample Variance of a sequence of nullable decimal values that are obtained by invoking a transform function on each element of the input sequence.

```
public static decimal? Variance<TSource>(this IEnumerable<TSource> source, Func<TSource, decimal?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [decimal](#)?>

A transform function to apply to each element.

Returns

[decimal](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance<TSource>(IEnumerable<TSource>, Func<TSource, double?>)

Computes the sample Variance of a sequence of nullable double values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Variance<TSource>(this IEnumerable<TSource> source, Func<TSource, double?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [double](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance<TSource>(IEnumerable<TSource>, Func<TSource, int?>)

Computes the sample Variance of a sequence of nullable int values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Variance<TSource>(this IEnumerable<TSource> source, Func<TSource, int?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [int](#)?>

A transform function to apply to each element.

Returns

[double](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance<TSource>(IEnumerable<TSource>, Func<TSource, long?>)

Computes the sample Variance of a sequence of nullable long values that are obtained by invoking a transform function on each element of the input sequence.

```
public static double? Variance<TSource>(this IEnumerable<TSource> source, Func<TSource, long?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, long?>

A transform function to apply to each element.

Returns

[double](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance<TSource>(IEnumerable<TSource>, Func<TSource, float?>)

Computes the sample Variance of a sequence of nullable float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float? Variance<TSource>(this IEnumerable<TSource> source, Func<TSource, float?> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [float](#)?>

A transform function to apply to each element.

Returns

[float](#)?

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Variance<TSource>(IEnumerable<TSource>, Func<TSource, float>)

Computes the sample Variance of a sequence of float values that are obtained by invoking a transform function on each element of the input sequence.

```
public static float Variance<TSource>(this IEnumerable<TSource> source, Func<TSource, float> selector)
```

Parameters

source [IEnumerable](#)<TSource>

The sequence of elements.

selector [Func](#)<TSource, [float](#)>

A transform function to apply to each element.

Returns

[float](#)

The Variance of the sequence of values.

Type Parameters

TSource

The type of the elements of source.

Remarks

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

Class ItemCount<T>

Namespace: [LinqStatistics](#)

Assembly: LinqStatistics.dll

Represents the count of an item in a collection

```
public class ItemCount<T> : IEquatable<ItemCount<T>>
```

Type Parameters

T

The type of the item

Inheritance

[object](#) ← ItemCount<T>

Implements

[IEquatable](#)<[ItemCount](#)<T>>

Derived

[Bin](#)

Inherited Members

[object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#)

Remarks

ctor

Constructors

ItemCount(T, int)

Represents the count of an item in a collection

```
public ItemCount(T v, int count)
```

Parameters

v T

count [int](#)

Remarks

ctor

Properties

Count

The number of times RepresentativeValue appears in the source data

```
public int Count { get; }
```

Property Value

[int](#)

Represents the count of an item in a collection

RepresentativeValue

The value represented by the bin

```
public T RepresentativeValue { get; }
```

Property Value

T

Represents the count of an item in a collection

Methods

Equals(ItemCount<T>)

[Equals\(I\)](#) ↗

```
public bool Equals(ItemCount<T> other)
```

Parameters

other [ItemCount](#)<T>

Returns

[bool](#) ↗

Equals(object)

[Equals\(object\)](#)

```
public override bool Equals(object obj)
```

Parameters

obj [object](#) ↗

The item to compare to.

Returns

[bool](#) ↗

True if obj is a Bin{T} and Value and Count are equal

GetHashCode()

[GetHashCode\(\)](#)

```
public override int GetHashCode()
```

Returns

[int](#)

Hash of Value and Count

ToString()

[ToString\(\)](#)

```
public override string ToString()
```

Returns

[string](#)

Operators

operator ==(ItemCount<T>, ItemCount<T>)

```
public static bool operator ==(ItemCount<T> lhs, ItemCount<T> rhs)
```

Parameters

lhs [ItemCount](#)<T>

rhs [ItemCount](#)<T>

Returns

[bool](#)

operator !=(ItemCount<T>, ItemCount<T>)

```
public static bool operator !=(ItemCount<T> lhs, ItemCount<T> rhs)
```

Parameters

`lhs ItemCount<T>`

`rhs ItemCount<T>`

Returns

`bool` ↗

Struct LeastSquares

Namespace: [LinqStatistics](#)

Assembly: LinqStatistics.dll

Represents the result of a LeastSquares calculation of the form $y = mX + b$

```
public struct LeastSquares : IFormattable, IEquatable<LeastSquares>
```

