Package ‘ssdtools’

October 21, 2024

Title Species Sensitivity Distributions

Version 2.1.0

Description Species sensitivity distributions are cumulative probability distributions which are fitted to toxicity concentrations for different species as described by Posthuma et al.(2001) <isbn:9781566705783>. The ssdtools package uses Maximum Likelihood to fit distributions such as the gamma, log-logistic, log-normal and log-normal log-normal mixture. Multiple distributions can be averaged using Akaike Information Criteria. Confidence intervals on hazard concentrations and proportions are produced by parametric bootstrapping.

License Apache License (== 2.0) | file LICENSE

URL `https://github.com/bcgov/ssdtools`, `https://bcgov.github.io/ssdtools/`

BugReports `https://github.com/bcgov/ssdtools/issues`

Depends R (>= 4.1)

Imports abind, chk, furrr, generics, ggplot2, glue, goftest, graphics, grid, lifecycle, parallel, plyr, purrr, Rcpp, rlang, scales, ssddata, stats, stringr, tibble, TMB, universals, utils, withr

Suggests actuar, covr, doFuture, dplyr, EnvStats, extraDistr, fitdistrplus, foreach, future, grDevices, knitr, latex2exp, magrittr, mle.tools, patchwork, R.rsp, readr, reshape2, rmarkdown, testthat (>= 3.0.0), tidyr, tidyselect, tinytex, VGAM

LinkingTo Rcpp, RcppEigen, TMB

VignetteBuilder knitr, R.rsp

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.2

NeedsCompilation yes

Author Joe Thorley [aut, cre] (<https://orcid.org/0000-0002-7683-4592>),
Rebecca Fisher [aut],
David Fox [aut],
Carl Schwarz [aut],
Angeline Tillmanns [ctb],
Seb Dalgarno [ctb] (<https://orcid.org/0000-0002-3658-4517>),
Kathleen McTavish [ctb],
Heather Thompson [ctb],
Doug Spry [ctb],
Rick van Dam [ctb],
Graham Batley [ctb],
Tony Bigwood [ctb],
Ali Azizishirazi [ctb],
Nadine Hussein [ctb] (<https://orcid.org/0000-0003-4470-8361>),
Sarah Lyons [ctb] (<https://orcid.org/0000-0002-6745-6796>),
Stephanie Hazlitt [ctb],
Hadley Wickham [ctb],
Sergio Ibarra Espinosa [ctb],
Andy Teucher [ctb],
Emilie Doussantousse [ctb],
Nan-Hung Hsieh [ctb],
Province of British Columbia [fnd, cph],
Environment and Climate Change Canada [fnd, cph],
Australian Government Department of Climate Change, Energy, the Environment and Water [fnd, cph]

Maintainer Joe Thorley <`joe@poissonconsulting.ca`>

Repository CRAN

Date/Publication 2024-10-21 13:30:08 UTC

Contents

`augment.fitdists` *Augmented Data from fitdists Object*

Description

Get a tibble of the original data with augmentation.

Usage

```
## S3 method for class 'fitdists'
augment(x, ...)
```

Arguments

`x`	The object.
`...`	Unused.

Value

A tibble of the agumented data.

See Also

`ssd_data()`

Other generics: `glance.fitdists()`, `tidy.fitdists()`

Examples

```
fits <- ssd_fit_dists(ssddata::ccme_boron)
augment(fits)
```

`autoplot.fitdists` *Plot a fitdists Object*

Description

A wrapper on `ssd_plot_cdf()`.

Usage

```
## S3 method for class 'fitdists'
autoplot(object, ...)
```

Arguments

`object`	The object.
`...`	Unused.

Value

A ggplot object.

See Also

`ssd_plot_cdf()`

Examples

```
fits <- ssd_fit_dists(ssddata::ccme_boron)
autoplot(fits)
```

`boron_pred` *Model Averaged Predictions for CCME Boron Data*

Description

A data frame of the predictions based on 1,000 bootstrap iterations.

Usage

```
boron_pred
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 99 rows and 11 columns.

Details

proportion The proportion of species affected (int).

est The estimated concentration (dbl).

se The standard error of the estimate (dbl).

lcl The lower confidence limit (dbl).

se The upper confidence limit (dbl).

dist The distribution (chr).

Examples

```
## Not run:
 fits <- ssd_fit_dists(ssddata::ccme_boron)
 set.seed(99)
 boron_pred <- predict(fits, ci = TRUE)

## End(Not run)
head(boron_pred)
```

`coef.fitdists` *Turn a fitdists Object into a Tidy Tibble*

Description

A wrapper on `tidy.fitdists()`.

Usage

```
## S3 method for class 'fitdists'
coef(object, ...)
```

Arguments

`object`	The object.
`...`	Unused.

See Also

`tidy.fitdists()`

Examples

```
fits <- ssd_fit_dists(ssddata::ccme_boron)
coef(fits)
```

`comma_signif` *Comma and Significance Formatter*

Description

Comma and Significance Formatter

Usage

```
comma_signif(x, digits = 3, ..., big.mark = ",")
```

Arguments

`x`	A numeric vector to format.
`digits`	A whole number specifying the number of significant figures.
`...`	Unused.
`big.mark`	A string specifying used between every 3 digits to separate thousands on the x-axis.

Value

A character vector.

See Also

`ssd_label_comma()`

Examples

```
## Not run:
comma_signif(c(0.1, 1, 10, 1000, 10000))

## End(Not run)
```

`dgompertz` *Gompertz Probability Density* **[Deprecated]**

Description

Gompertz Probability Density **[Deprecated]**

Usage

```
dgompertz(x, llocation = 0, lshape = 0, log = FALSE)
```

Arguments

`x`	A numeric vector of values.
`llocation`	location parameter on the log scale.
`lshape`	shape parameter on the log scale.
`log`	logical; if TRUE, probabilities p are given as log(p).

Value

A numeric vector.

`dist_data` *Distribution Data*

Description

A data frame of information on the implemented distributions.

Usage

```
dist_data
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 10 rows and 4 columns.

Details

dist The distribution (chr).

npars The number of parameters (int).

tails Whether the distribution has both tails (flag).

stable Whether the distribution is numerically stable (flag).

bcanz Whether the distribution belongs to the set of distributions approved by BC, Canada, Australia and New Zealand for official guidelines (flag).

See Also

Other dists: `ssd_dists()`, `ssd_dists_all()`

Examples

```
dist_data
```

`dlgumbel` *Log-Gumbel (Inverse Weibull) Probability Density* **[Deprecated]**

Description

Log-Gumbel (Inverse Weibull) Probability Density **[Deprecated]**

Usage

```
dlgumbel(x, locationlog = 0, scalelog = 1, log = FALSE)
```

Arguments

`x`	A numeric vector of values.
`locationlog`	location on the log scale parameter.
`scalelog`	scale on log scale parameter.
`log`	logical; if TRUE, probabilities p are given as log(p).

Value

A numeric vector.

`estimates.fitdists` *Estimates for fitdists Object*

Description

Gets a named list of the estimated weights and parameters.

Usage

```
## S3 method for class 'fitdists'
estimates(x, all_estimates = FALSE, ...)
```

Arguments

`x`	The object.
`all_estimates`	A flag specifying whether to calculate estimates for all implemented distributions.
`...`	Unused.

Value

A named list of the estimates.

See Also

`tidy.fitdists()`, `ssd_match_moments()`, `ssd_hc()` and `ssd_plot_cdf()`

Examples

```
fits <- ssd_fit_dists(ssddata::ccme_boron)
estimates(fits)
```

`geom_hcintersect` *Species Sensitivity Hazard Concentration Intersection*

Description

Plots the intersection between each `xintercept` and `yintercept` value.

Usage

```
geom_hcintersect(
  mapping = NULL,
  data = NULL,
  ...,
  xintercept,
  yintercept,
  na.rm = FALSE,
  show.legend = NA
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`...`	Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`xintercept`	The x-value for the intersect
`yintercept`	The y-value for the intersect.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

See Also

`ssd_plot_cdf()`

Other ggplot: `geom_ssdpoint()`, `geom_ssdsegment()`, `geom_xribbon()`, `scale_colour_ssd()`, `ssd_pal()`

Examples

```
ggplot2::ggplot(ssddata::ccme_boron, ggplot2::aes(x = Conc)) +
  geom_ssdpoint() +
  geom_hcintersect(xintercept = 1.5, yintercept = 0.05)
```

---

geom_ssd *Species Sensitivity Data Points* **[Deprecated]**

Description

`geom_ssd()` has been deprecated for `geom_ssdpoint()`.

Usage

```
geom_ssd(
  mapping = NULL,
  data = NULL,
  stat = "ssdpoint",
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`stat`	The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used the override the default coupling between geoms and stats. The `stat` argument accepts the following: • A `Stat` ggproto subclass, for example `StatCount`. • A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`. • For more information and other ways to specify the stat, see the layer stat documentation.
`position`	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following: • The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`. • For more information and other ways to specify the position, see the layer position documentation.

`...`	Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept. • The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

Examples

```
## Not run:
ggplot2::ggplot(ssddata::ccme_boron, ggplot2::aes(x = Conc)) +
  geom_ssd()

## End(Not run)
```

`geom_ssdpoint` *Species Sensitivity Data Points*

Description

Uses the empirical cumulative distribution to create scatterplot of points `x`.

Usage

```
geom_ssdpoint(
  mapping = NULL,
  data = NULL,
  stat = "ssdpoint",
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`stat`	The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used the override the default coupling between geoms and stats. The `stat` argument accepts the following: • A `Stat` ggproto subclass, for example `StatCount`. • A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`. • For more information and other ways to specify the stat, see the layer stat documentation.
`position`	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following: • The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`. • For more information and other ways to specify the position, see the layer position documentation.

`...`	Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept. • The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

See Also

`ssd_plot_cdf()`

Other ggplot: `geom_hcintersect()`, `geom_ssdsegment()`, `geom_xribbon()`, `scale_colour_ssd()`, `ssd_pal()`

Examples

```
ggplot2::ggplot(ssddata::ccme_boron, ggplot2::aes(x = Conc)) +
  geom_ssdpoint()
```

`geom_ssdsegment` *Species Sensitivity Censored Segments*

Description

Uses the empirical cumulative distribution to draw lines between points `x` and `xend`.

Usage

```
geom_ssdsegment(
  mapping = NULL,
  data = NULL,
  stat = "ssdsegment",
  position = "identity",
  ...,
  arrow = NULL,
  arrow.fill = NULL,
  lineend = "butt",
  linejoin = "round",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`stat`	The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used the override the default coupling between geoms and stats. The `stat` argument accepts the following: • A `Stat` ggproto subclass, for example `StatCount`. • A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.

	• For more information and other ways to specify the stat, see the layer stat documentation.
`position`	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following: • The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`. • For more information and other ways to specify the position, see the layer position documentation.
`...`	Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept. • The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
`arrow`	specification for arrow heads, as created by `grid::arrow()`.
`arrow.fill`	fill colour to use for the arrow head (if closed). `NULL` means use `colour` aesthetic.
`lineend`	Line end style (round, butt, square).
`linejoin`	Line join style (round, mitre, bevel).
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

See Also

`ssd_plot_cdf()`

Other ggplot: `geom_hcintersect()`, `geom_ssdpoint()`, `geom_xribbon()`, `scale_colour_ssd()`, `ssd_pal()`

Examples

```
ggplot2::ggplot(ssddata::ccme_boron, ggplot2::aes(x = Conc, xend = Conc * 2)) +
  geom_ssdsegment()
```

`geom_xribbon` *Ribbon on X-Axis*

Description

Plots the `x` interval defined by `xmin` and `xmax`.

