

# Package ‘TidyDensity’

January 9, 2024

**Title** Functions for Tidy Analysis and Generation of Random Data

**Version** 1.3.0

**Description** To make it easy to generate random numbers based upon the underlying stats distribution functions. All data is returned in a tidy and structured format making working with the data simple and straight forward. Given that the data is returned in a tidy 'tibble' it lends itself to working with the rest of the 'tidyverse'.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**URL** <https://github.com/spsanderson/TidyDensity>

**BugReports** <https://github.com/spsanderson/TidyDensity/issues>

**Depends** R (>= 4.1.0)

**Imports** magrittr, rlang (>= 0.4.11), dplyr, ggplot2, plotly, tidyr, purrr, actuar, methods, stats, patchwork, survival, nloptr, broom, tidyselect, data.table

**Suggests** rmarkdown, knitr, roxygen2, EnvStats

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Steven Sanderson [aut, cre, cph]  
(<https://orcid.org/0009-0006-7661-8247>)

**Maintainer** Steven Sanderson <spsanderson@gmail.com>

**Repository** CRAN

**Date/Publication** 2024-01-09 11:10:19 UTC

## R topics documented:

bootstrap_density_augment . . . . .	4
bootstrap_p_augment . . . . .	5
bootstrap_p_vec . . . . .	6

bootstrap_q_augment . . . . .	7
bootstrap_q_vec . . . . .	8
bootstrap_stat_plot . . . . .	9
bootstrap_unnest_tbl . . . . .	10
cgmean . . . . .	11
chmean . . . . .	12
ci_hi . . . . .	13
ci_lo . . . . .	14
ckurtosis . . . . .	15
cmean . . . . .	16
cmedian . . . . .	17
color_blind . . . . .	18
convert_to_ts . . . . .	18
csd . . . . .	19
cskewness . . . . .	20
cvar . . . . .	21
dist_type_extractor . . . . .	22
td_scale_color_colorblind . . . . .	23
td_scale_fill_colorblind . . . . .	23
tidyautoplot . . . . .	24
tidy_bernoulli . . . . .	25
tidy_beta . . . . .	27
tidy_binomial . . . . .	28
tidy_bootstrap . . . . .	30
tidy_burr . . . . .	31
tidy_cauchy . . . . .	32
tidy_chisquare . . . . .	34
tidy_combinedautoplot . . . . .	35
tidy_combine_distributions . . . . .	37
tidy_distribution_comparison . . . . .	38
tidy_distribution_summary_tbl . . . . .	40
tidy_empirical . . . . .	42
tidy_exponential . . . . .	43
tidy_f . . . . .	44
tidy_fourautoplot . . . . .	46
tidy_gamma . . . . .	47
tidy_generalized_beta . . . . .	49
tidy_generalized_pareto . . . . .	50
tidy_geometric . . . . .	52
tidy_hypergeometric . . . . .	53
tidy_inverse_burr . . . . .	55
tidy_inverse_exponential . . . . .	57
tidy_inverse_gamma . . . . .	58
tidy_inverse_normal . . . . .	60
tidy_inverse_pareto . . . . .	61
tidy_inverse_weibull . . . . .	63
tidy_kurtosis_vec . . . . .	64
tidy_logistic . . . . .	65

tidy_lognormal . . . . .	67
tidy_mixture_density . . . . .	68
tidy_multi_dist_autoplot . . . . .	69
tidy_multi_single_dist . . . . .	71
tidy_negative_binomial . . . . .	72
tidy_normal . . . . .	74
tidy_paralogistic . . . . .	75
tidy_pareto . . . . .	77
tidy_pareto1 . . . . .	78
tidy_poisson . . . . .	80
tidy_random_walk . . . . .	81
tidy_random_walk_autoplot . . . . .	82
tidy_range_statistic . . . . .	84
tidy_scale_zero_one_vec . . . . .	85
tidy_skewness_vec . . . . .	86
tidy_stat_tbl . . . . .	87
tidy_t . . . . .	88
tidy_triangular . . . . .	90
tidy_uniform . . . . .	91
tidy_weibull . . . . .	92
tidy_zero_truncated_binomial . . . . .	94
tidy_zero_truncated_geometric . . . . .	95
tidy_zero_truncated_negative_binomial . . . . .	97
tidy_zero_truncated_poisson . . . . .	98
triangle_plot . . . . .	100
util_bernoulli_param_estimate . . . . .	101
util_bernoulli_stats_tbl . . . . .	102
util_beta_param_estimate . . . . .	103
util_beta_stats_tbl . . . . .	104
util_binomial_param_estimate . . . . .	105
util_binomial_stats_tbl . . . . .	107
util_burr_param_estimate . . . . .	108
util_burr_stats_tbl . . . . .	109
util_cauchy_param_estimate . . . . .	110
util_cauchy_stats_tbl . . . . .	111
util_chisquare_stats_tbl . . . . .	112
util_exponential_param_estimate . . . . .	113
util_exponential_stats_tbl . . . . .	115
util_f_stats_tbl . . . . .	116
util_gamma_param_estimate . . . . .	117
util_gamma_stats_tbl . . . . .	118
util_geometric_param_estimate . . . . .	119
util_geometric_stats_tbl . . . . .	120
util_hypergeometric_param_estimate . . . . .	121
util_hypergeometric_stats_tbl . . . . .	123
util_logistic_param_estimate . . . . .	124
util_logistic_stats_tbl . . . . .	125
util_lognormal_param_estimate . . . . .	126

util_lognormal_stats_tbl . . . . .	128
util_negative_binomial_param_estimate . . . . .	129
util_negative_binomial_stats_tbl . . . . .	130
util_normal_param_estimate . . . . .	131
util_normal_stats_tbl . . . . .	133
util_pareto_param_estimate . . . . .	134
util_pareto_stats_tbl . . . . .	135
util_poisson_param_estimate . . . . .	137
util_poisson_stats_tbl . . . . .	138
util_triangular_param_estimate . . . . .	139
util_triangular_stats_tbl . . . . .	140
util_t_stats_tbl . . . . .	141
util_uniform_param_estimate . . . . .	142
util_uniform_stats_tbl . . . . .	144
util_weibull_param_estimate . . . . .	145
util_weibull_stats_tbl . . . . .	146

<b>Index</b>	<b>148</b>
--------------	------------

---

bootstrap\_density\_augment  
*Bootstrap Density Tibble*

---

## Description

Add density information to the output of `tidy_bootstrap()`, and `bootstrap_unnest_tbl()`.

## Usage

```
bootstrap_density_augment(.data)
```

## Arguments

<code>.data</code>	The data that is passed from the <code>tidy_bootstrap()</code> or <code>bootstrap_unnest_tbl()</code> functions.
--------------------	--

## Details

This function takes as input the output of the `tidy_bootstrap()` or `bootstrap_unnest_tbl()` and returns an augmented tibble that has the following columns added to it: `x`, `y`, `dx`, and `dy`.

It looks for an attribute that comes from using `tidy_bootstrap()` or `bootstrap_unnest_tbl()` so it will not work unless the data comes from one of those functions.

## Value

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Bootstrap: [bootstrap\\_p\\_augment\(\)](#), [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_augment\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [bootstrap\\_stat\\_plot\(\)](#), [bootstrap\\_unnest\\_tbl\(\)](#), [tidy\\_bootstrap\(\)](#)

Other Augment Function: [bootstrap\\_p\\_augment\(\)](#), [bootstrap\\_q\\_augment\(\)](#)

**Examples**

```
x <- mtcars$mpg

tidy_bootstrap(x) |>
  bootstrap_density_augment()

tidy_bootstrap(x) |>
  bootstrap_unnest_tbl() |>
  bootstrap_density_augment()
```

---

bootstrap\_p\_augment    *Augment Bootstrap P*

---

**Description**

Takes a numeric vector and will return the ecdf probability.

**Usage**

```
bootstrap_p_augment(.data, .value, .names = "auto")
```

**Arguments**

<code>.data</code>	The data being passed that will be augmented by the function.
<code>.value</code>	This is passed <a href="#">rlang::enquo()</a> to capture the vectors you want to augment.
<code>.names</code>	The default is "auto"

**Details**

Takes a numeric vector and will return the ecdf probability of that vector. This function is intended to be used on its own in order to add columns to a tibble.

**Value**

A augmented tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Augment Function: [bootstrap\\_density\\_augment\(\)](#), [bootstrap\\_q\\_augment\(\)](#)

Other Bootstrap: [bootstrap\\_density\\_augment\(\)](#), [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_augment\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [bootstrap\\_stat\\_plot\(\)](#), [bootstrap\\_unnest\\_tbl\(\)](#), [tidy\\_bootstrap\(\)](#)

**Examples**

```
x <- mtcars$mpg
tidy_bootstrap(x) |>
  bootstrap_unnest_tbl() |>
  bootstrap_p_augment(y)
```

---

bootstrap\_p\_vec

*Compute Bootstrap P of a Vector*

---

**Description**

This function takes in a vector as it's input and will return the ecdf probability of a vector.

**Usage**

```
bootstrap_p_vec(.x)
```

**Arguments**

`.x` A numeric

**Details**

A function to return the ecdf probability of a vector.

**Value**

A vector

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Bootstrap: [bootstrap\\_density\\_augment\(\)](#), [bootstrap\\_p\\_augment\(\)](#), [bootstrap\\_q\\_augment\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [bootstrap\\_stat\\_plot\(\)](#), [bootstrap\\_unnest\\_tbl\(\)](#), [tidy\\_bootstrap\(\)](#)

Other Vector Function: [bootstrap\\_q\\_vec\(\)](#), [cgmean\(\)](#), [chmean\(\)](#), [ckurtosis\(\)](#), [cmean\(\)](#), [cmedian\(\)](#), [csd\(\)](#), [cskewness\(\)](#), [cvar\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_scale\\_zero\\_one\\_vec\(\)](#), [tidy\\_skewness\\_vec\(\)](#)

**Examples**

```
x <- mtcars$mpg
bootstrap_p_vec(x)
```

---

bootstrap\_q\_augment     *Augment Bootstrap Q*

---

**Description**

Takes a numeric vector and will return the quantile.

**Usage**

```
bootstrap_q_augment(.data, .value, .names = "auto")
```

**Arguments**

<code>.data</code>	The data being passed that will be augmented by the function.
<code>.value</code>	This is passed <a href="#">rlang::enquo()</a> to capture the vectors you want to augment.
<code>.names</code>	The default is "auto"

**Details**

Takes a numeric vector and will return the quantile of that vector. This function is intended to be used on its own in order to add columns to a tibble.

**Value**

A augmented tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Augment Function: [bootstrap\\_density\\_augment\(\)](#), [bootstrap\\_p\\_augment\(\)](#)

Other Bootstrap: [bootstrap\\_density\\_augment\(\)](#), [bootstrap\\_p\\_augment\(\)](#), [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [bootstrap\\_stat\\_plot\(\)](#), [bootstrap\\_unnest\\_tbl\(\)](#), [tidy\\_bootstrap\(\)](#)

## Examples

```
x <- mtcars$mpg

tidy_bootstrap(x) |>
  bootstrap_unnest_tbl() |>
  bootstrap_q_augment(y)
```

---

bootstrap_q_vec	<i>Compute Bootstrap Q of a Vector</i>
-----------------	--

---

## Description

This function takes in a vector as it's input and will return the quantile of a vector.

## Usage

```
bootstrap_q_vec(.x)
```

## Arguments

`.x` A numeric

## Details

A function to return the quantile of a vector.

## Value

A vector

## Author(s)

Steven P. Sanderson II, MPH

## See Also

Other Bootstrap: [bootstrap\\_density\\_augment\(\)](#), [bootstrap\\_p\\_augment\(\)](#), [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_augment\(\)](#), [bootstrap\\_stat\\_plot\(\)](#), [bootstrap\\_unnest\\_tbl\(\)](#), [tidy\\_bootstrap\(\)](#)

Other Vector Function: [bootstrap\\_p\\_vec\(\)](#), [cgmean\(\)](#), [chmean\(\)](#), [ckurtosis\(\)](#), [cmean\(\)](#), [cmedian\(\)](#), [csd\(\)](#), [cskewness\(\)](#), [cvar\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_scale\\_zero\\_one\\_vec\(\)](#), [tidy\\_skewness\\_vec\(\)](#)

## Examples

```
x <- mtcars$mpg

bootstrap_q_vec(x)
```



---

bootstrap\_stat\_plot *Bootstrap Stat Plot*

---

## Description

This function produces a plot of a cumulative statistic function applied to the bootstrap variable from `tidy_bootstrap()` or after `bootstrap_unnest_tbl()` has been applied to it.

## Usage

```
bootstrap_stat_plot(  
  .data,  
  .value,  
  .stat = "cmean",  
  .show_groups = FALSE,  
  .show_ci_labels = TRUE,  
  .interactive = FALSE  
)
```

## Arguments

<code>.data</code>	The data that comes from either <code>tidy_bootstrap()</code> or after <code>bootstrap_unnest_tbl()</code> is applied to it.
<code>.value</code>	The value column that the calculations are being applied to.
<code>.stat</code>	The cumulative statistic function being applied to the <code>.value</code> column. It must be quoted. The default is "cmean".
<code>.show_groups</code>	The default is FALSE, set to TRUE to get output of all simulations of the bootstrap data.
<code>.show_ci_labels</code>	The default is TRUE, this will show the last value of the upper and lower quantile.
<code>.interactive</code>	The default is FALSE, set to TRUE to get a plotly plot object back.

## Details

This function will take in data from either `tidy_bootstrap()` directly or after apply `bootstrap_unnest_tbl()` to its output. There are several different cumulative functions that can be applied to the data. The accepted values are:

- "cmean" - Cumulative Mean
- "chmean" - Cumulative Harmonic Mean
- "cgmean" - Cumulative Geometric Mean
- "csum" = Cumulative Sum
- "cmedian" = Cumulative Median

- "cmax" = Cumulative Max
- "cmin" = Cumulative Min
- "cprod" = Cumulative Product
- "csd" = Cumulative Standard Deviation
- "cvar" = Cumulative Variance
- "c skewness" = Cumulative Skewness
- "ckurtosis" = Cumulative Kurtosis

**Value**

A plot either ggplot2 or plotly.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Bootstrap: [bootstrap\\_density\\_augment\(\)](#), [bootstrap\\_p\\_augment\(\)](#), [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_augment\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [bootstrap\\_unnest\\_tbl\(\)](#), [tidy\\_bootstrap\(\)](#)

Other Autoplot: [tidyautoplot\(\)](#), [tidy\\_combinedautoplot\(\)](#), [tidy\\_fourautoplot\(\)](#), [tidy\\_multi\\_distautoplot\(\)](#), [tidy\\_random\\_walkautoplot\(\)](#)

**Examples**

```
x <- mtcars$mpg

tidy_bootstrap(x) |>
  bootstrap_stat_plot(y, "cmean")

tidy_bootstrap(x, .num_sims = 10) |>
  bootstrap_stat_plot(y,
    .stat = "chmean", .show_groups = TRUE,
    .show_ci_label = FALSE
  )
```

---

bootstrap\_unnest\_tbl *Unnest Tidy Bootstrap Tibble*

---

**Description**

Unnest the data output from `tidy_bootstrap()`.

**Usage**

```
bootstrap_unnest_tbl(.data)
```

**Arguments**

`.data` The data that is passed from the `tidy_bootstrap()` function.

**Details**

This function takes as input the output of the `tidy_bootstrap()` function and returns a two column tibble. The columns are `sim_number` and `y`

It looks for an attribute that comes from using `tidy_bootstrap()` so it will not work unless the data comes from that function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Bootstrap: [bootstrap\\_density\\_augment\(\)](#), [bootstrap\\_p\\_augment\(\)](#), [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_augment\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [bootstrap\\_stat\\_plot\(\)](#), [tidy\\_bootstrap\(\)](#)

**Examples**

```
tb <- tidy_bootstrap(.x = mtcars$mpg)
bootstrap_unnest_tbl(tb)

bootstrap_unnest_tbl(tb) |>
  tidy_distribution_summary_tbl(sim_number)
```

---

cgmean

*Cumulative Geometric Mean*

---

**Description**

A function to return the cumulative geometric mean of a vector.

**Usage**

```
cgmean(.x)
```

**Arguments**

`.x` A numeric vector

**Details**

A function to return the cumulative geometric mean of a vector. `exp(cummean(log(.x)))`

**Value**

A numeric vector

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Vector Function: [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [chmean\(\)](#), [ckurtosis\(\)](#), [cmean\(\)](#), [cmedian\(\)](#), [csd\(\)](#), [cskewness\(\)](#), [cvar\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_scale\\_zero\\_one\\_vec\(\)](#), [tidy\\_skewness\\_vec\(\)](#)

**Examples**

```
x <- mtcars$mpg  
  
cgmean(x)
```

---

chmean

*Cumulative Harmonic Mean*

---

**Description**

A function to return the cumulative harmonic mean of a vector.

**Usage**

```
chmean(.x)
```

**Arguments**

`.x` A numeric vector

**Details**

A function to return the cumulative harmonic mean of a vector. `1 / (cumsum(1 / .x))`

**Value**

A numeric vector

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Vector Function: [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [cgmean\(\)](#), [ckurtosis\(\)](#), [cmean\(\)](#), [cmedian\(\)](#), [csd\(\)](#), [cskewness\(\)](#), [cvar\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_scale\\_zero\\_one\\_vec\(\)](#), [tidy\\_skewness\\_vec\(\)](#)

**Examples**

```
x <- mtcars$mpg
chmean(x)
```

---

ci\_hi

*Confidence Interval Generic*

---

**Description**

Gets the upper 97.5% quantile of a numeric vector.

**Usage**

```
ci_hi(.x, .na_rm = FALSE)
```

**Arguments**

`.x` A vector of numeric values  
`.na_rm` A Boolean, defaults to FALSE. Passed to the quantile function.

**Details**

Gets the upper 97.5% quantile of a numeric vector.

**Value**

A numeric value.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Statistic: [ci\\_lo\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_range\\_statistic\(\)](#), [tidy\\_skewness\\_vec\(\)](#), [tidy\\_stat\\_tbl\(\)](#)

**Examples**

```
x <- mtcars$mpg
ci_hi(x)
```

---

`ci_lo`*Confidence Interval Generic*

---

**Description**

Gets the lower 2.5% quantile of a numeric vector.

**Usage**

```
ci_lo(.x, .na_rm = FALSE)
```

**Arguments**

`.x` A vector of numeric values  
`.na_rm` A Boolean, defaults to FALSE. Passed to the quantile function.

**Details**

Gets the lower 2.5% quantile of a numeric vector.

**Value**

A numeric value.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Statistic: [ci\\_hi\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_range\\_statistic\(\)](#), [tidy\\_skewness\\_vec\(\)](#), [tidy\\_stat\\_tbl\(\)](#)

**Examples**

```
x <- mtcars$mpg
ci_lo(x)
```

---

ckurtosis	<i>Cumulative Kurtosis</i>
-----------	----------------------------

---

**Description**

A function to return the cumulative kurtosis of a vector.

**Usage**

```
ckurtosis(.x)
```

**Arguments**

`.x` A numeric vector

**Details**

A function to return the cumulative kurtosis of a vector.

**Value**

A numeric vector

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Vector Function: [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [cgmean\(\)](#), [chmean\(\)](#), [cmean\(\)](#), [cmedian\(\)](#), [csd\(\)](#), [cskewness\(\)](#), [cvar\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_scale\\_zero\\_one\\_vec\(\)](#), [tidy\\_skewness\\_vec\(\)](#)

**Examples**

```
x <- mtcars$mpg  
  
ckurtosis(x)
```

---

`cmean`*Cumulative Mean*

---

**Description**

A function to return the cumulative mean of a vector.

**Usage**

```
cmean(.x)
```

**Arguments**

`.x` A numeric vector

**Details**

A function to return the cumulative mean of a vector. It uses `dplyr::cummean()` as the basis of the function.

**Value**

A numeric vector

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Vector Function: [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [cgmean\(\)](#), [chmean\(\)](#), [ckurtosis\(\)](#), [cmedian\(\)](#), [csd\(\)](#), [cskewness\(\)](#), [cvar\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_scale\\_zero\\_one\\_vec\(\)](#), [tidy\\_skewness\\_vec\(\)](#)

**Examples**

```
x <- mtcars$mpg
```

```
cmean(x)
```



---

cmedian	<i>Cumulative Median</i>
---------	--------------------------

---

### Description

A function to return the cumulative median of a vector.

### Usage

```
cmedian(.x)
```

### Arguments

`.x` A numeric vector

### Details

A function to return the cumulative median of a vector.

### Value

A numeric vector

### Author(s)

Steven P. Sanderson II, MPH

### See Also

Other Vector Function: [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [cgmean\(\)](#), [chmean\(\)](#), [ckurtosis\(\)](#), [cmean\(\)](#), [csd\(\)](#), [cskewness\(\)](#), [cvar\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_scale\\_zero\\_one\\_vec\(\)](#), [tidy\\_skewness\\_vec\(\)](#)

### Examples

```
x <- mtcars$mpg  
  
cmedian(x)
```

---

color_blind	<i>Provide Colorblind Compliant Colors</i>
-------------	--

---

**Description**

8 Hex RGB color definitions suitable for charts for colorblind people.

**Usage**

```
color_blind()
```

---

convert_to_ts	<i>Convert Data to Time Series Format</i>
---------------	---

---

**Description**

This function converts data in a data frame or tibble into a time series format. It is designed to work with data generated from `tidy_` distribution functions. The function can return time series data, pivot it into long format, or both.

**Usage**

```
convert_to_ts(.data, .return_ts = TRUE, .pivot_longer = FALSE)
```

**Arguments**

<code>.data</code>	A data frame or tibble to be converted into a time series format.
<code>.return_ts</code>	A logical value indicating whether to return the time series data. Default is TRUE.
<code>.pivot_longer</code>	A logical value indicating whether to pivot the data into long format. Default is FALSE.

**Details**

The function takes a data frame or tibble as input and processes it based on the specified options. It performs the following actions:

1. Checks if the input is a data frame or tibble; otherwise, it raises an error.
2. Checks if the data comes from a `tidy_` distribution function; otherwise, it raises an error.
3. Converts the data into a time series format, grouping it by "sim\_number" and transforming the "y" column into a time series.
4. Returns the result based on the chosen options:
  - If `ret_ts` is set to TRUE, it returns the time series data.
  - If `pivot_longer` is set to TRUE, it pivots the data into long format.
  - If both options are set to FALSE, it returns the data as a tibble.

**Value**

The function returns the processed data based on the chosen options:

- If `ret_ts` is set to `TRUE`, it returns time series data.
- If `pivot_longer` is set to `TRUE`, it returns the data in long format.
- If both options are set to `FALSE`, it returns the data as a tibble.