Implements

[IFormattable](#), [IEquatable](#)<[LeastSquares](#)>

Inherited Members

[object.Equals\(object, object\)](#), [object.GetType\(\)](#), [object.ReferenceEquals\(object, object\)](#)

Constructors

LeastSquares(double, double)

```
public LeastSquares(double m, double b)
```

Parameters

m [double](#)

The slope

b [double](#)

The intercept

LeastSquares(double, double, double)

```
public LeastSquares(double m, double b, double r2)
```

Parameters

m [double](#)

The slope

b [double](#)

The intercept

r2 [double](#)

The R Squared correlation coefficient

Fields

Empty

Empty instance - returned when EnumerableStats.LeastSquares is passed a singular matrix. All members are zero.

```
public static readonly LeastSquares Empty
```

Field Value

[LeastSquares](#)

Represents the result of a LeastSquares calculation of the form $y = mX + b$

Properties

B

B Coefficient for $y = Mx + B$ (i.e. y intercept)

```
public readonly double B { get; }
```

Property Value

[double](#)

Represents the result of a LeastSquares calculation of the form $y = mX + b$

M

M Coefficient for $y = Mx + B$ (i.e. slope)

```
public readonly double M { get; }
```

Property Value

[double](#)

Represents the result of a LeastSquares calculation of the form $y = mX + b$

RSquared

The R Squared correlation coefficient

```
public readonly double RSquared { get; }
```

Property Value

[double](#)

Represents the result of a LeastSquares calculation of the form $y = mX + b$

Methods

Equals(LeastSquares)

[Equals\(I\)](#)

```
public readonly bool Equals(LeastSquares other)
```

Parameters

other [LeastSquares](#)

Returns

[bool](#)

True if other has equal m and b values

Equals(object)

[Equals\(object\)](#)

```
public override readonly bool Equals(object obj)
```

Parameters

[obj](#) [object](#)

The object to compare to

Returns

[bool](#)

True if obj is a LeastSquares and has equal m and b values

GetHashCode()

[GetHashCode\(\)](#)

```
public override readonly int GetHashCode()
```

Returns

[int](#)

Hash code of the instance

IsNaN(LeastSquares)

Determines if the [LeastSquares](#) argument has Not a Number elements

```
public static bool IsNaN(LeastSquares ls)
```

Parameters

ls [LeastSquares](#)

Returns

[bool](#)

True if either [M](#), [B](#) or [RSquared](#) is [NaN](#)

Solve(double)

Uses the calculated coefficients to solve Y for inputted X

```
public readonly double Solve(double x)
```

Parameters

x [double](#)

X value to solve for

Returns

[double](#)

Y value ($y = Mx + B$)

SolveForX(double)

Uses the calculated coefficients to solve X for inputted Y

```
public readonly double SolveForX(double y)
```

Parameters

y [double](#)

Y value to solve for

Returns

[double](#)

X value ($x = (y - b) / m$)

ToFunc(LeastSquares)

Casts a LeastSquares to a Function [Solve\(double\)](#).

```
public static Func<double, double> ToFunc(LeastSquares ls)
```

Parameters

ls [LeastSquares](#)

The LeastSquares instance

Returns

[Func](#)<[double](#), [double](#)>

Represents the result of a LeastSquares calculation of the form $y = mX + b$

ToString()

[ToString\(\)](#)

```
public override readonly string ToString()
```

Returns

[string](#)

ToString(IFormatProvider)

```
public readonly string ToString(IFormatProvider provider)
```

Parameters

provider [IFormatProvider](#)

Returns

[string](#)

ToString(string)

```
public readonly string ToString(string format)
```

Parameters

format [string](#)

Returns

[string](#)

ToString(string, IFormatProvider)

```
public readonly string ToString(string format, IFormatProvider formatProvider)
```

Parameters

format [string](#)

formatProvider [IFormatProvider](#)

Returns

[string](#)

Operators

operator ==(LeastSquares, LeastSquares)

Equality operator (M, B and RSquared must be equal)

```
public static bool operator ==(LeastSquares lhs, LeastSquares rhs)
```

Parameters

lhs [LeastSquares](#)

rhs [LeastSquares](#)

Returns

[bool](#)

implicit operator Func<double, double>(LeastSquares)

Casts a LeastSquares to a Function [Solve\(double\)](#)

```
public static implicit operator Func<double, double>(LeastSquares ls)
```

Parameters

ls [LeastSquares](#)