Usage

```
geom_xribbon(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).

`stat`

The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used the override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position`

A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...`

Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

See Also

`ssd_plot_cdf()`

Other ggplot: `geom_hcintersect()`, `geom_ssdpoint()`, `geom_ssdsegment()`, `scale_colour_ssd()`, `ssd_pal()`

Examples

```
gp <- ggplot2::ggplot(boron_pred) +
  geom_xribbon(ggplot2::aes(xmin = lcl, xmax = ucl, y = proportion))
```

`glance.fitdists` *Get a tibble summarizing each distribution*

Description

Gets a tibble with a single row for each distribution.

Usage

```
## S3 method for class 'fitdists'
glance(x, ...)
```

Arguments

`x`	The object.
`...`	Unused.

Value

A tidy tibble of the distributions.

See Also

`ssd_gof()`

Other generics: `augment.fitdists()`, `tidy.fitdists()`

Examples

```
fits <- ssd_fit_dists(ssddata::ccme_boron)
glance(fits)
```

`is.fitdists` *Is fitdists Object*

Description

Tests whether x is a fitdists Object.

Usage

```
is.fitdists(x)
```

Arguments

`x`	The object.

Value

A flag specifying whether x is a fitdists Object.

Examples

```
fits <- ssd_fit_dists(ssddata::ccme_boron)
is.fitdists(fits)
```

`is_censored` *Is Censored* **[Deprecated]**

Description

Deprecated for `ssd_is_censored()`.

Usage

```
is_censored(x)
```

Arguments

`x`	A fitdists object.

Value

A flag indicating if the data is censored.

See Also

`ssd_is_censored()`

`params` *Parameter Descriptions for ssdtools Functions*

Description

Parameter Descriptions for ssdtools Functions

Arguments

`...`	Unused.
`add_x`	The value to add to the label x values (before multiplying by `shift_x`).
`all`	A flag specifying whether to also return transformed parameters.
`all_dists`	A flag specifying whether all the named distributions must fit successfully.
`at_boundary_ok`	A flag specifying whether a model with one or more parameters at the boundary should be considered to have converged (default = FALSE).
`average`	A flag specifying whether to provide model averaged values as opposed to a value for each distribution.
`bcanz`	A flag or NULL specifying whether to only include distributions in the set that is approved by BC, Canada, Australia and New Zealand for official guidelines.
`big.mark`	A string specifying used between every 3 digits to separate thousands on the x-axis.
`breaks`	A character vector
`bounds`	A named non-negative numeric vector of the left and right bounds for uncensored missing (0 and Inf) data in terms of the orders of magnitude relative to the extremes for non-missing values.
`chk`	A flag specifying whether to check the arguments.
`ci`	A flag specifying whether to estimate confidence intervals (by bootstrapping).
`censoring`	A numeric vector of the left and right censoring values.
`color`	A string of the column in data for the color aesthetic.
`computable`	A flag specifying whether to only return fits with numerically computable standard errors.
`conc`	A numeric vector of concentrations to calculate the hazard proportions for.
`control`	A list of control parameters passed to `stats::optim()`.
`data`	A data frame.
`delta`	A non-negative number specifying the maximum absolute AIC difference cutoff. Distributions with an absolute AIC difference greater than delta are excluded from the calculations.

`digits`	A whole number specifying the number of significant figures.
`dists`	A character vector of the distribution names.
`fitdists`	An object of class fitdists.
`hc`	A value between 0 and 1 indicating the proportion hazard concentration (or NULL).
`label`	A string of the column in data with the labels.
`left`	A string of the column in data with the concentrations.
`level`	A number between 0 and 1 of the confidence level of the interval.
`linecolor`	A string of the column in pred to use for the line color.
`linetype`	A string of the column in pred to use for the linetype.
`llocation`	location parameter on the log scale.
`location`	location parameter.
`locationlog`	location on the log scale parameter.
`locationlog1`	locationlog1 parameter.
`locationlog2`	locationlog2 parameter.
`log`	logical; if TRUE, probabilities p are given as log(p).
`log.p`	logical; if TRUE, probabilities p are given as log(p).
`lscale`	scale parameter on the log scale.
`lshape`	shape parameter on the log scale.
`lshape1`	shape1 parameter on the log scale.
`lshape2`	shape2 parameter on the log scale.
`lower.tail`	logical; if TRUE (default), probabilities are `P[X <= x]`, otherwise, `P[X > x]`.
`meanlog`	mean on log scale parameter.
`meanlog1`	mean on log scale parameter.
`meanlog2`	mean on log scale parameter.
`min_pboot`	A number between 0 and 1 of the minimum proportion of bootstrap samples that must successfully fit (return a likelihood) to report the confidence intervals.
`min_pmix`	A number between 0 and 0.5 specifying the minimum proportion in mixture models.
`npars`	A whole numeric vector specifying which distributions to include based on the number of parameters.
`all_estimates`	A flag specifying whether to calculate estimates for all implemented distributions.
`ci_method`	A string specifying which method to use for estimating the bootstrap values. Possible values are "multi_free" and "multi_fixed" which treat the distributions as constituting a single distribution but differ in whether the model weights are fixed and "weighted_samples" and "weighted_arithmetic" take bootstrap samples from each distribution proportional to its weight versus calculating the weighted arithmetic means of the lower and upper confidence limits.

`multi_est`	A flag specifying whether to treat the distributions as constituting a single distribution (as opposed to taking the mean) when calculating model averaged estimates.
`na.rm`	A flag specifying whether to silently remove missing values or remove them with a warning.
`n`	positive number of observations.
`nboot`	A count of the number of bootstrap samples to use to estimate the confidence limits. A value of 10,000 is recommended for official guidelines.
`nrow`	A positive whole number of the minimum number of non-missing rows.
`nsim`	A positive whole number of the number of simulations to generate.
`object`	The object.
`parametric`	A flag specifying whether to perform parametric bootstrapping as opposed to non-parametrically resampling the original data with replacement.
`p`	vector of probabilities.
`percent`	A numeric vector of percent values to estimate hazard concentrations for. Soft-deprecated for `proportion = 0.05`.
`pmix`	Proportion mixture parameter.
`proportion`	A numeric vector of proportion values to estimate hazard concentrations for.
`pvalue`	A flag specifying whether to return p-values or the statistics (default) for the various tests.
`pred`	A data frame of the predictions.
`q`	vector of quantiles.
`range_shape1`	A numeric vector of length two of the lower and upper bounds for the shape1 parameter.
`range_shape2`	shape2 parameter.
`reweight`	A flag specifying whether to reweight weights by dividing by the largest weight.
`rescale`	A flag specifying whether to rescale concentration values by dividing by the geometric mean of the minimum and maximum positive finite values.
`ribbon`	A flag indicating whether to plot the confidence interval as a grey ribbon as opposed to green solid lines.
`right`	A string of the column in data with the right concentration values.
`save_to`	NULL or a string specifying a directory to save where the bootstrap datasets and parameter estimates (when successfully converged) to.
`samples`	A flag specfying whether to include a numeric vector of the bootstrap samples as a list column in the output.
`scale`	scale parameter.
`scalelog1`	scalelog1 parameter.
`scalelog2`	scalelog2 parameter.
`scalelog`	scale on log scale parameter.
`sdlog`	standard deviation on log scale parameter.

`sdlog1`	standard deviation on log scale parameter.
`sdlog2`	standard deviation on log scale parameter.
`select`	A character vector of the distributions to select.
`shape`	shape parameter.
`shape1`	shape1 parameter.
`shape2`	shape2 parameter.
`shift_x`	The value to multiply the label x values by (after adding `add_x`).
`silent`	A flag indicating whether fits should fail silently.
`size`	A number for the size of the labels.
`suffix`	Additional text to display after the number on the y-axis.
`tails`	A flag or NULL specifying whether to only include distributions with both tails.
`trans`	A string which transformation to use by default `"log10"`.
`weight`	A string of the numeric column in data with positive weights less than or equal to 1,000 or NULL.
`x`	The object.
`xbreaks`	The x-axis breaks as one of: • `NULL` for no breaks • `waiver()` for the default breaks • A numeric vector of positions
`xintercept`	The x-value for the intersect
`xlab`	A string of the x-axis label.
`yintercept`	The y-value for the intersect.
`ylab`	A string of the x-axis label.
`burrIII3.weight`	weight parameter for the Burr III distribution.
`burrIII3.shape1`	shape1 parameter for the Burr III distribution.
`burrIII3.shape2`	shape2 parameter for the Burr III distribution.
`burrIII3.scale`	scale parameter for the Burr III distribution.
`gamma.weight`	weight parameter for the gamma distribution.
`gamma.shape`	shape parameter for the gamma distribution.
`gamma.scale`	scale parameter for the gamma distribution.
`gompertz.weight`	weight parameter for the Gompertz distribution.
`gompertz.location`	location parameter for the Gompertz distribution.
`gompertz.shape`	shape parameter for the Gompertz distribution.
`invpareto.weight`	weight parameter for the inverse Pareto distribution.

`invpareto.shape`	shape parameter for the inverse Pareto distribution.
`invpareto.scale`	scale parameter for the inverse Pareto distribution.
`lgumbel.weight`	weight parameter for the log-Gumbel distribution.
`lgumbel.locationlog`	location parameter for the log-Gumbel distribution.
`lgumbel.scalelog`	scale parameter for the log-Gumbel distribution.
`llogis.weight`	weight parameter for the log-logistic distribution.
`llogis.locationlog`	location parameter for the log-logistic distribution.
`llogis.scalelog`	scale parameter for the log-logistic distribution.
`llogis_llogis.weight`	weight parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.locationlog1`	locationlog1 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.scalelog1`	scalelog1 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.locationlog2`	locationlog2 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.scalelog2`	scalelog2 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.pmix`	pmix parameter for the log-logistic log-logistic mixture distribution.
`lnorm.weight`	weight parameter for the log-normal distribution.
`lnorm.meanlog`	meanlog parameter for the log-normal distribution.
`lnorm.sdlog`	sdlog parameter for the log-normal distribution.
`lnorm_lnorm.weight`	weight parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.meanlog1`	meanlog1 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.sdlog1`	sdlog1 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.meanlog2`	meanlog2 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.sdlog2`	sdlog2 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.pmix`	pmix parameter for the log-normal log-normal mixture distribution.
`weibull.weight`	weight parameter for the Weibull distribution.
`weibull.shape`	shape parameter for the Weibull distribution.
`weibull.scale`	scale parameter for the Weibull distribution.

`pgompertz` *Cumulative Distribution Function for Gompertz Distribution* **[Deprecated]**

Description

Cumulative Distribution Function for Gompertz Distribution **[Deprecated]**

Usage

```
pgompertz(q, llocation = 0, lshape = 0, lower.tail = TRUE, log.p = FALSE)
```

Arguments

`q`	vector of quantiles.
`llocation`	location parameter on the log scale.
`lshape`	shape parameter on the log scale.
`lower.tail`	logical; if TRUE (default), probabilities are `P[X <= x]`, otherwise, `P[X > x]`.
`log.p`	logical; if TRUE, probabilities p are given as log(p).

`plgumbel` *Cumulative Distribution Function for Log-Gumbel Distribution* **[Deprecated]**

Description

Cumulative Distribution Function for Log-Gumbel Distribution **[Deprecated]**

Usage

```
plgumbel(q, locationlog = 0, scalelog = 1, lower.tail = TRUE, log.p = FALSE)
```

Arguments

`q`	vector of quantiles.
`locationlog`	location on the log scale parameter.
`scalelog`	scale on log scale parameter.
`lower.tail`	logical; if TRUE (default), probabilities are `P[X <= x]`, otherwise, `P[X > x]`.
`log.p`	logical; if TRUE, probabilities p are given as log(p).

`predict.fitburrlioz` *Predict Hazard Concentrations of fitburrlioz Object*

Description

A wrapper on `ssd_hc()` that by default calculates all hazard concentrations from 1 to 99%.

Usage

