





**An Environmetrics Australia, Australian Institute of Marine Science and Poisson Consulting Report**

# Final report of the joint investigation into SSD modelling and ssdtools implementation for the derivation of toxicant guidelines values in Australia and New Zealand

**19 April 2024**

www.environmetrics.net.au
www.aims.gov.au
www.poissonconsulting.ca

*This page intentionally blank*

**Environmetrics Australia**

PO Box 7117
Beaumaris VIC, 3193
+61-3-9018-7121
www.environmetrics.net.au

**Australian Institute of Marine Science**

PMB No 3
Townsville MC Qld 4810

PO Box 41775
Casuarina NT 0811

Indian Ocean Marine Research Centre
University of Western Australia, M096
Crawley WA 6009

This report should be cited as:

*Fox D.R., Fisher R., Thorley J.L. (2024) Final report of the joint investigation into SSD modelling and ssdtools implementation for the derivation of toxicant guidelines values in Australia and New Zealand. Report prepared for the Department of Climate Change, Energy, the Environment and Water. Environmetrics Australia, Beaumaris, Vic and the Australian Institute of Marine Science, Perth, WA. (183 pp).*



**DISCLAIMER**

While reasonable efforts have been made to ensure that the contents of this document are factually correct, neither Environmetrics Australia nor AIMS make any representation or give any warranty regarding the accuracy, completeness, currency or suitability for any particular purpose of the information or statements contained in this document. To the extent permitted by law Environmetrics Australia and/or AIMS shall not be liable for any loss, damage, cost or expense that may be occasioned directly or indirectly through the use of or reliance on the contents of this document.

| Project Leader shall ensure that documents have been fully checked and approved prior to submittal to client | | | | |
|---|---|---|---|---|
| **Revision History**: | | *Name* | *Date* | *Comments* |
| 0 | Prepared by: | *David Fox, Rebecca Fisher, and Joe Thorley* | *12/04/2024* | |
| | Reviewed by: | *Ross Jones* | *16/04/2024* | |
| | Approved by: | *Claire Streten* | *19/4/2024* | |
| 1 | Prepared by: | *David Fox, Rebecca Fisher, and Joe Thorley* | *16/05/2024* | |

# Table of Contents

**TABLE OF FIGURES**

# 1. EXECUTIVE SUMMARY AND RECOMMENDATIONS

This report summarises the results of the final set of investigations, additional studies, and software fixes prior to the release of `ssdtools 2.0` and companion on-line implementation, `shinyssdtools`. It is intended that the release of `ssdtools 2.0` will coincide with its adoption by Australian, New Zealand, and Canadian jurisdictions as the recommended software tool for establishing default guideline values (DGVs) for concentrations of contaminants in natural aquatic systems. This signifies the achievement the collaborative project's over-riding objective of harmonising the statistical and computational basis of guideline value derivation across all three jurisdictions. In the context of SSD modelling, we believe this is the first time such collaboration and coordination among non-EU countries has occurred.

Over the past 4.5 years of this project, the core development team has made numerous changes, enhancements, and updates to the `ssdtools` software resulting in a tool that is easy to use, incorporates the latest developments in SSD modelling, and is computationally stable and efficient. This has required many decisions to be made along the way and while we are confident that those decisions have been based on sound science supported with well documented study results, future decisions regarding technical, computational, and functional issues will be made by a newly established Technical Advisory Group (TAG) whose membership introduces a wider range of interests and representation across the three jurisdictions.

A listing and brief description of all 8 tasks comprising this final phase of the collaborative project are listed in the table below.

| Category | Task ID | Description |
|---|---|---|
| **Software (S)** | **S1** | Explore and implement computationally efficient methods for obtaining confidence intervals for model-averaged HCx values. |
| | **S2** | Output customisation: improvements to plot formatting, options, report production. |
| | **S3** | Resolve the hc() - hp() inconsistency in ssdtools. |
| | **S4** | Documentation updates: new and revised help material, vignettes, new feature summary. |
| | **S5** | Investigations into stability issues with distribution fitting and confirmation of final set of stable distributions. |
| | **S6** | Fix hc() and hp() documentation |
| **Mathematical/Statistical (M)** | **M1** | Integrity checks; review decision rules for minimum data requirements; other misc. |
| | **M2** | Investigate and resolve convergence issues with lnorm-lnorm mixture distribution. |

## *Recommendations*

### *Recommendation #1*

*The CRAN-version of* `ssdtools` *be updated to incorporate the new functions and associated* `uniroot` *procedures for model averaged HCx and HPx estimation that consider the model set as a joint probability distribution with a mixing proportion based on AIC model weights, thereby resolving the* `hc()-hp()` *inconsistency identified in Task S3. These functions include:* `ssd_qmulti()`, `ssd_pmulti()`,`ssd_rmulti()`; `ssd_dmulti()`.

### *Recommendation #2*

*Subsequent releases of* `ssdtools` *use the* `weighted_sample` *method to determine confidence intervals for the HCx estimated by the procedures specified in Recommendation #1.*

### *Recommendation #3*

*To ensure reliable convergence when fitting a lognormal mixture model the minimum bound for the mixing proportion (*`pmix`*) should be set at 3/N where N is the sample size of the input data.*

### *Recommendation #4*

*The final set of 'default' distributions remains unchanged and consists of:*

- *Gamma*
- *Log-Gumbel*
- *Log-logistic*
- *Lognormal*
- *Lognormal_lognormal*
- *Weibull*

*And the distributions utilised in any analysis from this default set be subject to minimum sample size requirements, according to recommendation #5 below.*

### *Recommendation #5*

*The required minimum sample sizes are: **7** for two parameter distributions (Gamma, Log-Gumbel, Log-logistic, Lognormal and Weibull); **10** for three parameter distributions (Burr III, not a default distribution), and **16** for five parameter distributions (lognormal-lognormal, log-logistic – log-logistic). Adoption of these recommendations for use with* `ssd_fit_bcanz()` *function will require endorsement by the technical advisory group as this needs to be harmonised with the revised advice in Warne et al. (2018), and also be informed by CCME (2007). Given this recommendation represents a slight departure from previous sample-size advice, we further recommend that, if adopted, the revised sample-size requirements be applied prospectively and not retrospectively.*

### *Recommendation #6*

*The preferred minimum sample sizes are: **11** for two parameter distributions (Gamma, Log-Gumbel, Log-logistic, Lognormal and Weibull); **16** for three parameter distributions (Burr III, not a default distribution), and **26** for five parameter distributions (lognormal-lognormal, log-logistic – log-logistic). As for recommendation #5, this will require endorsement by the technical advisory group.*

### *Recommendation #7*

*Decisions regarding censoring options to be made available in* `ssdtools` *should be referred to the technical advisory group.*

# 2. INTRODUCTION

This final report is provided in fulfilment of the reporting requirements specified in Commonwealth Contract TM_2023_0499 between the Australian Government and Environmetrics Australia P/L and Contract ATM_2023_0469 between the Australian Government and the Australian Institute of Marine Science.

Contracts TM_2023_0499 and ATM_2023_0469 are for the provision of consultancy services to assist with the resolution and implementation of critical remaining technical details required to ensure the species sensitivity distribution (SSD) statistical modelling method based on the ssdtools R package and associated Shiny App is suitable for the generation of toxicant default guideline values (DGVs) for the Australian and New Zealand Guidelines for Fresh and Marine Water Quality.

This review of technical issues builds upon and formalises the collaborative research efforts of the following individuals and organisations: Professor David Fox (*Environmetrics Australia and the University of Melbourne*); Dr. Rebecca Fisher (*Australian Institute of Marine Science and University of Western Australia Oceans Institute and School of Plant Biology*); Dr. Carl Schwarz (*StatMathComp Consulting, Vancouver, BC, Canada, now retired*); and Dr. Joe Thorley (*Poisson Consulting, Nelson, BC, Canada*).

The following individuals and organisations have been closely involved and instrumental in the preparatory technical work leading up to this Australian-Canadian collaboration: Dr. Rick van Dam (*WQadvice*); Dr. Graeme Batley (*CSIRO Land and Water*); Dr. Angeline Tillmanns (*British Columbia Ministry of Environment and Climate Change Strategy*); and Doug Spry and Kathleen McTavish (*Environment and Climate Change Canada, Gatineau, Quebec, Canada*).

Yulia Cuthbertson, Tony Bigwood and Michael Antenucci from the Department of Climate Change, Energy, the Environment and Water have, and continue to provide administrative and contractual support.

Professor David Fox (Environmetrics Australia P/L.)
Dr. Rebecca Fisher (Australian Institute of Marine Science)
Dr. Joe Thorley (Poisson Consulting Ltd.)
19 April 2024

*This page intentionally blank*

# 3. BACKGROUND

The genesis of this 4.5 yr collaborative research effort dates to a workshop held between March 27–29, 2019 at the Australian Institute of Marine Science (AIMS) in Townsville, Queensland, to discuss key issues associated with the derivation of water quality guideline values (GVs). The workshop was attended by 14 national experts covering the fields of Ecotoxicology, Chemistry, statistical science, biostatistics and ecological modelling and aquatic ecology. The workshop arose following discussions in 2018 among some of the workshop attendees on potential improvements to GV derivation methods in use at the time, and an associated presentation at the SETAC Europe conference in Rome, in May 2018. The workshop report made 15 recommendations with respect to the need for more robust statistical techniques, investigations into 'new' modes of SSD modelling and analysis, software development, minimum data requirements, candidate models, and opportunities for collaboration (Fisher et al. 2019).

Later that year, Professor David Fox attended a three-day workshop held in the offices of Ministry of Environment and Climate Change Strategy, Victoria, BC, Canada (November 25-27, 2019) (Phase I). The workshop was attended by scientists, technical experts, and policymakers from Australia and Canada with the primary aim of identifying R&D priorities associated with methodologies for deriving water quality guideline values. All parties expressed a strong desire to contemplate a collaborative framework to facilitate these R&D efforts with the goal of harmonising methodological and computational approaches across jurisdictions – particularly with respect to the use of R, `ssdtools`, model-averaged SSDs, and `shinyssdtools`.

Discussions between Australian-New Zealand and Canadian jurisdictions continued throughout 2020 culminating in the awarding of Commonwealth contracts to Environmetrics Australia and AIMS in April 2021 to undertake the first tranche of work . This was conducted in collaboration with Canadian researchers to improve the statistical underpinnings and functionalities of the R package `ssdtools` and its associated shiny app (Phase II). Specific tasks included investigations into stability issues with the Burr III distribution, identification of benchmark datasets, refinement of statistical mixture models, identification of alternatives to bootstrapping for CI and HCx estimation, and considerations of the default distribution set. A comprehensive summary of this work was provided in the Project report (Fox et al. 2021) which recommended that the Australian-New Zealand and Canadian jurisdictions formally adopt the R package `ssdtools` (and its on-line implementation `shinysssdtools`) as the default software tools for fitting SSDs to toxicity data for the purpose of deriving 'safe' or 'protective' concentrations of chemicals in natural aquatic environments. Further investigations into the default distribution set, inconsistencies between `Burrlioz` and `ssdtools` output, numerical instabilities, and convergence issues were recommended.