**Author(s)**

Steven P. Sanderson II, MPH

**Examples**

```
# Example 1: Convert data to time series format without returning time series data
x <- tidy_normal()
result <- convert_to_ts(x, FALSE)
head(result)

# Example 2: Convert data to time series format and pivot it into long format
x <- tidy_normal()
result <- convert_to_ts(x, FALSE, TRUE)
head(result)

# Example 3: Convert data to time series format and return the time series data
x <- tidy_normal()
result <- convert_to_ts(x)
head(result)
```

---

csd

*Cumulative Standard Deviation*

---

**Description**

A function to return the cumulative standard deviation of a vector.

**Usage**

```
csd(.x)
```

**Arguments**

`.x` A numeric vector

**Details**

A function to return the cumulative standard deviation of a vector.

**Value**

A numeric vector. Note: The first entry will always be NaN.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Vector Function: [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [cgmean\(\)](#), [chmean\(\)](#), [ckurtosis\(\)](#), [cmean\(\)](#), [cmedian\(\)](#), [cskewness\(\)](#), [cvar\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_scale\\_zero\\_one\\_vec\(\)](#), [tidy\\_skewness\\_vec\(\)](#)

**Examples**

```
x <- mtcars$mpg  
  
csd(x)
```

---

cskewness

*Cumulative Skewness*

---

**Description**

A function to return the cumulative skewness of a vector.

**Usage**

```
cskewness(.x)
```

**Arguments**

.x                    A numeric vector

**Details**

A function to return the cumulative skewness of a vector.

**Value**

A numeric vector

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Vector Function: [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [cgmean\(\)](#), [chmean\(\)](#), [ckurtosis\(\)](#), [cmean\(\)](#), [cmedian\(\)](#), [csd\(\)](#), [cvar\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_scale\\_zero\\_one\\_vec\(\)](#), [tidy\\_skewness\\_vec\(\)](#)

**Examples**

```
x <- mtcars$mpg
cskewness(x)
```

---

cvar	<i>Cumulative Variance</i>
------	----------------------------

---

**Description**

A function to return the cumulative variance of a vector.

**Usage**

```
cvar(.x)
```

**Arguments**

.x                    A numeric vector

**Details**

A function to return the cumulative variance of a vector. `exp(cummean(log(.x)))`

**Value**

A numeric vector. Note: The first entry will always be NaN.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Vector Function: [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [cgmean\(\)](#), [chmean\(\)](#), [ckurtosis\(\)](#), [cmean\(\)](#), [cmedian\(\)](#), [csd\(\)](#), [cskewness\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_scale\\_zero\\_one\\_vec\(\)](#), [tidy\\_skewness\\_vec\(\)](#)

**Examples**

```
x <- mtcars$mpg
cvar(x)
```

---

dist\_type\_extractor     *Extract Distribution Type from Tidy Distribution Object*

---

### Description

Get the distribution name in title case from the tidy\_ distribution function.

### Usage

```
dist_type_extractor(.x)
```

### Arguments

.x                    The attribute list passed from a tidy\_ distribution function.

### Details

This will extract the distribution type from a tidy\_ distribution function output using the attributes of that object. You must pass the attribute directly to the function. It is meant really to be used internally.

You should be passing if using manually the \$tibble\_type attribute.

### Value

A character string

### Author(s)

Steven P. Sanderson II,

### Examples

```
tn <- tidy_normal()
atb <- attributes(tn)
dist_type_extractor(atb$tibble_type)
```

---

`td_scale_color_colorblind`*Provide Colorblind Compliant Colors*

---

**Description**

Provide Colorblind Compliant Colors

**Usage**

```
td_scale_color_colorblind(..., theme = "td")
```

**Arguments**

<code>...</code>	Data passed to the function
<code>theme</code>	This defaults to <code>td</code> and that is the only allowed value

---

`td_scale_fill_colorblind`*Provide Colorblind Compliant Colors*

---

**Description**

Provide Colorblind Compliant Colors

**Usage**

```
td_scale_fill_colorblind(..., theme = "td")
```

**Arguments**

<code>...</code>	Data passed to the function
<code>theme</code>	This defaults to <code>td</code> and that is the only allowed value

tidy\_autoplot

*Automatic Plot of Density Data***Description**

This is an auto plotting function that will take in a tidy\_ distribution function and a few arguments, one being the plot type, which is a quoted string of one of the following:

- density
- quantile
- probablity
- qq
- mcmc

If the number of simulations exceeds 9 then the legend will not print. The plot subtitle is put together by the attributes of the table passed to the function.

**Usage**

```
tidy_autoplot(
  .data,
  .plot_type = "density",
  .line_size = 0.5,
  .geom_point = FALSE,
  .point_size = 1,
  .geom_rug = FALSE,
  .geom_smooth = FALSE,
  .geom_jitter = FALSE,
  .interactive = FALSE
)
```

**Arguments**

.data	The data passed in from a tidy_distribution function like tidy_normal()
.plot_type	This is a quoted string like 'density'
.line_size	The size param ggplot
.geom_point	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return a plot with ggplot2::geom_point()
.point_size	The point size param for ggplot
.geom_rug	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return the use of ggplot2::geom_rug()
.geom_smooth	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return the use of ggplot2::geom_smooth() The aes parameter of group is set to FALSE. This ensures a single smoothing band returned with SE also set to FALSE. Color is set to 'black' and linetype is 'dashed'.



- `.geom_jitter` A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return the use of `ggplot2::geom_jitter()`
- `.interactive` A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return an interactive plotly plot.

### Details

This function will spit out one of the following plots:

- density
- quantile
- probability
- qq
- mcmc

### Value

A ggplot or a plotly plot.

### Author(s)

Steven P. Sanderson II, MPH

### See Also

Other Autoplot: [bootstrap\\_stat\\_plot\(\)](#), [tidy\\_combined\\_autoplot\(\)](#), [tidy\\_four\\_autoplot\(\)](#), [tidy\\_multi\\_dist\\_autoplot\(\)](#), [tidy\\_random\\_walk\\_autoplot\(\)](#)

### Examples

```
tidy_normal(.num_sims = 5) |>
  tidy_autoplot()

tidy_normal(.num_sims = 20) |>
  tidy_autoplot(.plot_type = "qq")
```

---

tidy\_bernoulli

*Tidy Randomly Generated Bernoulli Distribution Tibble*

---

### Description

This function will generate  $n$  random points from a Bernoulli distribution with a user provided, `.prob`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the  $x$  column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of `n` for the current simulation.
- `y` The randomly generated data point.
- `dx` The `x` value from the `stats::density()` function.
- `dy` The `y` value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

### Usage

```
tidy_bernoulli(.n = 50, .prob = 0.1, .num_sims = 1, .return_tibble = TRUE)
```

### Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.prob</code>	The probability of success/failure.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

### Details

This function uses the `rbinom()`, and its underlying `p`, `d`, and `q` functions. The *Bernoulli* distribution is a special case of the *Binomial* distribution with `size = 1` hence this is why the `binom` functions are used and set to `size = 1`.

### Value

A tibble of randomly generated data.

### Author(s)

Steven P. Sanderson II, MPH

### See Also

[https://en.wikipedia.org/wiki/Bernoulli\\_distribution](https://en.wikipedia.org/wiki/Bernoulli_distribution)

Other Discrete Distribution: `tidy_binomial()`, `tidy_hypergeometric()`, `tidy_negative_binomial()`, `tidy_poisson()`, `tidy_zero_truncated_binomial()`, `tidy_zero_truncated_negative_binomial()`, `tidy_zero_truncated_poisson()`

Other Bernoulli: `util_bernoulli_param_estimate()`, `util_bernoulli_stats_tbl()`

### Examples

```
tidy_bernoulli()
```

## Description

This function will generate  $n$  random points from a beta distribution with a user provided, `.shape1`, `.shape2`, `.ncp` or non-centrality parameter, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

## Usage

```
tidy_beta(  
  .n = 50,  
  .shape1 = 1,  
  .shape2 = 1,  
  .ncp = 0,  
  .num_sims = 1,  
  .return_tibble = TRUE  
)
```

## Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.shape1</code>	A non-negative parameter of the Beta distribution.
<code>.shape2</code>	A non-negative parameter of the Beta distribution.
<code>.ncp</code>	The non-centrality parameter of the Beta distribution.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

## Details

This function uses the underlying `stats::rbeta()`, and its underlying `p`, `d`, and `q` functions. For more information please see `stats::rbeta()`

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://statisticsglobe.com/beta-distribution-in-r-dbeta-pbeta-qbeta-rbeta>

[https://en.wikipedia.org/wiki/Beta\\_distribution](https://en.wikipedia.org/wiki/Beta_distribution)

Other Continuous Distribution: `tidy_burr()`, `tidy_cauchy()`, `tidy_chisquare()`, `tidy_exponential()`, `tidy_f()`, `tidy_gamma()`, `tidy_generalized_beta()`, `tidy_generalized_pareto()`, `tidy_geometric()`, `tidy_inverse_burr()`, `tidy_inverse_exponential()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_inverse_weibull()`, `tidy_logistic()`, `tidy_lognormal()`, `tidy_normal()`, `tidy_paralogistic()`, `tidy_pareto1()`, `tidy_pareto()`, `tidy_triangular()`, `tidy_t()`, `tidy_uniform()`, `tidy_weibull()`, `tidy_zero_truncated_geometric()`

Other Beta: `tidy_generalized_beta()`, `util_beta_param_estimate()`, `util_beta_stats_tbl()`

**Examples**

```
tidy_beta()
```

---

tidy\_binomial

*Tidy Randomly Generated Binomial Distribution Tibble*

---

**Description**

This function will generate  $n$  random points from a binomial distribution with a user provided, `.size`, `.prob`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

**Usage**

```
tidy_binomial(  
  .n = 50,  
  .size = 0,  
  .prob = 1,  
  .num_sims = 1,  
  .return_tibble = TRUE  
)
```

**Arguments**

<code>.n</code>	The number of randomly generated points you want.
<code>.size</code>	Number of trials, zero or more.
<code>.prob</code>	Probability of success on each trial.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `stats::rbinom()`, and its underlying `p`, `d`, and `q` functions. For more information please see [stats::rbinom\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://www.itl.nist.gov/div898/handbook/eda/section3/eda366i.htm>

Other Discrete Distribution: [tidy\\_bernoulli\(\)](#), [tidy\\_hypergeometric\(\)](#), [tidy\\_negative\\_binomial\(\)](#), [tidy\\_poisson\(\)](#), [tidy\\_zero\\_truncated\\_binomial\(\)](#), [tidy\\_zero\\_truncated\\_negative\\_binomial\(\)](#), [tidy\\_zero\\_truncated\\_poisson\(\)](#)

Other Binomial: [tidy\\_negative\\_binomial\(\)](#), [tidy\\_zero\\_truncated\\_binomial\(\)](#), [tidy\\_zero\\_truncated\\_negative\\_binomial\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#)

**Examples**

```
tidy_binomial()
```

---

`tidy_bootstrap`*Bootstrap Empirical Data*

---

## Description

Takes an input vector of numeric data and produces a bootstrapped nested tibble by simulation number.

## Usage

```
tidy_bootstrap(  
  .x,  
  .num_sims = 2000,  
  .proportion = 0.8,  
  .distribution_type = "continuous"  
)
```

## Arguments

<code>.x</code>	The vector of data being passed to the function. Must be a numeric vector.
<code>.num_sims</code>	The default is 2000, can be set to anything desired. A warning will pass to the console if the value is less than 2000.
<code>.proportion</code>	How much of the original data do you want to pass through to the sampling function. The default is 0.80 (80%)
<code>.distribution_type</code>	This can either be 'continuous' or 'discrete'

## Details

This function will take in a numeric input vector and produce a tibble of bootstrapped values in a list. The table that is output will have two columns: `sim_number` and `bootstrap_samples`

The `sim_number` corresponds to how many times you want the data to be resampled, and the `bootstrap_samples` column contains a list of the bootstrapped resampled data.

## Value

A nested tibble

## Author(s)

Steven P. Sanderson II, MPH

## See Also

Other Bootstrap: [bootstrap\\_density\\_augment\(\)](#), [bootstrap\\_p\\_augment\(\)](#), [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_augment\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [bootstrap\\_stat\\_plot\(\)](#), [bootstrap\\_unnest\\_tbl\(\)](#)

## Examples

```
x <- mtcars$mpg
tidy_bootstrap(x)
```

---

tidy\_burr

*Tidy Randomly Generated Burr Distribution Tibble*

---

## Description

This function will generate  $n$  random points from a Burr distribution with a user provided, `.shape1`, `.shape2`, `.scale`, `.rate`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

## Usage

```
tidy_burr(  
  .n = 50,  
  .shape1 = 1,  
  .shape2 = 1,  
  .rate = 1,  
  .scale = 1/.rate,  
  .num_sims = 1,  
  .return_tibble = TRUE  
)
```

## Arguments

- |                      |   |
|----------------------|---|
| <code>.n</code>      | The number of randomly generated points you want.       |
| <code>.shape1</code> | Must be strictly positive.                              |
| <code>.shape2</code> | Must be strictly positive.                              |
| <code>.rate</code>   | An alternative way to specify the <code>.scale</code> . |

.scale            Must be strictly positive.  
 .num\_sims        The number of randomly generated simulations you want.  
 .return\_tibble   A logical value indicating whether to return the result as a tibble. Default is TRUE.

### Details

This function uses the underlying `actuar::rburr()`, and its underlying `p`, `d`, and `q` functions. For more information please see [actuar::rburr\(\)](#)

### Value

A tibble of randomly generated data.

### Author(s)

Steven P. Sanderson II, MPH

### See Also

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Continuous Distribution: [tidy\\_beta\(\)](#), [tidy\\_cauchy\(\)](#), [tidy\\_chisquare\(\)](#), [tidy\\_exponential\(\)](#), [tidy\\_f\(\)](#), [tidy\\_gamma\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_geometric\(\)](#), [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_logistic\(\)](#), [tidy\\_lognormal\(\)](#), [tidy\\_normal\(\)](#), [tidy\\_paralogistic\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [tidy\\_triangular\(\)](#), [tidy\\_t\(\)](#), [tidy\\_uniform\(\)](#), [tidy\\_weibull\(\)](#), [tidy\\_zero\\_truncated\\_geometric\(\)](#)

Other Burr: [tidy\\_inverse\\_burr\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#)

### Examples

```
tidy_burr()
```

---

tidy\_cauchy

*Tidy Randomly Generated Cauchy Distribution Tibble*

---

### Description

This function will generate `n` random points from a cauchy distribution with a user provided, `.location`, `.scale`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the `n` randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.



- x The current value of n for the current simulation.
- y The randomly generated data point.
- dx The x value from the `stats::density()` function.
- dy The y value from the `stats::density()` function.
- p The values from the resulting p\_ function of the distribution family.
- q The values from the resulting q\_ function of the distribution family.

### Usage

```
tidy_cauchy(
  .n = 50,
  .location = 0,
  .scale = 1,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

### Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.location</code>	The location parameter.
<code>.scale</code>	The scale parameter, must be greater than or equal to 0.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

### Details

This function uses the underlying `stats::rcauchy()`, and its underlying p, d, and q functions. For more information please see `stats::rcauchy()`

### Value

A tibble of randomly generated data.

### Author(s)

Steven P. Sanderson II, MPH

### See Also

<https://www.itl.nist.gov/div898/handbook/eda/section3/eda3663.htm>

Other Continuous Distribution: `tidy_beta()`, `tidy_burr()`, `tidy_chisquare()`, `tidy_exponential()`, `tidy_f()`, `tidy_gamma()`, `tidy_generalized_beta()`, `tidy_generalized_pareto()`, `tidy_geometric()`, `tidy_inverse_burr()`, `tidy_inverse_exponential()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_inverse_weibull()`, `tidy_logistic()`, `tidy_lognormal()`, `tidy_normal()`,

`tidy_paralogistic()`, `tidy_pareto1()`, `tidy_pareto()`, `tidy_triangular()`, `tidy_t()`, `tidy_uniform()`,  
`tidy_weibull()`, `tidy_zero_truncated_geometric()`

Other Cauchy: `util_cauchy_param_estimate()`, `util_cauchy_stats_tbl()`

## Examples

```
tidy_cauchy()
```

---

<code>tidy_chisquare</code>	<i>Tidy Randomly Generated Chisquare (Non-Central) Distribution Tibble</i>
-----------------------------	--

---

## Description

This function will generate `n` random points from a chisquare distribution with a user provided, `.df`, `.ncp`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the `n` randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of `n` for the current simulation.
- `y` The randomly generated data point.
- `dx` The `x` value from the `stats::density()` function.
- `dy` The `y` value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

## Usage

```
tidy_chisquare(  
  .n = 50,  
  .df = 1,  
  .ncp = 1,  
  .num_sims = 1,  
  .return_tibble = TRUE  
)
```

## Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.df</code>	Degrees of freedom (non-negative but can be non-integer)
<code>.ncp</code>	Non-centrality parameter, must be non-negative.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `stats::rchisq()`, and its underlying p, d, and q functions. For more information please see [stats::rchisq\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://www.itl.nist.gov/div898/handbook/eda/section3/eda3666.htm>

Other Continuous Distribution: [tidy\\_beta\(\)](#), [tidy\\_burr\(\)](#), [tidy\\_cauchy\(\)](#), [tidy\\_exponential\(\)](#), [tidy\\_f\(\)](#), [tidy\\_gamma\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_geometric\(\)](#), [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_logistic\(\)](#), [tidy\\_lognormal\(\)](#), [tidy\\_normal\(\)](#), [tidy\\_paralogistic\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [tidy\\_triangular\(\)](#), [tidy\\_t\(\)](#), [tidy\\_uniform\(\)](#), [tidy\\_weibull\(\)](#), [tidy\\_zero\\_truncated\\_geometric\(\)](#)

Other Chisquare: [util\\_chisquare\\_stats\\_tbl\(\)](#)

**Examples**

```
tidy_chisquare()
```

---

```
tidy_combinedautoplot
```

*Automatic Plot of Combined Multi Dist Data*

---

**Description**

This is an auto plotting function that will take in a `tidy_` distribution function and a few arguments, one being the plot type, which is a quoted string of one of the following:

- density
- quantile
- probability
- qq

If the number of simulations exceeds 9 then the legend will not print. The plot subtitle is put together by the attributes of the table passed to the function.

**Usage**

```
tidy_combined_autoplot(
  .data,
  .plot_type = "density",
  .line_size = 0.5,
  .geom_point = FALSE,
  .point_size = 1,
  .geom_rug = FALSE,
  .geom_smooth = FALSE,
  .geom_jitter = FALSE,
  .interactive = FALSE
)
```

**Arguments**

<code>.data</code>	The data passed in from a the function <code>tidy_multi_dist()</code>
<code>.plot_type</code>	This is a quoted string like 'density'
<code>.line_size</code>	The size param ggplot
<code>.geom_point</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return a plot with <code>ggplot2::geom_point()</code>
<code>.point_size</code>	The point size param for ggplot
<code>.geom_rug</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return the use of <code>ggplot2::geom_rug()</code>
<code>.geom_smooth</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return the use of <code>ggplot2::geom_smooth()</code> The aes parameter of group is set to FALSE. This ensures a single smoothing band returned with SE also set to FALSE. Color is set to 'black' and linetype is 'dashed'.
<code>.geom_jitter</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return the use of <code>ggplot2::geom_jitter()</code>
<code>.interactive</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return an interactive plotly plot.

**Details**

This function will spit out one of the following plots:

- density
- quantile
- probability
- qq

**Value**

A ggplot or a plotly plot.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Autoplot: [bootstrap\\_stat\\_plot\(\)](#), [tidy\\_autoplot\(\)](#), [tidy\\_four\\_autoplot\(\)](#), [tidy\\_multi\\_dist\\_autoplot\(\)](#), [tidy\\_random\\_walk\\_autoplot\(\)](#)

**Examples**

```
combined_tbl <- tidy_combine_distributions(  
  tidy_normal(),  
  tidy_gamma(),  
  tidy_beta()  
)  
  
combined_tbl  
  
combined_tbl |>  
  tidy_combined_autoplot()  
  
combined_tbl |>  
  tidy_combined_autoplot(.plot_type = "qq")
```

---

tidy\_combine\_distributions

*Combine Multiple Tidy Distributions of Different Types*

---

**Description**

This allows a user to specify any n number of tidy\_ distributions that can be combined into a single tibble. This is the preferred method for combining multiple distributions of different types, for example a Gaussian distribution and a Beta distribution.

This generates a single tibble with an added column of dist\_type that will give the distribution family name and its associated parameters.

**Usage**

```
tidy_combine_distributions(...)
```

**Arguments**

...                   The ... is where you can place your different distributions

**Details**

Allows a user to generate a tibble of different tidy\_ distributions

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Multiple Distribution: [tidy\\_multi\\_single\\_dist\(\)](#)

**Examples**

```
tn <- tidy_normal()
tb <- tidy_beta()
tc <- tidy_cauchy()

tidy_combine_distributions(tn, tb, tc)

## OR

tidy_combine_distributions(
  tidy_normal(),
  tidy_beta(),
  tidy_cauchy(),
  tidy_logistic()
)
```

---

tidy\_distribution\_comparison

*Compare Empirical Data to Distributions*

---

**Description**

Compare some empirical data set against different distributions to help find the distribution that could be the best fit.

**Usage**

```
tidy_distribution_comparison(
  .x,
  .distribution_type = "continuous",
  .round_to_place = 3
)
```

## Arguments

- `.x` The data set being passed to the function
- `.distribution_type` What kind of data is it, can be one of continuous or discrete
- `.round_to_place` How many decimal places should the parameter estimates be rounded off to for distribution construction. The default is 3

## Details

The purpose of this function is to take some data set provided and to try to find a distribution that may fit the best. A parameter of `.distribution_type` must be set to either continuous or discrete in order for this the function to try the appropriate types of distributions.

The following distributions are used:

Continuous:

- `tidy_beta`
- `tidy_cauchy`
- `tidy_exponential`
- `tidy_gamma`
- `tidy_logistic`
- `tidy_lognormal`
- `tidy_normal`
- `tidy_pareto`
- `tidy_uniform`
- `tidy_weibull`

Discrete:

- `tidy_binomial`
- `tidy_geometric`
- `tidy_hypergeometric`
- `tidy_poisson`

The function itself returns a list output of tibbles. Here are the tibbles that are returned:

- `comparison_tbl`
- `deviance_tbl`
- `total_deviance_tbl`
- `aic_tbl`
- `kolmogorov_smirnov_tbl`
- `multi_metric_tbl`

The `comparison_tbl` is a long tibble that lists the values of the density function against the given data.

The `deviance_tbl` and the `total_deviance_tbl` just give the simple difference from the actual density to the estimated density for the given estimated distribution.

The `aic_tbl` will provide the AIC for a `lm` model of the estimated density against the empirical density.

The `kolmogorov_smirnov_tbl` for now provides a `two.sided` estimate of the `ks.test` of the estimated density against the empirical.

The `multi_metric_tbl` will summarise all of these metrics into a single tibble.

### Value

An invisible list object. A tibble is printed.

### Author(s)

Steven P. Sanderson II, MPH

### Examples