```
## S3 method for class 'fitburrlioz'
predict(
  object,
  percent,
  proportion = 1:99/100,
  ci = FALSE,
  level = 0.95,
  nboot = 1000,
  min_pboot = 0.95,
  parametric = TRUE,
  ...
)
```

Arguments

`object`	The object.
`percent`	A numeric vector of percent values to estimate hazard concentrations for. Soft-deprecated for `proportion = 0.05`.
`proportion`	A numeric vector of proportion values to estimate hazard concentrations for.
`ci`	A flag specifying whether to estimate confidence intervals (by bootstrapping).
`level`	A number between 0 and 1 of the confidence level of the interval.
`nboot`	A count of the number of bootstrap samples to use to estimate the confidence limits. A value of 10,000 is recommended for official guidelines.
`min_pboot`	A number between 0 and 1 of the minimum proportion of bootstrap samples that must successfully fit (return a likelihood) to report the confidence intervals.
`parametric`	A flag specifying whether to perform parametric bootstrapping as opposed to non-parametrically resampling the original data with replacement.
`...`	Unused.

Details

It is useful for plotting purposes.

See Also

`ssd_hc()` and `ssd_plot()`

Examples

```
fits <- ssd_fit_burrlioz(ssddata::ccme_boron)
predict(fits)
```

`predict.fitdists` *Predict Hazard Concentrations of fitdists Object*

Description

A wrapper on `ssd_hc()` that by default calculates all hazard concentrations from 1 to 99%.

Usage

```
## S3 method for class 'fitdists'
predict(
  object,
  percent,
  proportion = 1:99/100,
  average = TRUE,
  ci = FALSE,
  level = 0.95,
  nboot = 1000,
  min_pboot = 0.95,
  multi_est = TRUE,
  ci_method = "weighted_samples",
  parametric = TRUE,
  delta = 9.21,
  control = NULL,
  ...
)
```

Arguments

`object`	The object.
`percent`	A numeric vector of percent values to estimate hazard concentrations for. Soft-deprecated for `proportion = 0.05`.
`proportion`	A numeric vector of proportion values to estimate hazard concentrations for.
`average`	A flag specifying whether to provide model averaged values as opposed to a value for each distribution.
`ci`	A flag specifying whether to estimate confidence intervals (by bootstrapping).
`level`	A number between 0 and 1 of the confidence level of the interval.
`nboot`	A count of the number of bootstrap samples to use to estimate the confidence limits. A value of 10,000 is recommended for official guidelines.
`min_pboot`	A number between 0 and 1 of the minimum proportion of bootstrap samples that must successfully fit (return a likelihood) to report the confidence intervals.

`multi_est`	A flag specifying whether to treat the distributions as constituting a single distribution (as opposed to taking the mean) when calculating model averaged estimates.
`ci_method`	A string specifying which method to use for estimating the bootstrap values. Possible values are "multi_free" and "multi_fixed" which treat the distributions as constituting a single distribution but differ in whether the model weights are fixed and "weighted_samples" and "weighted_arithmetic" take bootstrap samples from each distribution proportional to its weight versus calculating the weighted arithmetic means of the lower and upper confidence limits.
`parametric`	A flag specifying whether to perform parametric bootstrapping as opposed to non-parametrically resampling the original data with replacement.
`delta`	A non-negative number specifying the maximum absolute AIC difference cutoff. Distributions with an absolute AIC difference greater than delta are excluded from the calculations.
`control`	A list of control parameters passed to `stats::optim()`.
`...`	Unused.

Details

It is useful for plotting purposes.

See Also

`ssd_hc()` and `ssd_plot()`

Examples

```
fits <- ssd_fit_dists(ssddata::ccme_boron)
predict(fits)
```

`qgompertz` *Quantile Function for Gompertz Distribution* **[Deprecated]**

Description

Quantile Function for Gompertz Distribution **[Deprecated]**

Usage

```
qgompertz(p, llocation = 0, lshape = 0, lower.tail = TRUE, log.p = FALSE)
```

Arguments

`p`	vector of probabilities.
`llocation`	location parameter on the log scale.
`lshape`	shape parameter on the log scale.
`lower.tail`	logical; if TRUE (default), probabilities are `P[X <= x]`, otherwise, `P[X > x]`.
`log.p`	logical; if TRUE, probabilities p are given as log(p).

`qlgumbel` *Quantile Function for Log-Gumbel Distribution* **[Deprecated]**

Description

Quantile Function for Log-Gumbel Distribution **[Deprecated]**

Usage

```
qlgumbel(p, locationlog = 0, scalelog = 1, lower.tail = TRUE, log.p = FALSE)
```

Arguments

`p`	vector of probabilities.
`locationlog`	location on the log scale parameter.
`scalelog`	scale on log scale parameter.
`lower.tail`	logical; if TRUE (default), probabilities are `P[X <= x]`, otherwise, `P[X > x]`.
`log.p`	logical; if TRUE, probabilities p are given as log(p).

`rgompertz` *Random Generation for Gompertz Distribution* **[Deprecated]**

Description

Random Generation for Gompertz Distribution **[Deprecated]**

Usage

```
rgompertz(n, llocation = 0, lshape = 0)
```

Arguments

`n`	positive number of observations.
`llocation`	location parameter on the log scale.
`lshape`	shape parameter on the log scale.

`rlgumbel` *Random Generation for log-Gumbel Distribution* **[Deprecated]**

Description

Random Generation for log-Gumbel Distribution **[Deprecated]**

Usage

```
rlgumbel(n, locationlog = 0, scalelog = 1)
```

Arguments

`n`	positive number of observations.
`locationlog`	location on the log scale parameter.
`scalelog`	scale on log scale parameter.

`scale_colour_ssd` *Discrete color-blind scale for SSD Plots*

Description

Discrete color-blind scale for SSD Plots

Usage

```
scale_colour_ssd(...)

scale_color_ssd(...)

scale_fill_ssd(...)
```

Arguments

`...`	Arguments passed to `ggplot2::discrete_scale()`.

Functions

- `scale_color_ssd()`: Discrete color-blind scale for SSD Plots
- `scale_fill_ssd()`: Discrete color-blind scale for SSD Plots

See Also

Other ggplot: `geom_hcintersect()`, `geom_ssdpoint()`, `geom_ssdsegment()`, `geom_xribbon()`, `ssd_pal()`

Examples

```
ssd_plot(ssddata::ccme_boron, boron_pred, shape = "Group") +
  scale_colour_ssd()
```

`ssdtools-ggproto` *ggproto Classes for Plotting Species Sensitivity Data and Distributions*

Description

ggproto Classes for Plotting Species Sensitivity Data and Distributions

Usage

```
StatSsdpoint

StatSsdsegment

GeomSsdpoint

GeomSsdsegment

GeomHcintersect

GeomXribbon
```

Format

An object of class `StatSsdpoint` (inherits from `Stat`, `ggproto`, `gg`) of length 4.

An object of class `StatSsdsegment` (inherits from `Stat`, `ggproto`, `gg`) of length 4.

An object of class `GeomSsdpoint` (inherits from `GeomPoint`, `Geom`, `ggproto`, `gg`) of length 1.

An object of class `GeomSsdsegment` (inherits from `GeomSegment`, `Geom`, `ggproto`, `gg`) of length 1.

An object of class `GeomHcintersect` (inherits from `Geom`, `ggproto`, `gg`) of length 5.

An object of class `GeomXribbon` (inherits from `Geom`, `ggproto`, `gg`) of length 6.

See Also

`ggplot2::ggproto()` and `ssd_plot_cdf()`

`ssd_censor_data` *Censor Data*

Description

Censor Data

Usage

```
ssd_censor_data(data, left = "Conc", ..., right = left, censoring = c(0, Inf))
```

Arguments

`data`	A data frame.
`left`	A string of the column in data with the concentrations.
`...`	Unused.
`right`	A string of the column in data with the right concentration values.
`censoring`	A numeric vector of the left and right censoring values.

Value

A tibble of the censored data.

Examples

```
ssd_censor_data(ssddata::ccme_boron, censoring = c(2.5, Inf))
```

`ssd_data` *Data from fitdists Object*

Description

Get a tibble of the original data.

Usage

```
ssd_data(x)
```

Arguments

`x`	The object.

Value

A tibble of the original data.

See Also

`augment.fitdists()`, `ssd_ecd_data()` and `ssd_sort_data()`

Examples

```
fits <- ssd_fit_dists(ssddata::ccme_boron)
ssd_data(fits)
```

`ssd_dists` *Species Sensitivity Distributions*

Description

Gets a character vector of the names of the available distributions.

Usage

```
ssd_dists(bcanz = NULL, tails = NULL, npars = 2:5)
```

Arguments

`bcanz`	A flag or NULL specifying whether to only include distributions in the set that is approved by BC, Canada, Australia and New Zealand for official guidelines.
`tails`	A flag or NULL specifying whether to only include distributions with both tails.
`npars`	A whole numeric vector specifying which distributions to include based on the number of parameters.

Value

A unique, sorted character vector of the distributions.

See Also

Other dists: `dist_data`, `ssd_dists_all()`

Examples

```
ssd_dists()
ssd_dists(bcanz = TRUE)
ssd_dists(tails = FALSE)
ssd_dists(npars = 5)
```

`ssd_dists_all` *All Species Sensitivity Distributions*

Description

Gets a character vector of the names of all the available distributions.

Usage

```
ssd_dists_all()
```

Value

A unique, sorted character vector of the distributions.

See Also

Other dists: `dist_data`, `ssd_dists()`

Examples

```
ssd_dists_all()
```

`ssd_dists_bcanz` *BCANZ Distributions*

Description

Gets a character vector of the names of the distributions adopted by BC, Canada, Australia and New Zealand for official guidelines.

Usage

```
ssd_dists_bcanz(npars = c(2L, 5L))
```

Arguments

`npars`	A whole numeric vector specifying which distributions to include based on the number of parameters.

Value

A unique, sorted character vector of the distributions.

See Also

`ssd_dists()`

Examples

```
ssd_dists_bcanz()
ssd_dists_bcanz(npars = 2)
```

`ssd_eburrIII3` *Default Parameter Estimates*

Description

Default Parameter Estimates

Usage

```
ssd_eburrIII3()

ssd_egamma()

ssd_egompertz()

ssd_einvpareto()

ssd_elgumbel()

ssd_elgumbel()

ssd_ellogis_llogis()

ssd_ellogis()

ssd_elnorm_lnorm()

ssd_elnorm()

ssd_emulti()

ssd_eweibull()
```

Functions

- `ssd_eburrIII3()`: Default Parameter Values for BurrIII Distribution
- `ssd_egamma()`: Default Parameter Values for Gamma Distribution
- `ssd_egompertz()`: Default Parameter Values for Gompertz Distribution
- `ssd_einvpareto()`: Default Parameter Values for Inverse Pareto Distribution
- `ssd_elgumbel()`: Default Parameter Values for Log-Gumbel Distribution
- `ssd_elgumbel()`: Default Parameter Values for log-Gumbel Distribution

- `ssd_ellogis_llogis()`: Default Parameter Values for Log-Logistic/Log-Logistic Mixture Distribution
- `ssd_ellogis()`: Default Parameter Values for Log-Logistic Distribution
- `ssd_elnorm_lnorm()`: Default Parameter Values for Log-Normal/Log-Normal Mixture Distribution
- `ssd_elnorm()`: Default Parameter Values for Log-Normal Distribution
- `ssd_emulti()`: Default Parameter Values for Multiple Distributions
- `ssd_eweibull()`: Default Parameter Values for Log-Normal Distribution

See Also

`ssd_p` and `ssd_q`

Examples