Further contracts between the Commonwealth (represented by the Department of Climate Change, Energy, the Environment and Water) and Environmetrics Australia and AIMS were let in October 2023 (Phase III) to complete all outstanding investigations prior to formal adoption by the three jurisdictions and a major upgrade of the `ssdtools` package. The Phase III project comprised 8 Tasks broadly classified as 'Software Issues' and 'Mathematical/Statistical Issues'. This Final Report provides details of those investigations and a synthesis of all work undertaken.

# 4. Summary of Phase III investigations.

Table 1 summarises all Phase III investigations, including a description and the outcome.

**Table 1. Summary of preliminary investigations into issues associated with fitting Burr distributions and related activities**

| Task ID | Issue explored | Description | Actions/Outcome |
|---|---|---|---|
| S1 | Implement weighted bootstrap sample method for estimating CI | The current approach adopted in `ssdtools` for calculating model averaged confidence intervals based on a simple weighted arithmetic mean yields coverage values that fall substantially short of their notional 95% level, even at quite high samples sizes. A sample proportional to the weight of each distribution can be drawn such that the total bootstrap samples equals the number required. This will substantially speed up bootstrap based CI estimation, whilst still yielding a robust result. | Comparison of alternative CI estimation strategies showed that the weighted sample method previously identified, along with a newly developed method based on the functions and procedures implemented to resolve the `hc()-hp()` inconsistency (see S3) both result in similar CI values, with much better coverage than the weighted arithmetic mean currently implemented in ssdtools. The weighted sample method is faster and is therefore recommended for adoption. |
| S2 | Output customisation | Identify all the 'tweaks' to plot formatting (e.g. HCx/PCx, dashed lines at HCx-FA,font-sizes), labelling (toxicant name, units, data point label font/colour), axes scaling (set min/max, and ticks), statistical information reporting levels (e.g. x values for HCx), report generation options and formats (e.g. produce a PDF with user-specified elements), colour schemes etc. that would enhance user experience. | These customisations have been made, including a new draft Shiny App and pdf report suitable for use in DGV derivation. The French translations are being incorporated so that it can be made available in Canada. |
| S3 | Inconsistency between model averaged ssd_hc() and ssd_hp() | As it currently stands, if you use the `ssd_hc()` function to estimate an HCx and then use that HCx as the argument to the `ssd_hp()` function, the value returned is not x. This is an artefact of the way `ssdtools` currently computes the HCx | `ssdtools` was updated to incorporate the functions and associated `uniroot` procedures for HCx estimation: `ssd_qmulti()`, `ssd_pmulti()`,`ssd_rmulti()`; `ssd_dmulti()` thereby resolving the `hc()-hp()` inconsistency. |
| S4 | Update documentation | Add new material to outline and explain new features. Update vignette. | Additional documentation has been added to the `ssdtools` github website, including updated and detailed vignettes. |
| S5 | Review 'stable' and 'unstable' distributions | Should the invpareto be among the stable distributions? Note also other tasks associated with stability. The final set of "stable" distributions needs discussion and resolution. | All distributions showing instability were examined. The inverse Pareto will not be included based on theoretical grounds. The final set of distributions remains as originally identified. |
| S6 | Fix documentation for ssd_hc() and ssd_hp() | `ssd_hc()` and `ssd_hp()` are missing descriptions of arguments. | This has now been described in detail. |
| M1 | Integrity checks | Review and update/modify where necessary decision rules for minimum data requirements with and without mixtures; inclusion/exclusion rules based on data attributes (e.g. left and right censoring) | A review of the minimum data requirements was undertaken, with the recommendation that this should be set at $n \geq 3k+1$, where k is the number of estimated parameters. The preferred minimum should be set at $n \geq 5k+1$. |
| M2 | Convergence issues with lnorm-lnorm | Currently, to ensure reasonable behaviour `ssdtools` has lowered the `pboot` tolerance from 0.99 to 0.95 for the `bcanz` function to ensure the lnorm mixture is reliably included, because this does show instances of instability (a small percentage of failed bootstrap samples). This has not yet been investigated. In addition, we need to decide if it is necessary to implement a minimum sample size cut-off criterion for inclusion of this mixture. | Convergence in the lnorm-lnorm mixture were related to difficulty in estimating the mixing parameter (pmix) at the bounds of 0 and 1 on the logit scale. The `ssdtools` code was refactored to estimate `pmix` on its bounded natural 0-1 scale, which improved convergence by 20%. Issues remain with estimation of bounds at 0. A pragmatic solution is to set the bound at 3/N, which has been shown to resolve most convergence issues. |

## 4.1 DETAILED TASK SUMMARIES AND RESULTS

### 4.1.1 Task S1: Weighted bootstrap estimation for HCx confidence intervals

Tasks S1 (HCx interval estimation strategies evaluation) and S3 (HCx – FA inconsistency) are related. The stated aim of Task S1 was to address inadequacies (both mathematical and computational) of the current `ssdtools` methodology for obtaining confidence intervals for an HCx. As originally conceived, Task S1 would evaluate a 'weighted samples` method for this purpose. While computationally efficient, subsequent investigations revealed that this approach may yield biased estimates of an HCx. Accordingly, the scope of Task S1 was broadened to investigate and compare several alternative strategies for bootstrap estimation, particularly considering modifications to `ssdtools` to resolve the `hc()-hp()` inversion issues (section 4.1.2) for model average distributions. Here we outline pseudo code for the various possible estimation strategies that could be used for obtaining model-averaged HCx estimates. Because of the mathematical complexity and analytical intractability, evaluation of all methods could only be undertaken using computer simulations, and the adopted methodology is outlined below.

#### *Bootstrapping methods*

The ssdtools bootwtmerge branch `https://github.com/open-AIMS/ssdtools/tree/bootwtmerge (8853b3b)` was expanded to include four candidate averaging methods for estimating HC confidence interval:

*`averaging_method == "weighted_sample"`*

Method `weighted_sample` parametrically bootstraps from each of the set of distributions individually, taking a weighted sample (number of samples proportional to the relative weight of the distribution) from each, and combining these into a pooled bootstrap sample for estimation of CIs. This is the method that was compared to the original averaging method (see below) following Phase II of the `ssdtools` adoption project that indicated there was very poor coverage for the currently adopted averaging method in `ssdtools`.

PSEUDO CODE:

- For each distribution in the fitdists object, the proportional number of bootstrap samples to draw (`nboot_vals`) is found using `round(nboot * weight)`, where `nboot` is the total number of bootstrap samples and weight are the AICc based model weights for each distribution based on the original `ssd_fitdist` fit.
- For each of the `nboot_vals` for each distribution, a random sample of size N is drawn (the total number of original data points included in the original SSD fit) based on the estimated parameters from the original data for that distribution.
- The random sample is re-fitted using that distribution, and HCx is estimated from the re-fitted bootstrap fit.
- The HCx estimates for all `nboot_vals` for each distribution are then pooled across all distributions, and quantile is used to determine the lower and upper confidence bounds for this pooled weighted bootstrap sample of HCx values.

### *averaging_method == "uniroot"*

Method `uniroot` parametrically bootstraps from the complete set of distributions, by taking `nboot` bootstrap samples of size N from each distribution (based on the original estimated parameters), refitting each distribution independently, and using `uniroot` to estimate a joint HCx value, based on the original AICc based model weights.

PSEUDO CODE:

For each distribution in the fitdists object a new sample of size N (the total number of original data points included in the original SSD fit) is drawn, based on the original parameter estimates.

- Each distribution is re-fitted to the new bootstrap sample (independently).

Based on the new parameter estimates for each distribution at each iteration of `nboot`, `uniroot` is used to estimate the joint HCx value, using the original AICc model weights.

This is repeated a total of `nboot` times (total number of bootstrap samples), and quantile is used to determine the lower and upper confidence bounds from the resulting sample of HCx values.

### *averaging_method == "arithmetic" || averaging_method == "geometric"*

Method arithmetic is the currently implemented bootstrapping method in the CRAN version of `ssdtools`. This method takes a parametric bootstrap sample for each distribution separately, finds upper and lower confidence intervals independently and calculates a weighted arithmetic mean. Method geometric is equivalent to the arithmetic method, but uses a weighted geometric mean instead, as this is likely more appropriate given the skewed distributions represented by SSDs.

PSEUDO CODE:

For each distribution in the fitdists object a new sample of size N (the total number of original data points included in the original SSD fit) is drawn, based on the original parameter estimates.

- Each distribution is re-fitted to the new bootstrap sample (independently).

Based on the new parameter estimates for each distribution at each iteration of `nboot`, the HCx value is estimated.

This is repeated a total of `nboot` times (total number of bootstrap samples), and quantile is used to determine the lower and upper confidence bounds from the resulting sample of HCx values for each distribution independently.

- A weighted mean (arithmetic or geometric) is then applied to the upper and lower confidence intervals, to obtain model averaged confidence estimates.

### *The r_multi method*

A more theoretically correct way of obtaining ci and se values is to consider the model average set as a mixture distribution (see above, and the model averaging vignette, Appendix I). This method was implemented directly into the development branch of `ssdtools`. When we consider the model set as a mixture distribution, bootstrapping is achieved by resampling from the model set according to the AICc based model weights. A method for sampling from mixture distributions has been

implemented in ssdtools, via the function ssd_rmulti() which will generate random samples from a mixture of any combination of distributions currently implemented in ssdtools. Setting multi_ci = TRUE in the ssd_hc() call will ensure that bootstrap samples are drawn from a mixture distribution, instead of individual candidate distributions.

As mentioned above, when bootstrapping from the mixture distribution, the question arises whether the model weights should be re-estimated for every bootstrap sample or fixed at the values estimated from the models fitted to the original sample of toxicity data. Using simulation studies, we explored the coverage and bias of ci values obtained with and without fixing the distribution weights, and results indicate little difference.

If treating the distributions as a single mixture distribution when calculating model average confidence intervals (i.e. with multi_ci = TRUE), then setting weighted = FALSE specifies to use the original model weights. Setting weighted = TRUE will result in bootstrapping that will re-estimate weights for each bootstrap sample.

*Simulation study methodology*

A series of simulation studies (see 'Datasets' below) was used to test the following (revised) CI estimation strategies.

"weighted_sample", "uniroot", "arithmetic" || averaging_method = "geometric" (see section 4.1.2.1) as implemented on the 'bootwtmerge' branch on open-aims:

```
remotes::install_github('open-aims/ssdtools', ref = 'bootwtmerge', dependencies = TRUE)
```

and methods rmulti, and rmulti_fixed as implemented on the a development branch of ssdtools on bcgov: remotes::install_github('bcgov/ssdtools@b903b3d', dependencies = TRUE)

The bcgov/ssdtools@b903b3d version was also used to run the original method (implemented in the CRAN version 1.0.6 of ssdtools) and is equivalent to the arithmetic method from the bootwtmerge branch on open-aims. This was implemented to ensure there was consistency across the two versions of ssdtools.