```
xc <- mtcars$mpg
output_c <- tidy_distribution_comparison(xc, "continuous")

xd <- trunc(xc)
output_d <- tidy_distribution_comparison(xd, "discrete")

output_c
```

---

`tidy_distribution_summary_tbl`

*Tidy Distribution Summary Statistics Tibble*

---

### Description

This function returns a summary statistics tibble. It will use the `y` column from the `tidy_distribution` function.

### Usage

```
tidy_distribution_summary_tbl(.data, ...)
```

### Arguments

`.data` The data that is going to be passed from a `tidy_distribution` function.

`...` This is the grouping variable that gets passed to `dplyr::group_by()` and `dplyr::select()`.



**Details**

This function takes in a `tidy_distribution` table and will return a tibble of the following information:

- `sim_number`
- `mean_val`
- `median_val`
- `std_val`
- `min_val`
- `max_val`
- `skewness`
- `kurtosis`
- `range`
- `iqr`
- `variance`
- `ci_hi`
- `ci_lo`

The kurtosis and skewness come from the package `healthyR.ai`

**Value**

A summary stats tibble

**Author(s)**

Steven P. Sanderson II, MPH

**Examples**

```
library(dplyr)

tn <- tidy_normal(.num_sims = 5)
tb <- tidy_beta(.num_sims = 5)

tidy_distribution_summary_tbl(tn)
tidy_distribution_summary_tbl(tn, sim_number)

data_tbl <- tidy_combine_distributions(tn, tb)

tidy_distribution_summary_tbl(data_tbl)
tidy_distribution_summary_tbl(data_tbl, dist_type)
```

---

tidy_empirical	<i>Tidy Empirical</i>
----------------	-----------------------

---

### Description

This function takes in a single argument of `.x` a vector and will return a tibble of information similar to the `tidy_` distribution functions. The `y` column is set equal to `dy` from the density function.

### Usage

```
tidy_empirical(.x, .num_sims = 1, .distribution_type = "continuous")
```

### Arguments

<code>.x</code>	A vector of numbers
<code>.num_sims</code>	How many simulations should be run, defaults to 1.
<code>.distribution_type</code>	A string of either "continuous" or "discrete". The function will default to "continuous"

### Details

This function takes in a single argument of `.x` a vector

### Value

A tibble

### Author(s)

Steven P. Sanderson II, MPH

### Examples

```
x <- mtcars$mpg
tidy_empirical(.x = x, .distribution_type = "continuous")
tidy_empirical(.x = x, .num_sims = 10, .distribution_type = "continuous")
```

---

tidy_exponential	<i>Tidy Randomly Generated Exponential Distribution Tibble</i>
------------------	--

---

## Description

This function will generate `n` random points from a exponential distribution with a user provided, `.rate`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the `n` randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of `n` for the current simulation.
- `y` The randomly generated data point.
- `dx` The `x` value from the `stats::density()` function.
- `dy` The `y` value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

## Usage

```
tidy_exponential(.n = 50, .rate = 1, .num_sims = 1, .return_tibble = TRUE)
```

## Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.rate</code>	A vector of rates
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

## Details

This function uses the underlying `stats::rexp()`, and its underlying `p`, `d`, and `q` functions. For more information please see `stats::rexp()`

## Value

A tibble of randomly generated data.

## Author(s)

Steven P. Sanderson II, MPH

**See Also**

<https://www.itl.nist.gov/div898/handbook/eda/section3/eda3667.htm>

Other Continuous Distribution: `tidy_beta()`, `tidy_burr()`, `tidy_cauchy()`, `tidy_chisquare()`, `tidy_f()`, `tidy_gamma()`, `tidy_generalized_beta()`, `tidy_generalized_pareto()`, `tidy_geometric()`, `tidy_inverse_burr()`, `tidy_inverse_exponential()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_inverse_weibull()`, `tidy_logistic()`, `tidy_lognormal()`, `tidy_normal()`, `tidy_paralogistic()`, `tidy_pareto1()`, `tidy_pareto()`, `tidy_triangular()`, `tidy_t()`, `tidy_uniform()`, `tidy_weibull()`, `tidy_zero_truncated_geometric()`

Other Exponential: `tidy_inverse_exponential()`, `util_exponential_param_estimate()`, `util_exponential_stats_`

**Examples**

```
tidy_exponential()
```

---

```
tidy_f
```

*Tidy Randomly Generated F Distribution Tibble*

---

**Description**

This function will generate  $n$  random points from a  $rf$  distribution with a user provided,  $df1$ ,  $df2$ , and  $ncp$ , and number of random simulations to be produced. The function returns a tibble with the simulation number column the  $x$  column which corresponds to the  $n$  randomly generated points, the  $d_$ ,  $p_$  and  $q_$  data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting  $p_$  function of the distribution family.
- `q` The values from the resulting  $q_$  function of the distribution family.

**Usage**

```
tidy_f(
  .n = 50,
  .df1 = 1,
  .df2 = 1,
  .ncp = 0,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

**Arguments**

<code>.n</code>	The number of randomly generated points you want.
<code>.df1</code>	Degrees of freedom, Inf is allowed.
<code>.df2</code>	Degrees of freedom, Inf is allowed.
<code>.ncp</code>	Non-centrality parameter.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `stats::rf()`, and its underlying p, d, and q functions. For more information please see [stats::rf\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://www.itl.nist.gov/div898/handbook/eda/section3/eda3665.htm>

Other Continuous Distribution: [tidy\\_beta\(\)](#), [tidy\\_burr\(\)](#), [tidy\\_cauchy\(\)](#), [tidy\\_chisquare\(\)](#), [tidy\\_exponential\(\)](#), [tidy\\_gamma\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_geometric\(\)](#), [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_logistic\(\)](#), [tidy\\_lognormal\(\)](#), [tidy\\_normal\(\)](#), [tidy\\_paralogistic\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [tidy\\_triangular\(\)](#), [tidy\\_t\(\)](#), [tidy\\_uniform\(\)](#), [tidy\\_weibull\(\)](#), [tidy\\_zero\\_truncated\\_geometric\(\)](#)

Other F Distribution: [util\\_f\\_stats\\_tbl\(\)](#)

**Examples**

```
tidy_f()
```

---

tidy\_four\_autoplot      *Automatic Plot of Density Data*


---

### Description

This is an auto plotting function that will take in a `tidy_` distribution function and a few arguments, one being the plot type, which is a quoted string of one of the following:

- density
- quantile
- probablity
- qq

If the number of simulations exceeds 9 then the legend will not print. The plot subtitle is put together by the attributes of the table passed to the function.

### Usage

```
tidy_four_autoplot(
  .data,
  .line_size = 0.5,
  .geom_point = FALSE,
  .point_size = 1,
  .geom_rug = FALSE,
  .geom_smooth = FALSE,
  .geom_jitter = FALSE,
  .interactive = FALSE
)
```

### Arguments

<code>.data</code>	The data passed in from a <code>tidy_distribution</code> function like <code>tidy_normal()</code>
<code>.line_size</code>	The size param <code>ggplot</code>
<code>.geom_point</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return a plot with <code>ggplot2::geom_point()</code>
<code>.point_size</code>	The point size param for <code>ggplot</code>
<code>.geom_rug</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return the use of <code>ggplot2::geom_rug()</code>
<code>.geom_smooth</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return the use of <code>ggplot2::geom_smooth()</code> The <code>aes</code> parameter of <code>group</code> is set to FALSE. This ensures a single smoothing band returned with <code>SE</code> also set to FALSE. Color is set to 'black' and <code>linetype</code> is 'dashed'.
<code>.geom_jitter</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return the use of <code>ggplot2::geom_jitter()</code>
<code>.interactive</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return an interactive <code>plotly</code> plot.

## Details

This function will spit out one of the following plots:

- density
- quantile
- probability
- qq

## Value

A ggplot or a plotly plot.

## Author(s)

Steven P. Sanderson II, MPH

## See Also

Other Autoplot: [bootstrap\\_stat\\_plot\(\)](#), [tidyautoplot\(\)](#), [tidy\\_combined\\_autoplot\(\)](#), [tidy\\_multi\\_dist\\_autoplot\(\)](#), [tidy\\_random\\_walk\\_autoplot\(\)](#)

## Examples

```
tidy_normal(.num_sims = 5) |>
  tidy_four_autoplot()
```

---

tidy\_gamma

*Tidy Randomly Generated Gamma Distribution Tibble*

---

## Description

This function will generate n random points from a gamma distribution with a user provided, `.shape`, `.scale`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the x column which corresponds to the n randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of n for the current simulation.
- `y` The randomly generated data point.
- `dx` The x value from the `stats::density()` function.
- `dy` The y value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

## Usage

```
tidy_gamma(  
  .n = 50,  
  .shape = 1,  
  .scale = 0.3,  
  .num_sims = 1,  
  .return_tibble = TRUE  
)
```

## Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.shape</code>	This is strictly 0 to infinity.
<code>.scale</code>	The standard deviation of the randomly generated data. This is strictly from 0 to infinity.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

## Details

This function uses the underlying `stats::rgamma()`, and its underlying `p`, `d`, and `q` functions. For more information please see [stats::rgamma\(\)](#)

## Value

A tibble of randomly generated data.

## Author(s)

Steven P. Sanderson II, MPH

## See Also

<https://www.statology.org/fit-gamma-distribution-to-dataset-in-r/>

[https://en.wikipedia.org/wiki/Gamma\\_distribution](https://en.wikipedia.org/wiki/Gamma_distribution)

Other Continuous Distribution: [tidy\\_beta\(\)](#), [tidy\\_burr\(\)](#), [tidy\\_cauchy\(\)](#), [tidy\\_chisquare\(\)](#), [tidy\\_exponential\(\)](#), [tidy\\_f\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_geometric\(\)](#), [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_logistic\(\)](#), [tidy\\_lognormal\(\)](#), [tidy\\_normal\(\)](#), [tidy\\_paralogistic\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [tidy\\_triangular\(\)](#), [tidy\\_t\(\)](#), [tidy\\_uniform\(\)](#), [tidy\\_weibull\(\)](#), [tidy\\_zero\\_truncated\\_geometric\(\)](#)

Other Gamma: [tidy\\_inverse\\_gamma\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#)

## Examples

```
tidy_gamma()
```



---

tidy\_generalized\_beta *Tidy Randomly Generated Generalized Beta Distribution Tibble*

---

### Description

This function will generate  $n$  random points from a generalized beta distribution with a user provided, `.shape1`, `.shape2`, `.shape3`, `.rate`, and/or `.scale`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

### Usage

```
tidy_generalized_beta(
  .n = 50,
  .shape1 = 1,
  .shape2 = 1,
  .shape3 = 1,
  .rate = 1,
  .scale = 1/.rate,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

### Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.shape1</code>	A non-negative parameter of the Beta distribution.
<code>.shape2</code>	A non-negative parameter of the Beta distribution.
<code>.shape3</code>	A non-negative parameter of the Beta distribution.
<code>.rate</code>	An alternative way to specify the <code>.scale</code> parameter.
<code>.scale</code>	Must be strictly positive.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `stats::rbeta()`, and its underlying `p`, `d`, and `q` functions. For more information please see [stats::rbeta\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://statisticsglobe.com/beta-distribution-in-r-dbeta-pbeta-qbeta-rbeta>

[https://en.wikipedia.org/wiki/Beta\\_distribution](https://en.wikipedia.org/wiki/Beta_distribution)

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Continuous Distribution: [tidy\\_beta\(\)](#), [tidy\\_burr\(\)](#), [tidy\\_cauchy\(\)](#), [tidy\\_chisquare\(\)](#), [tidy\\_exponential\(\)](#), [tidy\\_f\(\)](#), [tidy\\_gamma\(\)](#), [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_geometric\(\)](#), [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_logistic\(\)](#), [tidy\\_lognormal\(\)](#), [tidy\\_normal\(\)](#), [tidy\\_paralogistic\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [tidy\\_triangular\(\)](#), [tidy\\_t\(\)](#), [tidy\\_uniform\(\)](#), [tidy\\_weibull\(\)](#), [tidy\\_zero\\_truncated\\_geometric\(\)](#)

Other Beta: [tidy\\_beta\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#)

**Examples**

```
tidy_generalized_beta()
```

---

```
tidy_generalized_pareto
```

*Tidy Randomly Generated Generalized Pareto Distribution Tibble*

---

**Description**

This function will generate `n` random points from a generalized Pareto distribution with a user provided, `.shape1`, `.shape2`, `.rate` or `.scale` and number of `#'` random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the `n` randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of `n` for the current simulation.

- y The randomly generated data point.
- dx The x value from the `stats::density()` function.
- dy The y value from the `stats::density()` function.
- p The values from the resulting p\_ function of the distribution family.
- q The values from the resulting q\_ function of the distribution family.

### Usage

```
tidy_generalized_pareto(  
  .n = 50,  
  .shape1 = 1,  
  .shape2 = 1,  
  .rate = 1,  
  .scale = 1/.rate,  
  .num_sims = 1,  
  .return_tibble = TRUE  
)
```

### Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.shape1</code>	Must be positive.
<code>.shape2</code>	Must be positive.
<code>.rate</code>	An alternative way to specify the <code>.scale</code> argument
<code>.scale</code>	Must be positive.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

### Details

This function uses the underlying `actuar::rgenpareto()`, and its underlying p, d, and q functions. For more information please see [actuar::rgenpareto\(\)](#)

### Value

A tibble of randomly generated data.

### Author(s)

Steven P. Sanderson II, MPH

**See Also**

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Continuous Distribution: `tidy_beta()`, `tidy_burr()`, `tidy_cauchy()`, `tidy_chisquare()`, `tidy_exponential()`, `tidy_f()`, `tidy_gamma()`, `tidy_generalized_beta()`, `tidy_geometric()`, `tidy_inverse_burr()`, `tidy_inverse_exponential()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_inverse_weibull()`, `tidy_logistic()`, `tidy_lognormal()`, `tidy_normal()`, `tidy_paralogistic()`, `tidy_pareto1()`, `tidy_pareto()`, `tidy_triangular()`, `tidy_t()`, `tidy_uniform()`, `tidy_weibull()`, `tidy_zero_truncated_geometric()`

Other Pareto: `tidy_inverse_pareto()`, `tidy_pareto1()`, `tidy_pareto()`, `util_pareto_param_estimate()`, `util_pareto_stats_tbl()`

**Examples**

```
tidy_generalized_pareto()
```

---

tidy\_geometric

*Tidy Randomly Generated Geometric Distribution Tibble*

---

**Description**

This function will generate  $n$  random points from a geometric distribution with a user provided, `.prob`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

**Usage**

```
tidy_geometric(.n = 50, .prob = 1, .num_sims = 1, .return_tibble = TRUE)
```

**Arguments**

<code>.n</code>	The number of randomly generated points you want.
<code>.prob</code>	A probability of success in each trial $0 < \text{prob} \leq 1$ .
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `stats::rgeom()`, and its underlying `p`, `d`, and `q` functions. For more information please see [stats::rgeom\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

[https://en.wikipedia.org/wiki/Geometric\\_distribution](https://en.wikipedia.org/wiki/Geometric_distribution)

Other Continuous Distribution: `tidy_beta()`, `tidy_burr()`, `tidy_cauchy()`, `tidy_chisquare()`, `tidy_exponential()`, `tidy_f()`, `tidy_gamma()`, `tidy_generalized_beta()`, `tidy_generalized_pareto()`, `tidy_inverse_burr()`, `tidy_inverse_exponential()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_inverse_weibull()`, `tidy_logistic()`, `tidy_lognormal()`, `tidy_normal()`, `tidy_paralogistic()`, `tidy_pareto1()`, `tidy_pareto()`, `tidy_triangular()`, `tidy_t()`, `tidy_uniform()`, `tidy_weibull()`, `tidy_zero_truncated_geometric()`

Other Geometric: `tidy_zero_truncated_geometric()`, `util_geometric_param_estimate()`, `util_geometric_stats_tbl()`

**Examples**

```
tidy_geometric()
```

**Description**

This function will generate  $n$  random points from a hypergeometric distribution with a user provided,  $m$ ,  $nn$ , and  $k$ , and number of random simulations to be produced. The function returns a tibble with the simulation number column the  $x$  column which corresponds to the  $n$  randomly generated points, the  $d_$ ,  $p_$  and  $q_$  data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting  $p_$  function of the distribution family.
- `q` The values from the resulting  $q_$  function of the distribution family.

**Usage**

```
tidy_hypergeometric(
  .n = 50,
  .m = 0,
  .nn = 0,
  .k = 0,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

**Arguments**

<code>.n</code>	The number of randomly generated points you want.
<code>.m</code>	The number of white balls in the urn
<code>.nn</code>	The number of black balls in the urn
<code>.k</code>	The number of balls drawn fro the urn.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `stats::rhyper()`, and its underlying  $p$ ,  $d$ , and  $q$  functions. For more information please see `stats::rhyper()`

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

[https://en.wikipedia.org/wiki/Hypergeometric\\_distribution](https://en.wikipedia.org/wiki/Hypergeometric_distribution)

Other Discrete Distribution: `tidy_bernoulli()`, `tidy_binomial()`, `tidy_negative_binomial()`, `tidy_poisson()`, `tidy_zero_truncated_binomial()`, `tidy_zero_truncated_negative_binomial()`, `tidy_zero_truncated_poisson()`

Other Hypergeometric: `util_hypergeometric_param_estimate()`, `util_hypergeometric_stats_tbl()`

**Examples**

```
tidy_hypergeometric()
```

---

tidy_inverse_burr	<i>Tidy Randomly Generated Inverse Burr Distribution Tibble</i>
-------------------	---

---

**Description**

This function will generate  $n$  random points from an Inverse Burr distribution with a user provided, `.shape1`, `.shape2`, `.scale`, `.rate`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

**Usage**

```
tidy_inverse_burr(
  .n = 50,
  .shape1 = 1,
  .shape2 = 1,
  .rate = 1,
  .scale = 1/.rate,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

**Arguments**

<code>.n</code>	The number of randomly generated points you want.
<code>.shape1</code>	Must be strictly positive.
<code>.shape2</code>	Must be strictly positive.
<code>.rate</code>	An alternative way to specify the <code>.scale</code> .
<code>.scale</code>	Must be strictly positive.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `actuar::rinvburr()`, and its underlying `p`, `d`, and `q` functions. For more information please see `actuar::rinvburr()`

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Continuous Distribution: `tidy_beta()`, `tidy_burr()`, `tidy_cauchy()`, `tidy_chisquare()`, `tidy_exponential()`, `tidy_f()`, `tidy_gamma()`, `tidy_generalized_beta()`, `tidy_generalized_pareto()`, `tidy_geometric()`, `tidy_inverse_exponential()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_inverse_weibull()`, `tidy_logistic()`, `tidy_lognormal()`, `tidy_normal()`, `tidy_paralogistic()`, `tidy_pareto1()`, `tidy_pareto()`, `tidy_triangular()`, `tidy_t()`, `tidy_uniform()`, `tidy_weibull()`, `tidy_zero_truncated_geometric()`

Other Burr: `tidy_burr()`, `util_burr_param_estimate()`, `util_burr_stats_tbl()`

Other Inverse Distribution: `tidy_inverse_exponential()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_inverse_weibull()`

**Examples**

```
tidy_inverse_burr()
```



---

`tidy_inverse_exponential`*Tidy Randomly Generated Inverse Exponential Distribution Tibble*

---

## Description

This function will generate  $n$  random points from an inverse exponential distribution with a user provided, `.rate` or `.scale` and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

## Usage

```
tidy_inverse_exponential(  
  .n = 50,  
  .rate = 1,  
  .scale = 1/.rate,  
  .num_sims = 1,  
  .return_tibble = TRUE  
)
```

## Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.rate</code>	An alternative way to specify the <code>.scale</code>
<code>.scale</code>	Must be strictly positive.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

## Details

This function uses the underlying `actuar::rinvexp()`, and its underlying `p`, `d`, and `q` functions. For more information please see `actuar::rinvexp()`

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Continuous Distribution: `tidy_beta()`, `tidy_burr()`, `tidy_cauchy()`, `tidy_chisquare()`, `tidy_exponential()`, `tidy_f()`, `tidy_gamma()`, `tidy_generalized_beta()`, `tidy_generalized_pareto()`, `tidy_geometric()`, `tidy_inverse_burr()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_inverse_weibull()`, `tidy_logistic()`, `tidy_lognormal()`, `tidy_normal()`, `tidy_paralogistic()`, `tidy_pareto1()`, `tidy_pareto()`, `tidy_triangular()`, `tidy_t()`, `tidy_uniform()`, `tidy_weibull()`, `tidy_zero_truncated_geometric()`

Other Exponential: `tidy_exponential()`, `util_exponential_param_estimate()`, `util_exponential_stats_tbl()`

Other Inverse Distribution: `tidy_inverse_burr()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_inverse_weibull()`

**Examples**

```
tidy_inverse_exponential()
```

---

<code>tidy_inverse_gamma</code>	<i>Tidy Randomly Generated Inverse Gamma Distribution Tibble</i>
---------------------------------	--

---

**Description**

This function will generate  $n$  random points from an inverse gamma distribution with a user provided, `.shape`, `.rate`, `.scale`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

**Usage**

```
tidy_inverse_gamma(
  .n = 50,
  .shape = 1,
  .rate = 1,
  .scale = 1/.rate,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

**Arguments**

<code>.n</code>	The number of randomly generated points you want.
<code>.shape</code>	Must be strictly positive.
<code>.rate</code>	An alternative way to specify the <code>.scale</code>
<code>.scale</code>	Must be strictly positive.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `actuar::rinvgamma()`, and its underlying `p`, `d`, and `q` functions. For more information please see [actuar::rinvgamma\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Continuous Distribution: [tidy\\_beta\(\)](#), [tidy\\_burr\(\)](#), [tidy\\_cauchy\(\)](#), [tidy\\_chisquare\(\)](#), [tidy\\_exponential\(\)](#), [tidy\\_f\(\)](#), [tidy\\_gamma\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_geometric\(\)](#), [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_logistic\(\)](#), [tidy\\_lognormal\(\)](#), [tidy\\_normal\(\)](#), [tidy\\_paralogistic\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [tidy\\_triangular\(\)](#), [tidy\\_t\(\)](#), [tidy\\_uniform\(\)](#), [tidy\\_weibull\(\)](#), [tidy\\_zero\\_truncated\\_geometric\(\)](#)

Other Gamma: [tidy\\_gamma\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#)

Other Inverse Distribution: [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#)

**Examples**

```
tidy_inverse_gamma()
```

---

```
tidy_inverse_normal
```

*Tidy Randomly Generated Inverse Gaussian Distribution Tibble*

---

**Description**

This function will generate  $n$  random points from an Inverse Gaussian distribution with a user provided, `.mean`, `.shape`, `.dispersion`. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