```
ssd_eburrIII3()

ssd_egamma()

ssd_egompertz()

ssd_einvpareto()

ssd_einvpareto()

ssd_elgumbel()

ssd_ellogis_llogis()

ssd_ellogis()

ssd_elnorm_lnorm()

ssd_elnorm()

ssd_emulti()

ssd_eweibull()
```

`ssd_ecd` *Empirical Cumulative Density*

Description

Empirical Cumulative Density

Usage

```
ssd_ecd(x, ties.method = "first")
```

Arguments

`x`	a numeric, complex, character or logical vector.
`ties.method`	a character string specifying how ties are treated, see ‘Details’; can be abbreviated.

Value

A numeric vector of the empirical cumulative density.

Examples

```
ssd_ecd(1:10)
```

`ssd_ecd_data` *Empirical Cumulative Density for Species Sensitivity Data*

Description

Empirical Cumulative Density for Species Sensitivity Data

Usage

```
ssd_ecd_data(
  data,
  left = "Conc",
  right = left,
  bounds = c(left = 1, right = 1)
)
```

Arguments

`data`	A data frame.
`left`	A string of the column in data with the concentrations.
`right`	A string of the column in data with the right concentration values.
`bounds`	A named non-negative numeric vector of the left and right bounds for uncensored missing (0 and Inf) data in terms of the orders of magnitude relative to the extremes for non-missing values.

Value

A numeric vector of the empirical cumulative density for the rows in data.

See Also

`ssd_ecd()` and `ssd_data()`

Examples

```
ssd_ecd_data(ssddata::ccme_boron)
```

`ssd_exposure` *Proportion Exposure*

Description

Calculates average proportion exposed based on log-normal distribution of concentrations.

Usage

```
ssd_exposure(x, meanlog = 0, sdlog = 1, nboot = 1000)
```

Arguments

`x`	The object.
`meanlog`	The mean of the exposure concentrations on the log scale.
`sdlog`	The standard deviation of the exposure concentrations on the log scale.
`nboot`	The number of samples to use to calculate the exposure.

Value

The proportion exposed.

Examples

```
## Not run:
  fits <- ssd_fit_dists(ssddata::ccme_boron, dists = "lnorm")
  set.seed(10)
  ssd_exposure(fits)
  ssd_exposure(fits, meanlog = 1)
  ssd_exposure(fits, meanlog = 1, sdlog = 1)

## End(Not run)
```

`ssd_fit_bcanz` *Fit BCANZ Distributions*

Description

Fits distributions using settings adopted by BC, Canada, Australia and New Zealand for official guidelines.

Usage

```
ssd_fit_bcanz(data, left = "Conc", dists = ssd_dists_bcanz())
```

Arguments

`data`	A data frame.
`left`	A string of the column in data with the concentrations.
`dists`	A character vector of the distribution names.

Value

An object of class fitdists.

See Also

`ssd_fit_dists()`

Other BCANZ: `ssd_hc_bcanz()`, `ssd_hp_bcanz()`

Examples

```
ssd_fit_bcanz(ssddata::ccme_boron)
```

`ssd_fit_burrlioz` *Fit Burrlioz Distributions*

Description

Fits ’burrIII3’ distribution. If shape1 parameter is at boundary returns ’lgumbel’ (which is equivalent to inverse Weibull). Else if shape2 parameter is at a boundary returns ’invpareto’. Otherwise returns ’burrIII3’

Usage

```
ssd_fit_burrlioz(
  data,
  left = "Conc",
  rescale = FALSE,
  control = list(),
  silent = FALSE
)
```

Arguments

`data`	A data frame.
`left`	A string of the column in data with the concentrations.
`rescale`	A flag specifying whether to rescale concentration values by dividing by the geometric mean of the minimum and maximum positive finite values.
`control`	A list of control parameters passed to `stats::optim()`.
`silent`	A flag indicating whether fits should fail silently.

Value

An object of class fitdists.

See Also

`ssd_fit_dists()`

Examples

```
ssd_fit_burrlioz(ssddata::ccme_boron)
```

`ssd_fit_dists` *Fit Distributions*

Description

Fits one or more distributions to species sensitivity data.

Usage

```
ssd_fit_dists(
  data,
  left = "Conc",
  right = left,
  weight = NULL,
  dists = ssd_dists_bcanz(),
  nrow = 6L,
```

```
  rescale = FALSE,
  reweight = FALSE,
  computable = FALSE,
  at_boundary_ok = TRUE,
  all_dists = FALSE,
  min_pmix = ssd_min_pmix(nrow(data)),
  range_shape1 = c(0.05, 20),
  range_shape2 = range_shape1,
  control = list(),
  silent = FALSE
)
```

Arguments

`data`	A data frame.
`left`	A string of the column in data with the concentrations.
`right`	A string of the column in data with the right concentration values.
`weight`	A string of the numeric column in data with positive weights less than or equal to 1,000 or NULL.
`dists`	A character vector of the distribution names.
`nrow`	A positive whole number of the minimum number of non-missing rows.
`rescale`	A flag specifying whether to rescale concentration values by dividing by the geometric mean of the minimum and maximum positive finite values.
`reweight`	A flag specifying whether to reweight weights by dividing by the largest weight.
`computable`	A flag specifying whether to only return fits with numerically computable standard errors.
`at_boundary_ok`	A flag specifying whether a model with one or more parameters at the boundary should be considered to have converged (default = FALSE).
`all_dists`	A flag specifying whether all the named distributions must fit successfully.
`min_pmix`	A number between 0 and 0.5 specifying the minimum proportion in mixture models.
`range_shape1`	A numeric vector of length two of the lower and upper bounds for the shape1 parameter.
`range_shape2`	shape2 parameter.
`control`	A list of control parameters passed to `stats::optim()`.
`silent`	A flag indicating whether fits should fail silently.

Details

By default the ’gamma’, ’lgumbel’, ’llogis’, ’lnorm’, ’lnorm_lnorm’ and ’weibull’ distributions are fitted to the data. For a complete list of the distributions that are currently implemented in `ssdtools` see `ssd_dists_all()`.

If weight specifies a column in the data frame with positive numbers, weighted estimation occurs. However, currently only the resultant parameter estimates are available.

If the `right` argument is different to the `left` argument then the data are considered to be censored.

Value

An object of class fitdists.

See Also

`ssd_plot_cdf()` and `ssd_hc()`

Examples

```
fits <- ssd_fit_dists(ssddata::ccme_boron)
fits
ssd_plot_cdf(fits)
ssd_hc(fits)
```

---

ssd_gof *Goodness of Fit*

Description

Returns a tbl data frame with the following columns

dist The distribution name (chr)

aic Akaike's Information Criterion (dbl)

bic Bayesian Information Criterion (dbl)

and if the data are non-censored

aicc Akaike's Information Criterion corrected for sample size (dbl)

and if there are 8 or more samples

ad Anderson-Darling statistic (dbl)

ks Kolmogorov-Smirnov statistic (dbl)

cvm Cramer-von Mises statistic (dbl)

In the case of an object of class fitdists the function also returns

delta The Information Criterion differences (dbl)

weight The Information Criterion weights (dbl)

where `delta` and `weight` are based on `aic` for censored data and `aicc` for non-censored data.

Usage

```
ssd_gof(x, ...)

## S3 method for class 'fitdists'
ssd_gof(x, pvalue = FALSE, ...)
```

Arguments

`x`	The object.
`...`	Unused.
`pvalue`	A flag specifying whether to return p-values or the statistics (default) for the various tests.

Value

A tbl data frame of the gof statistics.

Methods (by class)

- `ssd_gof(fitdists)`: Goodness of Fit

See Also

`glance.fitdists()`

Examples

```
fits <- ssd_fit_dists(ssddata::ccme_boron)
ssd_gof(fits)
ssd_gof(fits)
```

`ssd_hc` *Hazard Concentrations for Species Sensitivity Distributions*

Description

Calculates concentration(s) with bootstrap confidence intervals that protect specified proportion(s) of species for individual or model-averaged distributions using parametric or non-parametric bootstrapping.

Usage

```
ssd_hc(x, ...)

## S3 method for class 'list'
ssd_hc(x, percent, proportion = 0.05, ...)

## S3 method for class 'fitdists'
ssd_hc(
  x,
  percent,
  proportion = 0.05,
  average = TRUE,
  ci = FALSE,
```

```
  level = 0.95,
  nboot = 1000,
  min_pboot = 0.95,
  multi_est = TRUE,
  ci_method = "weighted_samples",
  parametric = TRUE,
  delta = 9.21,
  samples = FALSE,
  save_to = NULL,
  control = NULL,
  ...
)

## S3 method for class 'fitburrlioz'
ssd_hc(
  x,
  percent,
  proportion = 0.05,
  ci = FALSE,
  level = 0.95,
  nboot = 1000,
  min_pboot = 0.95,
  parametric = FALSE,
  samples = FALSE,
  save_to = NULL,
  ...
)
```

Arguments

`x`	The object.
`...`	Unused.
`percent`	A numeric vector of percent values to estimate hazard concentrations for. Soft-deprecated for `proportion = 0.05`.
`proportion`	A numeric vector of proportion values to estimate hazard concentrations for.
`average`	A flag specifying whether to provide model averaged values as opposed to a value for each distribution.
`ci`	A flag specifying whether to estimate confidence intervals (by bootstrapping).
`level`	A number between 0 and 1 of the confidence level of the interval.
`nboot`	A count of the number of bootstrap samples to use to estimate the confidence limits. A value of 10,000 is recommended for official guidelines.
`min_pboot`	A number between 0 and 1 of the minimum proportion of bootstrap samples that must successfully fit (return a likelihood) to report the confidence intervals.
`multi_est`	A flag specifying whether to treat the distributions as constituting a single distribution (as opposed to taking the mean) when calculating model averaged estimates.

`ci_method`	A string specifying which method to use for estimating the bootstrap values. Possible values are "multi_free" and "multi_fixed" which treat the distributions as constituting a single distribution but differ in whether the model weights are fixed and "weighted_samples" and "weighted_arithmetic" take bootstrap samples from each distribution proportional to its weight versus calculating the weighted arithmetic means of the lower and upper confidence limits.
`parametric`	A flag specifying whether to perform parametric bootstrapping as opposed to non-parametrically resampling the original data with replacement.
`delta`	A non-negative number specifying the maximum absolute AIC difference cutoff. Distributions with an absolute AIC difference greater than delta are excluded from the calculations.
`samples`	A flag specfying whether to include a numeric vector of the bootstrap samples as a list column in the output.
`save_to`	NULL or a string specifying a directory to save where the bootstrap datasets and parameter estimates (when successfully converged) to.
`control`	A list of control parameters passed to `stats::optim()`.

Details

Model-averaged estimates and/or confidence intervals (including standard error) can be calculated by treating the distributions as constituting a single mixture distribution versus 'taking the mean'. When calculating the model averaged estimates treating the distributions as constituting a single mixture distribution ensures that `ssd_hc()` is the inverse of `ssd_hp()`.

If treating the distributions as constituting a single mixture distribution when calculating model average confidence intervals then `weighted` specifies whether to use the original model weights versus re-estimating for each bootstrap sample unless 'taking the mean' in which case `weighted` specifies whether to take bootstrap samples from each distribution proportional to its weight (so that they sum to `nboot`) versus calculating the weighted arithmetic means of the lower and upper confidence limits based on `nboot` samples for each distribution.

Distributions with an absolute AIC difference greater than a delta of by default 7 have considerably less support (weight < 0.01) and are excluded prior to calculation of the hazard concentrations to reduce the run time.

Value

A tibble of corresponding hazard concentrations.

Methods (by class)

- `ssd_hc(list)`: Hazard Concentrations for Distributional Estimates
- `ssd_hc(fitdists)`: Hazard Concentrations for fitdists Object
- `ssd_hc(fitburrlioz)`: Hazard Concentrations for fitburrlioz Object

References

Burnham, K.P., and Anderson, D.R. 2002. Model Selection and Multimodel Inference. Springer New York, New York, NY. doi:10.1007/b97636.

See Also

`predict.fitdists()` and `ssd_hp()`.

Examples

