Package ‘ssdtools’

May 15, 2022

Title  Species Sensitivity Distributions
Version  1.0.2

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 Weibull to censored and/or weighted data. 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

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

Depends  R (>= 4.1)
Imports  abind, chk (>= 0.7.0), doFuture, foreach, furrr, generics, ggplot2, goftest, graphics, grid, lifecycle, parallel, plyr, purrr, Rcpp, scales, ssddata, stats, stringr, tibble, TMB (>= 1.7.20), universals, utils, VGAM

Suggests  covr, dplyr, fitdistrplus, future, glue, grDevices, knitr, magrittr, mle.tools, R.rsp, readr, reshape2, rlang, rmarkdown, testthat, tidyr, tidyverse, withr

LinkingTo  Rcpp, TMB
VignetteBuilder  knitr, R.rsp
Config/testthat/edition  3
Encoding  UTF-8
Language  en-US
LazyData  true
RoxygenNote  7.2.0
NeedsCompilation  yes
Author  Joe Thorley [aut, cre, ctr] (<https://orcid.org/0000-0002-7683-4592>),
        Carl Schwarz [aut, ctr],
        Rebecca Fisher [ctb],
        David Fox [ctb],
        Angeline Tillmans [ctb],
        Ali Azizishirazi [ctb],
        Kathleen McTavish [ctb],
        Heather Thompson [ctb],
        Andy Teucher [ctb],
        Seb Dalgarno [ctb] (<https://orcid.org/0000-0002-3658-4517>),
        Emilie Doussantousse [ctb],
        Stephanie Hazlitt [ctb],
        Nadine Hussein [ctb],
        Nan-Hung Hsieh [ctb],
        Sergio Ibarra Espinosa [ctb],
        Province of British Columbia [cph],
        Environment and Climate Change Canada [cph]

Maintainer Joe Thorley <joe@poissonconsulting.ca>

Repository CRAN

Date/Publication 2022-05-14 23:50:02 UTC

R topics documented:

augment.fitdists ................................. 3
autoplot.fitdists ................................ 4
boron_data ........................................ 5
boron_pred ......................................... 6
ccme_data .......................................... 6
coeff.fitdists ..................................... 7
comma_signif ...................................... 8
dgompertz .......................................... 8
dist_data .......................................... 9
dlgumbel ........................................... 10
estimates.fitdists ................................. 10
geom_hcintersect ................................ 11
geom_ssdpoint .................................... 12
geom_ssdsegment ................................ 14
geom_xribbon ..................................... 15
glance.fitdists ................................... 17
is.fitdists ........................................ 18
is_censored ....................................... 18
predict.fitburrloz ................................ 19
predict.fitdists .................................. 20
scale_colour_ssd ................................ 21
ssdtools-ggproto ................................ 22
ssd_data .......................................... 22
ssd_dists ......................................... 23
**augment.fitdists**

**Augmented Data from fitdists Object**

### Description
Get a tibble of the original data with augmentation.

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

### Arguments
- `x`: The object.
- `...`: Unused.
Value
A tibble of the augmented data.

See Also
- `ssd_data()`
Other generics: `glance.fitdists()`, `tidy.fitdists()`

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

Description
A wrapper on `ssd_plot_cdf()`.

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

Arguments
- `object` The object.
- `...` Unused.

Value
A ggplot object.

See Also
- `ssd_plot_cdf()`

Examples
```r
fits <- ssd_fit_dists(ssddata::ccme_boron)
augment(fits)
```
boron_data

**CCME Species Sensitivity Data for Boron**

**Description**
Species Sensitivity Data from the Canadian Council of Ministers of the Environment. Please use `[ssddata::ccme_data]` instead.

**Usage**
boron_data

**Format**
An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 28 rows and 5 columns.

**Details**

The columns are as follows

- **Chemical** The chemical (chr).
- **Species** The species binomial name (chr).
- **Concentration** The chemical concentration (dbl).
- **Units** The units (chr).
- **Group** The taxonomic group (fctr).

**See Also**
- `ccme_data()`

Other boron: `boron_pred`

**Examples**
```r
head(ccme_data)
```
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 8 columns.

Details
- **percent**  The percent of species affected (int).
- **est**    The estimated concentration (dbl).
- **se**    The standard error of the estimate (dbl).
- **lcl**  The lower confidence limit (dbl).
- **ucl**  The upper confidence limit (dbl).
- **dist**  The distribution (chr).