**Usage**

```
tidy_inverse_normal(
  .n = 50,
  .mean = 1,
  .shape = 1,
  .dispersion = 1/.shape,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

**Arguments**

- |                             |   |
|-----------------------------|---|
| <code>.n</code>             | The number of randomly generated points you want.                                     |
| <code>.mean</code>          | Must be strictly positive.  |
| <code>.shape</code>         | Must be strictly positive.  |
| <code>.dispersion</code>    | An alternative way to specify the <code>.shape</code> .                               |
| <code>.num_sims</code>      | The number of randomly generated simulations you want.                                |
| <code>.return_tibble</code> | A logical value indicating whether to return the result as a tibble. Default is TRUE. |

**Details**

This function uses the underlying actuar::rinvgauss(). For more information please see [rinvgauss\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Continuous Distribution: [tidy\\_beta\(\)](#), [tidy\\_burr\(\)](#), [tidy\\_cauchy\(\)](#), [tidy\\_chisquare\(\)](#), [tidy\\_exponential\(\)](#), [tidy\\_f\(\)](#), [tidy\\_gamma\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_geometric\(\)](#), [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_logistic\(\)](#), [tidy\\_lognormal\(\)](#), [tidy\\_normal\(\)](#), [tidy\\_paralogistic\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [tidy\\_triangular\(\)](#), [tidy\\_t\(\)](#), [tidy\\_uniform\(\)](#), [tidy\\_weibull\(\)](#), [tidy\\_zero\\_truncated\\_geometric\(\)](#)

Other Gaussian: [tidy\\_normal\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#)

Other Inverse Distribution: [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#)

**Examples**

```
tidy_inverse_normal()
```

---

tidy\_inverse\_pareto     *Tidy Randomly Generated Inverse Pareto Distribution Tibble*

---

**Description**

This function will generate n random points from an inverse pareto distribution with a user provided, .shape, .scale, and number of random simulations to be produced. The function returns a tibble with the simulation number column the x column which corresponds to the n randomly generated points, the d\_, p\_ and q\_ data points as well.

The data is returned un-grouped.

The columns that are output are:

- sim\_number The current simulation number.
- x The current value of n for the current simulation.
- y The randomly generated data point.
- dx The x value from the [stats::density\(\)](#) function.
- dy The y value from the [stats::density\(\)](#) function.
- p The values from the resulting p\_ function of the distribution family.
- q The values from the resulting q\_ function of the distribution family.

**Usage**

```
tidy_inverse_pareto(
  .n = 50,
  .shape = 1,
  .scale = 1,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

**Arguments**

<code>.n</code>	The number of randomly generated points you want.
<code>.shape</code>	Must be positive.
<code>.scale</code>	Must be positive.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `actuar::rinvpareto()`, and its underlying `p`, `d`, and `q` functions. For more information please see [actuar::rinvpareto\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Continuous Distribution: [tidy\\_beta\(\)](#), [tidy\\_burr\(\)](#), [tidy\\_cauchy\(\)](#), [tidy\\_chisquare\(\)](#), [tidy\\_exponential\(\)](#), [tidy\\_f\(\)](#), [tidy\\_gamma\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_geometric\(\)](#), [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_logistic\(\)](#), [tidy\\_lognormal\(\)](#), [tidy\\_normal\(\)](#), [tidy\\_paralogistic\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [tidy\\_triangular\(\)](#), [tidy\\_t\(\)](#), [tidy\\_uniform\(\)](#), [tidy\\_weibull\(\)](#), [tidy\\_zero\\_truncated\\_geometric\(\)](#)

Other Pareto: [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#)

Other Inverse Distribution: [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_weibull\(\)](#)

**Examples**

```
tidy_inverse_pareto()
```

---

tidy\_inverse\_weibull *Tidy Randomly Generated Inverse Weibull Distribution Tibble*

---

**Description**

This function will generate  $n$  random points from a weibull distribution with a user provided, `.shape`, `.scale`, `.rate`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

**Usage**

```
tidy_inverse_weibull(
  .n = 50,
  .shape = 1,
  .rate = 1,
  .scale = 1/.rate,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

**Arguments**

<code>.n</code>	The number of randomly generated points you want.
<code>.shape</code>	Must be strictly positive.
<code>.rate</code>	An alternative way to specify the <code>.scale</code> .
<code>.scale</code>	Must be strictly positive.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `actuar::rinvweibull()`, and its underlying p, d, and q functions. For more information please see `actuar::rinvweibull()`

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Continuous Distribution: `tidy_beta()`, `tidy_burr()`, `tidy_cauchy()`, `tidy_chisquare()`, `tidy_exponential()`, `tidy_f()`, `tidy_gamma()`, `tidy_generalized_beta()`, `tidy_generalized_pareto()`, `tidy_geometric()`, `tidy_inverse_burr()`, `tidy_inverse_exponential()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_logistic()`, `tidy_lognormal()`, `tidy_normal()`, `tidy_paralogistic()`, `tidy_pareto1()`, `tidy_pareto()`, `tidy_triangular()`, `tidy_t()`, `tidy_uniform()`, `tidy_weibull()`, `tidy_zero_truncated_geometric()`

Other Weibull: `tidy_weibull()`, `util_weibull_param_estimate()`, `util_weibull_stats_tbl()`

Other Inverse Distribution: `tidy_inverse_burr()`, `tidy_inverse_exponential()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`

**Examples**

```
tidy_inverse_weibull()
```

---

tidy_kurtosis_vec	<i>Compute Kurtosis of a Vector</i>
-------------------	-------------------------------------

---

**Description**

This function takes in a vector as it's input and will return the kurtosis of that vector. The length of this vector must be at least four numbers. The kurtosis explains the sharpness of the peak of a distribution of data.

$$\left(\frac{1}{n} * \sum(x - \mu)^4\right) / \left(\left(\frac{1}{n} * \sum(x - \mu)^2\right)^2\right)$$

**Usage**

```
tidy_kurtosis_vec(.x)
```

**Arguments**

`.x` A numeric vector of length four or more.



**Details**

A function to return the kurtosis of a vector.

**Value**

The kurtosis of a vector

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://en.wikipedia.org/wiki/Kurtosis>

Other Vector Function: `bootstrap_p_vec()`, `bootstrap_q_vec()`, `cgmean()`, `chmean()`, `ckurtosis()`, `cmean()`, `cmedian()`, `csd()`, `cskewness()`, `cvar()`, `tidy_scale_zero_one_vec()`, `tidy_skewness_vec()`

Other Statistic: `ci_hi()`, `ci_lo()`, `tidy_range_statistic()`, `tidy_skewness_vec()`, `tidy_stat_tbl()`

Other Vector Function: `bootstrap_p_vec()`, `bootstrap_q_vec()`, `cgmean()`, `chmean()`, `ckurtosis()`, `cmean()`, `cmedian()`, `csd()`, `cskewness()`, `cvar()`, `tidy_scale_zero_one_vec()`, `tidy_skewness_vec()`

**Examples**

```
tidy_kurtosis_vec(rnorm(100, 3, 2))
```

---

tidy\_logistic

*Tidy Randomly Generated Logistic Distribution Tibble*

---

**Description**

This function will generate  $n$  random points from a logistic distribution with a user provided, `.location`, `.scale`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

**Usage**

```
tidy_logistic(  
  .n = 50,  
  .location = 0,  
  .scale = 1,  
  .num_sims = 1,  
  .return_tibble = TRUE  
)
```

**Arguments**

<code>.n</code>	The number of randomly generated points you want.
<code>.location</code>	The location parameter
<code>.scale</code>	The scale parameter
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `stats::rlogis()`, and its underlying `p`, `d`, and `q` functions. For more information please see [stats::rlogis\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

[https://en.wikipedia.org/wiki/Logistic\\_distribution](https://en.wikipedia.org/wiki/Logistic_distribution)

Other Continuous Distribution: [tidy\\_beta\(\)](#), [tidy\\_burr\(\)](#), [tidy\\_cauchy\(\)](#), [tidy\\_chisquare\(\)](#), [tidy\\_exponential\(\)](#), [tidy\\_f\(\)](#), [tidy\\_gamma\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_geometric\(\)](#), [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_lognormal\(\)](#), [tidy\\_normal\(\)](#), [tidy\\_paralogistic\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [tidy\\_triangular\(\)](#), [tidy\\_t\(\)](#), [tidy\\_uniform\(\)](#), [tidy\\_weibull\(\)](#), [tidy\\_zero\\_truncated\\_geometric\(\)](#)

Other Logistic: [tidy\\_paralogistic\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#)

**Examples**

```
tidy_logistic()
```

---

`tidy_lognormal`*Tidy Randomly Generated Lognormal Distribution Tibble*

---

## Description

This function will generate  $n$  random points from a lognormal distribution with a user provided, `.meanlog`, `.sdlog`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

## Usage

```
tidy_lognormal(  
  .n = 50,  
  .meanlog = 0,  
  .sdlog = 1,  
  .num_sims = 1,  
  .return_tibble = TRUE  
)
```

## Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.meanlog</code>	Mean of the distribution on the log scale with default 0
<code>.sdlog</code>	Standard deviation of the distribution on the log scale with default 1
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

## Details

This function uses the underlying `stats::rlnorm()`, and its underlying `p`, `d`, and `q` functions. For more information please see `stats::rlnorm()`

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://www.itl.nist.gov/div898/handbook/eda/section3/eda3669.htm>

Other Continuous Distribution: `tidy_beta()`, `tidy_burr()`, `tidy_cauchy()`, `tidy_chisquare()`, `tidy_exponential()`, `tidy_f()`, `tidy_gamma()`, `tidy_generalized_beta()`, `tidy_generalized_pareto()`, `tidy_geometric()`, `tidy_inverse_burr()`, `tidy_inverse_exponential()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_inverse_weibull()`, `tidy_logistic()`, `tidy_normal()`, `tidy_paralogistic()`, `tidy_pareto1()`, `tidy_pareto()`, `tidy_triangular()`, `tidy_t()`, `tidy_uniform()`, `tidy_weibull()`, `tidy_zero_truncated_geometric()`

Other Lognormal: `util_lognormal_param_estimate()`, `util_lognormal_stats_tbl()`

**Examples**

```
tidy_lognormal()
```

---

tidy\_mixture\_density *Tidy Mixture Data*

---

**Description**

Create mixture model data and resulting density and line plots.

**Usage**

```
tidy_mixture_density(...)
```

**Arguments**

... The random data you want to pass. Example `rnorm(50,0,1)` or something like `tidy_normal(.mean = 5, .sd = 1)`

**Details**

This function allows you to make mixture model data. It allows you to produce density data and plots for data that is not strictly of one family or of one single type of distribution with a given set of parameters.

For example this function will allow you to mix say `tidy_normal(.mean = 0, .sd = 1)` and `tidy_normal(.mean = 5, .sd = 1)` or you can mix and match distributions.

The output is a list object with three components.

### 1. Data

- input\_data (The random data passed)
- dist\_tbl (A tibble of the passed random data)
- density\_tbl (A tibble of the x and y data from `stats::density()`)

### 1. Plots

- line\_plot - Plots the dist\_tbl
- dens\_plot - Plots the density\_tbl

### 1. Input Functions

- input\_fns - A list of the functions and their parameters passed to the function itself

## Value

A list object

## Author(s)

Steven P. Sanderson II, MPH

## Examples

```
output <- tidy_mixture_density(rnorm(100, 0, 1), tidy_normal(.mean = 5, .sd = 1))
```

```
output$data
```

```
output$plots
```

```
output$input_fns
```

---

```
tidy_multi_dist_autoplot
```

*Automatic Plot of Multi Dist Data*

---

## Description

This is an auto plotting function that will take in a `tidy_` distribution function and a few arguments, one being the plot type, which is a quoted string of one of the following:

- density
- quantile
- probability
- qq

If the number of simulations exceeds 9 then the legend will not print. The plot subtitle is put together by the attributes of the table passed to the function.

**Usage**

```
tidy_multi_dist_autoplot(
  .data,
  .plot_type = "density",
  .line_size = 0.5,
  .geom_point = FALSE,
  .point_size = 1,
  .geom_rug = FALSE,
  .geom_smooth = FALSE,
  .geom_jitter = FALSE,
  .interactive = FALSE
)
```

**Arguments**

<code>.data</code>	The data passed in from a the function <code>tidy_multi_dist()</code>
<code>.plot_type</code>	This is a quoted string like 'density'
<code>.line_size</code>	The size param ggplot
<code>.geom_point</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return a plot with <code>ggplot2::geom_point()</code>
<code>.point_size</code>	The point size param for ggplot
<code>.geom_rug</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return the use of <code>ggplot2::geom_rug()</code>
<code>.geom_smooth</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return the use of <code>ggplot2::geom_smooth()</code> The aes parameter of group is set to FALSE. This ensures a single smoothing band returned with SE also set to FALSE. Color is set to 'black' and linetype is 'dashed'.
<code>.geom_jitter</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return the use of <code>ggplot2::geom_jitter()</code>
<code>.interactive</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return an interactive plotly plot.

**Details**

This function will spit out one of the following plots:

- density
- quantile
- probability
- qq

**Value**

A ggplot or a plotly plot.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Autoplot: [bootstrap\\_stat\\_plot\(\)](#), [tidy\\_autoplot\(\)](#), [tidy\\_combined\\_autoplot\(\)](#), [tidy\\_four\\_autoplot\(\)](#), [tidy\\_random\\_walk\\_autoplot\(\)](#)

**Examples**

```
tn <- tidy_multi_single_dist(
  .tidy_dist = "tidy_normal",
  .param_list = list(
    .n = 100,
    .mean = c(-2, 0, 2),
    .sd = 1,
    .num_sims = 5,
    .return_tibble = TRUE
  )
)

tn |>
  tidy_multi_dist_autoplot()

tn |>
  tidy_multi_dist_autoplot(.plot_type = "qq")
```

---

tidy\_multi\_single\_dist

*Generate Multiple Tidy Distributions of a single type*

---

**Description**

Generate multiple distributions of data from the same tidy\_ distribution function.

**Usage**

```
tidy_multi_single_dist(.tidy_dist = NULL, .param_list = list())
```

**Arguments**

<code>.tidy_dist</code>	The type of tidy_ distribution that you want to run. You can only choose one.
<code>.param_list</code>	This must be a <code>list()</code> object of the parameters that you want to pass through to the TidyDensity tidy_ distribution function.

## Details

Generate multiple distributions of data from the same `tidy_` distribution function. This allows you to simulate multiple distributions of the same family in order to view how shapes change with parameter changes. You can then visualize the differences however you choose.

## Value

A tibble

## Author(s)

Steven P. Sanderson II, MPH

## See Also

Other Multiple Distribution: [tidy\\_combine\\_distributions\(\)](#)

## Examples

```
tidy_multi_single_dist(  
  .tidy_dist = "tidy_normal",  
  .param_list = list(  
    .n = 50,  
    .mean = c(-1, 0, 1),  
    .sd = 1,  
    .num_sims = 3,  
    .return_tibble = TRUE  
  )  
)
```

---

tidy\_negative\_binomial

*Tidy Randomly Generated Negative Binomial Distribution Tibble*

---

## Description

This function will generate `n` random points from a negative binomial distribution with a user provided, `.size`, `.prob`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the `n` randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of `n` for the current simulation.
- `y` The randomly generated data point.



- dx The x value from the `stats::density()` function.
- dy The y value from the `stats::density()` function.
- p The values from the resulting `p_` function of the distribution family.
- q The values from the resulting `q_` function of the distribution family.

### Usage

```
tidy_negative_binomial(
  .n = 50,
  .size = 1,
  .prob = 0.1,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

### Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.size</code>	target for number of successful trials, or dispersion parameter (the shape parameter of the gamma mixing distribution). Must be strictly positive, need not be integer.
<code>.prob</code>	Probability of success on each trial where $0 < .prob \leq 1$ .
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

### Details

This function uses the underlying `stats::rnbinom()`, and its underlying `p`, `d`, and `q` functions. For more information please see `stats::rnbinom()`

### Value

A tibble of randomly generated data.

### Author(s)

Steven P. Sanderson II, MPH

### See Also

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Discrete Distribution: `tidy_bernoulli()`, `tidy_binomial()`, `tidy_hypergeometric()`, `tidy_poisson()`, `tidy_zero_truncated_binomial()`, `tidy_zero_truncated_negative_binomial()`, `tidy_zero_truncated_poisson()`

Other Binomial: `tidy_binomial()`, `tidy_zero_truncated_binomial()`, `tidy_zero_truncated_negative_binomial()`, `util_binomial_param_estimate()`, `util_binomial_stats_tbl()`, `util_negative_binomial_param_estimate()`

**Examples**

```
tidy_negative_binomial()
```

---

tidy\_normal

*Tidy Randomly Generated Gaussian Distribution Tibble*


---

**Description**

This function will generate  $n$  random points from a Gaussian distribution with a user provided, `.mean`, `.sd` - standard deviation and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `dnorm`, `pnorm` and `qnorm` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

**Usage**

```
tidy_normal(.n = 50, .mean = 0, .sd = 1, .num_sims = 1, .return_tibble = TRUE)
```

**Arguments**

<code>.n</code>	The number of randomly generated points you want.
<code>.mean</code>	The mean of the randomly generated data.
<code>.sd</code>	The standard deviation of the randomly generated data.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `stats::rnorm()`, `stats::pnorm()`, and `stats::qnorm()` functions to generate data from the given parameters. For more information please see `stats::rnorm()`

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Continuous Distribution: `tidy_beta()`, `tidy_burr()`, `tidy_cauchy()`, `tidy_chisquare()`, `tidy_exponential()`, `tidy_f()`, `tidy_gamma()`, `tidy_generalized_beta()`, `tidy_generalized_pareto()`, `tidy_geometric()`, `tidy_inverse_burr()`, `tidy_inverse_exponential()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_inverse_weibull()`, `tidy_logistic()`, `tidy_lognormal()`, `tidy_paralogistic()`, `tidy_pareto1()`, `tidy_pareto()`, `tidy_triangular()`, `tidy_t()`, `tidy_uniform()`, `tidy_weibull()`, `tidy_zero_truncated_geometric()`

Other Gaussian: `tidy_inverse_normal()`, `util_normal_param_estimate()`, `util_normal_stats_tbl()`

**Examples**

```
tidy_normal()
```

---

tidy_paralogistic	<i>Tidy Randomly Generated Paralogistic Distribution Tibble</i>
-------------------	---

---

**Description**

This function will generate `n` random points from a paralogistic distribution with a user provided, `.shape`, `.rate`, `.scale` and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the `n` randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of `n` for the current simulation.
- `y` The randomly generated data point.
- `dx` The `x` value from the `stats::density()` function.
- `dy` The `y` value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

**Usage**

```
tidy_paralogistic(
  .n = 50,
  .shape = 1,
  .rate = 1,
  .scale = 1/.rate,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

**Arguments**

<code>.n</code>	The number of randomly generated points you want.
<code>.shape</code>	Must be strictly positive.
<code>.rate</code>	An alternative way to specify the <code>.scale</code>
<code>.scale</code>	Must be strictly positive.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `actuar::rparalogis()`, and its underlying `p`, `d`, and `q` functions. For more information please see [actuar::rparalogis\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

[https://en.wikipedia.org/wiki/Logistic\\_distribution](https://en.wikipedia.org/wiki/Logistic_distribution)

Other Continuous Distribution: [tidy\\_beta\(\)](#), [tidy\\_burr\(\)](#), [tidy\\_cauchy\(\)](#), [tidy\\_chisquare\(\)](#), [tidy\\_exponential\(\)](#), [tidy\\_f\(\)](#), [tidy\\_gamma\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_geometric\(\)](#), [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_logistic\(\)](#), [tidy\\_lognormal\(\)](#), [tidy\\_normal\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [tidy\\_triangular\(\)](#), [tidy\\_t\(\)](#), [tidy\\_uniform\(\)](#), [tidy\\_weibull\(\)](#), [tidy\\_zero\\_truncated\\_geometric\(\)](#)

Other Logistic: [tidy\\_logistic\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#)

**Examples**

```
tidy_paralogistic()
```

---

`tidy_pareto`*Tidy Randomly Generated Pareto Distribution Tibble*

---

## Description

This function will generate  $n$  random points from a pareto distribution with a user provided, `.shape`, `.scale`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

## Usage

```
tidy_pareto(  
  .n = 50,  
  .shape = 10,  
  .scale = 0.1,  
  .num_sims = 1,  
  .return_tibble = TRUE  
)
```

## Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.shape</code>	Must be positive.
<code>.scale</code>	Must be positive.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

## Details

This function uses the underlying `actuar::rpareto()`, and its underlying `p`, `d`, and `q` functions. For more information please see `actuar::rpareto()`

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Continuous Distribution: `tidy_beta()`, `tidy_burr()`, `tidy_cauchy()`, `tidy_chisquare()`, `tidy_exponential()`, `tidy_f()`, `tidy_gamma()`, `tidy_generalized_beta()`, `tidy_generalized_pareto()`, `tidy_geometric()`, `tidy_inverse_burr()`, `tidy_inverse_exponential()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_inverse_weibull()`, `tidy_logistic()`, `tidy_lognormal()`, `tidy_normal()`, `tidy_paralogistic()`, `tidy_pareto1()`, `tidy_triangular()`, `tidy_t()`, `tidy_uniform()`, `tidy_weibull()`, `tidy_zero_truncated_geometric()`

Other Pareto: `tidy_generalized_pareto()`, `tidy_inverse_pareto()`, `tidy_pareto1()`, `util_pareto_param_estimat`, `util_pareto_stats_tbl()`

**Examples**

```
tidy_pareto()
```

---

tidy\_pareto1

*Tidy Randomly Generated Pareto Single Parameter Distribution Tib-  
ble*

---

**Description**

This function will generate n random points from a single parameter pareto distribution with a user provided, `.shape`, `.min`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the x column which corresponds to the n randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of n for the current simulation.
- `y` The randomly generated data point.
- `dx` The x value from the `stats::density()` function.
- `dy` The y value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

**Usage**

```
tidy_pareto1(
  .n = 50,
  .shape = 1,
  .min = 1,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

**Arguments**

<code>.n</code>	The number of randomly generated points you want.
<code>.shape</code>	Must be positive.
<code>.min</code>	The lower bound of the support of the distribution.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `actuar::rpareto1()`, and its underlying p, d, and q functions. For more information please see [actuar::rpareto1\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Continuous Distribution: [tidy\\_beta\(\)](#), [tidy\\_burr\(\)](#), [tidy\\_cauchy\(\)](#), [tidy\\_chisquare\(\)](#), [tidy\\_exponential\(\)](#), [tidy\\_f\(\)](#), [tidy\\_gamma\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_geometric\(\)](#), [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_logistic\(\)](#), [tidy\\_lognormal\(\)](#), [tidy\\_normal\(\)](#), [tidy\\_paralogistic\(\)](#), [tidy\\_pareto\(\)](#), [tidy\\_triangular\(\)](#), [tidy\\_t\(\)](#), [tidy\\_uniform\(\)](#), [tidy\\_weibull\(\)](#), [tidy\\_zero\\_truncated\\_geometric\(\)](#)

Other Pareto: [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_pareto\(\)](#), [util\\_pareto\\_param\\_estimate\\_util\\_pareto\\_stats\\_tbl\(\)](#)

**Examples**

```
tidy_pareto1()
```

---

`tidy_poisson`*Tidy Randomly Generated Poisson Distribution Tibble*

---

## Description

This function will generate `n` random points from a Poisson distribution with a user provided, `.lambda`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the `n` randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of `n` for the current simulation.
- `y` The randomly generated data point.
- `dx` The `x` value from the `stats::density()` function.
- `dy` The `y` value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

## Usage

```
tidy_poisson(.n = 50, .lambda = 1, .num_sims = 1, .return_tibble = TRUE)
```

## Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.lambda</code>	A vector of non-negative means.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

## Details

This function uses the underlying `stats::rpois()`, and its underlying `p`, `d`, and `q` functions. For more information please see `stats::rpois()`

## Value

A tibble of randomly generated data.