Both methods rmulti and rmulti_fixed undertake bootstrapping using the new internal function ssd_rmulti() that generates random data from the *single* mixture *cdf* constructed from the candidate distributions and whose weights are determined from the model-averaged fits. This same distribution is re-fitted to the generated data using both rmulti and rmulti_fixed. The difference between the two strategies is that while rmulti re-estimates the AICc weights for each bootstrap sample rmulti_fixed fixes the weights to those used in the *cdf* from which the data were generated.

Note that in the release version ssdtools 2.0, the averaging method will be controlled by the ci_method argument in ssdtools, taking a character vector indicating the method.

The four possible values are multi_free (equivalent to rmulti) and multi_fixed (equivalent to rmulti_fixed) as well as weighted_arithmetic for the method used in the previous versions

and `weighted_samples` for the new default method which as described above takes a sample from each distribution proportional to the distributions weight..

### *Datasets*

Three different simulations were undertaken – one each for simulated datasets generated from the following *'parent'* distributions:

A. *Burr III distribution*

The Burr III was chosen as it is currently *not* in the candidate model set and is the only three-parameter, univariate distribution in `ssdtools`. Eight different Burr III datasets were simulated, based on those from the `ssddata` package that showed successful convergence of the Burr III using Burrlioz.

B. *Lognormal-lognormal mixture distribution*

This distribution was included as it provides an indication of how well the model averaging can represent data drawn from a mixture distribution. This distribution was not included in the candidate set during simulations due to stability issues that had yet to be resolved (See 4.1.8). A total of seven different datasets were simulated from lognormal-lognormal fits to the `ccme` datasets available in the `ssddata` package.

C. *log-logistic distribution*

This distribution was used to assess how well the different methods identified above perform for data simulated from a stable distribution and can therefore be included in the candidate set. Seven log-logistic datasets were simulated from log-logistic fits to the `ccme` datasets contained in the `ssddata` package.

200 data sets for each of the sample sizes *n* = 6, 8, 16, 24, 32 were generated from distributions *A*, *B*, and *C* above. An SSD comprised of a set of 4 known stable distributions (gamma, log-gumbel, log-logistic, and lognormal) was fitted to each of the synthetic dataset generated from all *dataset x sample_size x distribution* combinations.

All methods were tested using 1,000 bootstrap samples, for all simulated datasets, and for HC values of 1, 5, 10 and 20% – resulting in a total of 527, 960 HC estimates from 131,990 fitted `ssd_fit_dist` objects.

It is evident from the description of the simulation methodology that this was both a complex and large undertaking. Notwithstanding the sheer number of datasets, iterations, and multiple computations, the *evaluation* of the results is based on considerations of confidence interval *bias, coverage*, and *width.*

**Bias** is a measure of the degree to which the estimated HCx consistently *over-* or *under-estimates* the *true* value. An *unbiased* estimator is one for which the *average bias* is zero. We use the bias criterion as a *first-order* screening consideration. That is, a method for which the magnitude of the average bias is unacceptably high will be eliminated from further consideration.

**Coverage** is defined as the *proportion of bootstrap samples for which the computed confidence interval 'captured' the true HCx value.* Thus, for a 95% confidence interval, we expect 95% of all bootstrap confidence intervals to contain the true HCx value.

**Width** is simply the difference between the upper and lower limits of the confidence interval.

Our criteria for determining what constitutes a 'good' methodology for CI estimation is one for which:

- The *magnitude* of the average bias is close to zero;
- The *coverage* for a $(1-\alpha)100\%$ confidence interval is close to $(1-\alpha)100\%$ ; and
- The *width* of the confidence interval is as small as possible.

## *Simulation study results*

### *Bias*[1]

Overall, our results suggest (for this particular simulation scenario) that all HCx estimation methods are *positively biased* (i.e. the estimated HC value tends to be *higher* than the true HC value) for *x = 1, 5, 10,* and 20 (Figure 1). Bias decreases rapidly with increasing sample size for all methods, highlighting the importance of obtaining the largest possible sample sizes in SSD modelling (Figure 1). Bias was always lowest for the *rmulti* methods (Figure 1) and on that basis, is the *preferred approach for HCx point estimation*. Note that full details for each simulation source dataset are shown in Appendix A.

### *Coverage*

The picture with respect to coverage is less well-defined (Figure 2). Overall, the coverage of the `weighted_sample` and the two methods using the `ssd_qmulti()` and `ssd_pmulti()` functions (`rmulti` and `rmulti_fixed`) yield intervals whose coverage is closest to the nominal 95% (Figure 2). However, this is not uniform across the 4 HCx levels considered (Figure 2). We see that the `weighted_sample` method provides superior coverage for the estimation of $HC_1$ and $HC_5$ with relatively small differences in coverage among all methods for the estimation of $HC_{10}$ and $HC_{20}$ (Figure 2).

The two averaging methods (`arithmetic` and `geometric`) are the worst in terms of coverage (Figure 2). The CRAN version 1.0.6 of `ssdtools` uses arithmetic averaging which is why this investigation was required, and the results here confirm the need to replace this method with a superior approach.

Fixing the AICc weights for the methods based on bootstrap samples taken from the candidate set as a mixture distribution had little effect on coverage, and these methods appear to yield very similar results (`rmulti` and `rmulti_fixed`, Figure 2).

Note that full details for each simulation source dataset are shown in Appendix B.

[1] *The* `uniroot` *method performed very poorly in terms of both coverage and bias and was removed from plotting so that the pattern across the remaining methods could be better observed.*

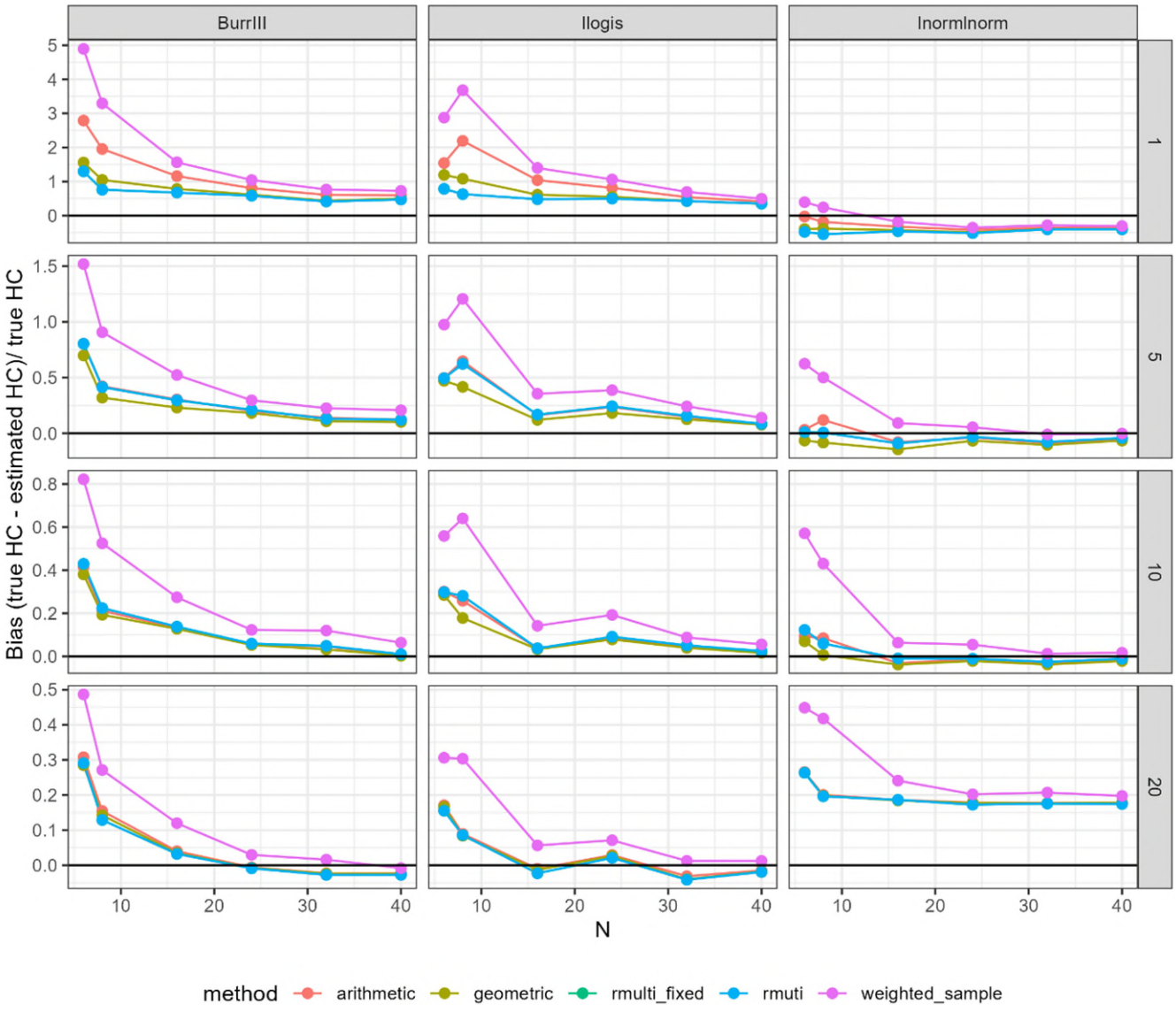



**Figure 1. Mean bias as a function of sample size (N = 6, 8, 16, 24, 32, and 40) across all datasets and simulation runs for each of three simulation scenarios (Burr III, log-logistic and lognormal-lognormal mixture distributions, plot columns). Bias was calculated as (estimated hazard concentration – true hazard concentration)/true HC. Four hazard concentration values were considered (1, 5, 10 and 20, plot rows). Fitted candidate distributions included 'gamma', 'lgumbel', 'llogis', and 'lnorm.**