See Also
Other boron: boron_data

Examples
head(boron_pred)

ccme_data

CCME Species Sensitivity Data

Description
Species Sensitivity Data from the Canadian Council of Ministers of the Environment. The taxonomic groups are Amphibian, Fish, Invertebrate and Plant. Plants includes freshwater algae. Please use [ssd::ccme_data] instead.

Usage
ccme_data
Format

An object of class tbl_df (inherits from tbl.data.frame) with 144 rows and 5 columns.

Details

Additional information on each of the chemicals is available from the CCME website.


**Chemical**  The chemical (chr).
**Species**  The species binomial name (chr).
**Conc**  The chemical concentration (dbl).
**Group**  The taxonomic group (fctr).
**Units**  The units (chr).

Examples

```r
head(ccme_data)
```

---

### coef.fitdists  

*Turn a fitdists Object into a Tidy Tibble*

**Description**

A wrapper on `tidy.fitdists()`.

**Usage**

```r
## 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**

By default the numeric vectors are first rounded to three significant figures. Then scales::comma is only applied to values greater than or equal to 1000 to ensure that labels are permitted to have different numbers of decimal places.

**Usage**

comma_signif(x, digits = 3, ...)

**Arguments**

- **x**
  A numeric vector to format.

- **digits**
  A whole number specifying the number of significant figures

- **...**
  Additional arguments passed to scales::comma.

**Value**

A character vector.

**Examples**

comma_signif(c(0.1, 1, 10, 1000))
scales::comma(c(0.1, 1, 10, 1000))

---

**dgompertz**

*Gompertz Probability Density*

**Description**

[Deprecated]

**Usage**

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

- **x**: A numeric vector of values.
- **location**: location parameter on the log scale.
- **shape**: 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**

<table>
<thead>
<tr>
<th>dist_data</th>
</tr>
</thead>
</table>

#### 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_all()`, `ssd_dists()`

#### Examples

dist
**dlgumbel**  
*Log-Gumbel (Inverse Weibull) Probability Density*

**Description**

[Deprecated]

**Usage**

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

**Arguments**

- `x`: A numeric vector of values.
- `locationlog`: location on 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 values by distribution and term.

**Usage**

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

**Arguments**

- `x`: The object.
- `...`: Unused.

**Value**

A named list of the estimates.

**See Also**

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

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

---

### Description

Plots the intersection between each xintercept and yintercept value.

### Usage

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

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapping</td>
<td>Set of aesthetic mappings created by <code>aes()</code> or <code>aes()</code>. If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply <code>mapping</code> if there is no plot mapping.</td>
</tr>
<tr>
<td>data</td>
<td>The data to be displayed in this layer. There are three options:</td>
</tr>
<tr>
<td></td>
<td>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code>.</td>
</tr>
<tr>
<td></td>
<td>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created.</td>
</tr>
<tr>
<td></td>
<td>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</td>
</tr>
<tr>
<td></td>
<td>Other arguments passed on to <code>layer()</code>. These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = &quot;red&quot;</code> or <code>size = 3</code>. They may also be parameters to the paired geom/stat.</td>
</tr>
<tr>
<td>xintercept</td>
<td>The x-value for the intersect</td>
</tr>
<tr>
<td>yintercept</td>
<td>The y-value for the intersect</td>
</tr>
</tbody>
</table>
If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

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)

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
)

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()` or `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, as a string.

position  
Position adjustment, either as a string, or the result of a call to a position adjustment function.

...  
Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

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()`.

Details

details

geom_ssd() has been deprecated for geom_ssdpoint().

Functions

- geom_ssd: Species Sensitivity Data Points
  Uses the empirical cumulative distribution to create scatterplot of points x. [Deprecated]

See Also

- `ssd_plot_cdf()`

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

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

## Not run:
## Not run:

## End(Not run)
```

---

### geom_ssdsegment

**Species Sensitivity Censored Segments**

#### Description

Uses the empirical cumulative distribution to draw lines between points \( x \) and \( x_{\text{end}} \).