```
ssd_hc(ssd_match_moments())

fits <- ssd_fit_dists(ssddata::ccme_boron)
ssd_hc(fits)

fit <- ssd_fit_burrlioz(ssddata::ccme_boron)
ssd_hc(fit)
```

`ssd_hc_bcanz` *BCANZ Hazard Concentrations*

Description

Gets hazard concentrations with confidence intervals that protect 1, 5, 10 and 20% of species using settings adopted by BC, Canada, Australia and New Zealand for official guidelines. This function can take several minutes to run with recommended 10,000 iterations.

Usage

```
ssd_hc_bcanz(x, nboot = 10000, min_pboot = 0.95)
```

Arguments

`x`	The object.
`nboot`	A count of the number of bootstrap samples to use to estimate the confidence limits. A value of 10,000 is recommended for official guidelines.
`min_pboot`	A number between 0 and 1 of the minimum proportion of bootstrap samples that must successfully fit (return a likelihood) to report the confidence intervals.

Value

A tibble of corresponding hazard concentrations.

See Also

`ssd_hc()`.

Other BCANZ: `ssd_fit_bcanz()`, `ssd_hp_bcanz()`

Examples

```
fits <- ssd_fit_bcanz(ssddata::ccme_boron)
ssd_hc_bcanz(fits, nboot = 100)
```

`ssd_hc_burrlioz` *Hazard Concentrations for Burrlioz Fit* **[Deprecated]**

Description

Deprecated for `ssd_hc()`.

Usage

```
ssd_hc_burrlioz(
  x,
  percent,
  proportion = 0.05,
  ci = FALSE,
  level = 0.95,
  nboot = 1000,
  min_pboot = 0.95,
  parametric = FALSE
)
```

Arguments

`x`	The object.
`percent`	A numeric vector of percent values to estimate hazard concentrations for. Soft-deprecated for `proportion = 0.05`.
`proportion`	A numeric vector of proportion values to estimate hazard concentrations for.
`ci`	A flag specifying whether to estimate confidence intervals (by bootstrapping).
`level`	A number between 0 and 1 of the confidence level of the interval.
`nboot`	A count of the number of bootstrap samples to use to estimate the confidence limits. A value of 10,000 is recommended for official guidelines.
`min_pboot`	A number between 0 and 1 of the minimum proportion of bootstrap samples that must successfully fit (return a likelihood) to report the confidence intervals.
`parametric`	A flag specifying whether to perform parametric bootstrapping as opposed to non-parametrically resampling the original data with replacement.

Value

A tibble of corresponding hazard concentrations.

`ssd_hp` *Hazard Proportion*

Description

Calculates proportion of species affected at specified concentration(s) with quantile based bootstrap confidence intervals for individual or model-averaged distributions using parametric or non-parametric bootstrapping. For more information see the inverse function `ssd_hc()`.

Usage

```
ssd_hp(x, ...)

## S3 method for class 'fitdists'
ssd_hp(
  x,
  conc = 1,
  average = TRUE,
  ci = FALSE,
  level = 0.95,
  nboot = 1000,
  min_pboot = 0.95,
  multi_est = TRUE,
  ci_method = "weighted_samples",
  parametric = TRUE,
  delta = 9.21,
  samples = FALSE,
  save_to = NULL,
  control = NULL,
  ...
)

## S3 method for class 'fitburrlioz'
ssd_hp(
  x,
  conc = 1,
  ci = FALSE,
  level = 0.95,
  nboot = 1000,
  min_pboot = 0.95,
  parametric = FALSE,
  samples = FALSE,
  save_to = NULL,
  ...
)
```

Arguments

`x`	The object.
`...`	Unused.
`conc`	A numeric vector of concentrations to calculate the hazard proportions for.
`average`	A flag specifying whether to provide model averaged values as opposed to a value for each distribution.
`ci`	A flag specifying whether to estimate confidence intervals (by bootstrapping).
`level`	A number between 0 and 1 of the confidence level of the interval.
`nboot`	A count of the number of bootstrap samples to use to estimate the confidence limits. A value of 10,000 is recommended for official guidelines.
`min_pboot`	A number between 0 and 1 of the minimum proportion of bootstrap samples that must successfully fit (return a likelihood) to report the confidence intervals.
`multi_est`	A flag specifying whether to treat the distributions as constituting a single distribution (as opposed to taking the mean) when calculating model averaged estimates.
`ci_method`	A string specifying which method to use for estimating the bootstrap values. Possible values are "multi_free" and "multi_fixed" which treat the distributions as constituting a single distribution but differ in whether the model weights are fixed and "weighted_samples" and "weighted_arithmetic" take bootstrap samples from each distribution proportional to its weight versus calculating the weighted arithmetic means of the lower and upper confidence limits.
`parametric`	A flag specifying whether to perform parametric bootstrapping as opposed to non-parametrically resampling the original data with replacement.
`delta`	A non-negative number specifying the maximum absolute AIC difference cutoff. Distributions with an absolute AIC difference greater than delta are excluded from the calculations.
`samples`	A flag specfying whether to include a numeric vector of the bootstrap samples as a list column in the output.
`save_to`	NULL or a string specifying a directory to save where the bootstrap datasets and parameter estimates (when successfully converged) to.
`control`	A list of control parameters passed to `stats::optim()`.

Value

A tibble of corresponding hazard proportions.

Methods (by class)

- `ssd_hp(fitdists)`: Hazard Proportions for fitdists Object
- `ssd_hp(fitburrlioz)`: Hazard Proportions for fitburrlioz Object

See Also

`ssd_hc()`

Examples

```
fits <- ssd_fit_dists(ssddata::ccme_boron)
ssd_hp(fits, conc = 1)

fit <- ssd_fit_burrlioz(ssddata::ccme_boron)
ssd_hp(fit)
```

`ssd_hp_bcanz` *BCANZ Hazard Proportion*

Description

Gets proportion of species affected at specified concentration(s) using settings adopted by BC, Canada, Australia and New Zealand for official guidelines. This function can take several minutes to run with recommended 10,000 iterations.

Usage

```
ssd_hp_bcanz(x, conc = 1, nboot = 10000, min_pboot = 0.95)
```

Arguments

`x`	The object.
`conc`	A numeric vector of concentrations to calculate the hazard proportions for.
`nboot`	A count of the number of bootstrap samples to use to estimate the confidence limits. A value of 10,000 is recommended for official guidelines.
`min_pboot`	A number between 0 and 1 of the minimum proportion of bootstrap samples that must successfully fit (return a likelihood) to report the confidence intervals.

Value

A tibble of corresponding hazard concentrations.

See Also

`ssd_hp()`.

Other BCANZ: `ssd_fit_bcanz()`, `ssd_hc_bcanz()`

Examples

```
fits <- ssd_fit_bcanz(ssddata::ccme_boron)
ssd_hp_bcanz(fits, nboot = 100)
```

`ssd_is_censored` *Is Censored*

Description

Tests if an object has censored data.

Test if a data frame is censored.

Test if a fitdists object is censored.

Usage

```
ssd_is_censored(x, ...)

## S3 method for class 'data.frame'
ssd_is_censored(x, left = "Conc", right = left, ...)

## S3 method for class 'fitdists'
ssd_is_censored(x, ...)
```

Arguments

`x`	The object.
`...`	Unused.
`left`	A string of the column in data with the concentrations.
`right`	A string of the column in data with the right concentration values.

Value

A flag indicating whether an object is censored.

Examples

```
ssd_is_censored(ssddata::ccme_boron)
ssd_is_censored(data.frame(Conc = 1, right = 2), right = "right")

fits <- ssd_fit_dists(ssddata::ccme_boron)
ssd_is_censored(fits)
```

`ssd_label_comma` *Label numbers with significant digits and comma*

Description

Label numbers with significant digits and comma

Usage

```
ssd_label_comma(digits = 3, ..., big.mark = ",")
```

Arguments

`digits`	A whole number specifying the number of significant figures.
`...`	Unused.
`big.mark`	A string specifying used between every 3 digits to separate thousands on the x-axis.

Value

A "labelling" function that takes a vector x and returns a character vector of `length(x)` giving a label for each input value.

See Also

`scales::label_comma()`

Examples

```
ggplot2::ggplot(data = ssddata::anon_e, ggplot2::aes(x = Conc / 10)) +
  geom_ssdpoint() +
  ggplot2::scale_x_log10(labels = ssd_label_comma())
```

`ssd_licensing_md` *Licensing Markdown*

Description

A string of markdown code indicating the licensing of the code and documentation

Usage

```
ssd_licensing_md()
```

Examples

```
ssd_licensing_md()
```

`ssd_match_moments` *Match Moments*

Description

Gets a named list of the values that produce the moment values (meanlog and sdlog) by distribution and term.

Usage

```
ssd_match_moments(
  dists = ssd_dists_bcanz(),
  meanlog = 1,
  sdlog = 1,
  nsim = 1e+05
)
```

Arguments

`dists`	A character vector of the distribution names.
`meanlog`	The mean on the log scale.
`sdlog`	The standard deviation on the log scale.
`nsim`	A positive whole number of the number of simulations to generate.

Value

a named list of the values that produce the moment values by distribution and term.

See Also

`estimates.fitdists()`, `ssd_hc()` and `ssd_plot_cdf()`

Examples

```
moments <- ssd_match_moments()
print(moments)
ssd_hc(moments)
ssd_plot_cdf(moments)
```

`ssd_min_pmix` *Calculate Minimum Proportion in Mixture Models*

Description

Calculate Minimum Proportion in Mixture Models

Usage

```
ssd_min_pmix(n)
```

Arguments

`n`	positive number of observations.

Value

A number between 0 and 0.5 of the minimum proportion in mixture models.

See Also

`ssd_fit_dists()`

Examples

```
ssd_min_pmix(6)
ssd_min_pmix(50)
```

`ssd_pal` *Color-blind Palette for SSD Plots*

Description

Color-blind Palette for SSD Plots

Usage

```
ssd_pal()
```

Value

A character vector of a color blind palette with 8 colors.

See Also

Other ggplot: `geom_hcintersect()`, `geom_ssdpoint()`, `geom_ssdsegment()`, `geom_xribbon()`, `scale_colour_ssd()`

Examples

```
ssd_pal()
```

`ssd_pburrIII3` *Cumulative Distribution Function*

Description

Cumulative Distribution Function

Usage

```
ssd_pburrIII3(
  q,
  shape1 = 1,
  shape2 = 1,
  scale = 1,
  lower.tail = TRUE,
  log.p = FALSE
)

ssd_pgamma(q, shape = 1, scale = 1, lower.tail = TRUE, log.p = FALSE)

ssd_pgompertz(q, location = 1, shape = 1, lower.tail = TRUE, log.p = FALSE)

ssd_pinvpareto(q, shape = 3, scale = 1, lower.tail = TRUE, log.p = FALSE)

ssd_plgumbel(
  q,
  locationlog = 0,
  scalelog = 1,
  lower.tail = TRUE,
  log.p = FALSE
)

ssd_pllogis_llogis(
  q,
  locationlog1 = 0,
  scalelog1 = 1,
  locationlog2 = 1,
  scalelog2 = 1,
  pmix = 0.5,
  lower.tail = TRUE,
  log.p = FALSE
)

ssd_pllogis(q, locationlog = 0, scalelog = 1, lower.tail = TRUE, log.p = FALSE)
```