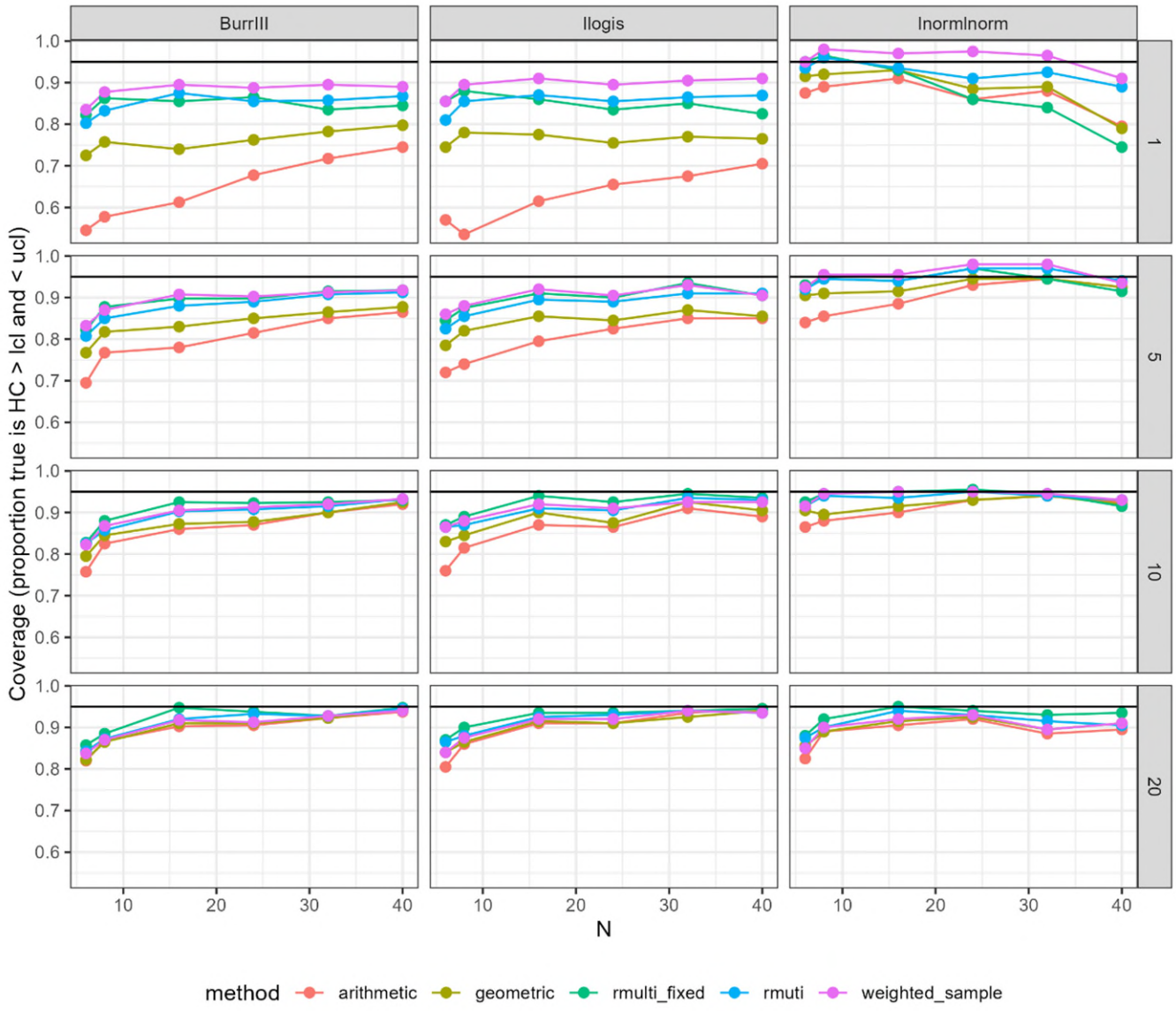



**Figure 2. Mean coverage as a function of sample size (N = 6, 8, 16, 24, 32, and 40) across all datasets and simulation runs for each of three simulation scenarios (Burr III, log-logistic and lognormal-lognormal mixture distributions, plot columns). Coverage was calculated as the proportion of times the true HC fell within the 95% confidence interval. Four hazard concentration values were considered (1, 5, 10 and 20, plot rows). Fitted candidate distributions included 'gamma', 'lgumbel', 'llogis', and 'lnorm.**

### *Width*

Confidence interval widths were computed for each of the 500 bootstrap samples, replicated 200 times for each of the 8 data sets, across all sample sizes and for $HC_x$ ; $\{x = 1,5,10,20\}$ where the parent distribution was a Burr III distribution (Figure 3); a lognormal mixture distribution (Figure 4); and a log-logistic distribution (Figure 5).

The most obvious, and striking feature of the plots of Figure 3, Figure 4 and Figure 5 is the almost identical *average* confidence interval widths over all combinations of datasets, parent distribution for simulated data, HCx values, and sample size. Thus, as a criterion for assessing the performance of different CI computational methods, confidence interval width can be ignored, leaving considerations of *bias* and *coverage* as screening criteria.

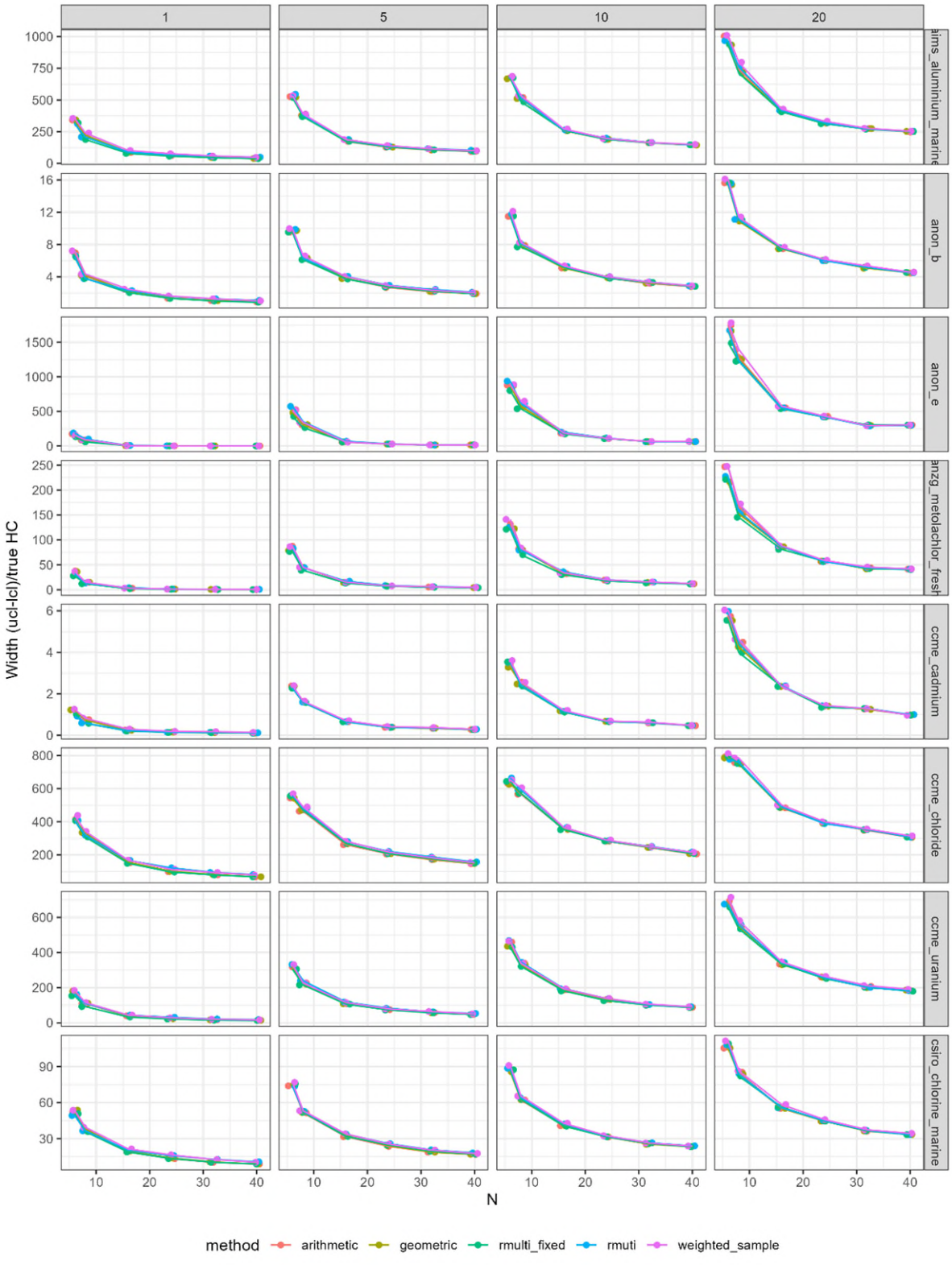



**Figure 3. Confidence interval width (ucl-lcl) for 7 simulation datasets based on a Burr III distribution (plot rows) across a range of sample sizes (6, 8, 16, 24, 32, and 40), for hazard concentration values of 1, 5, 10 and 20 (plot columns) for a range of different methods (see text). Each dataset was simulated 200 times, for each dataset, and a minimum of 1000 bootstrap samples were used to estimate confidence intervals. Fitted candidate distributions included 'gamma', 'lgumbel', 'llogis', and 'lnorm'.**

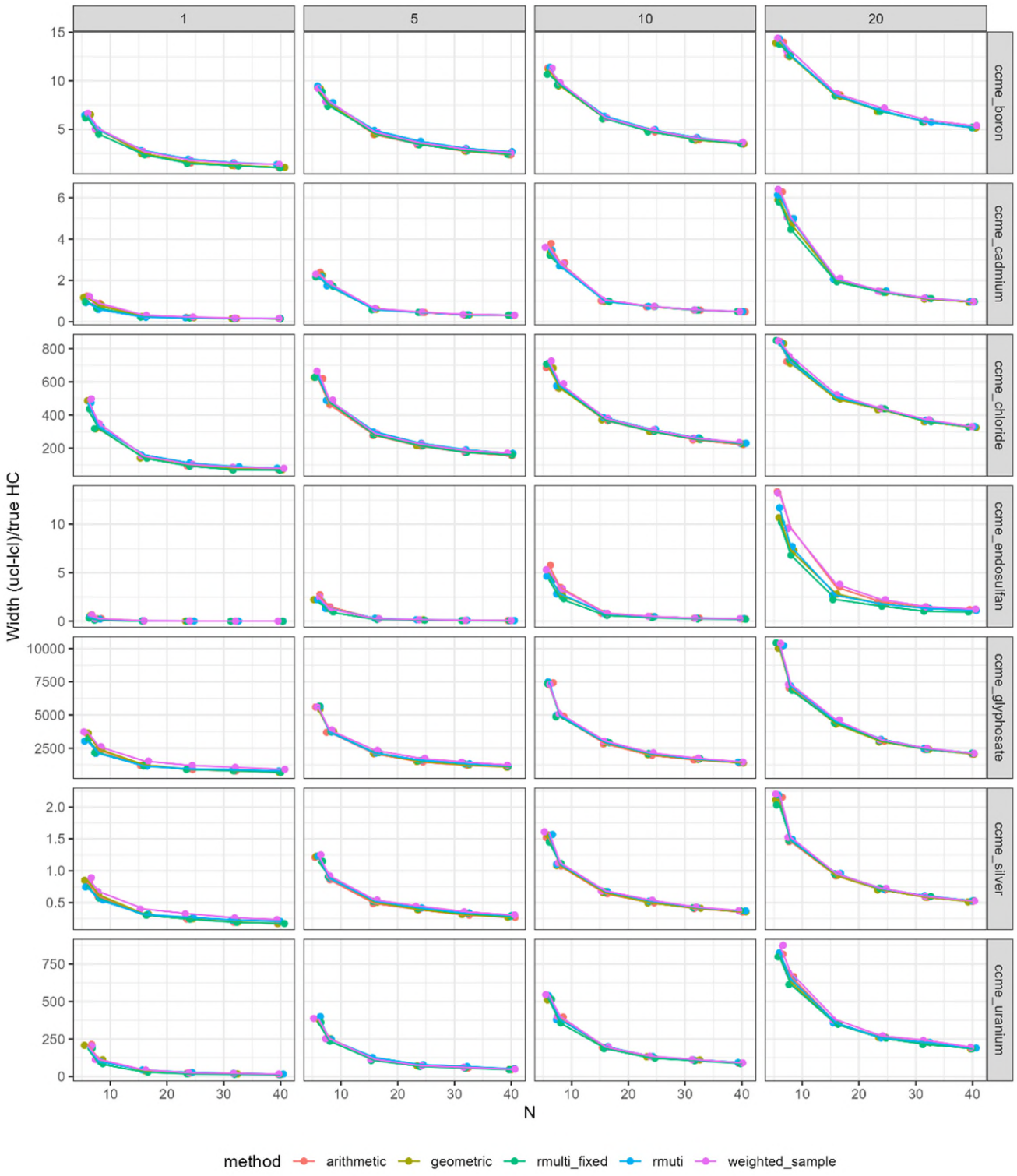



**Figure 4. Confidence interval width (ucl-lcl) for 6 simulation datasets based on a log-normal log-normal distribution (plot rows) across a range of sample sizes (6, 8, 16, 24, 32, and 40), for hazard concentration values of 1, 5, 10 and 20 (plot columns) for a range of different methods (see text). Each dataset was simulated 200 times, for each dataset, and a minimum of 500 bootstrap samples were used to estimate confidence intervals. Fitted candidate distributions included 'gamma', 'lgumbel', 'llogis', and 'lnorm'.**