## Author(s)

Steven P. Sanderson II, MPH



**See Also**

<https://r-coder.com/poisson-distribution-r/>

[https://en.wikipedia.org/wiki/Poisson\\_distribution](https://en.wikipedia.org/wiki/Poisson_distribution)

Other Poisson: `tidy_zero_truncated_poisson()`, `util_poisson_param_estimate()`, `util_poisson_stats_tbl()`

Other Discrete Distribution: `tidy_bernoulli()`, `tidy_binomial()`, `tidy_hypergeometric()`, `tidy_negative_binomial()`, `tidy_zero_truncated_binomial()`, `tidy_zero_truncated_negative_binomial()`, `tidy_zero_truncated_poisson()`

**Examples**

```
tidy_poisson()
```

---

tidy_random_walk	<i>Tidy Random Walk</i>
------------------	-------------------------

---

**Description**

Takes in the data from a `tidy_` distribution function and applies a random walk calculation of either `cum_prod` or `cum_sum` to `y`.

**Usage**

```
tidy_random_walk(  
  .data,  
  .initial_value = 0,  
  .sample = FALSE,  
  .replace = FALSE,  
  .value_type = "cum_prod"  
)
```

**Arguments**

<code>.data</code>	The data that is being passed from a <code>tidy_</code> distribution function.
<code>.initial_value</code>	The default is 0, this can be set to whatever you want.
<code>.sample</code>	This is a boolean value TRUE/FALSE. The default is FALSE. If set to TRUE then the <code>y</code> value from the <code>tidy_</code> distribution function is sampled.
<code>.replace</code>	This is a boolean value TRUE/FALSE. The default is FALSE. If set to TRUE AND <code>.sample</code> is set to TRUE then the <code>replace</code> parameter of the <code>sample</code> function will be set to TRUE.
<code>.value_type</code>	This can take one of three different values for now. These are the following: <ul style="list-style-type: none"> <li>"cum_prod" - This will take the cumprod of <code>y</code></li> <li>"cum_sum" - This will take the cumsum of <code>y</code></li> </ul>

## Details

Monte Carlo simulations were first formally designed in the 1940's while developing nuclear weapons, and since have been heavily used in various fields to use randomness solve problems that are potentially deterministic in nature. In finance, Monte Carlo simulations can be a useful tool to give a sense of how assets with certain characteristics might behave in the future. While there are more complex and sophisticated financial forecasting methods such as ARIMA (Auto-Regressive Integrated Moving Average) and GARCH (Generalised Auto-Regressive Conditional Heteroskedasticity) which attempt to model not only the randomness but underlying macro factors such as seasonality and volatility clustering, Monte Carlo random walks work surprisingly well in illustrating market volatility as long as the results are not taken too seriously.

## Value

An ungrouped tibble.

## Author(s)

Steven P. Sanderson II, MPH

## Examples

```
tidy_normal(.sd = .1, .num_sims = 25) %>%  
  tidy_random_walk()
```

---

tidy\_random\_walk\_autoplot

*Automatic Plot of Random Walk Data*

---

## Description

This is an auto-plotting function that will take in a tidy\_ distribution function and a few arguments with regard to the output of the visualization.

If the number of simulations exceeds 9 then the legend will not print. The plot subtitle is put together by the attributes of the table passed to the function.

## Usage

```
tidy_random_walk_autoplot(  
  .data,  
  .line_size = 0.5,  
  .geom_rug = FALSE,  
  .geom_smooth = FALSE,  
  .interactive = FALSE  
)
```

### Arguments

<code>.data</code>	The data passed in from a tidy_distribution function like tidy_normal()
<code>.line_size</code>	The size param ggplot
<code>.geom_rug</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return the use of ggplot2::geom_rug()
<code>.geom_smooth</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return the use of ggplot2::geom_smooth() The aes parameter of group is set to FALSE. This ensures a single smoothing band returned with SE also set to FALSE. Color is set to 'black' and linetype is 'dashed'.
<code>.interactive</code>	A Boolean value of TRUE/FALSE, FALSE is the default. TRUE will return an interactive plotly plot.

### Details

This function will produce a simple random walk plot from a tidy\_ distribution function.

### Value

A ggplot or a plotly plot.

### Author(s)

Steven P. Sanderson II, MPH

### See Also

Other Autoplot: [bootstrap\\_stat\\_plot\(\)](#), [tidy\\_autoplot\(\)](#), [tidy\\_combined\\_autoplot\(\)](#), [tidy\\_four\\_autoplot\(\)](#), [tidy\\_multi\\_dist\\_autoplot\(\)](#)

### Examples

```
tidy_normal(sd = .1, .num_sims = 5) |>
  tidy_random_walk(.value_type = "cum_sum") |>
  tidy_random_walk_autoplot()
```

```
tidy_normal(sd = .1, .num_sims = 20) |>
  tidy_random_walk(.value_type = "cum_sum", .sample = TRUE, .replace = TRUE) |>
  tidy_random_walk_autoplot()
```

---

tidy\_range\_statistic *Get the range statistic*

---

### Description

Takes in a numeric vector and returns back the range of that vector

### Usage

```
tidy_range_statistic(.x)
```

### Arguments

`.x` A numeric vector

### Details

Takes in a numeric vector and returns the range of that vector using the `diff` and `range` functions.

### Value

A single number, the range statistic

### Author(s)

Steven P. Sandeson II, MPH

### See Also

Other Statistic: [ci\\_hi\(\)](#), [ci\\_lo\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_skewness\\_vec\(\)](#), [tidy\\_stat\\_tbl\(\)](#)

### Examples

```
tidy_range_statistic(seq(1:10))
```

---

`tidy_scale_zero_one_vec`*Vector Function Scale to Zero and One*

---

**Description**

Takes a numeric vector and will return a vector that has been scaled from  $[0, 1]$

**Usage**

```
tidy_scale_zero_one_vec(.x)
```

**Arguments**

`.x` A numeric vector to be scaled from  $[0, 1]$  inclusive.

**Details**

Takes a numeric vector and will return a vector that has been scaled from  $[0, 1]$  The input vector must be numeric. The computation is fairly straightforward. This may be helpful when trying to compare the distributions of data where a distribution like beta which requires data to be between 0 and 1

$$y[h] = (x - \min(x)) / (\max(x) - \min(x))$$

**Value**

A numeric vector

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Vector Function: [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [cgmean\(\)](#), [chmean\(\)](#), [ckurtosis\(\)](#), [cmean\(\)](#), [cmedian\(\)](#), [csd\(\)](#), [cskewness\(\)](#), [cvar\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_skewness\\_vec\(\)](#)

**Examples**

```
vec_1 <- rnorm(100, 2, 1)
vec_2 <- tidy_scale_zero_one_vec(vec_1)

dens_1 <- density(vec_1)
dens_2 <- density(vec_2)
max_x <- max(dens_1$x, dens_2$x)
max_y <- max(dens_1$y, dens_2$y)
plot(dens_1,
```

```

  asp = max_y / max_x, main = "Density vec_1 (Red) and vec_2 (Blue)",
  col = "red", xlab = "", ylab = "Density of Vec 1 and Vec 2"
)
lines(dens_2, col = "blue")

```

---

tidy\_skewness\_vec      *Compute Skewness of a Vector*

---

### Description

This function takes in a vector as it's input and will return the skewness of that vector. The length of this vector must be at least four numbers. The skewness explains the 'tailedness' of the distribution of data.

$$\frac{((1/n) * \sum(x - \mu)^3)}{(((1/n) * \sum(x - \mu)^2)^{3/2})}$$

### Usage

```
tidy_skewness_vec(.x)
```

### Arguments

.x                    A numeric vector of length four or more.

### Details

A function to return the skewness of a vector.

### Value

The skewness of a vector

### Author(s)

Steven P. Sanderson II, MPH

### See Also

<https://en.wikipedia.org/wiki/Skewness>

Other Statistic: [ci\\_hi\(\)](#), [ci\\_lo\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_range\\_statistic\(\)](#), [tidy\\_stat\\_tbl\(\)](#)

Other Vector Function: [bootstrap\\_p\\_vec\(\)](#), [bootstrap\\_q\\_vec\(\)](#), [cgmean\(\)](#), [chmean\(\)](#), [ckurtosis\(\)](#), [cmean\(\)](#), [cmedian\(\)](#), [csd\(\)](#), [cskewness\(\)](#), [cvar\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_scale\\_zero\\_one\\_vec\(\)](#)

### Examples

```
tidy_skewness_vec(rnorm(100, 3, 2))
```

tidy\_stat\_tbl

*Tidy Stats of Tidy Distribution***Description**

A function to return the stat function values of a given tidy\_ distribution output.

**Usage**

```
tidy_stat_tbl(
  .data,
  .x = y,
  .fns,
  .return_type = "vector",
  .use_data_table = FALSE,
  ...
)
```

**Arguments**

<code>.data</code>	The input data coming from a tidy_ distribution function.
<code>.x</code>	The default is <code>y</code> but can be one of the other columns from the input data.
<code>.fns</code>	The default is <code>IQR</code> , but this can be any stat function like <code>quantile</code> or <code>median</code> etc.
<code>.return_type</code>	The default is "vector" which returns an <code>sapply</code> object.
<code>.use_data_table</code>	The default is <code>FALSE</code> , <code>TRUE</code> will use <code>data.table</code> under the hood and still return a tibble. If this argument is set to <code>TRUE</code> then the <code>.return_type</code> parameter will be ignored.
<code>...</code>	Addition function arguments to be supplied to the parameters of <code>.fns</code>

**Details**

A function to return the value(s) of a given tidy\_ distribution function output and chosen column from it. This function will only work with tidy\_ distribution functions.

There are currently three different output types for this function. These are:

- "vector" - which gives an `sapply()` output
- "list" - which gives an `lapply()` output, and
- "tibble" - which returns a tibble in long format.

Currently you can pass any stat function that performs an operation on a vector input. This means you can pass things like `IQR`, `quantile` and their associated arguments in the `...` portion of the function.

This function also by default will rename the value column of the tibble to the name of the function. This function will also give the column name of sim\_number for the tibble output with the corresponding simulation numbers as the values.

For the sapply and lapply outputs the column names will also give the simulation number information by making column names like sim\_number\_1 etc.

There is an option of .use\_data\_table which can greatly enhance the speed of the calculations performed if used while still returning a tibble. The calculations are performed after turning the input data into a data.table object, performing the necessary calculation and then converting back to a tibble object.

### Value

A return of object of either sapply lapply or tibble based upon user input.

### Author(s)

Steven P. Sanderson II, MPH

### See Also

Other Statistic: [ci\\_hi\(\)](#), [ci\\_lo\(\)](#), [tidy\\_kurtosis\\_vec\(\)](#), [tidy\\_range\\_statistic\(\)](#), [tidy\\_skewness\\_vec\(\)](#)

### Examples

```
tn <- tidy_normal(.num_sims = 3)

p <- c(0.025, 0.25, 0.5, 0.75, 0.95)

tidy_stat_tbl(tn, y, quantile, "vector", probs = p, na.rm = TRUE)
tidy_stat_tbl(tn, y, quantile, "list", probs = p)
tidy_stat_tbl(tn, y, quantile, "tibble", probs = p)
tidy_stat_tbl(tn, y, quantile, .use_data_table = TRUE, probs = p, na.rm = TRUE)
```

---

tidy\_t

*Tidy Randomly Generated T Distribution Tibble*

---

### Description

This function will generate n random points from a rt distribution with a user provided, df, ncp, and number of random simulations to be produced. The function returns a tibble with the simulation number column the x column which corresponds to the n randomly generated points, the d\_, p\_ and q\_ data points as well.

The data is returned un-grouped.

The columns that are output are:

- sim\_number The current simulation number.



- x The current value of n for the current simulation.
- y The randomly generated data point.
- dx The x value from the `stats::density()` function.
- dy The y value from the `stats::density()` function.
- p The values from the resulting p\_ function of the distribution family.
- q The values from the resulting q\_ function of the distribution family.

### Usage

```
tidy_t(.n = 50, .df = 1, .ncp = 0, .num_sims = 1, .return_tibble = TRUE)
```

### Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.df</code>	Degrees of freedom, Inf is allowed.
<code>.ncp</code>	Non-centrality parameter.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

### Details

This function uses the underlying `stats::rt()`, and its underlying p, d, and q functions. For more information please see `stats::rt()`

### Value

A tibble of randomly generated data.

### Author(s)

Steven P. Sanderson II, MPH

### See Also

<https://www.itl.nist.gov/div898/handbook/eda/section3/eda3664.htm>

Other Continuous Distribution: `tidy_beta()`, `tidy_burr()`, `tidy_cauchy()`, `tidy_chisquare()`, `tidy_exponential()`, `tidy_f()`, `tidy_gamma()`, `tidy_generalized_beta()`, `tidy_generalized_pareto()`, `tidy_geometric()`, `tidy_inverse_burr()`, `tidy_inverse_exponential()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_inverse_weibull()`, `tidy_logistic()`, `tidy_lognormal()`, `tidy_normal()`, `tidy_paralogistic()`, `tidy_pareto1()`, `tidy_pareto()`, `tidy_triangular()`, `tidy_uniform()`, `tidy_weibull()`, `tidy_zero_truncated_geometric()`

Other T Distribution: `util_t_stats_tbl()`

### Examples

```
tidy_t()
```

---

tidy\_triangular      *Generate Tidy Data from Triangular Distribution*

---

### Description

This function generates tidy data from the triangular distribution.

### Usage

```
tidy_triangular(  
  .n = 50,  
  .min = 0,  
  .max = 1,  
  .mode = 1/2,  
  .num_sims = 1,  
  .return_tibble = TRUE  
)
```

### Arguments

<code>.n</code>	The number of x values for each simulation.
<code>.min</code>	The minimum value of the triangular distribution.
<code>.max</code>	The maximum value of the triangular distribution.
<code>.mode</code>	The mode (peak) value of the triangular distribution.
<code>.num_sims</code>	The number of simulations to perform.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

### Details

The function takes parameters for the triangular distribution (minimum, maximum, mode), the number of x values (`n`), the number of simulations (`num_sims`), and an option to return the result as a tibble (`return_tibble`). It performs various checks on the input parameters to ensure validity. The result is a data frame or tibble with tidy data for further analysis.

### Value

A tibble of randomly generated data.

### Author(s)

Steven P. Sanderson II, MPH

**See Also**

Other Continuous Distribution: `tidy_beta()`, `tidy_burr()`, `tidy_cauchy()`, `tidy_chisquare()`, `tidy_exponential()`, `tidy_f()`, `tidy_gamma()`, `tidy_generalized_beta()`, `tidy_generalized_pareto()`, `tidy_geometric()`, `tidy_inverse_burr()`, `tidy_inverse_exponential()`, `tidy_inverse_gamma()`, `tidy_inverse_normal()`, `tidy_inverse_pareto()`, `tidy_inverse_weibull()`, `tidy_logistic()`, `tidy_lognormal()`, `tidy_normal()`, `tidy_paralogistic()`, `tidy_pareto1()`, `tidy_pareto()`, `tidy_t()`, `tidy_uniform()`, `tidy_weibull()`, `tidy_zero_truncated_geometric()`

Other Triangular: `util_triangular_param_estimate()`, `util_triangular_stats_tbl()`

**Examples**

```
tidy_triangular(.return_tibble = TRUE)
```

---

tidy\_uniform

*Tidy Randomly Generated Uniform Distribution Tibble*


---

**Description**

This function will generate `n` random points from a uniform distribution with a user provided, `.min` and `.max` values, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the `n` randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of `n` for the current simulation.
- `y` The randomly generated data point.
- `dx` The `x` value from the `stats::density()` function.
- `dy` The `y` value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

**Usage**

```
tidy_uniform(.n = 50, .min = 0, .max = 1, .num_sims = 1, .return_tibble = TRUE)
```

**Arguments**

<code>.n</code>	The number of randomly generated points you want.
<code>.min</code>	A lower limit of the distribution.
<code>.max</code>	An upper limit of the distribution
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `stats::runif()`, and its underlying p, d, and q functions. For more information please see [stats::runif\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://www.itl.nist.gov/div898/handbook/eda/section3/eda3662.htm>

Other Continuous Distribution: [tidy\\_beta\(\)](#), [tidy\\_burr\(\)](#), [tidy\\_cauchy\(\)](#), [tidy\\_chisquare\(\)](#), [tidy\\_exponential\(\)](#), [tidy\\_f\(\)](#), [tidy\\_gamma\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_geometric\(\)](#), [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_logistic\(\)](#), [tidy\\_lognormal\(\)](#), [tidy\\_normal\(\)](#), [tidy\\_paralogistic\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [tidy\\_triangular\(\)](#), [tidy\\_t\(\)](#), [tidy\\_weibull\(\)](#), [tidy\\_zero\\_truncated\\_geometric\(\)](#)

Other Uniform: [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#)

**Examples**

```
tidy_uniform()
```

---

tidy\_weibull

*Tidy Randomly Generated Weibull Distribution Tibble*

---

**Description**

This function will generate n random points from a weibull distribution with a user provided, `.shape`, `.scale`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the x column which corresponds to the n randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of n for the current simulation.
- `y` The randomly generated data point.
- `dx` The x value from the [stats::density\(\)](#) function.
- `dy` The y value from the [stats::density\(\)](#) function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

**Usage**

```
tidy_weibull(  
  .n = 50,  
  .shape = 1,  
  .scale = 1,  
  .num_sims = 1,  
  .return_tibble = TRUE  
)
```

**Arguments**

<code>.n</code>	The number of randomly generated points you want.
<code>.shape</code>	Shape parameter defaults to 0.
<code>.scale</code>	Scale parameter defaults to 1.
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `stats::rweibull()`, and its underlying `p`, `d`, and `q` functions. For more information please see [stats::rweibull\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://www.itl.nist.gov/div898/handbook/eda/section3/eda3669.htm>

Other Continuous Distribution: [tidy\\_beta\(\)](#), [tidy\\_burr\(\)](#), [tidy\\_cauchy\(\)](#), [tidy\\_chisquare\(\)](#), [tidy\\_exponential\(\)](#), [tidy\\_f\(\)](#), [tidy\\_gamma\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_geometric\(\)](#), [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_logistic\(\)](#), [tidy\\_lognormal\(\)](#), [tidy\\_normal\(\)](#), [tidy\\_paralogistic\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [tidy\\_triangular\(\)](#), [tidy\\_t\(\)](#), [tidy\\_uniform\(\)](#), [tidy\\_zero\\_truncated\\_geometric\(\)](#)

Other Weibull: [tidy\\_inverse\\_weibull\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

**Examples**

```
tidy_weibull()
```

---

tidy\_zero\_truncated\_binomial

*Tidy Randomly Generated Binomial Distribution Tibble*


---

## Description

This function will generate  $n$  random points from a zero truncated binomial distribution with a user provided, `.size`, `.prob`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

## Usage

```
tidy_zero_truncated_binomial(
  .n = 50,
  .size = 1,
  .prob = 1,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

## Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.size</code>	Number of trials, zero or more.
<code>.prob</code>	Probability of success on each trial $0 \leq \text{prob} \leq 1$ .
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

## Details

This function uses the underlying `actuar::rztbinom()`, and its underlying `p`, `d`, and `q` functions. For more information please see `actuar::rztbinom()`

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Discrete Distribution: `tidy_bernoulli()`, `tidy_binomial()`, `tidy_hypergeometric()`, `tidy_negative_binomial()`, `tidy_poisson()`, `tidy_zero_truncated_negative_binomial()`, `tidy_zero_truncated_poisson()`

Other Binomial: `tidy_binomial()`, `tidy_negative_binomial()`, `tidy_zero_truncated_negative_binomial()`, `util_binomial_param_estimate()`, `util_binomial_stats_tbl()`, `util_negative_binomial_param_estimate()`

Other Zero Truncated Distribution: `tidy_zero_truncated_geometric()`, `tidy_zero_truncated_poisson()`

**Examples**

```
tidy_zero_truncated_binomial()
```

---

```
tidy_zero_truncated_geometric
```

*Tidy Randomly Generated Zero Truncated Geometric Distribution Tib-  
ble*

---

**Description**

This function will generate  $n$  random points from a zero truncated Geometric distribution with a user provided, `.prob`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

**Usage**

```
tidy_zero_truncated_geometric(
  .n = 50,
  .prob = 1,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

**Arguments**

`.n` The number of randomly generated points you want.

`.prob` A probability of success in each trial  $0 < \text{prob} \leq 1$ .

`.num_sims` The number of randomly generated simulations you want.

`.return_tibble` A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `actuar::rztgeom()`, and its underlying `p`, `d`, and `q` functions. For more information please see [actuar::rztgeom\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Geometric: [tidy\\_geometric\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#)

Other Continuous Distribution: [tidy\\_beta\(\)](#), [tidy\\_burr\(\)](#), [tidy\\_cauchy\(\)](#), [tidy\\_chisquare\(\)](#), [tidy\\_exponential\(\)](#), [tidy\\_f\(\)](#), [tidy\\_gamma\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_geometric\(\)](#), [tidy\\_inverse\\_burr\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [tidy\\_inverse\\_normal\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_logistic\(\)](#), [tidy\\_lognormal\(\)](#), [tidy\\_normal\(\)](#), [tidy\\_paralogistic\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [tidy\\_triangular\(\)](#), [tidy\\_t\(\)](#), [tidy\\_uniform\(\)](#), [tidy\\_weibull\(\)](#)

Other Zero Truncated Distribution: [tidy\\_zero\\_truncated\\_binomial\(\)](#), [tidy\\_zero\\_truncated\\_poisson\(\)](#)

**Examples**

```
tidy_zero_truncated_geometric()
```



---

tidy\_zero\_truncated\_negative\_binomial

*Tidy Randomly Generated Binomial Distribution Tibble*


---

## Description

This function will generate  $n$  random points from a zero truncated binomial distribution with a user provided, `.size`, `.prob`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

## Usage

```
tidy_zero_truncated_negative_binomial(
  .n = 50,
  .size = 0,
  .prob = 1,
  .num_sims = 1,
  .return_tibble = TRUE
)
```

## Arguments

<code>.n</code>	The number of randomly generated points you want.
<code>.size</code>	Number of trials, zero or more.
<code>.prob</code>	Probability of success on each trial $0 \leq \text{prob} \leq 1$ .
<code>.num_sims</code>	The number of randomly generated simulations you want.
<code>.return_tibble</code>	A logical value indicating whether to return the result as a tibble. Default is TRUE.