#### Usage

```r
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()` or `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()`.
  - If `data` is a `data.frame`, it 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_xribbon**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>stat</strong></td>
<td>The statistical transformation to use on the data for this layer, as a string.</td>
</tr>
<tr>
<td><strong>position</strong></td>
<td>Position adjustment, either as a string, or the result of a call to a position adjustment function.</td>
</tr>
<tr>
<td>...</td>
<td>Other arguments passed on to <code>layer()</code>. These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = &quot;red&quot;</code> or <code>size = 3</code>. They may also be parameters to the paired geom/stat.</td>
</tr>
<tr>
<td><strong>arrow</strong></td>
<td>Specification for arrow heads, as created by <code>arrow()</code>.</td>
</tr>
<tr>
<td><strong>arrow.fill</strong></td>
<td>Fill colour to use for the arrow head (if closed). NULL means use colour aesthetic.</td>
</tr>
<tr>
<td><strong>lineend</strong></td>
<td>Line end style (round, butt, square).</td>
</tr>
<tr>
<td><strong>linejoin</strong></td>
<td>Line join style (round, mitre, bevel).</td>
</tr>
<tr>
<td><strong>na.rm</strong></td>
<td>If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.</td>
</tr>
<tr>
<td><strong>show.legend</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>inherit.aes</strong></td>
<td>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. <code>borders()</code>.</td>
</tr>
</tbody>
</table>

**See Also**

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

**Examples**

```r
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() or 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, as a string.

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

... Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

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 = percent))

---

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

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()
Examples

defits <- ssd_fit_dists(ssddata::ccme_boron)
is_censored(fits)


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

```r
## S3 method for class 'fitburrlioz'
predict(
  object,
  percent = 1:99,
  ci = FALSE,
  level = 0.95,
  nboot = 1000,
  min_pboot = 0.99,
  parametric = TRUE,
  ...
)
```

Arguments

- `object`: The object.
- `percent`: A numeric vector of percentages.
- `ci`: A flag specifying whether to estimate confidence intervals (by parametric bootstrapping).
- `level`: A number between 0 and 1 of the confidence level.
- `nboot`: A count of the number of bootstrap samples to use to estimate the se and confidence limits. A value of 10000 is recommended for official guidelines.
- `min_pboot`: A number of the minimum proportion of bootstrap samples that must successfully fit in the sense of returning a likelihood.
- `parametric`: A flag specifying whether to perform parametric as opposed to non-parametric bootstrapping.
- `...`: Unused.

Details

It is useful for plotting purposes.
predict.fitdists

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 = 1:99,
  ci = FALSE,
  level = 0.95,
  nboot = 1000,
  average = TRUE,
  delta = 7,
  min_pboot = 0.99,
  parametric = TRUE,
  control = NULL,
  ...
)

Arguments

object  The object.

percent  A numeric vector of percentages.

ci  A flag specifying whether to estimate confidence intervals (by parametric bootstrapping).

level  A number between 0 and 1 of the confidence level.

nboot  A count of the number of bootstrap samples to use to estimate the se and confidence limits. A value of 10000 is recommended for official guidelines.

average  A flag specifying whether to model average the estimates.

delta  A non-negative number specifying the maximum absolute Akaike Information-theoretic Criterion difference cutoff. Distributions with an absolute difference from the best model greater than the cutoff are excluded.
scale_colour_ssd

- **min_pboot**: A number of the minimum proportion of bootstrap samples that must successfully fit in the sense of returning a likelihood.
- **parametric**: A flag specifying whether to perform parametric as opposed to non-parametric bootstrapping.
- **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**

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

---

**scale_colour_ssd**

*Discrete color-blind scale for SSD Plots*

**Description**

Discrete color-blind scale for SSD Plots

**Usage**

```r
scale_colour_ssd(...)  
scale_color_ssd(...)  
```

**Arguments**

```r
...  
```

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

**Functions**

- `scale_color_ssd`: Discrete color-blind scale for SSD Plots

**See Also**

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

**Examples**

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

Description

Data from fitdists Object

Usage

ssd_data(x)
ssd_dists

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)

Description

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

Usage

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

Arguments

bcanz

tails

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.

A flag or NULL specifying whether to only include distributions with both tails.

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**

```r
ssd_dists_all()
```

**Value**

A unique, sorted character vector of the distributions.

**See Also**

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

**Examples**

```r
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**

```r
ssd_dists_bcanz()
```

**Value**

A unique, sorted character vector of the distributions.

**See Also**

`ssd_dists()`

**Examples**

```r
ssd_dists_bcanz()
```
**ssd_ecd**

*Empirical Cumulative Density*

**Description**

Empirical Cumulative Density

**Usage**

```r
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**

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

---

**ssd_ecd_data**

*Empirical Cumulative Density for Species Sensitivity Data*

**Description**

Empirical Cumulative Density for Species Sensitivity Data

**Usage**

```r
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

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

---

<table>
<thead>
<tr>
<th>ssd_exposure</th>
<th>Percent Exposure</th>
</tr>
</thead>
</table>

Description

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

Usage

```r
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

```r
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)
```

ssd_fit_bcanz  Fit BCANZ Distributions

Description

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

Usage

```r
ssd_fit_bcanz(data, left = "Conc")
```

Arguments

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

Value

An object of class fitdists.