```
ssd_plnorm_lnorm(
  q,
  meanlog1 = 0,
  sdlog1 = 1,
  meanlog2 = 1,
  sdlog2 = 1,
  pmix = 0.5,
  lower.tail = TRUE,
  log.p = FALSE
)

ssd_plnorm(q, meanlog = 0, sdlog = 1, lower.tail = TRUE, log.p = FALSE)

ssd_pmulti(
  q,
  burrIII3.weight = 0,
  burrIII3.shape1 = 1,
  burrIII3.shape2 = 1,
  burrIII3.scale = 1,
  gamma.weight = 0,
  gamma.shape = 1,
  gamma.scale = 1,
  gompertz.weight = 0,
  gompertz.location = 1,
  gompertz.shape = 1,
  invpareto.weight = 0,
  invpareto.shape = 3,
  invpareto.scale = 1,
  lgumbel.weight = 0,
  lgumbel.locationlog = 0,
  lgumbel.scalelog = 1,
  llogis.weight = 0,
  llogis.locationlog = 0,
  llogis.scalelog = 1,
  llogis_llogis.weight = 0,
  llogis_llogis.locationlog1 = 0,
  llogis_llogis.scalelog1 = 1,
  llogis_llogis.locationlog2 = 1,
  llogis_llogis.scalelog2 = 1,
  llogis_llogis.pmix = 0.5,
  lnorm.weight = 0,
  lnorm.meanlog = 0,
  lnorm.sdlog = 1,
  lnorm_lnorm.weight = 0,
  lnorm_lnorm.meanlog1 = 0,
  lnorm_lnorm.sdlog1 = 1,
  lnorm_lnorm.meanlog2 = 1,
```

```
  lnorm_lnorm.sdlog2 = 1,
  lnorm_lnorm.pmix = 0.5,
  weibull.weight = 0,
  weibull.shape = 1,
  weibull.scale = 1,
  lower.tail = TRUE,
  log.p = FALSE
)

ssd_pmulti_fitdists(q, fitdists, lower.tail = TRUE, log.p = FALSE)

ssd_pweibull(q, shape = 1, scale = 1, lower.tail = TRUE, log.p = FALSE)
```

Arguments

`q`	vector of quantiles.
`shape1`	shape1 parameter.
`shape2`	shape2 parameter.
`scale`	scale parameter.
`lower.tail`	logical; if TRUE (default), probabilities are `P[X <= x]`, otherwise, `P[X > x]`.
`log.p`	logical; if TRUE, probabilities p are given as log(p).
`shape`	shape parameter.
`location`	location parameter.
`locationlog`	location on the log scale parameter.
`scalelog`	scale on log scale parameter.
`locationlog1`	locationlog1 parameter.
`scalelog1`	scalelog1 parameter.
`locationlog2`	locationlog2 parameter.
`scalelog2`	scalelog2 parameter.
`pmix`	Proportion mixture parameter.
`meanlog1`	mean on log scale parameter.
`sdlog1`	standard deviation on log scale parameter.
`meanlog2`	mean on log scale parameter.
`sdlog2`	standard deviation on log scale parameter.
`meanlog`	mean on log scale parameter.
`sdlog`	standard deviation on log scale parameter.
`burrIII3.weight`	weight parameter for the Burr III distribution.
`burrIII3.shape1`	shape1 parameter for the Burr III distribution.
`burrIII3.shape2`	shape2 parameter for the Burr III distribution.

`burrIII3.scale`	scale parameter for the Burr III distribution.
`gamma.weight`	weight parameter for the gamma distribution.
`gamma.shape`	shape parameter for the gamma distribution.
`gamma.scale`	scale parameter for the gamma distribution.
`gompertz.weight`	weight parameter for the Gompertz distribution.
`gompertz.location`	location parameter for the Gompertz distribution.
`gompertz.shape`	shape parameter for the Gompertz distribution.
`invpareto.weight`	weight parameter for the inverse Pareto distribution.
`invpareto.shape`	shape parameter for the inverse Pareto distribution.
`invpareto.scale`	scale parameter for the inverse Pareto distribution.
`lgumbel.weight`	weight parameter for the log-Gumbel distribution.
`lgumbel.locationlog`	location parameter for the log-Gumbel distribution.
`lgumbel.scalelog`	scale parameter for the log-Gumbel distribution.
`llogis.weight`	weight parameter for the log-logistic distribution.
`llogis.locationlog`	location parameter for the log-logistic distribution.
`llogis.scalelog`	scale parameter for the log-logistic distribution.
`llogis_llogis.weight`	weight parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.locationlog1`	locationlog1 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.scalelog1`	scalelog1 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.locationlog2`	locationlog2 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.scalelog2`	scalelog2 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.pmix`	pmix parameter for the log-logistic log-logistic mixture distribution.
`lnorm.weight`	weight parameter for the log-normal distribution.
`lnorm.meanlog`	meanlog parameter for the log-normal distribution.
`lnorm.sdlog`	sdlog parameter for the log-normal distribution.
`lnorm_lnorm.weight`	weight parameter for the log-normal log-normal mixture distribution.

`lnorm_lnorm.meanlog1`	meanlog1 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.sdlog1`	sdlog1 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.meanlog2`	meanlog2 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.sdlog2`	sdlog2 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.pmix`	pmix parameter for the log-normal log-normal mixture distribution.
`weibull.weight`	weight parameter for the Weibull distribution.
`weibull.shape`	shape parameter for the Weibull distribution.
`weibull.scale`	scale parameter for the Weibull distribution.
`fitdists`	An object of class fitdists.

Functions

- `ssd_pburrIII3()`: Cumulative Distribution Function for BurrIII Distribution
- `ssd_pgamma()`: Cumulative Distribution Function for Gamma Distribution
- `ssd_pgompertz()`: Cumulative Distribution Function for Gompertz Distribution
- `ssd_pinvpareto()`: Cumulative Distribution Function for Inverse Pareto Distribution
- `ssd_plgumbel()`: Cumulative Distribution Function for Log-Gumbel Distribution
- `ssd_pllogis_llogis()`: Cumulative Distribution Function for Log-Logistic/Log-Logistic Mixture Distribution
- `ssd_pllogis()`: Cumulative Distribution Function for Log-Logistic Distribution
- `ssd_plnorm_lnorm()`: Cumulative Distribution Function for Log-Normal/Log-Normal Mixture Distribution
- `ssd_plnorm()`: Cumulative Distribution Function for Log-Normal Distribution
- `ssd_pmulti()`: Cumulative Distribution Function for Multiple Distributions
- `ssd_pmulti_fitdists()`: Cumulative Distribution Function for Multiple Distributions
- `ssd_pweibull()`: Cumulative Distribution Function for Weibull Distribution

See Also

`ssd_q` and `ssd_r`

Examples

```
ssd_pburrIII3(1)

ssd_pgamma(1)

ssd_pgompertz(1)
```

```
ssd_pinvpareto(1)

ssd_plgumbel(1)

ssd_pllogis_llogis(1)

ssd_pllogis(1)

ssd_plnorm_lnorm(1)

ssd_plnorm(1)

# multi
ssd_pmulti(1, gamma.weight = 0.5, lnorm.weight = 0.5)

# multi fitdists
fit <- ssd_fit_dists(ssddata::ccme_boron)
ssd_pmulti_fitdists(1, fit)

ssd_pweibull(1)
```

`ssd_plot` *Plot Species Sensitivity Data and Distributions*

Description

Plots species sensitivity data and distributions.

Usage

```
ssd_plot(
  data,
  pred,
  left = "Conc",
  right = left,
  ...,
  label = NULL,
  shape = NULL,
  color = NULL,
  size = 2.5,
  linetype = NULL,
  linecolor = NULL,
  xlab = "Concentration",
  ylab = "Species Affected",
  ci = TRUE,
  ribbon = TRUE,
  hc = 0.05,
  shift_x = 3,
  add_x = 0,
```

```
  bounds = c(left = 1, right = 1),
  big.mark = ",",
  suffix = "%",
  trans = "log10",
  xbreaks = waiver()
)
```

Arguments

`data`	A data frame.
`pred`	A data frame of the predictions.
`left`	A string of the column in data with the concentrations.
`right`	A string of the column in data with the right concentration values.
`...`	Unused.
`label`	A string of the column in data with the labels.
`shape`	A string of the column in data for the shape aesthetic.
`color`	A string of the column in data for the color aesthetic.
`size`	A number for the size of the labels.
`linetype`	A string of the column in pred to use for the linetype.
`linecolor`	A string of the column in pred to use for the line color.
`xlab`	A string of the x-axis label.
`ylab`	A string of the x-axis label.
`ci`	A flag specifying whether to estimate confidence intervals (by bootstrapping).
`ribbon`	A flag indicating whether to plot the confidence interval as a grey ribbon as opposed to green solid lines.
`hc`	A value between 0 and 1 indicating the proportion hazard concentration (or NULL).
`shift_x`	The value to multiply the label x values by (after adding `add_x`).
`add_x`	The value to add to the label x values (before multiplying by `shift_x`).
`bounds`	A named non-negative numeric vector of the left and right bounds for uncensored missing (0 and Inf) data in terms of the orders of magnitude relative to the extremes for non-missing values.
`big.mark`	A string specifying used between every 3 digits to separate thousands on the x-axis.
`suffix`	Additional text to display after the number on the y-axis.
`trans`	A string which transformation to use by default `"log10"`.
`xbreaks`	The x-axis breaks as one of: • `NULL` for no breaks • `waiver()` for the default breaks • A numeric vector of positions

See Also

`ssd_plot_cdf()` and `geom_ssdpoint()`

Examples

```
ssd_plot(ssddata::ccme_boron, boron_pred, label = "Species", shape = "Group")
```

`ssd_plot_cdf` *Plot Cumulative Distribution Function (CDF)*

Description

Generic function to plots the cumulative distribution function (CDF).

Usage

```
ssd_plot_cdf(x, ...)