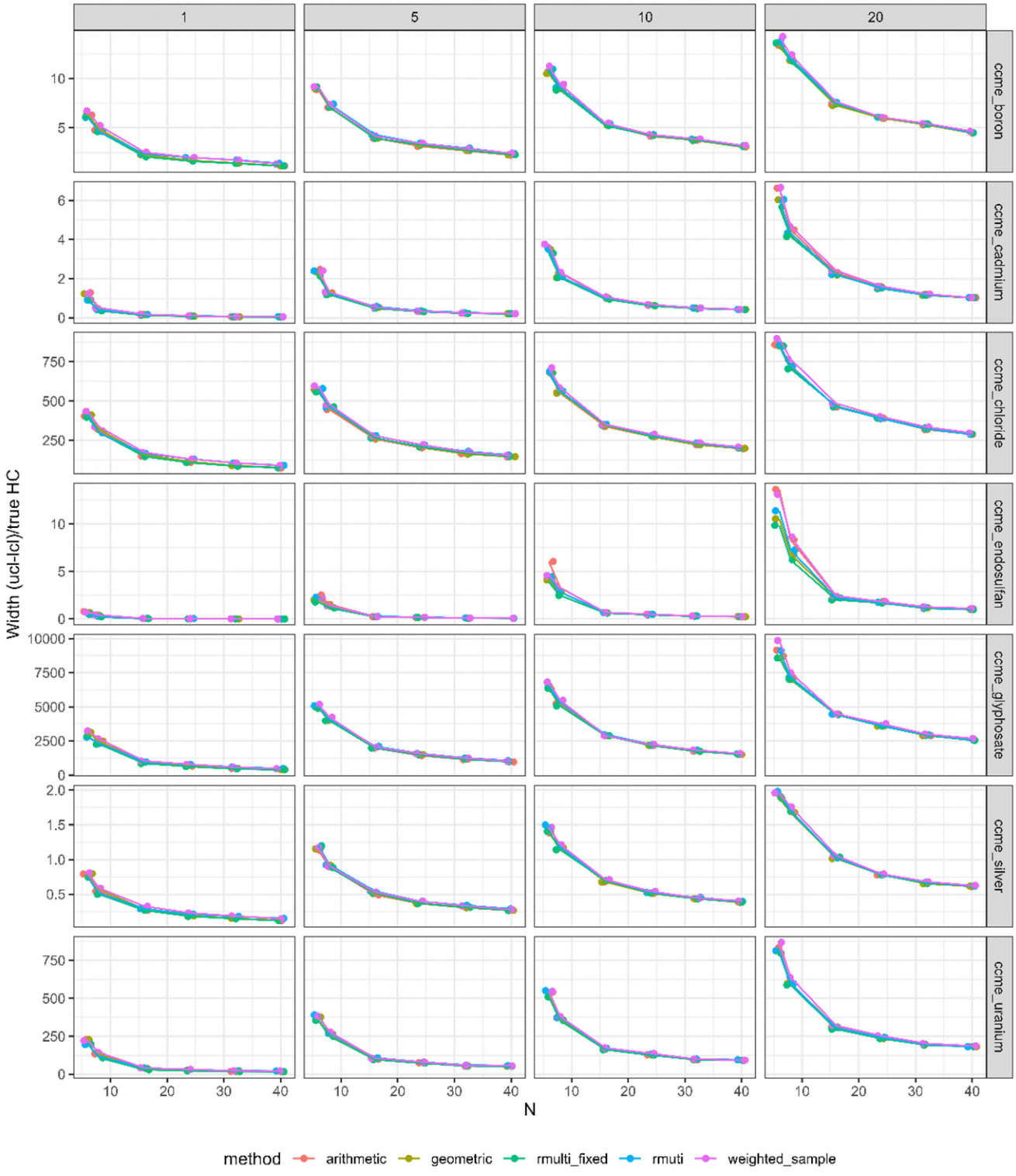



**Figure 5. Confidence interval width (ucl-lcl) for 7 simulation datasets based on a log-logistic distribution (plot rows) across a range of sample sizes (6, 8, 16, 24, 32, and 40), for hazard concentration values of 1, 5, 10 and 20 (plot columns) for a range of different methods (see text). Each dataset was simulated 200 times, for each dataset, and a minimum of 500 bootstrap samples were used to estimate confidence intervals. Fitted candidate distributions included 'gamma', 'lgumbel', 'llogis', and 'lnorm'.**

Our first-order consideration of *bias* ruled out the `weighted_sample` method for *point estimation* of the HCx. However, the `weighted-sample` method had good coverage properties and upon further benchmarking, was shown to be more than twice as fast as either the `rmulti` or `rmulti_fixed` methods (Figure 6).

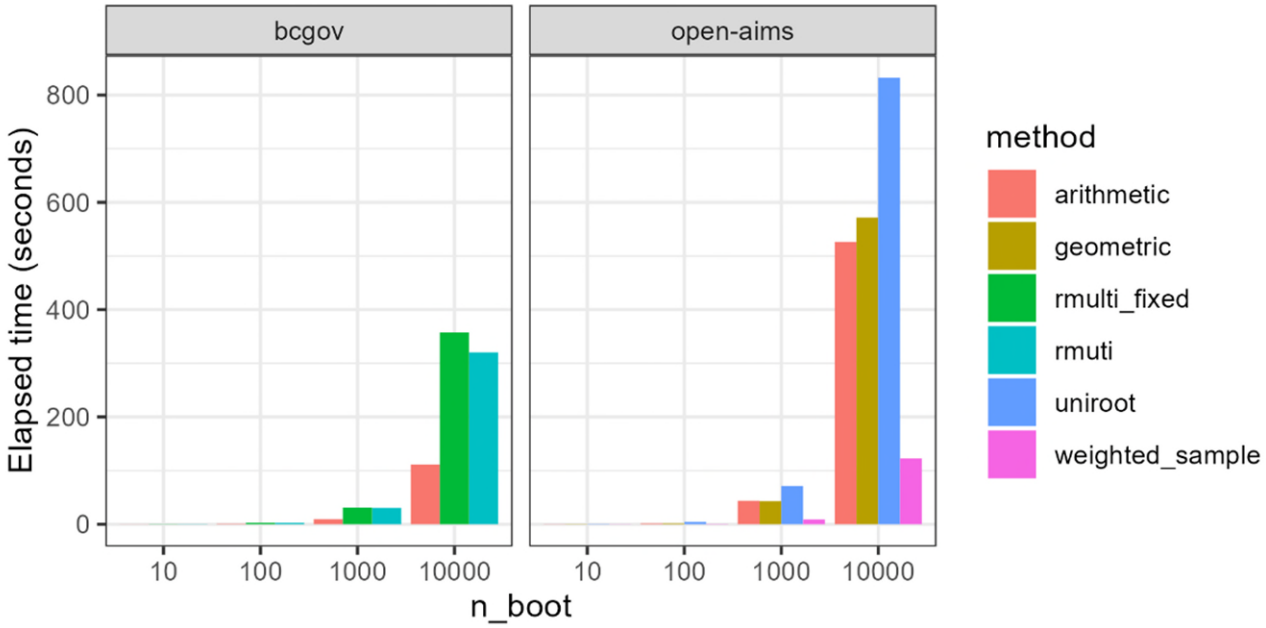



**Figure 6. Time taken to undertake bootstrap sampling for each of the methods across a range of different bootstrap sample sizes (n_boot) and ssdtools implementation – optimized version (bcgov panel) and development version (open-aims panel). Timing was undertaken using Dell Precision 7560, 11th Gen Intel® Core™ i9-11950H @ 2.6GHz and 32 GB RAM, running Microsoft Windows 10 Enterprise with an x64-based PC system.**

As remarked above, in terms of confidence interval width, there was negligible difference *on average* between the computational methods examined. What is now of interest is to compare the confidence limits obtained for both the `weighted_sample` and `rmulti` methods at an *individual sample* level. To facilitate this comparison we plotted separately, the upper CI limit from the `weighted sample` method against the upper CI limit for the `rmulti` method and the lower CI limit from the `weighted sample` method against the lower CI limit for the `rmulti` method (Figure 7).

Upon closer inspection, we see that *for any individual sample*, the upper and lower confidence interval limit derived from either of the `rmulti` methods is in almost perfect agreement with the upper or lower confidence interval limit derived from the `weighted_sample` method and that this observation holds true across all three simulation datasets and all 4 HCx values (Figure 7).

Together, these results suggest an effective HCx estimation strategy could be developed by integrating *both* approaches: (i) the *unbiased point estimation* of an HCx using the `ssd_qmulti()` function; and (ii) the *rapid confidence interval estimation* for that HCx using the `weighted_sample` method.