See Also

- `ssd_fit_dists()`
- Other BCANZ: `ssd_hc_bcanz()`

Examples

```r
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, 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 largest finite value.

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)
**Usage**

```r
ssd_fit_dists(
  data,
  left = "Conc",
  right = left,
  weight = NULL,
  dists = ssd_dists_bcanz(),
  nrow = 6L,
  rescale = FALSE,
  reweight = FALSE,
  computable = TRUE,
  at_boundary_ok = FALSE,
  min_pmix = 0,
  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 largest finite value.
- `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).
- `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`: A numeric vector of length two of the lower and upper bounds for the 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 'llogis', 'gamma' and 'lnorm' distributions are fitted to the data. For a complete list of the implemented distributions 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

```r
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.
### ssd_gof

#### Usage

```r
ssd_gof(x, ...)  
```

#### 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)

- `fitdists`: Goodness of Fit

#### See Also

- `glance.fitdists()`

#### Examples

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

### ssd_hc

#### Description

Gets concentration(s) that protect specified percentage(s) of species.

#### Usage

```r
ssd_hc(x, ...)  
```

#### Arguments

- `percent`: The percentage(s) of species to protect.
- `hc`: The hazard concentration.

#### Examples

```r
fits <- ssd_fit_dists(ssddata::ccme_boron)  
ssd_hc(fits)  
ssd_hc(fits)
```
x, percent = 5, hc = 5, ci = FALSE, level = 0.95, nboot = 1000, average = TRUE, delta = 7, min_pboot = 0.99, parametric = TRUE, control = NULL, ... )

## S3 method for class 'fitburrloiz'
ssd_hc(
  x, percent = 5, ci = FALSE, level = 0.95, nboot = 1000, min_pboot = 0.99, parametric = FALSE, ...
)

### Arguments

- **x**: The object.
- **...**: Unused.
- **percent**: A numeric vector of percentages.
- **hc**: A whole numeric vector between 1 and 99 indicating the percent hazard concentrations (deprecated for percent).
- **ci**: A flag specifying whether to estimate confidence intervals (by parametric bootstrapping).
- **level**: A number between 0 and 1 of the confidence level.
- **nboot**: A count of the number of bootstrap samples to use to estimate the se and confidence limits. A value of 10000 is recommended for official guidelines.
- **average**: A flag specifying whether to model average the estimates.
- **delta**: A non-negative number specifying the maximum absolute Akaike Information-theoretic Criterion difference cutoff. Distributions with an absolute difference from the best model greater than the cutoff are excluded.
- **min_pboot**: A number of the minimum proportion of bootstrap samples that must successfully fit in the sense of returning a likelihood.
- **parametric**: A flag specifying whether to perform parametric as opposed to non-parametric bootstrapping.
- **control**: A list of control parameters passed to `stats::optim()`.
Details

If \( ci = \text{TRUE} \) uses parametric bootstrapping to get confidence intervals on the hazard concentrations(s).

Value

A tibble of corresponding hazard concentrations.

Methods (by class)

- list: Hazard Concentrations for Distributional Estimates
- fitdists: Hazard Concentrations for fitdists Object
- fitburrlioz: Hazard Concentrations for fitburrlioz Object'

See Also

predict.fitdists() and ssd_hp().

Examples

```r
fits <- ssd_fit_dists(ssddata::ccme_boron)
ssd_hc(fits)
ssd_hc(estimates(fits))
ssd_hc(ssd_match_moments())
fit <- ssd_fit_burrlioz(ssddata::ccme_boron)
ssd_hc(fit)
```

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 required 10,000 iterations.

Usage