## Details

This function uses the underlying `actuar::rztbinom()`, and its underlying `p`, `d`, and `q` functions. For more information please see `actuar::rztbinom()`

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Discrete Distribution: `tidy_bernoulli()`, `tidy_binomial()`, `tidy_hypergeometric()`, `tidy_negative_binomial()`, `tidy_poisson()`, `tidy_zero_truncated_binomial()`, `tidy_zero_truncated_poisson()`

Other Binomial: `tidy_binomial()`, `tidy_negative_binomial()`, `tidy_zero_truncated_binomial()`, `util_binomial_param_estimate()`, `util_binomial_stats_tbl()`, `util_negative_binomial_param_estimate()`

**Examples**

```
tidy_zero_truncated_negative_binomial()
```

---

```
tidy_zero_truncated_poisson
```

*Tidy Randomly Generated Zero Truncated Poisson Distribution Tibble*

---

**Description**

This function will generate  $n$  random points from a Zero Truncated Poisson distribution with a user provided, `.lambda`, and number of random simulations to be produced. The function returns a tibble with the simulation number column the `x` column which corresponds to the  $n$  randomly generated points, the `d_`, `p_` and `q_` data points as well.

The data is returned un-grouped.

The columns that are output are:

- `sim_number` The current simulation number.
- `x` The current value of  $n$  for the current simulation.
- `y` The randomly generated data point.
- `dx` The  $x$  value from the `stats::density()` function.
- `dy` The  $y$  value from the `stats::density()` function.
- `p` The values from the resulting `p_` function of the distribution family.
- `q` The values from the resulting `q_` function of the distribution family.

**Usage**

```
tidy_zero_truncated_poisson(  
  .n = 50,  
  .lambda = 1,  
  .num_sims = 1,  
  .return_tibble = TRUE  
)
```

**Arguments**

`.n` The number of randomly generated points you want.

`.lambda` A vector of non-negative means.

`.num_sims` The number of randomly generated simulations you want.

`.return_tibble` A logical value indicating whether to return the result as a tibble. Default is TRUE.

**Details**

This function uses the underlying `actuar::rztpois()`, and its underlying `p`, `d`, and `q` functions. For more information please see [actuar::rztpois\(\)](#)

**Value**

A tibble of randomly generated data.

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

<https://openacttexts.github.io/Loss-Data-Analytics/ChapSummaryDistributions.html>

Other Poisson: [tidy\\_poisson\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#)

Other Zero Truncated Distribution: [tidy\\_zero\\_truncated\\_binomial\(\)](#), [tidy\\_zero\\_truncated\\_geometric\(\)](#)

Other Discrete Distribution: [tidy\\_bernoulli\(\)](#), [tidy\\_binomial\(\)](#), [tidy\\_hypergeometric\(\)](#), [tidy\\_negative\\_binomial\(\)](#), [tidy\\_poisson\(\)](#), [tidy\\_zero\\_truncated\\_binomial\(\)](#), [tidy\\_zero\\_truncated\\_negative](#)

**Examples**

```
tidy_zero_truncated_poisson()
```

---

triangle_plot	<i>Triangle Distribution PDF Plot</i>
---------------	---------------------------------------

---

### Description

This function generates a probability density function (PDF) plot for the triangular distribution.

### Usage

```
triangle_plot(.data, .interactive = FALSE)
```

### Arguments

<code>.data</code>	Tidy data from the <code>tidy_triangular</code> function.
<code>.interactive</code>	A logical value indicating whether to return an interactive plot using plotly. Default is FALSE.

### Details

The function checks if the input data is a data frame or tibble, and if it comes from the `tidy_triangular` function. It then extracts necessary attributes for the plot and creates a PDF plot using `ggplot2`. The plot includes data points and segments to represent the triangular distribution.

### Value

The function returns a `ggplot2` object representing the probability density function plot for the triangular distribution.

### Author(s)

Steven P. Sanderson II, MPH

### Examples

```
# Example: Generating a PDF plot for the triangular distribution
data <- tidy_triangular(.n = 50, .min = 0, .max = 1, .mode = 1/2, .num_sims = 1,
  .return_tibble = TRUE)
triangle_plot(data)
```

---

`util_bernoulli_param_estimate`*Estimate Bernoulli Parameters*

---

## Description

This function will attempt to estimate the Bernoulli prob parameter given some vector of values `.x`. The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to `TRUE` then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated Bernoulli data.

## Usage

```
util_bernoulli_param_estimate(.x, .auto_gen_empirical = TRUE)
```

## Arguments

`.x` The vector of data to be passed to the function. Must be non-negative integers.

`.auto_gen_empirical` This is a boolean value of `TRUE/FALSE` with default set to `TRUE`. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

## Details

This function will see if the given vector `.x` is a numeric vector. It will attempt to estimate the prob parameter of a Bernoulli distribution.

## Value

A tibble/list

## Author(s)

Steven P. Sanderson II, MPH

## See Also

Other Parameter Estimation: [util\\_beta\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Bernoulli: [tidy\\_bernoulli\(\)](#), [util\\_bernoulli\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)
library(ggplot2)

tb <- tidy_bernoulli(.prob = .1) |> pull(y)
output <- util_bernoulli_param_estimate(tb)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combinedautoplot()
```

---

util\_bernoulli\_stats\_tbl

*Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_bernoulli_stats_tbl(.data)
```

**Arguments**

`.data`            The data being passed from a tidy\_ distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of tidy\_ distribution. It is required that data be passed from a tidy\_ distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Bernoulli: [tidy\\_bernoulli\(\)](#), [util\\_bernoulli\\_param\\_estimate\(\)](#)

Other Distribution Statistics: [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)

tidy_bernoulli() |>
  util_bernoulli_stats_tbl() |>
  glimpse()
```

---

```
util_beta_param_estimate
```

*Estimate Beta Parameters*

---

**Description**

This function will automatically scale the data from 0 to 1 if it is not already. This means you can pass a vector like `mtcars$mpg` and not worry about it.

The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to `TRUE` then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated beta data.

Three different methods of shape parameters are supplied:

- Bayes
- NIST mme
- EnvStats mme, see [EnvStats::ebeta\(\)](#)

**Usage**

```
util_beta_param_estimate(.x, .auto_gen_empirical = TRUE)
```

**Arguments**

`.x` The vector of data to be passed to the function. Must be numeric, and all values must be  $0 \leq x \leq 1$

`.auto_gen_empirical` This is a boolean value of `TRUE/FALSE` with default set to `TRUE`. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

**Details**

This function will attempt to estimate the beta shape1 and shape2 parameters given some vector of values.

**Value**

A tibble/list

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Beta: [tidy\\_beta\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)
library(ggplot2)

x <- mtcars$mpg
output <- util_beta_param_estimate(x)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combinedautoplot()

tb <- rbeta(50, 2.5, 1.4)
util_beta_param_estimate(tb)$parameter_tbl
```

---

util\_beta\_stats\_tbl    *Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_beta_stats_tbl(.data)
```



## Arguments

`.data` The data being passed from a `tidy_` distribution function.

## Details

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

## Value

A tibble

## Author(s)

Steven P. Sanderson II, MPH

## See Also

Other Beta: [tidy\\_beta\(\)](#), [tidy\\_generalized\\_beta\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

## Examples

```
library(dplyr)

tidy_beta() |>
  util_beta_stats_tbl() |>
  glimpse()
```

---

util\_binomial\_param\_estimate

*Estimate Binomial Parameters*

---

## Description

This function will check to see if some given vector `.x` is either a numeric vector or a factor vector with at least two levels then it will cause an error and the function will abort. The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to `TRUE` then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated binomial data.

**Usage**

```
util_binomial_param_estimate(.x, .size = NULL, .auto_gen_empirical = TRUE)
```

**Arguments**

`.x` The vector of data to be passed to the function. Must be numeric, and all values must be  $0 \leq x \leq 1$

`.size` Number of trials, zero or more.

`.auto_gen_empirical`  
This is a boolean value of TRUE/FALSE with default set to TRUE. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

**Details**

This function will attempt to estimate the binomial  $p_{\hat{}}$  and size parameters given some vector of values.

**Value**

A tibble/list

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Binomial: [tidy\\_binomial\(\)](#), [tidy\\_negative\\_binomial\(\)](#), [tidy\\_zero\\_truncated\\_binomial\(\)](#), [tidy\\_zero\\_truncated\\_negative\\_binomial\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#)

**Examples**

```
library(dplyr)
library(ggplot2)

tb <- rbinom(50, 1, .1)
output <- util_binomial_param_estimate(tb)

output$parameter_tbl

output$combined_data_tbl |>
```

```
tidy_combinedautoplot()
```

---

```
util_binomial_stats_tbl
```

*Distribution Statistics*

---

## Description

Returns distribution statistics in a tibble.

## Usage

```
util_binomial_stats_tbl(.data)
```

## Arguments

`.data` The data being passed from a `tidy_` distribution function.

## Details

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

## Value

A tibble

## Author(s)

Steven P. Sanderson II, MPH

## See Also

Other Binomial: [tidy\\_binomial\(\)](#), [tidy\\_negative\\_binomial\(\)](#), [tidy\\_zero\\_truncated\\_binomial\(\)](#), [tidy\\_zero\\_truncated\\_negative\\_binomial\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

## Examples

```
library(dplyr)

tidy_binomial() |>
  util_binomial_stats_tbl() |>
  glimpse()
```

---

util\_burr\_param\_estimate

*Estimate Burr Parameters*

---

## Description

This function will attempt to estimate the Burr prob parameter given some vector of values `.x`. The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to TRUE then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated Burr data.

## Usage

```
util_burr_param_estimate(.x, .auto_gen_empirical = TRUE)
```

## Arguments

`.x` The vector of data to be passed to the function. Must be non-negative integers.

`.auto_gen_empirical` This is a boolean value of TRUE/FALSE with default set to TRUE. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

## Details

This function will see if the given vector `.x` is a numeric vector. It will attempt to estimate the prob parameter of a Burr distribution.

## Value

A tibble/list

## Author(s)

Steven P. Sanderson II, MPH

**See Also**

Other Parameter Estimation: `util_bernoulli_param_estimate()`, `util_beta_param_estimate()`, `util_binomial_param_estimate()`, `util_cauchy_param_estimate()`, `util_exponential_param_estimate()`, `util_gamma_param_estimate()`, `util_geometric_param_estimate()`, `util_hypergeometric_param_estimate()`, `util_logistic_param_estimate()`, `util_lognormal_param_estimate()`, `util_negative_binomial_param_estimate()`, `util_normal_param_estimate()`, `util_pareto_param_estimate()`, `util_poisson_param_estimate()`, `util_triangular_param_estimate()`, `util_uniform_param_estimate()`, `util_weibull_param_estimate()`

Other Burr: `tidy_burr()`, `tidy_inverse_burr()`, `util_burr_stats_tbl()`

**Examples**

```
library(dplyr)
library(ggplot2)

tb <- tidy_burr(.shape1 = 1, .shape2 = 2, .rate = .3) |> pull(y)
output <- util_burr_param_estimate(tb)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combined_autoplot()
```

---

util\_burr\_stats\_tbl     *Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_burr_stats_tbl(.data)
```

**Arguments**

`.data`             The data being passed from a `tidy_` distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Burr: `tidy_burr()`, `tidy_inverse_burr()`, `util_burr_param_estimate()`

Other Distribution Statistics: `util_bernoulli_stats_tbl()`, `util_beta_stats_tbl()`, `util_binomial_stats_tbl()`, `util_cauchy_stats_tbl()`, `util_chisquare_stats_tbl()`, `util_exponential_stats_tbl()`, `util_f_stats_tbl()`, `util_gamma_stats_tbl()`, `util_geometric_stats_tbl()`, `util_hypergeometric_stats_tbl()`, `util_logistic_stats_tbl()`, `util_lognormal_stats_tbl()`, `util_negative_binomial_stats_tbl()`, `util_normal_stats_tbl()`, `util_pareto_stats_tbl()`, `util_poisson_stats_tbl()`, `util_t_stats_tbl()`, `util_triangular_stats_tbl()`, `util_uniform_stats_tbl()`, `util_weibull_stats_tbl()`

**Examples**

```
library(dplyr)

tidy_burr() |>
  util_burr_stats_tbl() |>
  glimpse()
```

---

util\_cauchy\_param\_estimate

*Estimate Cauchy Parameters*

---

**Description**

The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to TRUE then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated cauchy data.

**Usage**

```
util_cauchy_param_estimate(.x, .auto_gen_empirical = TRUE)
```

**Arguments**

`.x` The vector of data to be passed to the function.

`.auto_gen_empirical`

This is a boolean value of TRUE/FALSE with default set to TRUE. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

**Details**

This function will attempt to estimate the cauchy location and scale parameters given some vector of values.

**Value**

A tibble/list

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Cauchy: [tidy\\_cauchy\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)
library(ggplot2)

x <- tidy_cauchy(.location = 0, .scale = 1)$y
output <- util_cauchy_param_estimate(x)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combined_autoplot()
```

---

util\_cauchy\_stats\_tbl *Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_cauchy_stats_tbl(.data)
```

**Arguments**

`.data` The data being passed from a `tidy_` distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Cauchy: [tidy\\_cauchy\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)

tidy_cauchy() |>
  util_cauchy_stats_tbl() |>
  glimpse()
```

---

util\_chisquare\_stats\_tbl

*Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_chisquare_stats_tbl(.data)
```



## Arguments

`.data` The data being passed from a `tidy_` distribution function.

## Details

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

## Value

A tibble

## Author(s)

Steven P. Sanderson II, MPH

## See Also

Other Chisquare: [tidy\\_chisquare\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

## Examples

```
library(dplyr)

tidy_chisquare() |>
  util_chisquare_stats_tbl() |>
  glimpse()
```

---

util\_exponential\_param\_estimate

*Estimate Exponential Parameters*

---

## Description

This function will attempt to estimate the exponential rate parameter given some vector of values. The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to `TRUE` then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated exponential data.

**Usage**

```
util_exponential_param_estimate(.x, .auto_gen_empirical = TRUE)
```

**Arguments**

`.x` The vector of data to be passed to the function. Must be numeric.

`.auto_gen_empirical` This is a boolean value of TRUE/FALSE with default set to TRUE. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

**Details**

This function will see if the given vector `.x` is a numeric vector.

**Value**

A tibble/list

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Exponential: [tidy\\_exponential\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)
library(ggplot2)

te <- tidy_exponential(.rate = .1) |> pull(y)
output <- util_exponential_param_estimate(te)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combinedautoplot()
```

---

`util_exponential_stats_tbl`*Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_exponential_stats_tbl(.data)
```

**Arguments**

`.data` The data being passed from a `tidy_` distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Exponential: [tidy\\_exponential\(\)](#), [tidy\\_inverse\\_exponential\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)

tidy_exponential() |>
  util_exponential_stats_tbl() |>
  glimpse()
```

---

util_f_stats_tbl	<i>Distribution Statistics</i>
------------------	--------------------------------

---

### Description

Returns distribution statistics in a tibble.

### Usage

```
util_f_stats_tbl(.data)
```

### Arguments

`.data` The data being passed from a `tidy_` distribution function.

### Details

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

### Value

A tibble

### Author(s)

Steven P. Sanderson II, MPH

### See Also

Other F Distribution: [tidy\\_f\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

### Examples

```
library(dplyr)

tidy_f() |>
  util_f_stats_tbl() |>
  glimpse()
```

---

`util_gamma_param_estimate`*Estimate Gamma Parameters*

---

### Description

This function will attempt to estimate the gamma shape and scale parameters given some vector of values. The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to `TRUE` then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated gamma data.

### Usage

```
util_gamma_param_estimate(.x, .auto_gen_empirical = TRUE)
```

### Arguments

`.x` The vector of data to be passed to the function. Must be numeric.

`.auto_gen_empirical` This is a boolean value of `TRUE/FALSE` with default set to `TRUE`. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

### Details

This function will see if the given vector `.x` is a numeric vector.

### Value

A tibble/list

### Author(s)

Steven P. Sanderson II, MPH

### See Also

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Gamma: [tidy\\_gamma\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#)

## Examples

```
library(dplyr)
library(ggplot2)

tg <- tidy_gamma(.shape = 1, .scale = .3) |> pull(y)
output <- util_gamma_param_estimate(tg)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combinedautoplot()
```

---

util\_gamma\_stats\_tbl *Distribution Statistics*

---

## Description

Returns distribution statistics in a tibble.

## Usage

```
util_gamma_stats_tbl(.data)
```

## Arguments

`.data` The data being passed from a `tidy_` distribution function.

## Details

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

## Value

A tibble

## Author(s)

Steven P. Sanderson II, MPH

## See Also

Other Gamma: [tidy\\_gamma\(\)](#), [tidy\\_inverse\\_gamma\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)

tidy_gamma() |>
  util_gamma_stats_tbl() |>
  glimpse()
```

---

util\_geometric\_param\_estimate  
*Estimate Geometric Parameters*

---

**Description**

This function will attempt to estimate the geometric prob parameter given some vector of values `.x`. The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to TRUE then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated geometric data.

**Usage**

```
util_geometric_param_estimate(.x, .auto_gen_empirical = TRUE)
```

**Arguments**

`.x` The vector of data to be passed to the function. Must be non-negative integers.

`.auto_gen_empirical` This is a boolean value of TRUE/FALSE with default set to TRUE. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

**Details**

This function will see if the given vector `.x` is a numeric vector. It will attempt to estimate the prob parameter of a geometric distribution.

**Value**

A tibble/list

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Parameter Estimation: `util_bernoulli_param_estimate()`, `util_beta_param_estimate()`, `util_binomial_param_estimate()`, `util_burr_param_estimate()`, `util_cauchy_param_estimate()`, `util_exponential_param_estimate()`, `util_gamma_param_estimate()`, `util_hypergeometric_param_estimate()`, `util_logistic_param_estimate()`, `util_lognormal_param_estimate()`, `util_negative_binomial_param_estimate()`, `util_normal_param_estimate()`, `util_pareto_param_estimate()`, `util_poisson_param_estimate()`, `util_triangular_param_estimate()`, `util_uniform_param_estimate()`, `util_weibull_param_estimate()`

Other Geometric: `tidy_geometric()`, `tidy_zero_truncated_geometric()`, `util_geometric_stats_tbl()`

**Examples**

```
library(dplyr)
library(ggplot2)

tg <- tidy_geometric(.prob = .1) |> pull(y)
output <- util_geometric_param_estimate(tg)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combinedautoplot()
```

---

```
util_geometric_stats_tbl
      Distribution Statistics
```

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_geometric_stats_tbl(.data)
```

**Arguments**

`.data`            The data being passed from a `tidy_` distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

**Value**

A tibble



**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Geometric: `tidy_geometric()`, `tidy_zero_truncated_geometric()`, `util_geometric_param_estimate()`

Other Distribution Statistics: `util_bernoulli_stats_tbl()`, `util_beta_stats_tbl()`, `util_binomial_stats_tbl()`, `util_burr_stats_tbl()`, `util_cauchy_stats_tbl()`, `util_chisquare_stats_tbl()`, `util_exponential_stats_tbl()`, `util_f_stats_tbl()`, `util_gamma_stats_tbl()`, `util_hypergeometric_stats_tbl()`, `util_logistic_stats_tbl()`, `util_lognormal_stats_tbl()`, `util_negative_binomial_stats_tbl()`, `util_normal_stats_tbl()`, `util_pareto_stats_tbl()`, `util_poisson_stats_tbl()`, `util_t_stats_tbl()`, `util_triangular_stats_tbl()`, `util_uniform_stats_tbl()`, `util_weibull_stats_tbl()`

**Examples**

```
library(dplyr)

tidy_geometric() |>
  util_geometric_stats_tbl() |>
  glimpse()
```

---

util\_hypergeometric\_param\_estimate

*Estimate Hypergeometric Parameters*

---

**Description**

This function will attempt to estimate the geometric prob parameter given some vector of values `.x`. Estimate `m`, the number of white balls in the urn, or `m+n`, the total number of balls in the urn, for a hypergeometric distribution.

**Usage**

```
util_hypergeometric_param_estimate(
  .x,
  .m = NULL,
  .total = NULL,
  .k,
  .auto_gen_empirical = TRUE
)
```

**Arguments**

- `.x` A non-negative integer indicating the number of white balls out of a sample of size `.k` drawn without replacement from the urn. You cannot have missing, undefined or infinite values.
- `.m` Non-negative integer indicating the number of white balls in the urn. You must supply `.m` or `.total`, but not both. You cannot have missing values.
- `.total` A positive integer indicating the total number of balls in the urn (i.e.,  $m+n$ ). You must supply `.m` or `.total`, but not both. You cannot have missing values.
- `.k` A positive integer indicating the number of balls drawn without replacement from the urn. You cannot have missing values.
- `.auto_gen_empirical`  
This is a boolean value of TRUE/FALSE with default set to TRUE. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

**Details**

This function will see if the given vector `.x` is a numeric integer. It will attempt to estimate the prob parameter of a geometric distribution. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. Let `.x` be an observation from a hypergeometric distribution with parameters `.m` = M, `.n` = N, and `.k` = K. In R nomenclature, `.x` represents the number of white balls drawn out of a sample of `.k` balls drawn without replacement from an urn containing `.m` white balls and `.n` black balls. The total number of balls in the urn is thus `.m + .n`. Denote the total number of balls by  $T = .m + .n$

**Value**

A tibble/list

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Hypergeometric: [tidy\\_hypergeometric\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)
library(ggplot2)

th <- rhyper(10, 20, 30, 5)
output <- util_hypergeometric_param_estimate(th, .total = 50, .k = 5)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combinedautoplot()
```

---

util\_hypergeometric\_stats\_tbl  
*Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_hypergeometric_stats_tbl(.data)
```

**Arguments**

`.data`            The data being passed from a `tidy_` distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Hypergeometric: [tidy\\_hypergeometric\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)

tidy_hypergeometric() |>
  util_hypergeometric_stats_tbl() |>
  glimpse()
```

---

```
util_logistic_param_estimate
```

*Estimate Logistic Parameters*

---

**Description**

The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to TRUE then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated logistic data.

Three different methods of shape parameters are supplied:

- MLE
- MME
- MMUE

**Usage**

```
util_logistic_param_estimate(.x, .auto_gen_empirical = TRUE)
```

**Arguments**

`.x` The vector of data to be passed to the function.

`.auto_gen_empirical`

This is a boolean value of TRUE/FALSE with default set to TRUE. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

**Details**

This function will attempt to estimate the logistic location and scale parameters given some vector of values.