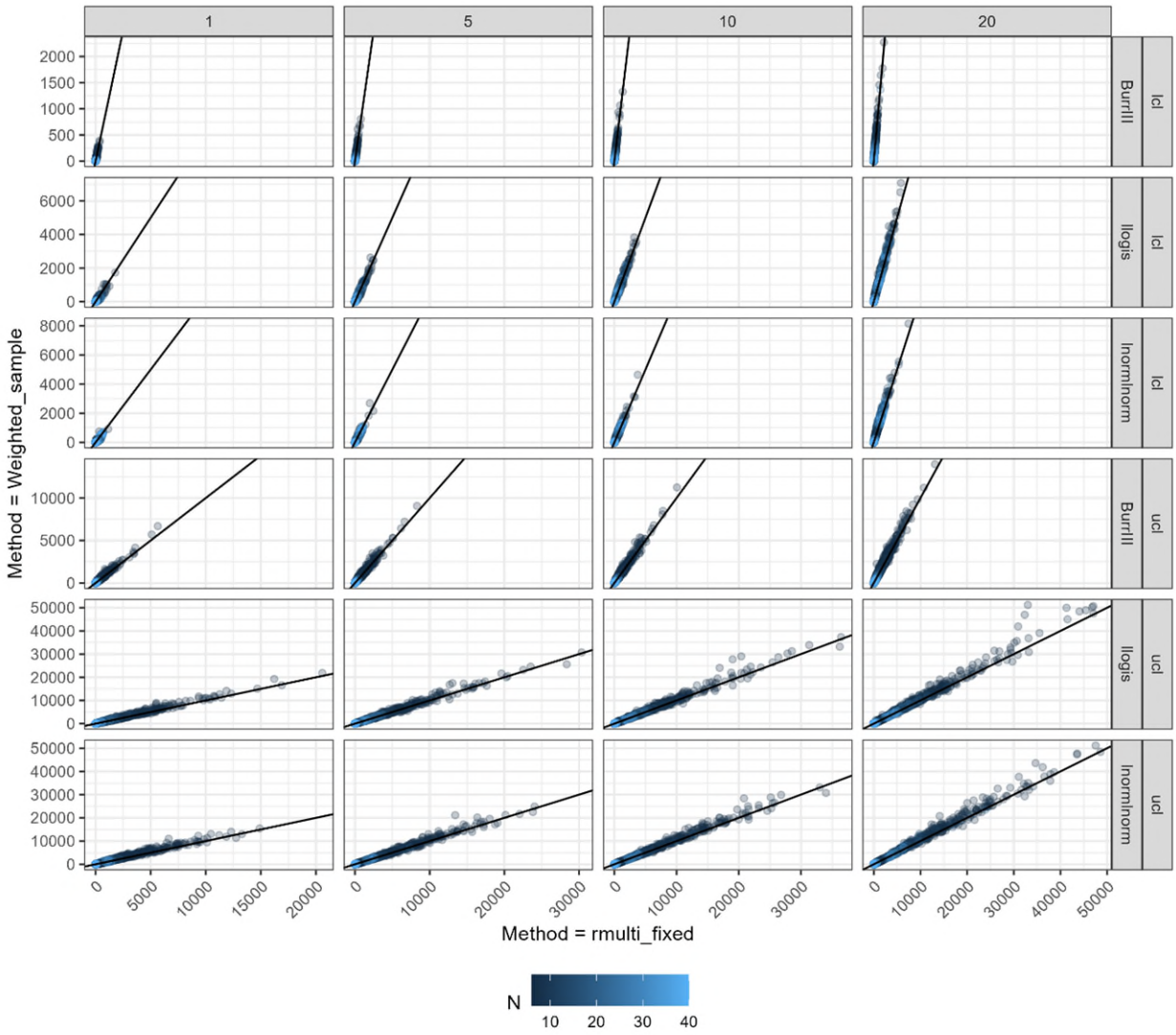



**Figure 7. A comparison of the lower 2.5% (lcl) and upper 97.5% estimated bootstrap confidence intervals between the weighted_bootstrap and rmulti methods, across all three simulation datasets, for a range of HC values (1, 5, 10 and 20%).**

### 4.1.2 Task S3: Inconsistency between model averaged ssd_hc() and ssd_hp()

During the course of Phase II an anomaly was discovered relating to the manner in which `ssdtools` was computing a model-averaged HCx that violated the 'inversion principle'. The inversion principle requires that, no matter how determined, an estimated HCx must satisfy the basic requirement that the estimated fraction of species affected at HCx is x. For HCx values obtained from a single cumulative distribution function (*cdf*), this property is guaranteed. However, in the case of an HCx derived from either a single *mixture* distribution or a *model-averaged* distribution, this requirement may not be met. Task S3 was designed to investigate this issue more fully and to provide a fix to the `ssdtools` code.

Here we describe weighted averaging and explain why simple weighted averages are not appropriate for estimating the joint *cdf* of a weighted set of candidate SSDs. The explanation leans heavily on the analogy between a statistical mixture distribution and a weighted set of candidate distributions, which is discussed in more detail later.

### *The process of averaging*

The process of averaging is a routine *data reduction* technique as well as the basis of much of statistical inference. In its simplest form, the sample *arithmetic average* or *mean* $\left(\bar{X}\right)$ is defined as

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Other types of means are available, including the geometric mean, the harmonic mean, and the weighted mean. The last of these is particularly pertinent to model averaging.

### *Weighted Averages*

In the computation of the simple arithmetic mean, the individual $X_i$ values all receive the same weight of $\frac{1}{n}$. While this is appropriate in many cases, it's not useful when the components contribute to *varying* degrees such as the *time-varying* concentration shown in Figure 8.

There are 5 values of concentration in Figure 8 (from *L* to *R*): {0.25, 0.95, 0.25, 0.12, 0.5}. The simple arithmetic mean of these concentrations is $\bar{X} = 0.414$. However, this ignores the different *durations* of each of the 5 concentrations. Of the 170 hours, 63 were at concentration 0.25, 25 at concentration 0.95, 23 at concentration 0.25, 23 at concentration 0.12, and 36 at concentration 0.50. So, if we were to *weight* these concentrations by time have:

$$\bar{X}_{TW} = \frac{\left(63\cdot 0.25 + 25\cdot 0.95 + 23\cdot 0.25 + 23\cdot 0.12 + 36\cdot 0.50\right)}{\left(63+25+23+23+36\right)} = \frac{56.01}{170} = 0.33$$

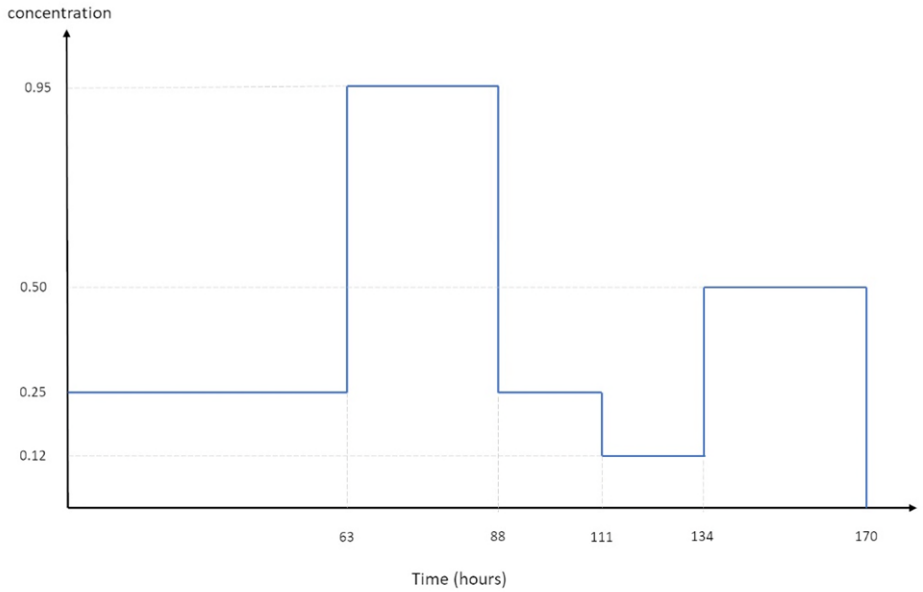



**Figure 8. Example of a concentration that varies over time.**

From this simple example, it is easy to see that the weights applied to each of the 5 concentrations are $\left\{\frac{63}{170},\frac{25}{170},\frac{23}{170},\frac{23}{170},\frac{36}{170}\right\}=\{0.371,0.147,0.135,0.135,0.212\}$. The formula for a *weighted average* is:

$$\bar{X}=\sum_{i=1}^{n} w_i X_i$$

with $0 \leq w_i \leq 1$ and $\sum_{i=1}^{n} w_i = 1$. Note, the simple arithmetic mean is just a special case of the weighted mean with $\sum_{i=1}^{n} w_i = \frac{1}{n}; \forall i = 1, \ldots, n$

### *Model Averaging Explained*

The *weighted average* acknowledges that the elements in the computation are not of equal 'importance'. In the example above, this importance was based on the *proportion of time* that the concentration was at a particular level. Bayesians are well-versed in this concept – the elicitation of *prior distributions* for model parameters provides a mechanism for weighting the degree to which the analysis is informed by existing knowledge versus using a purely data-driven approach. Model-averaging is an information-theoretic approach which attempts to resolve the ecotoxicological dilemma of having to choose a single probability model for the SSD. Model averaging has been historically usually used in the context of estimating model parameters, or quantities derived from a fitted model – for example an EC50 derived from a C-R model. Its use in model averaged SSD's is new, and there are technical issues with implementing model averaging in this setting. This is best explained with the use of the following small dataset of toxicity estimates for some chemical:

```
1.73 0.57 0.33 0.28 0.30 0.29 2.15 0.80 0.76 0.54 0.42 0.83 0.21 0.18 0.59
```

Suppose there are only two possibilities for fitting an SSD – both lognormal distributions. Model 1 is the LN(-1.067,0.414) distribution while Model 2 is the LN(-0.387,0.617) distribution. A plot of the empirical *cdf* and Models 1 and 2 is shown in Figure 9.

We see that Model 1 (green) fits well in the lower, left region and poorly in the upper region, while the reverse is true for Model 2 (blue). So, using *either* Model 1 *or* Model 2 is going to result in a poor fit overall. However, the obvious thing to do is to *combine* both models. We could try using 50% of Model 1 and 50% of Model 2, but that may be sub-optimal. It turns out that the best fit is obtained by using 44% of Model 1 and 56% of Model 2. The *weighted average* of Models 1 and 2 is shown in red (Figure 9).

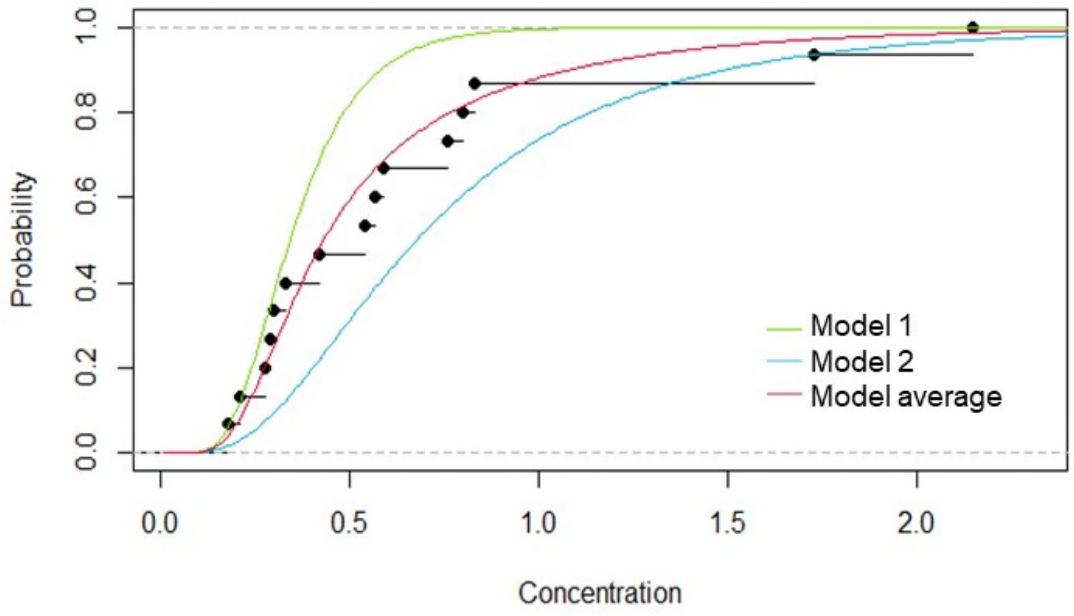



**Figure 9. Empirical cdf (black) with fitted cdfs overlayed, including: Model 1 (green), LN(-1.067,0.414); Model 2 (blue) LN(-0.387,0.617); and averaged Model (red).**

Clearly the 'appropriate[1]' combination of the two models provides a far better description of the data than either model alone.