```r
ssd_hc_bcanz(x, nboot = 10000, delta = 10, min_pboot = 0.9)
```
Arguments

x  The object.

nboot  A count of the number of bootstrap samples to use to estimate the se and confidence limits. A value of 10000 is recommended for official guidelines.

delta  A non-negative number specifying the maximum absolute Akaike Information-theoretic Criterion difference cutoff. Distributions with an absolute difference from the best model greater than the cutoff are excluded.

min_pboot  A number of the minimum proportion of bootstrap samples that must successfully fit in the sense of returning a likelihood.

Value

A tibble of corresponding hazard concentrations.

See Also

ssd_hc().

Other BCANZ: ssd_fit_bcanz()

Examples

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

ssd_hc_burrlioz  Hazard Concentrations for Burrlioz Fit

Description

Deprecated for ssd_hc().

Usage

ssd_hc_burrlioz(
  x,
  percent = 5,
  ci = FALSE,
  level = 0.95,
  nboot = 1000,
  min_pboot = 0.99,
  parametric = FALSE
)
ssd_hp

Arguments

- **x**: The object.
- **percent**: A numeric vector of percentages.
- **ci**: A flag specifying whether to estimate confidence intervals (by parametric bootstrapping).
- **level**: A number between 0 and 1 of the confidence level.
- **nboot**: A count of the number of bootstrap samples to use to estimate the se and confidence limits. A value of 10000 is recommended for official guidelines.
- **min_pboot**: A number of the minimum proportion of bootstrap samples that must successfully fit in the sense of returning a likelihood.
- **parametric**: A flag specifying whether to perform parametric as opposed to non-parametric bootstrapping.

Value

A tibble of corresponding hazard concentrations.

Examples

```r
fit <- ssd_fit_burrlioz(ssddata::ccme_boron)
ssd_hp_burrlioz(fit)
```

<table>
<thead>
<tr>
<th>Hazard Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

Description

Gets percent of species protected at specified concentration(s).

Usage

```r
ssd_hp(x, ...)
```

## S3 method for class 'fitdists'
```r
ssd_hp( 
  x,
  conc,
  ci = FALSE,
  level = 0.95,
  nboot = 1000,
  average = TRUE,
  delta = 7,
  min_pboot = 0.99,
  parametric = TRUE,
)```
Arguments

x The object.
...
conc A numeric vector of concentrations.

Arguments

x The object.
...
conc A numeric vector of concentrations.

conc A numeric vector of concentrations.

ci A flag specifying whether to estimate confidence intervals (by parametric bootstrapping).

level A number between 0 and 1 of the confidence level.

nboot A count of the number of bootstrap samples to use to estimate the se and confidence limits. A value of 10000 is recommended for official guidelines.

average A flag specifying whether to model average the estimates.

delta A non-negative number specifying the maximum absolute Akaike Information-theoretic Criterion difference cutoff. Distributions with an absolute difference from the best model greater than the cutoff are excluded.

min_pboot A number of the minimum proportion of bootstrap samples that must successfully fit in the sense of returning a likelihood.

parametric A flag specifying whether to perform parametric as opposed to non-parametric bootstrapping.

control A list of control parameters passed to `stats::optim()`.

Details

If `ci = TRUE` uses parameteric bootstrapping to get confidence intervals on the hazard percent(s).

Value

A tibble of corresponding hazard percents.

Methods (by class)

- `fitdists`: Hazard Percents for fitdists Object

See Also

`ssd_hc()`

Examples

```r
fits <- ssd_fit_dists(ssddata::ccme_boron)
ssd_hp(fits, conc = 1)
```
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_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_pal**  
*Color-blind Palette for SSD Plots*

**Description**  
Color-blind Palette for SSD Plots

**Usage**

```r
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**

```r
ssd_pal()
```

---

**ssd_pburrIII3**  
*Cumulative Distribution Function*

**Description**

Cumulative Distribution Function

**Usage**

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

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

```r
ssd_pgompertz(q, location = 1, shape = 1, lower.tail = TRUE, log.p = FALSE)
```
pgompertz(q, llocation = 0, lshape = 0, 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
)

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_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 \( P[X \leq x] \), otherwise, \( P[X > x] \).
log.p logical; if TRUE, probabilities \( p \) are given as \( \log(p) \).
shape A string of the column in data for the shape aesthetic.
location location parameter.
locationlog location parameter on the log scale.
locationlog1 locationlog1 parameter.
locationlog2 locationlog2 parameter.
scalelog scale on log scale parameter.
scalelog1 scalelog1 parameter.
scalelog2 scalelog2 parameter.

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
- `pgompertz`: Cumulative Distribution Function for Gompertz Distribution [Deprecated]
- `ssd_pinvpareto`: Cumulative Distribution Function for Inverse Pareto Distribution
- `ssd_plgumbel`: Cumulative Distribution Function for Log-Gumbel Distribution
- `plgumbel`: Cumulative Distribution Function for Log-Gumbel Distribution [Deprecated]
- `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_pweibull`: Cumulative Distribution Function for Weibull Distribution

See Also

`ssd_q` and `ssd_r`
Examples

ssd_pburrIII3()
ssd_pgamma()
ssd_pgomPERTZ()
ssd_pinvpareto()
ssd_plgumbel()
ssd_pllogis_llogis()
ssd_pllogis()
ssd_plnorm_lnorm()
ssd_plnorm()
ssd_pweibull()

---

**ssd_plot**  
Plot Species Sensitivity Data and Distributions

**Description**

Plots species sensitivity data and distributions.

**Usage**