**Value**

A tibble/list

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Logistic: [tidy\\_logistic\(\)](#), [tidy\\_paralogistic\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)
library(ggplot2)

x <- mtcars$mpg
output <- util_logistic_param_estimate(x)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combinedautoplot()

t <- rlogis(50, 2.5, 1.4)
util_logistic_param_estimate(t)$parameter_tbl
```

---

util\_logistic\_stats\_tbl

*Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_logistic_stats_tbl(.data)
```

**Arguments**

`.data`            The data being passed from a `tidy_` distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Logistic: [tidy\\_logistic\(\)](#), [tidy\\_paralogistic\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)

tidy_logistic() |>
  util_logistic_stats_tbl() |>
  glimpse()
```

---

util\_lognormal\_param\_estimate

*Estimate Lognormal Parameters*

---

## Description

The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to `TRUE` then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated lognormal data.

Three different methods of shape parameters are supplied:

- `mme`, see [EnvStats::elnorm\(\)](#)
- `mle`, see [EnvStats::elnorm\(\)](#)

## Usage

```
util_lognormal_param_estimate(.x, .auto_gen_empirical = TRUE)
```

## Arguments

`.x` The vector of data to be passed to the function.

`.auto_gen_empirical` This is a boolean value of `TRUE/FALSE` with default set to `TRUE`. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

## Details

This function will attempt to estimate the lognormal meanlog and log sd parameters given some vector of values.

## Value

A tibble/list

## Author(s)

Steven P. Sanderson II, MPH

## See Also

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_es](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Lognormal: [tidy\\_lognormal\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)
library(ggplot2)

x <- mtcars$mpg
output <- util_lognormal_param_estimate(x)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combinedautoplot()

tb <- tidy_lognormal(.meanlog = 2, .sdlog = 1) |> pull(y)
util_lognormal_param_estimate(tb)$parameter_tbl
```

---

util\_lognormal\_stats\_tbl

*Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_lognormal_stats_tbl(.data)
```

**Arguments**

`.data`            The data being passed from a `tidy_` distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH



**See Also**

Other Lognormal: [tidy\\_lognormal\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)

tidy_lognormal() |>
  util_lognormal_stats_tbl() |>
  glimpse()
```

---

```
util_negative_binomial_param_estimate
      Estimate Negative Binomial Parameters
```

---

**Description**

The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to TRUE then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated negative binomial data.

Two different methods of shape parameters are supplied:

- MLE/MME
- MMUE

**Usage**

```
util_negative_binomial_param_estimate(.x, .size, .auto_gen_empirical = TRUE)
```

**Arguments**

`.x` The vector of data to be passed to the function.

`.size` The size parameter.

`.auto_gen_empirical` This is a boolean value of TRUE/FALSE with default set to TRUE. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

**Details**

This function will attempt to estimate the negative binomial size and prob parameters given some vector of values.

**Value**

A tibble/list

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Binomial: [tidy\\_binomial\(\)](#), [tidy\\_negative\\_binomial\(\)](#), [tidy\\_zero\\_truncated\\_binomial\(\)](#), [tidy\\_zero\\_truncated\\_negative\\_binomial\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)
library(ggplot2)

x <- as.integer(mtcars$mpg)
output <- util_negative_binomial_param_estimate(x, .size = 1)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combinedautoplot()

t <- rnbinom(50, 1, .1)
util_negative_binomial_param_estimate(t, .size = 1)$parameter_tbl
```

---

util\_negative\_binomial\_stats\_tbl

*Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_negative_binomial_stats_tbl(.data)
```

**Arguments**

`.data`            The data being passed from a tidy\_ distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of tidy\_ distribution. It is required that data be passed from a tidy\_ distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)

tidy_negative_binomial() |>
  util_negative_binomial_stats_tbl() |>
  glimpse()
```

**Description**

The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to `TRUE` then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated normal data.

Three different methods of shape parameters are supplied:

- MLE/MME
- MVUE

**Usage**

```
util_normal_param_estimate(.x, .auto_gen_empirical = TRUE)
```

**Arguments**

`.x`                    The vector of data to be passed to the function.

`.auto_gen_empirical`  
                          This is a boolean value of `TRUE/FALSE` with default set to `TRUE`. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

**Details**

This function will attempt to estimate the normal gaussian mean and standard deviation parameters given some vector of values.

**Value**

A tibble/list

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Gaussian: [tidy\\_inverse\\_normal\(\)](#), [tidy\\_normal\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)
library(ggplot2)

x <- mtcars$mpg
output <- util_normal_param_estimate(x)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combinedautoplot()

t <- rnorm(50, 0, 1)
util_normal_param_estimate(t)$parameter_tbl
```

---

util\_normal\_stats\_tbl *Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_normal_stats_tbl(.data)
```

**Arguments**

`.data`            The data being passed from a `tidy_` distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Gaussian: `tidy_inverse_normal()`, `tidy_normal()`, `util_normal_param_estimate()`

Other Distribution Statistics: `util_bernoulli_stats_tbl()`, `util_beta_stats_tbl()`, `util_binomial_stats_tbl()`, `util_burr_stats_tbl()`, `util_cauchy_stats_tbl()`, `util_chisquare_stats_tbl()`, `util_exponential_stats_tbl()`, `util_f_stats_tbl()`, `util_gamma_stats_tbl()`, `util_geometric_stats_tbl()`, `util_hypergeometric_stats_tbl()`, `util_logistic_stats_tbl()`, `util_lognormal_stats_tbl()`, `util_negative_binomial_stats_tbl()`, `util_pareto_stats_tbl()`, `util_poisson_stats_tbl()`, `util_t_stats_tbl()`, `util_triangular_stats_tbl()`, `util_uniform_stats_tbl()`, `util_weibull_stats_tbl()`

**Examples**

```
library(dplyr)

tidy_normal() |>
  util_normal_stats_tbl() |>
  glimpse()
```

---

```
util_pareto_param_estimate
```

*Estimate Pareto Parameters*

---

**Description**

The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to TRUE then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated pareto data.

Two different methods of shape parameters are supplied:

- LSE
- MLE

**Usage**

```
util_pareto_param_estimate(.x, .auto_gen_empirical = TRUE)
```

**Arguments**

`.x` The vector of data to be passed to the function.

`.auto_gen_empirical`

This is a boolean value of TRUE/FALSE with default set to TRUE. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

**Details**

This function will attempt to estimate the pareto shape and scale parameters given some vector of values.

**Value**

A tibble/list

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Pareto: [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)
library(ggplot2)

x <- mtcars$mpg
output <- util_pareto_param_estimate(x)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combinedautoplot()

t <- tidy_pareto(50, 1, 1) |> pull(y)
util_pareto_param_estimate(t)$parameter_tbl
```

---

util\_pareto\_stats\_tbl *Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_pareto_stats_tbl(.data)
```

**Arguments**

`.data`            The data being passed from a `tidy_` distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Pareto: [tidy\\_generalized\\_pareto\(\)](#), [tidy\\_inverse\\_pareto\(\)](#), [tidy\\_pareto1\(\)](#), [tidy\\_pareto\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)

tidy_pareto() |>
  util_pareto_stats_tbl() |>
  glimpse()
```



---

`util_poisson_param_estimate`*Estimate Poisson Parameters*

---

## Description

The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to `TRUE` then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated poisson data.

## Usage

```
util_poisson_param_estimate(.x, .auto_gen_empirical = TRUE)
```

## Arguments

`.x` The vector of data to be passed to the function.

`.auto_gen_empirical` This is a boolean value of `TRUE/FALSE` with default set to `TRUE`. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

## Details

This function will attempt to estimate the pareto lambda parameter given some vector of values.

## Value

A tibble/list

## Author(s)

Steven P. Sanderson II, MPH

## See Also

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Poisson: [tidy\\_poisson\(\)](#), [tidy\\_zero\\_truncated\\_poisson\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)
library(ggplot2)

x <- as.integer(mtcars$mpg)
output <- util_poisson_param_estimate(x)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combinedautoplot()

t <- rpois(50, 5)
util_poisson_param_estimate(t)$parameter_tbl
```

---

util\_poisson\_stats\_tbl

*Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_poisson_stats_tbl(.data)
```

**Arguments**

`.data`            The data being passed from a `tidy_` distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Poisson: `tidy_poisson()`, `tidy_zero_truncated_poisson()`, `util_poisson_param_estimate()`

Other Distribution Statistics: `util_bernoulli_stats_tbl()`, `util_beta_stats_tbl()`, `util_binomial_stats_tbl()`, `util_burr_stats_tbl()`, `util_cauchy_stats_tbl()`, `util_chisquare_stats_tbl()`, `util_exponential_stats_tbl()`, `util_f_stats_tbl()`, `util_gamma_stats_tbl()`, `util_geometric_stats_tbl()`, `util_hypergeometric_stats_tbl()`, `util_logistic_stats_tbl()`, `util_lognormal_stats_tbl()`, `util_negative_binomial_stats_tbl()`, `util_normal_stats_tbl()`, `util_pareto_stats_tbl()`, `util_t_stats_tbl()`, `util_triangular_stats_tbl()`, `util_uniform_stats_tbl()`, `util_weibull_stats_tbl()`

**Examples**

```
library(dplyr)

tidy_poisson() |>
  util_poisson_stats_tbl() |>
  glimpse()
```

---

util\_triangular\_param\_estimate

*Estimate Triangular Parameters*

---

**Description**

This function will attempt to estimate the triangular min, mode, and max parameters given some vector of values.

The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to TRUE then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated beta data.

**Usage**

```
util_triangular_param_estimate(.x, .auto_gen_empirical = TRUE)
```

**Arguments**

`.x` The vector of data to be passed to the function. Must be numeric, and all values must be  $0 \leq x \leq 1$

`.auto_gen_empirical` This is a boolean value of TRUE/FALSE with default set to TRUE. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

**Details**

This function will attempt to estimate the triangular min, mode, and max parameters given some vector of values.

**Value**

A tibble/list

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Triangular: [tidy\\_triangular\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)
library(ggplot2)

x <- mtcars$mpg
output <- util_triangular_param_estimate(x)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combined_autoplot()

params <- tidy_triangular()$y |>
  util_triangular_param_estimate()
params$parameter_tbl
```

---

util\_triangular\_stats\_tbl

*Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_triangular_stats_tbl(.data)
```

**Arguments**

`.data` The data being passed from a `tidy_` distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Triangular: [tidy\\_triangular\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)

tidy_triangular() |>
  util_triangular_stats_tbl() |>
  glimpse()
```

---

`util_t_stats_tbl` *Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_t_stats_tbl(.data)
```

**Arguments**

`.data` The data being passed from a `tidy_` distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of tidy\_ distribution. It is required that data be passed from a tidy\_ distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other T Distribution: [tidy\\_t\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)

tidy_t() |>
  util_t_stats_tbl() |>
  glimpse()
```

---

util\_uniform\_param\_estimate

*Estimate Uniform Parameters*

---

**Description**

The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to `TRUE` then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated uniform data.

**Usage**

```
util_uniform_param_estimate(.x, .auto_gen_empirical = TRUE)
```

## Arguments

- `.x` The vector of data to be passed to the function.
- `.auto_gen_empirical` This is a boolean value of TRUE/FALSE with default set to TRUE. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

## Details

This function will attempt to estimate the uniform min and max parameters given some vector of values.

## Value

A tibble/list

## Author(s)

Steven P. Sanderson II, MPH

## See Also

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Uniform: [tidy\\_uniform\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#)

## Examples

```
library(dplyr)
library(ggplot2)

x <- tidy_uniform(.min = 1, .max = 3)$y
output <- util_uniform_param_estimate(x)

output$parameter_tbl

output$combined_data_tbl |>
  tidy_combinedautoplot()
```

---

`util_uniform_stats_tbl`*Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_uniform_stats_tbl(.data)
```

**Arguments**

`.data` The data being passed from a `tidy_` distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Uniform: [tidy\\_uniform\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)

tidy_uniform() |>
  util_uniform_stats_tbl() |>
  glimpse()
```



---

`util_weibull_param_estimate`*Estimate Weibull Parameters*

---

## Description

The function will return a list output by default, and if the parameter `.auto_gen_empirical` is set to `TRUE` then the empirical data given to the parameter `.x` will be run through the `tidy_empirical()` function and combined with the estimated weibull data.

## Usage

```
util_weibull_param_estimate(.x, .auto_gen_empirical = TRUE)
```

## Arguments

`.x` The vector of data to be passed to the function.

`.auto_gen_empirical` This is a boolean value of `TRUE/FALSE` with default set to `TRUE`. This will automatically create the `tidy_empirical()` output for the `.x` parameter and use the `tidy_combine_distributions()`. The user can then plot out the data using `$combined_data_tbl` from the function output.

## Details

This function will attempt to estimate the weibull shape and scale parameters given some vector of values.

## Value

A tibble/list

## Author(s)

Steven P. Sanderson II, MPH

## See Also

Other Parameter Estimation: [util\\_bernoulli\\_param\\_estimate\(\)](#), [util\\_beta\\_param\\_estimate\(\)](#), [util\\_binomial\\_param\\_estimate\(\)](#), [util\\_burr\\_param\\_estimate\(\)](#), [util\\_cauchy\\_param\\_estimate\(\)](#), [util\\_exponential\\_param\\_estimate\(\)](#), [util\\_gamma\\_param\\_estimate\(\)](#), [util\\_geometric\\_param\\_estimate\(\)](#), [util\\_hypergeometric\\_param\\_estimate\(\)](#), [util\\_logistic\\_param\\_estimate\(\)](#), [util\\_lognormal\\_param\\_estimate\(\)](#), [util\\_negative\\_binomial\\_param\\_estimate\(\)](#), [util\\_normal\\_param\\_estimate\(\)](#), [util\\_pareto\\_param\\_estimate\(\)](#), [util\\_poisson\\_param\\_estimate\(\)](#), [util\\_triangular\\_param\\_estimate\(\)](#), [util\\_uniform\\_param\\_estimate\(\)](#)

Other Weibull: [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_weibull\(\)](#), [util\\_weibull\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)
library(ggplot2)

x <- tidy_weibull(.shape = 1, .scale = 2)$y
output <- util_weibull_param_estimate(x)

output$parameter_tbl

output$combined_data_tbl %>%
  tidy_combinedautoplot()
```

---

util\_weibull\_stats\_tbl

*Distribution Statistics*

---

**Description**

Returns distribution statistics in a tibble.

**Usage**

```
util_weibull_stats_tbl(.data)
```

**Arguments**

`.data`            The data being passed from a `tidy_` distribution function.

**Details**

This function will take in a tibble and returns the statistics of the given type of `tidy_` distribution. It is required that data be passed from a `tidy_` distribution function.

**Value**

A tibble

**Author(s)**

Steven P. Sanderson II, MPH

**See Also**

Other Weibull: [tidy\\_inverse\\_weibull\(\)](#), [tidy\\_weibull\(\)](#), [util\\_weibull\\_param\\_estimate\(\)](#)

Other Distribution Statistics: [util\\_bernoulli\\_stats\\_tbl\(\)](#), [util\\_beta\\_stats\\_tbl\(\)](#), [util\\_binomial\\_stats\\_tbl\(\)](#), [util\\_burr\\_stats\\_tbl\(\)](#), [util\\_cauchy\\_stats\\_tbl\(\)](#), [util\\_chisquare\\_stats\\_tbl\(\)](#), [util\\_exponential\\_stats\\_tbl\(\)](#), [util\\_f\\_stats\\_tbl\(\)](#), [util\\_gamma\\_stats\\_tbl\(\)](#), [util\\_geometric\\_stats\\_tbl\(\)](#), [util\\_hypergeometric\\_stats\\_tbl\(\)](#), [util\\_logistic\\_stats\\_tbl\(\)](#), [util\\_lognormal\\_stats\\_tbl\(\)](#), [util\\_negative\\_binomial\\_stats\\_tbl\(\)](#), [util\\_normal\\_stats\\_tbl\(\)](#), [util\\_pareto\\_stats\\_tbl\(\)](#), [util\\_poisson\\_stats\\_tbl\(\)](#), [util\\_t\\_stats\\_tbl\(\)](#), [util\\_triangular\\_stats\\_tbl\(\)](#), [util\\_uniform\\_stats\\_tbl\(\)](#)

**Examples**

```
library(dplyr)

tidy_weibull() |>
  util_weibull_stats_tbl() |>
  glimpse()
```

# Index

- \* **Augment Function**
  - bootstrap\_density\_augment, 4
  - bootstrap\_p\_augment, 5
  - bootstrap\_q\_augment, 7
- \* **Autoplot**
  - bootstrap\_stat\_plot, 9
  - tidy\_autoplot, 24
  - tidy\_combined\_autoplot, 35
  - tidy\_four\_autoplot, 46
  - tidy\_multi\_dist\_autoplot, 69
  - tidy\_random\_walk\_autoplot, 82
- \* **Bernoulli**
  - tidy\_bernoulli, 25
  - util\_bernoulli\_param\_estimate, 101
  - util\_bernoulli\_stats\_tbl, 102
- \* **Beta**
  - tidy\_beta, 27
  - tidy\_generalized\_beta, 49
  - util\_beta\_param\_estimate, 103
  - util\_beta\_stats\_tbl, 104
- \* **Binomial**
  - util\_negative\_binomial\_stats\_tbl, 130
- \* **Binomial**
  - tidy\_binomial, 28
  - tidy\_negative\_binomial, 72
  - tidy\_zero\_truncated\_binomial, 94
  - tidy\_zero\_truncated\_negative\_binomial, 97
  - util\_binomial\_param\_estimate, 105
  - util\_binomial\_stats\_tbl, 107
  - util\_negative\_binomial\_param\_estimate, 129
- \* **Bootstrap**
  - bootstrap\_density\_augment, 4
  - bootstrap\_p\_augment, 5
  - bootstrap\_p\_vec, 6
  - bootstrap\_q\_augment, 7
  - bootstrap\_q\_vec, 8
  - bootstrap\_stat\_plot, 9
  - bootstrap\_unnest\_tbl, 10
  - tidy\_bootstrap, 30
- \* **Burr**
  - tidy\_burr, 31
  - tidy\_inverse\_burr, 55
  - util\_burr\_param\_estimate, 108
  - util\_burr\_stats\_tbl, 109
- \* **Cauchy**
  - tidy\_cauchy, 32
  - util\_cauchy\_param\_estimate, 110
  - util\_cauchy\_stats\_tbl, 111
- \* **Chisquare**
  - tidy\_chisquare, 34
  - util\_chisquare\_stats\_tbl, 112
- \* **Continuous Distribution**
  - tidy\_beta, 27
  - tidy\_burr, 31
  - tidy\_cauchy, 32
  - tidy\_chisquare, 34
  - tidy\_exponential, 43
  - tidy\_f, 44
  - tidy\_gamma, 47
  - tidy\_generalized\_beta, 49
  - tidy\_generalized\_pareto, 50
  - tidy\_geometric, 52
  - tidy\_inverse\_burr, 55
  - tidy\_inverse\_exponential, 57
  - tidy\_inverse\_gamma, 58
  - tidy\_inverse\_normal, 60
  - tidy\_inverse\_pareto, 61
  - tidy\_inverse\_weibull, 63
  - tidy\_logistic, 65
  - tidy\_lognormal, 67
  - tidy\_normal, 74
  - tidy\_paralogistic, 75
  - tidy\_pareto, 77
  - tidy\_pareto1, 78
  - tidy\_t, 88

- tidy\_triangular, 90
- tidy\_uniform, 91
- tidy\_weibull, 92
- tidy\_zero\_truncated\_geometric, 95
- \* **Discrete Distribution**
  - tidy\_bernoulli, 25
  - tidy\_binomial, 28
  - tidy\_hypergeometric, 53
  - tidy\_negative\_binomial, 72
  - tidy\_poisson, 80
  - tidy\_zero\_truncated\_binomial, 94
  - tidy\_zero\_truncated\_negative\_binomial, 97
  - tidy\_zero\_truncated\_poisson, 98
- \* **Distribution Statistics**
  - util\_bernoulli\_stats\_tbl, 102
  - util\_beta\_stats\_tbl, 104
  - util\_binomial\_stats\_tbl, 107
  - util\_burr\_stats\_tbl, 109
  - util\_cauchy\_stats\_tbl, 111
  - util\_chisquare\_stats\_tbl, 112
  - util\_exponential\_stats\_tbl, 115
  - util\_f\_stats\_tbl, 116
  - util\_gamma\_stats\_tbl, 118
  - util\_geometric\_stats\_tbl, 120
  - util\_hypergeometric\_stats\_tbl, 123
  - util\_logistic\_stats\_tbl, 125
  - util\_lognormal\_stats\_tbl, 128
  - util\_negative\_binomial\_stats\_tbl, 130
  - util\_normal\_stats\_tbl, 133
  - util\_pareto\_stats\_tbl, 135
  - util\_poisson\_stats\_tbl, 138
  - util\_t\_stats\_tbl, 141
  - util\_triangular\_stats\_tbl, 140
  - util\_uniform\_stats\_tbl, 144
  - util\_weibull\_stats\_tbl, 146
- \* **Empirical**
  - tidy\_distribution\_comparison, 38
- \* **Exponential**
  - tidy\_exponential, 43
  - tidy\_inverse\_exponential, 57
  - util\_exponential\_param\_estimate, 113
  - util\_exponential\_stats\_tbl, 115
- \* **F Distribution**
  - tidy\_f, 44
  - util\_f\_stats\_tbl, 116
- \* **Gamma**
  - tidy\_gamma, 47
  - tidy\_inverse\_gamma, 58
  - util\_gamma\_param\_estimate, 117
  - util\_gamma\_stats\_tbl, 118
- \* **Gaussian**
  - tidy\_inverse\_normal, 60
  - tidy\_normal, 74
  - util\_normal\_param\_estimate, 131
  - util\_normal\_stats\_tbl, 133
- \* **Geometric**
  - tidy\_geometric, 52
  - tidy\_zero\_truncated\_geometric, 95
  - util\_geometric\_param\_estimate, 119
  - util\_geometric\_stats\_tbl, 120
- \* **Helper**
  - dist\_type\_extractor, 22
- \* **Hypergeometric**
  - tidy\_hypergeometric, 53
  - util\_hypergeometric\_param\_estimate, 121
  - util\_hypergeometric\_stats\_tbl, 123
- \* **Inverse Distribution**
  - tidy\_inverse\_burr, 55
  - tidy\_inverse\_exponential, 57
  - tidy\_inverse\_gamma, 58
  - tidy\_inverse\_normal, 60
  - tidy\_inverse\_pareto, 61
  - tidy\_inverse\_weibull, 63
- \* **Logistic**
  - tidy\_logistic, 65
  - tidy\_paralogistic, 75
  - util\_logistic\_param\_estimate, 124
  - util\_logistic\_stats\_tbl, 125
- \* **Lognormal**
  - tidy\_lognormal, 67
  - util\_lognormal\_param\_estimate, 126
  - util\_lognormal\_stats\_tbl, 128
- \* **Mixture Data**
  - tidy\_mixture\_density, 68
- \* **Multiple Distribution**
  - tidy\_combine\_distributions, 37
  - tidy\_multi\_single\_dist, 71
- \* **Negative Binomial**
  - util\_negative\_binomial\_stats\_tbl, 130
- \* **Negative Distribution**
  - tidy\_negative\_binomial, 72