We next turn our attention to the estimation of an HCx – for example, the HC20. It's a simple matter to work out the *individual* HC20 values for Models 1 and 2 using the appropriate `qlnorm()` function in R:

```
# Model 1 HC20
cat("Model 1 HC20 =",qlnorm(0.2,-1.067,0.414))
#> Model 1 HC20 = 0.2428209

# Model 2 HC20
cat("Model 2 HC20 =",qlnorm(0.2,-0.387,0.617))
#> Model 2 HC20 = 0.4040243
```

An intuitively appealing approach to calculating the *model-averaged* HC20 is to apply the same weights to the individual HC20 values used to define the model-averaged SSD in Figure 2 (i.e. the red curve). This suggests a model-averaged HC20 of `0.44*0.2428209 + 0.56*0.4040243 = 0.33`. This how the original version of `ssdtools` was computing model averaged HCx values. For our simplistic example, the model-averaged HC20 estimate is 0.33. As a check, we can determine the *fraction affected* at concentration = 0.33 – it should be 20%. However, as can be seen from Figure 10, the actual fraction affected at concentration 0.33 is 30% – not the required 20%.

[1] By appropriate we mean that combination for which the resulting fit is best in some sense.

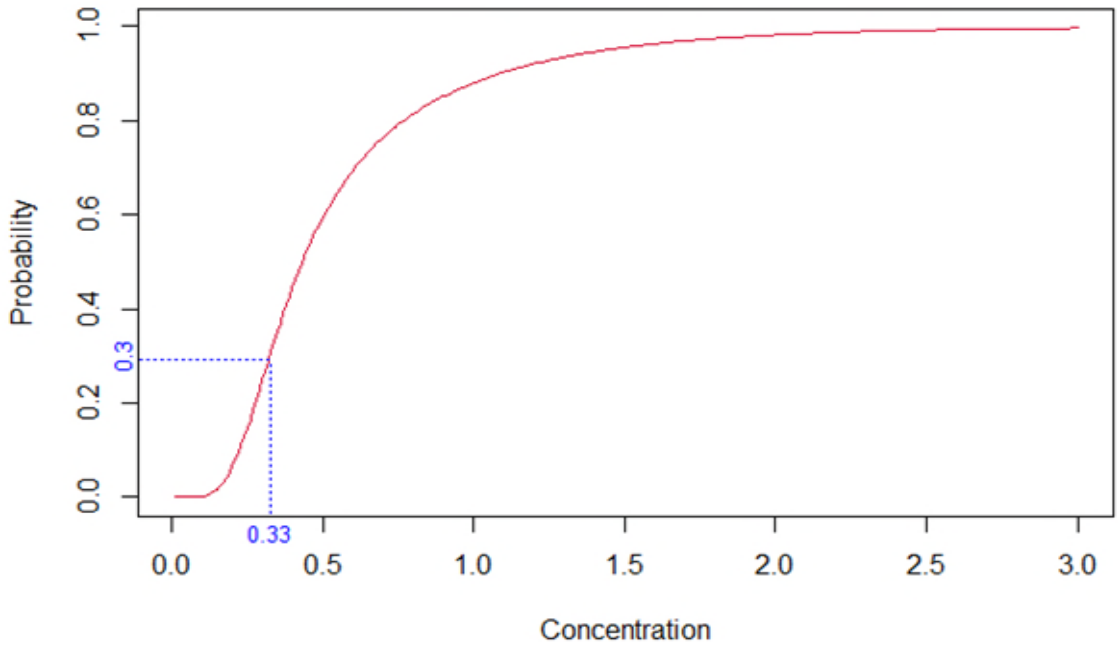



**Figure 10. Fitted cdf and showing the fraction effected back calculated using arithmetic averaging.**

### *Model Averaged SSDs*

As has been demonstrated for a simple numerical example, applying the model weights to component HCx values and summing did not produce the correct result. The reason for this is explained mathematically next.

#### *The fallacy of weighting individual HCx values*

The correct expression for a model-averaged SSD is:

$$G(x) = \sum_{i=1}^{k} w_i \, F_i(x)$$

where $F_i(\cdot)$ is the $i^{th}$ component SSD (i.e. *cdf*) and $w_i$ is the weight assigned to $F_i(\cdot)$. Notice that the function $G(x)$ is a proper *cumulative distribution function* (*cdf*) which means for a given quantile, *x*, $G(x)$ returns the *cumulative probability*:

$$P[X \leq x]$$

Now, the *incorrect* approach takes a weighted sum of the component *inverse cdf's*, that is:

$$H(p) = \sum_{i=1}^{k} w_i \, {F_i}^{-1}(p)$$

where ${F_i}^{-1}(\cdot)$ is the $i^{th}$ *inverse cdf*. Notice that $G_i(\cdot)$ is a function of a *quantile* and returns a probability while $H_i(\cdot)$ is a function of a *probability* and returns a quantile.

The correct method of determining the *HCx* is to work with the proper model-averaged *cdf* $G(x)$. This means finding the *inverse* function $G^{-1}(p)$. We'll address how this is done shortly.

The reason $H(p)$ does not return the correct result is because of the implicit assumption that the inverse of $G(x)$ is equivalent to $H(p)$. This is akin to stating the inverse of a *sum* is equal to the sum of the inverses i.e.

$$\sum_{i=1}^{n} \frac{1}{X_i} = \frac{1}{\sum_{i=1}^{n} X_i}$$

There are some very special cases where the above identity does in fact hold, but they involve complex numbers which do not arise in ecotoxicology. We therefore conclude:

$$G^{-1}(p) \neq H(p)$$

A visual demonstration can be seen in Figure 11, which is a plot of $G(x)$ and the *inverse* of $H(p)$ both as functions of *x* (a quantile) for our two-component lognormal distribution above (Figure 9).

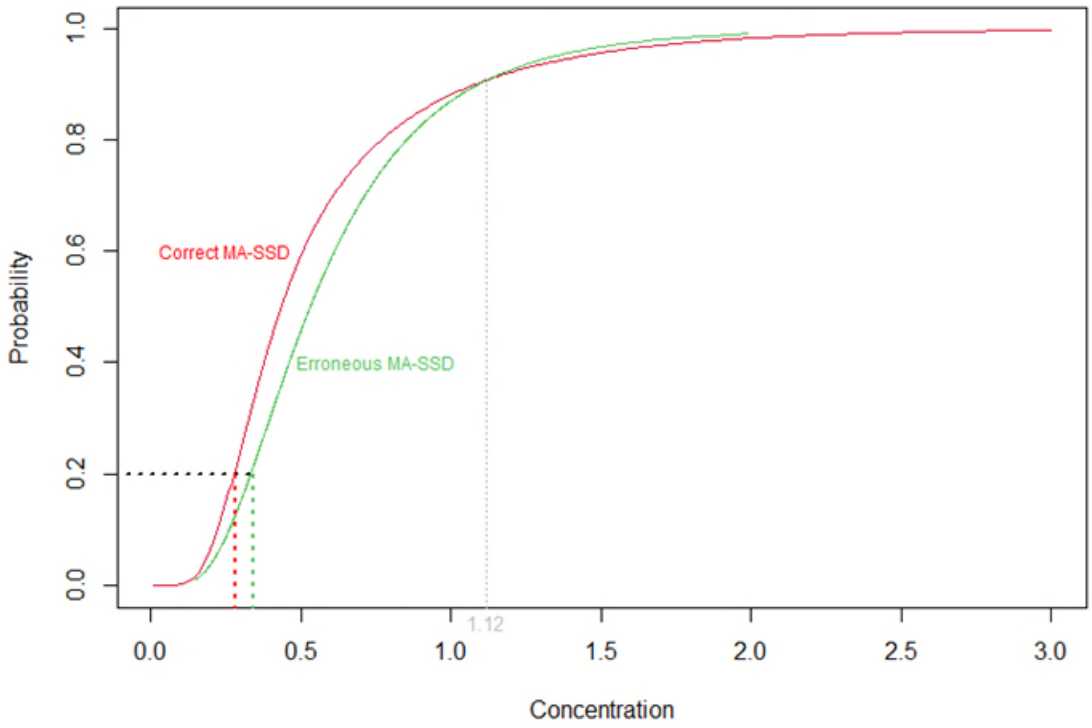



**Figure 11. Plot of $G(x)$ and the *inverse* of $H(p)$ both as functions of *x* (a quantile) for the two-component lognormal distribution above.**

Clearly, the two functions are not the same and thus HCx values derived from each will nearly always be different (as indicated by the positions of the vertical red and green dashed lines in Figure 11 corresponding to the 2 values of the HC20). Note that the two curves do cross over at a concentration of about 1.12 corresponding to the 90$^{th}$ percentile, but in the region of ecotoxicological interest, there is no such cross-over and so the two approaches will always yield different *HCx* values with this difference → 0 as x → 0 (Figure 11).

We next discuss the use of a model-averaged SSD to obtain the *correct* model-averaged HCx.

### *Computing a model-averaged HCx*

A proper *HCx* needs to satisfy *the inversion principle*. More formally, the inversion principle states that an *HCx* (denoted as $\varphi_x$) must satisfy the following:

$$df(\varphi_x) = x \qquad and \qquad qf(x) = \varphi_x$$

where $df(\cdot)$ is a model-averaged *distribution function* (i.e. SSD) and $qf(\cdot)$ is a model-averaged *quantile function*. For this equality to hold, it is necessary that $qf(p) = df^{-1}(p)$.

So, in our example above the green curve was taken to be $qf(x)$ and this was used to derive $\varphi_x$ but the *fraction affected* $\{= df(\varphi_x)\}$ at $\varphi_x$ is computed using the red curve (Figure 11).

In `ssdtools` the following is a check that the inversion principle holds:

```
# Obtain a model-averaged HCx using the ssd_hc() function
hcp<-ssd_hc(x, p = p)
# Check that the inversion principle holds
ssd_hp(x, hcp, multi_est = TRUE) == p
```

This should result in logical `TRUE`. If the `multi_est` argument is set to `FALSE` the test will fail. This is a new argument in the revised version of `ssdtools` that controls how the model averaged HCx estimate is calculated and is based on the `uniroot()` function described below.