```r
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 = FALSE,
  hc = 5L,
  shift_x = 3,
)```
SSD plot

    bounds = c(left = 1, right = 1),
    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.
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.
ici A flag specifying whether to estimate confidence intervals (by parametric bootstrapping).
ribbon A flag indicating whether to plot the confidence interval as a grey ribbon as opposed to green solid lines.
hc A count between 1 and 99 indicating the percent hazard concentration (or NULL).
shift_x The value to multiply the label x values by.
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.

See Also

ssd_plot_cdf() and geom_ssdpoint()

Examples

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

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

Usage

```r
ssd_plot_cdf(x, ...)
## S3 method for class 'fitdists'
ssd_plot_cdf(x, average = FALSE, delta = 7, ...)
## 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 model average the estimates.
- `delta` A non-negative number specifying the maximum absolute Akaike Information-theoretic Criterion difference cutoff. Distributions with an absolute difference from the best model greater than the cutoff are excluded.

Methods (by class)

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

See Also

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

Examples

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

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

### Cullen and Frey Plot

**Description**

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

**Usage**

```r
ssd_plot_cf(data, left = "Conc")
ssd Cfplot(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()`.

**Functions**

- `ssd_cfplot`: Defunct Cullen and Frey Plot

**Examples**

```r
ssd_plot_cf(ssddata::ccme_boron)
```

---

**ssd_plot_data**

### Plot Species Sensitivity Data

**Description**

Plots species sensitivity data.
Usage

```r
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,
  bounds = c(left = 1, right = 1),
  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.
- `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.
- `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.
- `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

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

Description

Quantile Function

Usage

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

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

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

```r
qgompertz(p, location = 0, shape = 0, lower.tail = TRUE, log.p = FALSE)
```

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

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

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

```r
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_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 \( P[X \leq x] \); otherwise, \( P[X > x] \).
- **log.p** logical; if TRUE, probabilities \( p \) are given as \( \log(p) \).
- **shape** A string of the column in data for the shape aesthetic.
- **location** location parameter.
- **locationlog** location parameter on the log scale.
- **lshape** shape parameter on the log scale.
- **locationlog1** locationlog1 parameter.
- **scalelog** scale on log scale parameter.
- **locationlog2** locationlog2 parameter.
- **scalelog1** scalelog1 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.
Functions

- `ssd_qburrIII3`: Quantile Function for BurrIII Distribution
- `ssd_qgamma`: Quantile Function for Gamma Distribution
- `ssd_qgompertz`: Quantile Function for Gompertz Distribution
- `qgompertz`: Quantile Function for Gompertz Distribution [Deprecated]
- `ssd_qinvpareto`: Quantile Function for Inverse Pareto Distribution
- `ssd qlgumbel`: Quantile Function for Log-Gumbel Distribution
- `qlgumbel`: Quantile Function for Log-Gumbel Distribution [Deprecated]
- `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 qweibull`: Cumulative Distribution Function for Weibull Distribution