## S3 method for class 'fitdists'
ssd_plot_cdf(x, average = FALSE, delta = 9.21, ...)

## S3 method for class 'list'
ssd_plot_cdf(x, ...)
```

Arguments

`x`	The object.
`...`	Additional arguments passed to `ssd_plot()`.
`average`	A flag specifying whether to provide model averaged values as opposed to a value for each distribution or if NA provides model averaged and individual values.
`delta`	A non-negative number specifying the maximum absolute AIC difference cutoff. Distributions with an absolute AIC difference greater than delta are excluded from the calculations.

Methods (by class)

- `ssd_plot_cdf(fitdists)`: Plot CDF for fitdists object
- `ssd_plot_cdf(list)`: Plot CDF for named list of distributional parameter values

See Also

`ssd_plot()`

`estimates.fitdists()` and `ssd_match_moments()`

Examples

```
fits <- ssd_fit_dists(ssddata::ccme_boron)
ssd_plot_cdf(fits)
ssd_plot_cdf(fits, average = NA)

ssd_plot_cdf(list(
  llogis = c(locationlog = 2, scalelog = 1),
  lnorm = c(meanlog = 2, sdlog = 2)
))
```

`ssd_plot_cf` *Cullen and Frey Plot* **[Deprecated]**

Description

Plots a Cullen and Frey graph of the skewness and kurtosis for non-censored data.

Usage

```
ssd_plot_cf(data, left = "Conc")
```

Arguments

`data`	A data frame.
`left`	A string of the column in data with the concentrations.

Details

Soft deprecated for direct call to `fitdistrplus::descdist()`.

`ssd_plot_data` *Plot Species Sensitivity Data*

Description

Plots species sensitivity data.

Usage

```
ssd_plot_data(
  data,
  left = "Conc",
  right = left,
  ...,
  label = NULL,
  shape = NULL,
  color = NULL,
  size = 2.5,
  xlab = "Concentration",
  ylab = "Species Affected",
  shift_x = 3,
  add_x = 0,
  big.mark = ",",
  suffix = "%",
  bounds = c(left = 1, right = 1),
  trans = "log10",
  xbreaks = waiver()
)
```

Arguments

`data`	A data frame.
`left`	A string of the column in data with the concentrations.
`right`	A string of the column in data with the right concentration values.
`...`	Unused.
`label`	A string of the column in data with the labels.
`shape`	A string of the column in data for the shape aesthetic.
`color`	A string of the column in data for the color aesthetic.
`size`	A number for the size of the labels.
`xlab`	A string of the x-axis label.
`ylab`	A string of the x-axis label.
`shift_x`	The value to multiply the label x values by (after adding `add_x`).
`add_x`	The value to add to the label x values (before multiplying by `shift_x`).
`big.mark`	A string specifying used between every 3 digits to separate thousands on the x-axis.
`suffix`	Additional text to display after the number on the y-axis.
`bounds`	A named non-negative numeric vector of the left and right bounds for uncensored missing (0 and Inf) data in terms of the orders of magnitude relative to the extremes for non-missing values.
`trans`	A string which transformation to use by default `"log10"`.
`xbreaks`	The x-axis breaks as one of:

- `NULL` for no breaks
- `waiver()` for the default breaks
- A numeric vector of positions

See Also

`ssd_plot()` and `geom_ssdpoint()`

Examples

```
ssd_plot_data(ssddata::ccme_boron, label = "Species", shape = "Group")
```

`ssd_qburrIII3` *Quantile Function*

Description

Quantile Function

Usage

```
ssd_qburrIII3(
  p,
  shape1 = 1,
  shape2 = 1,
  scale = 1,
  lower.tail = TRUE,
  log.p = FALSE
)

ssd_qgamma(p, shape = 1, scale = 1, lower.tail = TRUE, log.p = FALSE)

ssd_qgompertz(p, location = 1, shape = 1, lower.tail = TRUE, log.p = FALSE)

ssd_qinvpareto(p, shape = 3, scale = 1, lower.tail = TRUE, log.p = FALSE)

ssd_qlgumbel(
  p,
  locationlog = 0,
  scalelog = 1,
  lower.tail = TRUE,
  log.p = FALSE
)

ssd_qllogis_llogis(
  p,
  locationlog1 = 0,
```

```
  scalelog1 = 1,
  locationlog2 = 1,
  scalelog2 = 1,
  pmix = 0.5,
  lower.tail = TRUE,
  log.p = FALSE
)

ssd_qllogis(p, locationlog = 0, scalelog = 1, lower.tail = TRUE, log.p = FALSE)

ssd_qlnorm_lnorm(
  p,
  meanlog1 = 0,
  sdlog1 = 1,
  meanlog2 = 1,
  sdlog2 = 1,
  pmix = 0.5,
  lower.tail = TRUE,
  log.p = FALSE
)

ssd_qlnorm(p, meanlog = 0, sdlog = 1, lower.tail = TRUE, log.p = FALSE)

ssd_qmulti(
  p,
  burrIII3.weight = 0,
  burrIII3.shape1 = 1,
  burrIII3.shape2 = 1,
  burrIII3.scale = 1,
  gamma.weight = 0,
  gamma.shape = 1,
  gamma.scale = 1,
  gompertz.weight = 0,
  gompertz.location = 1,
  gompertz.shape = 1,
  invpareto.weight = 0,
  invpareto.shape = 3,
  invpareto.scale = 1,
  lgumbel.weight = 0,
  lgumbel.locationlog = 0,
  lgumbel.scalelog = 1,
  llogis.weight = 0,
  llogis.locationlog = 0,
  llogis.scalelog = 1,
  llogis_llogis.weight = 0,
  llogis_llogis.locationlog1 = 0,
  llogis_llogis.scalelog1 = 1,
  llogis_llogis.locationlog2 = 1,
```

```
    llogis_llogis.scalelog2 = 1,
    llogis_llogis.pmix = 0.5,
    lnorm.weight = 0,
    lnorm.meanlog = 0,
    lnorm.sdlog = 1,
    lnorm_lnorm.weight = 0,
    lnorm_lnorm.meanlog1 = 0,
    lnorm_lnorm.sdlog1 = 1,
    lnorm_lnorm.meanlog2 = 1,
    lnorm_lnorm.sdlog2 = 1,
    lnorm_lnorm.pmix = 0.5,
    weibull.weight = 0,
    weibull.shape = 1,
    weibull.scale = 1,
    lower.tail = TRUE,
    log.p = FALSE
  )

  ssd_qmulti_fitdists(p, fitdists, lower.tail = TRUE, log.p = FALSE)

  ssd_qweibull(p, shape = 1, scale = 1, lower.tail = TRUE, log.p = FALSE)
```

Arguments

`p`	vector of probabilities.
`shape1`	shape1 parameter.
`shape2`	shape2 parameter.
`scale`	scale parameter.
`lower.tail`	logical; if TRUE (default), probabilities are `P[X <= x]`, otherwise, `P[X > x]`.
`log.p`	logical; if TRUE, probabilities p are given as log(p).
`shape`	shape parameter.
`location`	location parameter.
`locationlog`	location on the log scale parameter.
`scalelog`	scale on log scale parameter.
`locationlog1`	locationlog1 parameter.
`scalelog1`	scalelog1 parameter.
`locationlog2`	locationlog2 parameter.
`scalelog2`	scalelog2 parameter.
`pmix`	Proportion mixture parameter.
`meanlog1`	mean on log scale parameter.
`sdlog1`	standard deviation on log scale parameter.
`meanlog2`	mean on log scale parameter.
`sdlog2`	standard deviation on log scale parameter.

`meanlog`	mean on log scale parameter.
`sdlog`	standard deviation on log scale parameter.
`burrIII3.weight`	weight parameter for the Burr III distribution.
`burrIII3.shape1`	shape1 parameter for the Burr III distribution.
`burrIII3.shape2`	shape2 parameter for the Burr III distribution.
`burrIII3.scale`	scale parameter for the Burr III distribution.
`gamma.weight`	weight parameter for the gamma distribution.
`gamma.shape`	shape parameter for the gamma distribution.
`gamma.scale`	scale parameter for the gamma distribution.
`gompertz.weight`	weight parameter for the Gompertz distribution.
`gompertz.location`	location parameter for the Gompertz distribution.
`gompertz.shape`	shape parameter for the Gompertz distribution.
`invpareto.weight`	weight parameter for the inverse Pareto distribution.
`invpareto.shape`	shape parameter for the inverse Pareto distribution.
`invpareto.scale`	scale parameter for the inverse Pareto distribution.
`lgumbel.weight`	weight parameter for the log-Gumbel distribution.
`lgumbel.locationlog`	location parameter for the log-Gumbel distribution.
`lgumbel.scalelog`	scale parameter for the log-Gumbel distribution.
`llogis.weight`	weight parameter for the log-logistic distribution.
`llogis.locationlog`	location parameter for the log-logistic distribution.
`llogis.scalelog`	scale parameter for the log-logistic distribution.
`llogis_llogis.weight`	weight parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.locationlog1`	locationlog1 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.scalelog1`	scalelog1 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.locationlog2`	locationlog2 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.scalelog2`	scalelog2 parameter for the log-logistic log-logistic mixture distribution.

`llogis_llogis.pmix`	pmix parameter for the log-logistic log-logistic mixture distribution.
`lnorm.weight`	weight parameter for the log-normal distribution.
`lnorm.meanlog`	meanlog parameter for the log-normal distribution.
`lnorm.sdlog`	sdlog parameter for the log-normal distribution.
`lnorm_lnorm.weight`	weight parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.meanlog1`	meanlog1 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.sdlog1`	sdlog1 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.meanlog2`	meanlog2 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.sdlog2`	sdlog2 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.pmix`	pmix parameter for the log-normal log-normal mixture distribution.
`weibull.weight`	weight parameter for the Weibull distribution.
`weibull.shape`	shape parameter for the Weibull distribution.
`weibull.scale`	scale parameter for the Weibull distribution.
`fitdists`	An object of class fitdists.

Functions

- `ssd_qburrIII3()`: Quantile Function for BurrIII Distribution
- `ssd_qgamma()`: Quantile Function for Gamma Distribution
- `ssd_qgompertz()`: Quantile Function for Gompertz Distribution
- `ssd_qinvpareto()`: Quantile Function for Inverse Pareto Distribution
- `ssd_qlgumbel()`: Quantile Function for Log-Gumbel Distribution
- `ssd_qllogis_llogis()`: Cumulative Distribution Function for Log-Logistic/Log-Logistic Mixture Distribution
- `ssd_qllogis()`: Cumulative Distribution Function for Log-Logistic Distribution
- `ssd_qlnorm_lnorm()`: Cumulative Distribution Function for Log-Normal/Log-Normal Mixture Distribution
- `ssd_qlnorm()`: Cumulative Distribution Function for Log-Normal Distribution
- `ssd_qmulti()`: Quantile Function for Multiple Distributions
- `ssd_qmulti_fitdists()`: Quantile Function for Multiple Distributions
- `ssd_qweibull()`: Cumulative Distribution Function for Weibull Distribution

See Also

`ssd_p` and `ssd_r`

Examples

```
ssd_qburrIII3(0.5)

ssd_qgamma(0.5)

ssd_qgompertz(0.5)

ssd_qinvpareto(0.5)

ssd_qlgumbel(0.5)

ssd_qllogis_llogis(0.5)

ssd_qllogis(0.5)

ssd_qlnorm_lnorm(0.5)

ssd_qlnorm(0.5)

# multi
ssd_qmulti(0.5, gamma.weight = 0.5, lnorm.weight = 0.5)

# multi fitdists
fit <- ssd_fit_dists(ssddata::ccme_boron)
ssd_qmulti_fitdists(0.5, fit)

ssd_qweibull(0.5)
```

`ssd_rburrIII3` *Random Number Generation*

Description

Random Number Generation

Usage

```
ssd_rburrIII3(n, shape1 = 1, shape2 = 1, scale = 1, chk = TRUE)

ssd_rgamma(n, shape = 1, scale = 1, chk = TRUE)

ssd_rgompertz(n, location = 1, shape = 1, chk = TRUE)

ssd_rinvpareto(n, shape = 3, scale = 1, chk = TRUE)

ssd_rlgumbel(n, locationlog = 0, scalelog = 1, chk = TRUE)

ssd_rllogis_llogis(
  n,
```

```
  locationlog1 = 0,
  scalelog1 = 1,
  locationlog2 = 1,
  scalelog2 = 1,
  pmix = 0.5,
  chk = TRUE
)

ssd_rllogis(n, locationlog = 0, scalelog = 1, chk = TRUE)

ssd_rlnorm_lnorm(
  n,
  meanlog1 = 0,
  sdlog1 = 1,
  meanlog2 = 1,
  sdlog2 = 1,
  pmix = 0.5,
  chk = TRUE
)

ssd_rlnorm(n, meanlog = 0, sdlog = 1, chk = TRUE)

ssd_rmulti(
  n,
  burrIII3.weight = 0,
  burrIII3.shape1 = 1,
  burrIII3.shape2 = 1,
  burrIII3.scale = 1,
  gamma.weight = 0,
  gamma.shape = 1,
  gamma.scale = 1,
  gompertz.weight = 0,
  gompertz.location = 1,
  gompertz.shape = 1,
  invpareto.weight = 0,
  invpareto.shape = 3,
  invpareto.scale = 1,
  lgumbel.weight = 0,
  lgumbel.locationlog = 0,
  lgumbel.scalelog = 1,
  llogis.weight = 0,
  llogis.locationlog = 0,
  llogis.scalelog = 1,
  llogis_llogis.weight = 0,
  llogis_llogis.locationlog1 = 0,
  llogis_llogis.scalelog1 = 1,
  llogis_llogis.locationlog2 = 1,
  llogis_llogis.scalelog2 = 1,
```