- \* **Parameter Estimation**
    - util\_bernoulli\_param\_estimate, 101
    - util\_beta\_param\_estimate, 103
    - util\_binomial\_param\_estimate, 105
    - util\_burr\_param\_estimate, 108
    - util\_cauchy\_param\_estimate, 110
    - util\_exponential\_param\_estimate, 113
    - util\_gamma\_param\_estimate, 117
    - util\_geometric\_param\_estimate, 119
    - util\_hypergeometric\_param\_estimate, 121
    - util\_logistic\_param\_estimate, 124
    - util\_lognormal\_param\_estimate, 126
    - util\_negative\_binomial\_param\_estimate, 129
    - util\_normal\_param\_estimate, 131
    - util\_pareto\_param\_estimate, 134
    - util\_poisson\_param\_estimate, 137
    - util\_triangular\_param\_estimate, 139
    - util\_uniform\_param\_estimate, 142
    - util\_weibull\_param\_estimate, 145
  - \* **Pareto**
    - tidy\_generalized\_pareto, 50
    - tidy\_inverse\_pareto, 61
    - tidy\_pareto, 77
    - tidy\_pareto1, 78
    - util\_pareto\_param\_estimate, 134
    - util\_pareto\_stats\_tbl, 135
  - \* **Poisson**
    - tidy\_poisson, 80
    - tidy\_zero\_truncated\_poisson, 98
    - util\_poisson\_param\_estimate, 137
    - util\_poisson\_stats\_tbl, 138
  - \* **Statistic**
    - ci\_hi, 13
    - ci\_lo, 14
    - tidy\_kurtosis\_vec, 64
    - tidy\_range\_statistic, 84
    - tidy\_skewness\_vec, 86
    - tidy\_stat\_tbl, 87
  - \* **Summary Statistics**
    - tidy\_distribution\_summary\_tbl, 40
  - \* **T Distribution**
    - tidy\_t, 88
    - util\_t\_stats\_tbl, 141
  - \* **Table Data**
    - tidy\_distribution\_summary\_tbl, 40
  - \* **Triangular**
    - tidy\_triangular, 90
    - util\_triangular\_param\_estimate, 139
    - util\_triangular\_stats\_tbl, 140
  - \* **Uniform**
    - tidy\_uniform, 91
    - util\_uniform\_param\_estimate, 142
    - util\_uniform\_stats\_tbl, 144
  - \* **Utility**
    - convert\_to\_ts, 18
  - \* **Vector Function**
    - bootstrap\_p\_vec, 6
    - bootstrap\_q\_vec, 8
    - cgmean, 11
    - chmean, 12
    - ckurtosis, 15
    - cmean, 16
    - cmedian, 17
    - csd, 19
    - cskewness, 20
    - cvar, 21
    - tidy\_kurtosis\_vec, 64
    - tidy\_scale\_zero\_one\_vec, 85
    - tidy\_skewness\_vec, 86
  - \* **Visualization**
    - triangle\_plot, 100
  - \* **Weibull**
    - tidy\_inverse\_weibull, 63
    - tidy\_weibull, 92
    - util\_weibull\_param\_estimate, 145
    - util\_weibull\_stats\_tbl, 146
  - \* **Zero Truncated Distribution**
    - tidy\_zero\_truncated\_binomial, 94
    - tidy\_zero\_truncated\_geometric, 95
    - tidy\_zero\_truncated\_poisson, 98
  - \* **Zero Truncated Negative Distribution**
    - tidy\_zero\_truncated\_negative\_binomial, 97
- actuar::rburr(), 32
- actuar::rgenpareto(), 51
- actuar::rinvburr(), 56
- actuar::rinvexp(), 57
- actuar::rinvgamma(), 59
- actuar::rinvpareto(), 62
- actuar::rinvweibull(), 64
- actuar::rparalogis(), 76

- actuar::rpareto(), 77
- actuar::rpareto1(), 79
- actuar::rztbinom(), 94
- actuar::rztgeom(), 96
- actuar::rztlnbinom(), 97
- actuar::rztpois(), 99
  
- bootstrap\_density\_augment, 4, 6–8, 10, 11, 30
- bootstrap\_p\_augment, 5, 5, 7, 8, 10, 11, 30
- bootstrap\_p\_vec, 5, 6, 6, 7, 8, 10–13, 15–17, 20, 21, 30, 65, 85, 86
- bootstrap\_q\_augment, 5–7, 7, 8, 10, 11, 30
- bootstrap\_q\_vec, 5–7, 8, 10–13, 15–17, 20, 21, 30, 65, 85, 86
- bootstrap\_stat\_plot, 5–8, 9, 11, 25, 30, 37, 47, 71, 83
- bootstrap\_unnest\_tbl, 5–8, 10, 10, 30
  
- cgmean, 7, 8, 11, 13, 15–17, 20, 21, 65, 85, 86
- chmean, 7, 8, 12, 12, 15–17, 20, 21, 65, 85, 86
- ci\_hi, 13, 14, 65, 84, 86, 88
- ci\_lo, 13, 14, 65, 84, 86, 88
- ckurtosis, 7, 8, 12, 13, 15, 16, 17, 20, 21, 65, 85, 86
- cmean, 7, 8, 12, 13, 15, 16, 17, 20, 21, 65, 85, 86
- cmedian, 7, 8, 12, 13, 15, 16, 17, 20, 21, 65, 85, 86
- color\_blind, 18
- convert\_to\_ts, 18
- csd, 7, 8, 12, 13, 15–17, 19, 21, 65, 85, 86
- cskewness, 7, 8, 12, 13, 15–17, 20, 20, 21, 65, 85, 86
- cvar, 7, 8, 12, 13, 15–17, 20, 21, 21, 65, 85, 86
  
- dist\_type\_extractor, 22
- dplyr::cummean(), 16
- dplyr::group\_by(), 40
- dplyr::select(), 40
  
- EnvStats::ebeta(), 103
- EnvStats::elnorm(), 127
  
- rinvgauss(), 61
- rlang::enquo(), 5, 7
  
- stats::density(), 26–28, 31, 33, 34, 43, 44, 47, 49, 51, 52, 54, 55, 57, 58, 60, 61, 63, 65, 67, 73–75, 77, 78, 80, 89, 91, 92, 94, 95, 97, 98
- stats::rbeta(), 27, 50
- stats::rbinom(), 29
- stats::rcauchy(), 33
- stats::rchisq(), 35
- stats::rexp(), 43
- stats::rf(), 45
- stats::rgamma(), 48
- stats::rgeom(), 53
- stats::rhyper(), 54
- stats::rlnorm(), 67
- stats::rlogis(), 66
- stats::rnbinom(), 73
- stats::rnorm(), 74
- stats::rpois(), 80
- stats::rt(), 89
- stats::runif(), 92
- stats::rweibull(), 93
  
- td\_scale\_color\_colorblind, 23
- td\_scale\_fill\_colorblind, 23
- tidyautoplot, 10, 24, 37, 47, 71, 83
- tidy\_bernoulli, 25, 29, 55, 73, 81, 95, 98, 99, 101, 103
- tidy\_beta, 27, 32, 33, 35, 44, 45, 48, 50, 52, 53, 56, 58, 59, 61, 62, 64, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 104, 105
- tidy\_binomial, 26, 28, 55, 73, 81, 95, 98, 99, 106, 107, 130
- tidy\_bootstrap, 5–8, 10, 11, 30
- tidy\_burr, 28, 31, 33, 35, 44, 45, 48, 50, 52, 53, 56, 58, 59, 61, 62, 64, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 109, 110
- tidy\_cauchy, 28, 32, 32, 35, 44, 45, 48, 50, 52, 53, 56, 58, 59, 61, 62, 64, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 111, 112
- tidy\_chisquare, 28, 32, 33, 34, 44, 45, 48, 50, 52, 53, 56, 58, 59, 61, 62, 64, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 113
- tidy\_combine\_distributions, 37, 72
- tidy\_combinedautoplot, 10, 25, 35, 47, 71, 83
- tidy\_distribution\_comparison, 38
- tidy\_distribution\_summary\_tbl, 40
- tidy\_empirical, 42
- tidy\_exponential, 28, 32, 33, 35, 43, 45, 48, 50, 52, 53, 56, 58, 59, 61, 62, 64, 66,

- 68, 75, 76, 78, 79, 89, 91–93, 96, 114, 115
- tidy\_f, 28, 32, 33, 35, 44, 44, 48, 50, 52, 53, 56, 58, 59, 61, 62, 64, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 116
- tidy\_four\_autoplot, 10, 25, 37, 46, 71, 83
- tidy\_gamma, 28, 32, 33, 35, 44, 45, 47, 50, 52, 53, 56, 58, 59, 61, 62, 64, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 117, 118
- tidy\_generalized\_beta, 28, 32, 33, 35, 44, 45, 48, 49, 52, 53, 56, 58, 59, 61, 62, 64, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 104, 105
- tidy\_generalized\_pareto, 28, 32, 33, 35, 44, 45, 48, 50, 50, 53, 56, 58, 59, 61, 62, 64, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 135, 136
- tidy\_geometric, 28, 32, 33, 35, 44, 45, 48, 50, 52, 52, 56, 58, 59, 61, 62, 64, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 120, 121
- tidy\_hypergeometric, 26, 29, 53, 73, 81, 95, 98, 99, 122, 124
- tidy\_inverse\_burr, 28, 32, 33, 35, 44, 45, 48, 50, 52, 53, 55, 58, 59, 61, 62, 64, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 109, 110
- tidy\_inverse\_exponential, 28, 32, 33, 35, 44, 45, 48, 50, 52, 53, 56, 57, 59, 61, 62, 64, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 114, 115
- tidy\_inverse\_gamma, 28, 32, 33, 35, 44, 45, 48, 50, 52, 53, 56, 58, 58, 61, 62, 64, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 117, 118
- tidy\_inverse\_normal, 28, 32, 33, 35, 44, 45, 48, 50, 52, 53, 56, 58, 59, 60, 62, 64, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 132, 134
- tidy\_inverse\_pareto, 28, 32, 33, 35, 44, 45, 48, 50, 52, 53, 56, 58, 59, 61, 61, 64, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 135, 136
- tidy\_inverse\_weibull, 28, 32, 33, 35, 44, 45, 48, 50, 52, 53, 56, 58, 59, 61, 62, 63, 66, 68, 75, 76, 78, 79, 89, 91–93, 96, 145, 147
- tidy\_kurtosis\_vec, 7, 8, 12–17, 20, 21, 64, 84–86, 88
- tidy\_logistic, 28, 32, 33, 35, 44, 45, 48, 50, 52, 53, 56, 58, 59, 61, 62, 64, 65, 68, 75, 76, 78, 79, 89, 91–93, 96, 125, 126
- tidy\_lognormal, 28, 32, 33, 35, 44, 45, 48, 50, 52, 53, 56, 58, 59, 61, 62, 64, 66, 67, 75, 76, 78, 79, 89, 91–93, 96, 127, 129
- tidy\_mixture\_density, 68
- tidy\_multi\_dist\_autoplot, 10, 25, 37, 47, 69, 83
- tidy\_multi\_single\_dist, 38, 71
- tidy\_negative\_binomial, 26, 29, 55, 72, 81, 95, 98, 99, 106, 107, 130
- tidy\_normal, 28, 32, 33, 35, 44, 45, 48, 50, 52, 53, 56, 58, 59, 61, 62, 64, 66, 68, 74, 76, 78, 79, 89, 91–93, 96, 132, 134
- tidy\_paralogistic, 28, 32, 34, 35, 44, 45, 48, 50, 52, 53, 56, 58, 59, 61, 62, 64, 66, 68, 75, 75, 78, 79, 89, 91–93, 96, 125, 126
- tidy\_pareto, 28, 32, 34, 35, 44, 45, 48, 50, 52, 53, 56, 58, 59, 61, 62, 64, 66, 68, 75, 76, 77, 79, 89, 91–93, 96, 135, 136
- tidy\_pareto1, 28, 32, 34, 35, 44, 45, 48, 50, 52, 53, 56, 58, 59, 61, 62, 64, 66, 68, 75, 76, 78, 78, 89, 91–93, 96, 135, 136
- tidy\_poisson, 26, 29, 55, 73, 80, 95, 98, 99, 137, 139
- tidy\_random\_walk, 81
- tidy\_random\_walk\_autoplot, 10, 25, 37, 47, 71, 82
- tidy\_range\_statistic, 13, 14, 65, 84, 86, 88
- tidy\_scale\_zero\_one\_vec, 7, 8, 12, 13, 15–17, 20, 21, 65, 85, 86
- tidy\_skewness\_vec, 7, 8, 12–17, 20, 21, 65, 84, 85, 86, 88
- tidy\_stat\_tbl, 13, 14, 65, 84, 86, 87
- tidy\_t, 28, 32, 34, 35, 44, 45, 48, 50, 52, 53, 56, 58, 59, 61, 62, 64, 66, 68, 75, 76, 78, 79, 88, 91–93, 96, 142
- tidy\_triangular, 28, 32, 34, 35, 44, 45, 48, 50, 52, 53, 56, 58, 59, 61, 62, 64, 66, 68, 75, 76, 78, 79, 89, 90, 92, 93, 96,



- 140, 141  
 tidy\_uniform, 28, 32, 34, 35, 44, 45, 48, 50,  
 52, 53, 56, 58, 59, 61, 62, 64, 66, 68,  
 75, 76, 78, 79, 89, 91, 91, 93, 96,  
 143, 144  
 tidy\_weibull, 28, 32, 34, 35, 44, 45, 48, 50,  
 52, 53, 56, 58, 59, 61, 62, 64, 66, 68,  
 75, 76, 78, 79, 89, 91, 92, 92, 96,  
 145, 147  
 tidy\_zero\_truncated\_binomial, 26, 29, 55,  
 73, 81, 94, 96, 98, 99, 106, 107, 130  
 tidy\_zero\_truncated\_geometric, 28, 32,  
 34, 35, 44, 45, 48, 50, 52, 53, 56, 58,  
 59, 61, 62, 64, 66, 68, 75, 76, 78, 79,  
 89, 91–93, 95, 95, 99, 120, 121  
 tidy\_zero\_truncated\_negative\_binomial,  
 26, 29, 55, 73, 81, 95, 97, 99, 106,  
 107, 130  
 tidy\_zero\_truncated\_poisson, 26, 29, 55,  
 73, 81, 95, 96, 98, 98, 137, 139  
 triangle\_plot, 100  
 util\_bernoulli\_param\_estimate, 26, 101,  
 103, 104, 106, 109, 111, 114, 117,  
 120, 122, 125, 127, 130, 132, 135,  
 137, 140, 143, 145  
 util\_bernoulli\_stats\_tbl, 26, 101, 102,  
 105, 107, 110, 112, 113, 115, 116,  
 118, 121, 124, 126, 129, 131, 134,  
 136, 139, 141, 142, 144, 147  
 util\_beta\_param\_estimate, 28, 50, 101,  
 103, 105, 106, 109, 111, 114, 117,  
 120, 122, 125, 127, 130, 132, 135,  
 137, 140, 143, 145  
 util\_beta\_stats\_tbl, 28, 50, 103, 104, 104,  
 107, 110, 112, 113, 115, 116, 118,  
 121, 124, 126, 129, 131, 134, 136,  
 139, 141, 142, 144, 147  
 util\_binomial\_param\_estimate, 29, 73, 95,  
 98, 101, 104, 105, 107, 109, 111,  
 114, 117, 120, 122, 125, 127, 130,  
 132, 135, 137, 140, 143, 145  
 util\_binomial\_stats\_tbl, 29, 73, 95, 98,  
 103, 105, 106, 107, 110, 112, 113,  
 115, 116, 118, 121, 124, 126,  
 129–131, 134, 136, 139, 141, 142,  
 144, 147  
 util\_burr\_param\_estimate, 32, 56, 101,  
 104, 106, 108, 110, 111, 114, 117,  
 120, 122, 125, 127, 130, 132, 135,  
 137, 140, 143, 145  
 util\_burr\_stats\_tbl, 32, 56, 103, 105, 107,  
 109, 109, 112, 113, 115, 116, 118,  
 121, 124, 126, 129, 131, 134, 136,  
 139, 141, 142, 144, 147  
 util\_cauchy\_param\_estimate, 34, 101, 104,  
 106, 109, 110, 112, 114, 117, 120,  
 122, 125, 127, 130, 132, 135, 137,  
 140, 143, 145  
 util\_cauchy\_stats\_tbl, 34, 103, 105, 107,  
 110, 111, 111, 113, 115, 116, 118,  
 121, 124, 126, 129, 131, 134, 136,  
 139, 141, 142, 144, 147  
 util\_chisquare\_stats\_tbl, 35, 103, 105,  
 107, 110, 112, 112, 115, 116, 118,  
 121, 124, 126, 129, 131, 134, 136,  
 139, 141, 142, 144, 147  
 util\_exponential\_param\_estimate, 44, 58,  
 101, 104, 106, 109, 111, 113, 115,  
 117, 120, 122, 125, 127, 130, 132,  
 135, 137, 140, 143, 145  
 util\_exponential\_stats\_tbl, 44, 58, 103,  
 105, 107, 110, 112–114, 115, 116,  
 118, 121, 124, 126, 129, 131, 134,  
 136, 139, 141, 142, 144, 147  
 util\_f\_stats\_tbl, 45, 103, 105, 107, 110,  
 112, 113, 115, 116, 118, 121, 124,  
 126, 129, 131, 134, 136, 139, 141,  
 142, 144, 147  
 util\_gamma\_param\_estimate, 48, 59, 101,  
 104, 106, 109, 111, 114, 117, 118,  
 120, 122, 125, 127, 130, 132, 135,  
 137, 140, 143, 145  
 util\_gamma\_stats\_tbl, 48, 59, 103, 105,  
 107, 110, 112, 113, 115–117, 118,  
 121, 124, 126, 129, 131, 134, 136,  
 139, 141, 142, 144, 147  
 util\_geometric\_param\_estimate, 53, 96,  
 101, 104, 106, 109, 111, 114, 117,  
 119, 121, 122, 125, 127, 130, 132,  
 135, 137, 140, 143, 145  
 util\_geometric\_stats\_tbl, 53, 96, 103,  
 105, 107, 110, 112, 113, 115, 116,  
 118, 120, 120, 124, 126, 129, 131,  
 134, 136, 139, 141, 142, 144, 147  
 util\_hypergeometric\_param\_estimate, 55,  
 101, 104, 106, 109, 111, 114, 117,

- 120, 121, 124, 125, 127, 130, 132, 135, 137, 140, 143, 145*  
*util\_hypergeometric\_stats\_tbl, 55, 103, 105, 107, 110, 112, 113, 115, 116, 118, 121, 122, 123, 126, 129, 131, 134, 136, 139, 141, 142, 144, 147*  
*util\_logistic\_param\_estimate, 66, 76, 101, 104, 106, 109, 111, 114, 117, 120, 122, 124, 126, 127, 130, 132, 135, 137, 140, 143, 145*  
*util\_logistic\_stats\_tbl, 66, 76, 103, 105, 107, 110, 112, 113, 115, 116, 118, 121, 124, 125, 125, 129, 131, 134, 136, 139, 141, 142, 144, 147*  
*util\_lognormal\_param\_estimate, 68, 101, 104, 106, 109, 111, 114, 117, 120, 122, 125, 126, 129, 130, 132, 135, 137, 140, 143, 145*  
*util\_lognormal\_stats\_tbl, 68, 103, 105, 107, 110, 112, 113, 115, 116, 118, 121, 124, 126, 127, 128, 131, 134, 136, 139, 141, 142, 144, 147*  
*util\_negative\_binomial\_param\_estimate, 29, 73, 95, 98, 101, 104, 106, 107, 109, 111, 114, 117, 120, 122, 125, 127, 129, 132, 135, 137, 140, 143, 145*  
*util\_negative\_binomial\_stats\_tbl, 103, 105, 107, 110, 112, 113, 115, 116, 118, 121, 124, 126, 129, 130, 134, 136, 139, 141, 142, 144, 147*  
*util\_normal\_param\_estimate, 61, 75, 101, 104, 106, 109, 111, 114, 117, 120, 122, 125, 127, 130, 131, 134, 135, 137, 140, 143, 145*  
*util\_normal\_stats\_tbl, 61, 75, 103, 105, 107, 110, 112, 113, 115, 116, 118, 121, 124, 126, 129, 131, 132, 133, 136, 139, 141, 142, 144, 147*  
*util\_pareto\_param\_estimate, 52, 62, 78, 79, 101, 104, 106, 109, 111, 114, 117, 120, 122, 125, 127, 130, 132, 134, 136, 137, 140, 143, 145*  
*util\_pareto\_stats\_tbl, 52, 62, 78, 79, 103, 105, 107, 110, 112, 113, 115, 116, 118, 121, 124, 126, 129, 131, 134, 135, 135, 139, 141, 142, 144, 147*  
*util\_poisson\_param\_estimate, 81, 99, 101, 104, 106, 109, 111, 114, 117, 120, 122, 125, 127, 130, 132, 135, 137, 139, 140, 143, 145*  
*util\_poisson\_stats\_tbl, 81, 99, 103, 105, 107, 110, 112, 113, 115, 116, 118, 121, 124, 126, 129, 131, 134, 136, 137, 138, 141, 142, 144, 147*  
*util\_t\_stats\_tbl, 89, 103, 105, 107, 110, 112, 113, 115, 116, 118, 121, 124, 126, 129, 131, 134, 136, 139, 141, 141, 144, 147*  
*util\_triangular\_param\_estimate, 91, 101, 104, 106, 109, 111, 114, 117, 120, 122, 125, 127, 130, 132, 135, 137, 139, 141, 143, 145*  
*util\_triangular\_stats\_tbl, 91, 103, 105, 107, 110, 112, 113, 115, 116, 118, 121, 124, 126, 129, 131, 134, 136, 139, 140, 140, 142, 144, 147*  
*util\_uniform\_param\_estimate, 92, 101, 104, 106, 109, 111, 114, 117, 120, 122, 125, 127, 130, 132, 135, 137, 140, 142, 144, 145*  
*util\_uniform\_stats\_tbl, 92, 103, 105, 107, 110, 112, 113, 115, 116, 118, 121, 124, 126, 129, 131, 134, 136, 139, 141–143, 144, 147*  
*util\_weibull\_param\_estimate, 64, 93, 101, 104, 106, 109, 111, 114, 117, 120, 122, 125, 127, 130, 132, 135, 137, 140, 143, 145, 147*  
*util\_weibull\_stats\_tbl, 64, 93, 103, 105, 107, 110, 112, 113, 115, 116, 118, 121, 124, 126, 129, 131, 134, 136, 139, 141, 142, 144, 145, 146*