See Also

`ssd_p` and `ssd_r`

Examples

```r
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)
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)

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

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

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

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

ssd_rlllogis_llogis(
  n,
  locationlog1 = 0,
  scalelog1 = 1,
  locationlog2 = 1,
  scalelog2 = 1,
  pmix = 0.5,
  chk = TRUE
)

ssd_rlllogis(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_rweibull(n, shape = 1, scale = 1, chk = TRUE)

Arguments

- **n**: number of observations.
- **shape1**: shape1 parameter.
- **shape2**: shape2 parameter.
- **scale**: scale parameter.
- **chk**: A flag specifying whether to check the arguments.
- **shape**: A string of the column in data for the shape aesthetic.
- **location**: location parameter.
- **location**: location parameter on the log scale.
- **lshape**: shape parameter on the log scale.
- **locationlog**: location on log scale parameter.
- **scalelog**: scale on log scale parameter.
- **locationlog1**: locationlog1 parameter.
- **scalelog1**: scalelog1 parameter.
- **locationlog2**: locationlog2 parameter.
- **scalelog2**: scalelog2 parameter.
- **p mix**: 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.

Functions

- **ssd_rburrIII3**: Random Generation for BurrIII Distribution
- **ssd_rgamma**: Random Generation for Gamma Distribution
- **ssd_rgompertz**: Random Generation for Gompertz Distribution
- **rgompertz**: Random Generation for Gompertz Distribution [Deprecated]
- **ssd_rinvpareto**: Random Generation for Inverse Pareto Distribution
- **ssd_rlgumbel1**: Random Generation for log-Gumbel Distribution
- **rlgumbel1**: Random Generation for log-Gumbel Distribution [Deprecated]
- **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_rweibull**: Random Generation for Weibull Distribution
See Also

`ssd_p` and `ssd_q`

Examples

```r
set.seed(50)
hist(ssd_rburrIII(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_rlllogis_llogis(10000), breaks = 1000)

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

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**

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

---

### 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
```r
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

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

Navigation

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

```r
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

```r
## Not run:
ssd_wqg_burrlioz(ssdata::ccme_boron)

## End(Not run)
```

---

### stat_ssd

#### Plot Species Sensitivity Data

#### Description

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

#### Usage

```r
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()` or `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 display the data
- **position**: Position adjustment, either as a string, or the result of a call to a position adjustment function.
Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

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.

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