```
  llogis_llogis.pmix = 0.5,
  lnorm.weight = 0,
  lnorm.meanlog = 0,
  lnorm.sdlog = 1,
  lnorm_lnorm.weight = 0,
  lnorm_lnorm.meanlog1 = 0,
  lnorm_lnorm.sdlog1 = 1,
  lnorm_lnorm.meanlog2 = 1,
  lnorm_lnorm.sdlog2 = 1,
  lnorm_lnorm.pmix = 0.5,
  weibull.weight = 0,
  weibull.shape = 1,
  weibull.scale = 1,
  chk = TRUE
)

ssd_rmulti_fitdists(n, fitdists, chk = TRUE)

ssd_rweibull(n, shape = 1, scale = 1, chk = TRUE)
```

Arguments

`n`	positive number of observations.
`shape1`	shape1 parameter.
`shape2`	shape2 parameter.
`scale`	scale parameter.
`chk`	A flag specifying whether to check the arguments.
`shape`	shape parameter.
`location`	location parameter.
`locationlog`	location on the log scale parameter.
`scalelog`	scale on log scale parameter.
`locationlog1`	locationlog1 parameter.
`scalelog1`	scalelog1 parameter.
`locationlog2`	locationlog2 parameter.
`scalelog2`	scalelog2 parameter.
`pmix`	Proportion mixture parameter.
`meanlog1`	mean on log scale parameter.
`sdlog1`	standard deviation on log scale parameter.
`meanlog2`	mean on log scale parameter.
`sdlog2`	standard deviation on log scale parameter.
`meanlog`	mean on log scale parameter.
`sdlog`	standard deviation on log scale parameter.

`burrIII3.weight`	weight parameter for the Burr III distribution.
`burrIII3.shape1`	shape1 parameter for the Burr III distribution.
`burrIII3.shape2`	shape2 parameter for the Burr III distribution.
`burrIII3.scale`	scale parameter for the Burr III distribution.
`gamma.weight`	weight parameter for the gamma distribution.
`gamma.shape`	shape parameter for the gamma distribution.
`gamma.scale`	scale parameter for the gamma distribution.
`gompertz.weight`	weight parameter for the Gompertz distribution.
`gompertz.location`	location parameter for the Gompertz distribution.
`gompertz.shape`	shape parameter for the Gompertz distribution.
`invpareto.weight`	weight parameter for the inverse Pareto distribution.
`invpareto.shape`	shape parameter for the inverse Pareto distribution.
`invpareto.scale`	scale parameter for the inverse Pareto distribution.
`lgumbel.weight`	weight parameter for the log-Gumbel distribution.
`lgumbel.locationlog`	location parameter for the log-Gumbel distribution.
`lgumbel.scalelog`	scale parameter for the log-Gumbel distribution.
`llogis.weight`	weight parameter for the log-logistic distribution.
`llogis.locationlog`	location parameter for the log-logistic distribution.
`llogis.scalelog`	scale parameter for the log-logistic distribution.
`llogis_llogis.weight`	weight parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.locationlog1`	locationlog1 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.scalelog1`	scalelog1 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.locationlog2`	locationlog2 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.scalelog2`	scalelog2 parameter for the log-logistic log-logistic mixture distribution.
`llogis_llogis.pmix`	pmix parameter for the log-logistic log-logistic mixture distribution.

`lnorm.weight`	weight parameter for the log-normal distribution.
`lnorm.meanlog`	meanlog parameter for the log-normal distribution.
`lnorm.sdlog`	sdlog parameter for the log-normal distribution.
`lnorm_lnorm.weight`	weight parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.meanlog1`	meanlog1 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.sdlog1`	sdlog1 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.meanlog2`	meanlog2 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.sdlog2`	sdlog2 parameter for the log-normal log-normal mixture distribution.
`lnorm_lnorm.pmix`	pmix parameter for the log-normal log-normal mixture distribution.
`weibull.weight`	weight parameter for the Weibull distribution.
`weibull.shape`	shape parameter for the Weibull distribution.
`weibull.scale`	scale parameter for the Weibull distribution.
`fitdists`	An object of class fitdists.

Functions

- `ssd_rburrIII3()`: Random Generation for BurrIII Distribution
- `ssd_rgamma()`: Random Generation for Gamma Distribution
- `ssd_rgompertz()`: Random Generation for Gompertz Distribution
- `ssd_rinvpareto()`: Random Generation for Inverse Pareto Distribution
- `ssd_rlgumbel()`: Random Generation for log-Gumbel Distribution
- `ssd_rllogis_llogis()`: Random Generation for Log-Logistic/Log-Logistic Mixture Distribution
- `ssd_rllogis()`: Random Generation for Log-Logistic Distribution
- `ssd_rlnorm_lnorm()`: Random Generation for Log-Normal/Log-Normal Mixture Distribution
- `ssd_rlnorm()`: Random Generation for Log-Normal Distribution
- `ssd_rmulti()`: Random Generation for Multiple Distributions
- `ssd_rmulti_fitdists()`: Random Generation for Multiple Distributions
- `ssd_rweibull()`: Random Generation for Weibull Distribution

See Also

`ssd_p` and `ssd_q`

Examples

```
set.seed(50)
hist(ssd_rburrIII3(10000), breaks = 1000)

set.seed(50)
hist(ssd_rgamma(10000), breaks = 1000)

set.seed(50)
hist(ssd_rgompertz(10000), breaks = 1000)

set.seed(50)
hist(ssd_rinvpareto(10000), breaks = 1000)

set.seed(50)
hist(ssd_rlgumbel(10000), breaks = 1000)

set.seed(50)
hist(ssd_rllogis_llogis(10000), breaks = 1000)

set.seed(50)
hist(ssd_rllogis(10000), breaks = 1000)

set.seed(50)
hist(ssd_rlnorm_lnorm(10000), breaks = 1000)

set.seed(50)
hist(ssd_rlnorm(10000), breaks = 1000)

# multi
set.seed(50)
hist(ssd_rmulti(1000, gamma.weight = 0.5, lnorm.weight = 0.5), breaks = 100)

# multi fitdists
fit <- ssd_fit_dists(ssddata::ccme_boron)
ssd_rmulti_fitdists(2, fit)

set.seed(50)
hist(ssd_rweibull(10000), breaks = 1000)
```

`ssd_sort_data` *Sort Species Sensitivity Data*

Description

Sorts Species Sensitivity Data by empirical cumulative density (ECD).

Usage

```
ssd_sort_data(data, left = "Conc", right = left)
```

Arguments

`data`	A data frame.
`left`	A string of the column in data with the concentrations.
`right`	A string of the column in data with the right concentration values.

Details

Useful for sorting data before using `geom_ssdpoint()` and `geom_ssdsegment()` to construct plots for censored data with `stat = identity` to ensure order is the same for the various components.

Value

data sorted by the empirical cumulative density.

See Also

`ssd_ecd_data()` and `ssd_data()`

Examples

```
ssd_sort_data(ssddata::ccme_boron)
```

`ssd_wqg_bc` *Water Quality Guideline for British Columbia*

Description

Calculates the 5% Hazard Concentration for British Columbia after rescaling the data based on the log-logistic, log-normal and gamma distributions using the parametric bootstrap and AICc model averaging.

Usage

```
ssd_wqg_bc(data, left = "Conc")
```

Arguments

`data`	A data frame.
`left`	A string of the column in data with the concentrations.

Details

Returns a tibble the model averaged 5% hazard concentration with standard errors, 95% lower and upper confidence limits and the number of bootstrap samples as well as the proportion of bootstrap samples that successfully returned a likelihood (convergence of the bootstrap sample is not required).

Value

A tibble of the 5% hazard concentration with 95% confidence intervals.

See Also

`ssd_fit_dists()` and `ssd_hc()`

Other wqg: `ssd_wqg_burrlioz()`

Examples

```
## Not run:
ssd_wqg_bc(ssddata::ccme_boron)

## End(Not run)
```

`ssd_wqg_burrlioz` *Water Quality Guideline for Burrlioz*

Description

Calculates the 5% Hazard Concentration (after rescaling the data) using the same approach as Burrlioz based on 10,000 non-parametric bootstrap samples.

Usage

```
ssd_wqg_burrlioz(data, left = "Conc")
```

Arguments

`data`	A data frame.
`left`	A string of the column in data with the concentrations.

Details

Returns a tibble the model averaged 5% hazard concentration with standard errors, 95% lower and upper confidence limits and the number of bootstrap samples as well as the proportion of bootstrap samples that successfully returned a likelihood (convergence of the bootstrap sample is not required).

Value

A tibble of the 5% hazard concentration with 95% confidence intervals.

See Also

`ssd_fit_burrlioz()` and `ssd_hc_burrlioz()`

Other wqg: `ssd_wqg_bc()`

Examples

```
## Not run:
ssd_wqg_burrlioz(ssddata::ccme_boron)

## End(Not run)
```

---

stat_ssd *Plot Species Sensitivity Data* **[Deprecated]**

Description

Uses the empirical cumulative density/distribution to visualize species sensitivity data.

Usage

```
stat_ssd(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`geom`	The geometric object to use to display the data for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following: • A `Geom` ggproto subclass, for example `GeomPoint`.

	• A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`. • For more information and other ways to specify the geom, see the layer geom documentation.
`position`	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following: • The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`. • For more information and other ways to specify the position, see the layer position documentation.
`...`	Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept. • The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

See Also

geom_ssdpoint()

Examples

```
## Not run:
ggplot2::ggplot(ssddata::ccme_boron, ggplot2::aes(x = Conc)) +
  stat_ssd()

## End(Not run)
```

---

subset.fitdists *Subset fitdists Object*

Description

Select a subset of distributions from a fitdists object. The Akaike Information-theoretic Criterion differences are calculated after selecting the distributions named in select.

Usage

```
## S3 method for class 'fitdists'
subset(x, select = names(x), delta = Inf, ...)
```

Arguments

`x`	The object.
`select`	A character vector of the distributions to select.
`delta`	A non-negative number specifying the maximum absolute AIC difference cutoff. Distributions with an absolute AIC difference greater than delta are excluded from the calculations.
`...`	Unused.

Examples

```
fits <- ssd_fit_dists(ssddata::ccme_boron)
subset(fits, c("gamma", "lnorm"))
```

`tidy.fitdists` *Turn a fitdists Object into a Tibble*

Description

Turns a fitdists object into a tidy tibble of the estimates (est) and standard errors (se) by the terms (term) and distributions (dist).

Usage

```
## S3 method for class 'fitdists'
tidy(x, all = FALSE, ...)
```

Arguments

`x`	The object.
`all`	A flag specifying whether to also return transformed parameters.
`...`	Unused.

Value

A tidy tibble of the estimates and standard errors.

See Also

`coef.fitdists()`

Other generics: `augment.fitdists`(), `glance.fitdists`()

Examples

```
fits <- ssd_fit_dists(ssddata::ccme_boron)
tidy(fits)
tidy(fits, all = TRUE)
```

Index