The LeastSquares instance

Returns

[Func](#)<[double](#), [double](#)>

Represents the result of a LeastSquares calculation of the form $y = mX + b$

operator !=(LeastSquares, LeastSquares)

Inequality operatory

```
public static bool operator !=(LeastSquares lhs, LeastSquares rhs)
```

Parameters

lhs [LeastSquares](#)

rhs [LeastSquares](#)

Returns

[bool](#) ↗

Struct Range<T>

Namespace: [LinqStatistics](#)

Assembly: LinqStatistics.dll

An ordered pair of values, representing a segment.

```
public struct Range<T> : IFormattable, IEquatable<Range<T>>, IComparable,
IComparable<Range<T>> where T : struct, IComparable<T>, IFormattable, IEquatable<T>
```

Type Parameters

T

An ordered pair of values, representing a segment.

Implements

[IFormattable](#), [IEquatable](#)<[Range](#)<T>>, [IComparable](#), [IComparable](#)<[Range](#)<T>>

Inherited Members

[object.Equals\(object, object\)](#), [object.GetType\(\)](#), [object.ReferenceEquals\(object, object\)](#)

Constructors

Range(T, T)

Initializes a new instance of the [Range](#)<T> struct.

```
public Range(T min, T max)
```

Parameters

min T

The minimal value of segment.

max T

The maximal value of segment.

Properties

Max

Gets the maximal value of segment.

```
public readonly T Max { get; }
```

Property Value

T

The Max.

Min

Gets the minimal value of segment.

```
public readonly T Min { get; }
```

Property Value

T

The Min.

Methods

CompareTo(Range<T>)

[CompareTo\(I\)](#)

```
public readonly int CompareTo(Range<T> other)
```

Parameters

other [Range<T>](#)

Returns

[int ↗](#)

CompareTo(object)

[CompareTo\(object\) ↗](#)

```
public int CompareTo(object obj)
```

Parameters

obj [object ↗](#)

Returns

[int ↗](#)

Equals(Range<T>)

Indicates whether the current object is equal to another object of the same type.

```
public readonly bool Equals(Range<T> other)
```

Parameters

other [Range<T>](#)

An object to compare with this object.

Returns

[bool ↗](#)

true if the current object is equal to the **other** parameter; otherwise, false.

Equals(object)

Indicates whether this instance and a specified object are equal.

```
public override readonly bool Equals(object obj)
```

Parameters

obj [object](#)

Another object to compare to.

Returns

[bool](#)

true if **obj** and this instance are the same type and represent the same value; otherwise, false.

GetHashCode()

Returns the hash code for this instance.

```
public override readonly int GetHashCode()
```

Returns

[int](#)

A 32-bit signed integer that is the hash code for this instance.

ToString()

Returns a string representation of the range

```
public override readonly string ToString()
```

Returns

[string](#)

A [string](#) representation of the range

ToString(IFormatProvider)

```
public readonly string ToString(IFormatProvider provider)
```

Parameters

provider [IFormatProvider](#)

Returns

[string](#)

ToString(string)

```
public readonly string ToString(string format)
```

Parameters

format [string](#)

Returns

[string](#)

ToString(string, IFormatProvider)

```
public readonly string ToString(string format, IFormatProvider formatProvider)
```

Parameters

format [string](#)

formatProvider [IFormatProvider](#)

Returns

[string](#)

Operators

operator ==(Range<T>, Range<T>)

```
public static bool operator ==(Range<T> first, Range<T> second)
```

Parameters

first [Range](#)<T>

second [Range](#)<T>

Returns

[bool](#)

operator >(Range<T>, Range<T>)

```
public static bool operator >(Range<T> first, Range<T> second)
```

Parameters

first [Range](#)<T>

second [Range](#)<T>

Returns

[bool](#)

operator >=(Range<T>, Range<T>)

```
public static bool operator >=(Range<T> first, Range<T> second)
```

Parameters

first [Range<T>](#)

second [Range<T>](#)

Returns

[bool](#)

operator !=(Range<T>, Range<T>)

```
public static bool operator !=(Range<T> first, Range<T> second)
```

Parameters

first [Range<T>](#)

second [Range<T>](#)

Returns

[bool](#)

operator <(Range<T>, Range<T>)

```
public static bool operator <(Range<T> first, Range<T> second)
```

Parameters

first [Range<T>](#)

second [Range<T>](#)

Returns

[bool](#)

operator <=(Range<T>, Range<T>)

```
public static bool operator <=(Range<T> first, Range<T> second)
```

Parameters

first [Range](#)<T>

second [Range](#)<T>

Returns

[bool](#)