```r
## 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**

```r
## 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 Akaike Information-theoretic Criterion difference cutoff. Distributions with an absolute difference from the best model greater than the cutoff are excluded.
- `...`  
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

coeff.fitdists()

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

Examples

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

* **BCANZ**
  ssd_fit_bcanz, 27
  ssd_HC_bcanz, 33

* **boron**
  boron_data, 5
  boron_pred, 6

* **datasets**
  boron_data, 5
  boron_pred, 6
  ccme_data, 6
  dist_data, 9
  ssdtools-ggproto, 22

* **dists BCANZ**
  ssd_dists_bcanz, 24

* **dists**
  dist_data, 9
  ssd_dists, 23
  ssd_dists_all, 24

* **generics**
  augment.fitdists, 3
  glance.fitdists, 17
  tidy.fitdists, 57

* **ggplot2**
  stat_ssd, 55

* **ggplot**
  geom_hcintersect, 11
  geom_ssdpoint, 12
  geom_ssdsegment, 14
  geom_xribbon, 15
  scale_colour_ssd, 21
  ssd_pal, 39

* **wq**
  ssd_wq_bc, 53
  ssd_wq_burrloiz, 54

aes(), 11, 13, 14, 16, 55
aes_(), 11, 13, 14, 16, 55
augment.fitdists, 3, 17, 57
augment.fitdists(), 23
autoplot.fitdists, 4
borders(), 13, 15, 16, 56
boron_data, 5, 6
boron_pred, 5, 6
ccme_data, 6
ccme_data(), 5
coef.fitdists, 7
coef.fitdists(), 57
comma_signif, 8
dgompertz, 8
dist_data, 9, 23, 24
dlgumbel, 10
estimates.fitdists, 10
estimates.fitdists(), 38, 44
fitdistrplus::descdist(), 45
fortify(), 11, 13, 14, 16, 55
geom_hcintersect, 11, 13, 15, 16, 21, 39
geom_ssd (geom_ssdpoint), 12
geom_ssdpoint, 12, 12, 15, 16, 21, 39
geom_ssdpoint(), 43, 46, 53, 56
geom_ssdsegment, 12, 13, 14, 16, 21, 39
geom_ssdsegment(), 53
geom_xribbon, 12, 13, 15, 15, 21, 39
GeomHcintersect (ssdtools-ggproto), 22
GeomSsdpoint (ssdtools-ggproto), 22
GeomSsdsegment (ssdtools-ggproto), 22
GeomXribbon (ssdtools-ggproto), 22
ggplot(), 11, 13, 14, 16, 55
ggplot2::discrete_scale(), 21
ggplot2::ggproto(), 22
glance.fitdists, 4, 17, 57
glance.fitdists(), 31
is.fitdists, 18
is_censored, 18
layer(), 11, 13, 15, 16, 56
INDEX

pgompertz (ssd_pburrIII3), 39
plgumbel (ssd_pburrIII3), 39
predict.fitburrlioz, 19
predict.fitdists, 20
predict.fitdists(), 33
qgompertz (ssd_qburrIII3), 47
qlgumbel (ssd_qburrIII3), 47
rgompertz (ssd_rburrIII3), 50
rlgumbel (ssd_rburrIII3), 50

scale_color_ssd (scale_colour_ssd), 21
scale_colour_ssd, 12, 13, 15, 16, 21, 39
scales::comma, 8
ssd_cfplot (ssd_plot_cf), 45
ssd_data, 22
ssd_data(), 4, 6, 26, 53
ssd_dists, 9, 23, 24
ssd_dists(), 24
ssd_dists_all, 9, 23, 24
ssd_dists_all(), 30
ssd_dists_bcanz, 24
ssd_ecd, 25
ssd_ecd(), 26
ssd_ecd_data, 25
ssd_ecd_data(), 23, 53
ssd_exposure, 26
ssd_fit_bcanz, 27, 34
ssd_fit_burrlioz, 28
ssd_fit_burrlioz(), 54
ssd_fit_dists, 28
ssd_fit_dists(), 27, 28, 54
ssd_gof, 30
ssd_gof(), 17
ssd_hc, 31
ssd_HC(), 10, 19–21, 30, 34, 36, 38, 54
ssd_hc_bcanz, 27, 33
ssd_hc_burrlioz, 34
ssd_hc_burrlioz(), 54
ssd_hp, 35
ssd_hp(), 33
ssd_is_censored, 37
ssd_is_censored(), 18
ssd_match_moments, 38
ssd_match_moments(), 10, 44
ssd_p, 49, 52
ssd_p (ssd_pburrIII3), 39
ssd_pal, 12, 13, 15, 16, 21, 39
ssd_pburrIII3, 39
ssd_pgamma (ssd_pburrIII3), 39
ssd_pgeom (ssd_pburrIII3), 39
ssd_pinvpareto (ssd_pburrIII3), 39
ssd_plgumbel (ssd_pburrIII3), 39
ssd_pllogis (ssd_pburrIII3), 39
ssd_pllogis_llogis (ssd_pburrIII3), 39
ssd_plnorm (ssd_pburrIII3), 39
ssd_plnorm_lnorm (ssd_pburrIII3), 39
ssd_plot, 42
ssd_plot, 20, 21, 44, 46
ssd_plot_cdf, 44
ssd_plot_cdf(), 4, 10, 12, 13, 15, 16, 22, 30, 38, 43
ssd_plot_cdf(), 4, 50
ssd_plot_data, 45
ssd_pweibull (ssd_pburrIII3), 39
ssd_q, 41, 52
ssd_q (ssd_qburrIII3), 47
ssd_qburrIII3, 47
ssd_qgamma (ssd_qburrIII3), 47
ssd_qgompertz (ssd_qburrIII3), 47
ssd_qinvpareto (ssd_qburrIII3), 47
ssd_qlgumbel (ssd_qburrIII3), 47
ssd_qlogis (ssd_qburrIII3), 47
ssd_qlogis_llogis (ssd_qburrIII3), 47
ssd_qlnorm (ssd_qburrIII3), 47
ssd_qlnorm_lnorm (ssd_qburrIII3), 47
ssd_qweibull (ssd_qburrIII3), 47
ssd_r, 41, 49
ssd_r (ssd_rburrIII3), 50
ssd_rburrIII3, 50
ssd_rgamma (ssd_rburrIII3), 50
ssd_rgeom (ssd_rburrIII3), 50
ssd_rinvpareto (ssd_rburrIII3), 50
ssd_rlgumbel (ssd_rburrIII3), 50
ssd_rlogis (ssd_rburrIII3), 50
ssd_rlogis_llogis (ssd_rburrIII3), 50
ssd_rlnorm (ssd_rburrIII3), 50
ssd_rlnorm_lnorm (ssd_rburrIII3), 50
ssd_rweibull (ssd_rburrIII3), 50
ssd_sort_data, 52
ssd_sort_data(), 23
ssd_wag_bc, 53, 54
ssd_wag_burrlioz, 54, 54
ssdtools_ggproto, 22
stat_ssd, 55
stats::optim(), 21, 29, 32, 36
StatSsdpoint (ssdtools-ggproto), 22
StatSsdsegment (ssdtools-ggproto), 22
subset.fitdists, 56

tidy.fitdists, 4, 17, 57
tidy.fitdists(), 7, 10