The *inversion principle* ensures that we only use a single distribution function to compute both the HCx *and* the fraction affected. Referring to Figure 12, the HCx is obtained from the MA-SSD (red curve) by following the → arrows while the fraction affected is obtained by following the ← arrows.

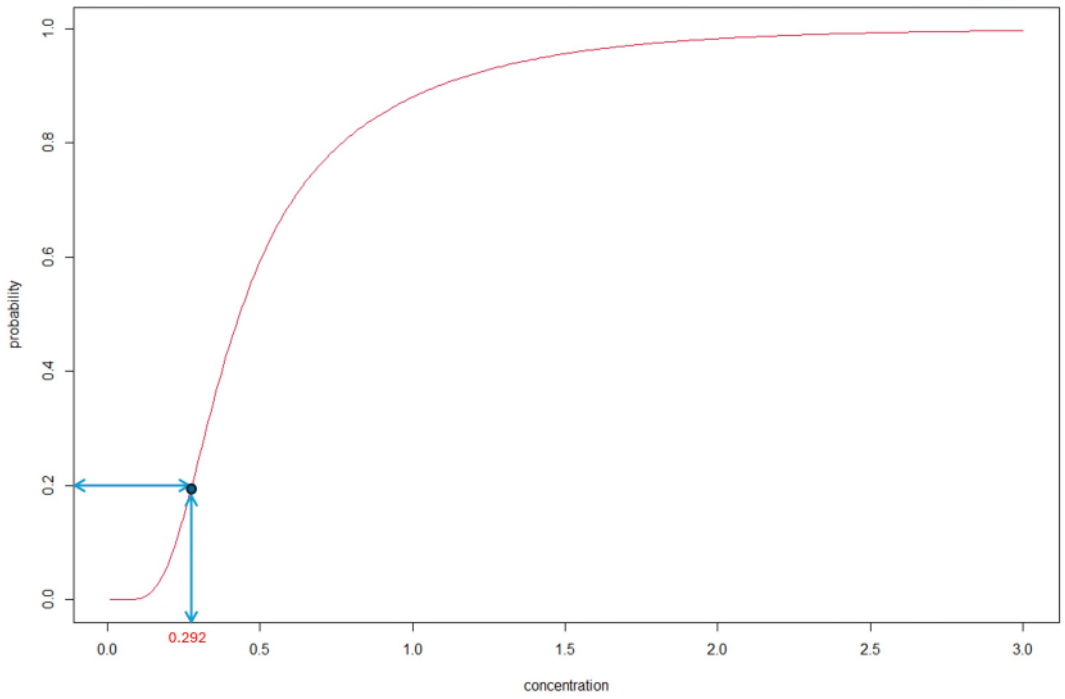



**Figure 12. Plot of a properly invertible cdf for two-component lognormal distribution show in Figure 9.**

Finally, we briefly discuss how the HCx is computed in R using the method that has been implemented in `ssdtools`. Recall, the MA-SSD is given as

$$G(x) = \sum_{i=1}^{k} w_i \, F_i(x)$$

and an HCx is obtained from the MA-SSD by essentially working 'in reverse' by starting at a value of $x$ on the vertical scale in Figure 12 and following the → arrows and reading off the corresponding value on the horizontal scale.

Obviously, we need to be able to 'codify' this process in R (or any other computer language). Mathematically this is equivalent to seeking a solution to the following equation:

$$x\colon G(x) = p$$

or, equivalently:

$$x\colon G(x) - p = 0$$

for some fraction affected, $p$.

Finding the solution to this last equation is referred to as finding the root(s) of the function $\boldsymbol{G(x)}$ or, as is made clear in Figure 13, finding the zero-crossing of the function $G(x) - p$ for the case $p = 0.2$. In R finding the roots of $x\colon G(x) - p = 0$ is achieved using the `uniroot()` function.

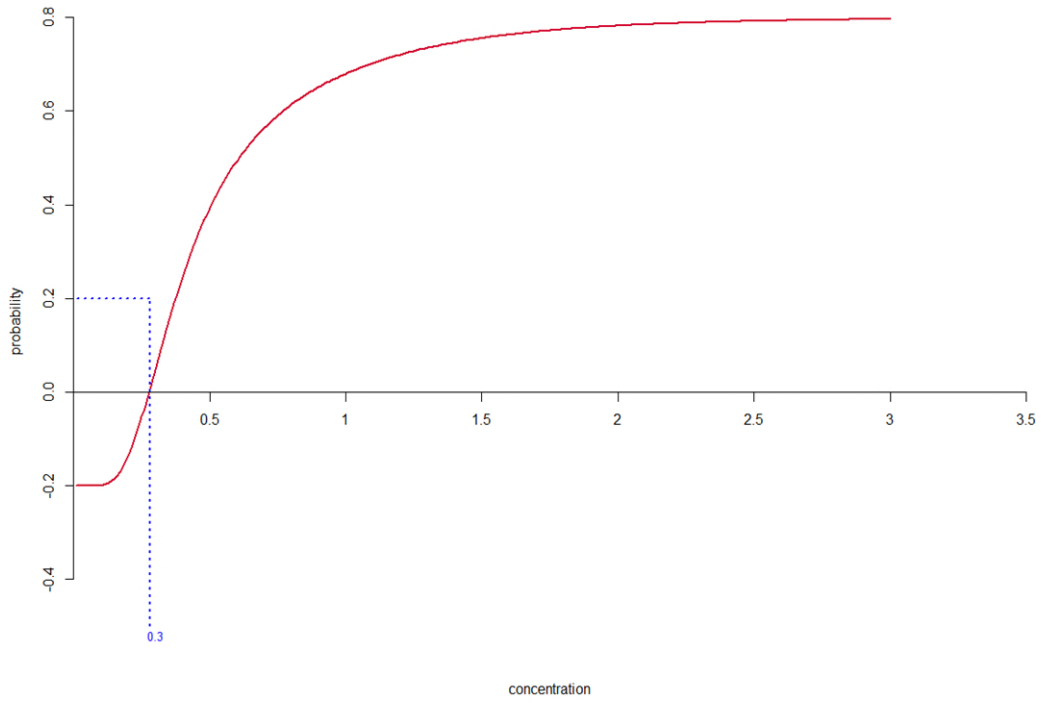


**Figure 13. Graphical representation of finding the zero-crossing of the function $G(x) - p$ for the case $p = 0.2$ for two-component lognormal distribution shown in Figure 9.**

### *Mixture models and Model Averaged SSDs: Philosophical and practical considerations*

The Project Team has had several detailed discussions about the differences and similarities between a *mixture* model SSD and a *model-averaged* SSD. Both are expressible as a weighted linear combination of individual theoretical probability density functions (*pdfs*) and as such the *mathematical* treatment is the same for both. However, what is different is the way in which the *weights* are computed. In the case of model averaging, the weights are determined *after* fitting all the candidate distributions and then re-scaling the individual goodness-of-fit metrics (as measured by the AICc), so they sum to unity. In mixture modelling, the weights are explicitly incorporated into the *pdf* as model parameters to be estimated along with the parameters of the component *pdfs* – subject to the constraints that each weight is between 0 and 1 and the sum of all weights is unity. Not surprisingly, the resultant model averaged (MA) and mixture distributions are not identical even though they have the same component *pdfs*. This distinction has important ramifications for subsequent parametric bootstrapping to estimate HCx confidence intervals. As an example, Figure 14 shows an empirical *cdf* with both a mixture model SSD and model-averaged SSD whose component distributions are the same (Weibull, log-normal, and log-logistic) but having weights estimated as described above.

A currently unresolved question is: should the model weights be re-estimated for every bootstrap sample, or should the model weights be fixed at the values estimated from the models fitted to the original sample of toxicity data (see below)? Our current thinking is that they should be fixed at their nominal values in the same way that the component distributions to be used in bootstrapping are informed by the fit to the sample toxicity data.

This contrasts with fitting a single *mixture* distribution where the weights are inherent model parameters and not lines of evidence/support for the true, underlying single distribution.

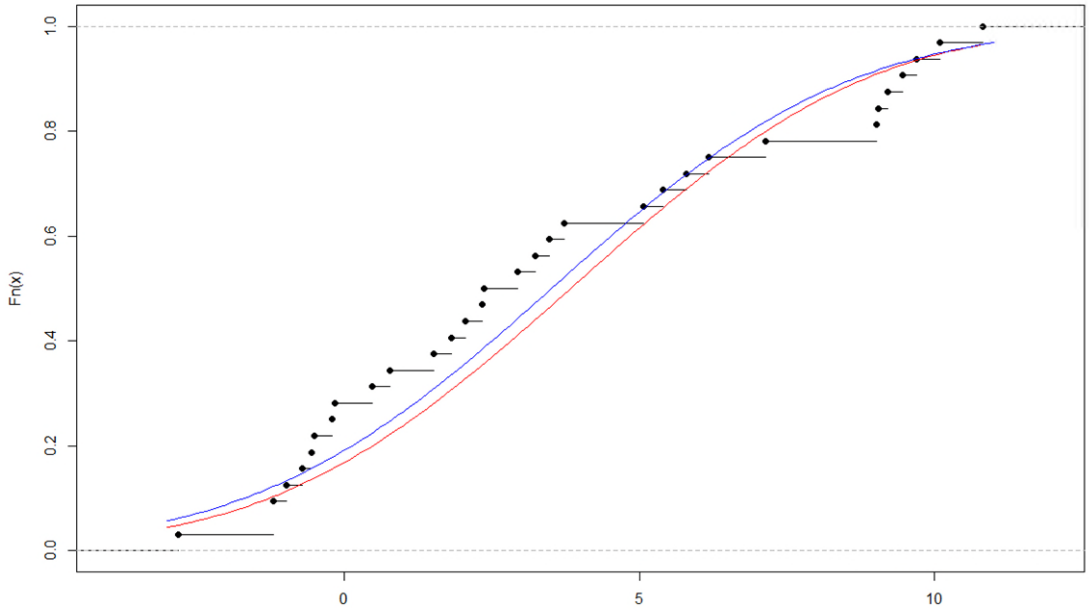



**Figure 14. Empirical cdf (step function) with fitted model-averaged cdf (comprised of Weibull, lognormal, and loglogistic distributions) (blue curve) and fitted mixture-model (comprised of Weibull, lognormal, and loglogistic distributions) (red curve) overlaid.**

### *New methods for CI estimation and SSD weighting*

Although somewhat out-of-scope, two new approaches for obtaining confidence interval estimates for an HCx have been developed that does not require bootstrapping.

The first method takes the confidence interval around the *fraction affected* (FA) (as determined by $\hat{p} \pm t_{n-p,1-\alpha/2} \cdot SE\left[\hat{p}\right]$ ) at the HCx and finds the corresponding interval on the abscissa (i.e. for the HCx). The method is illustrated in Figure 15. The procedure is entirely *analytic* and thus requires no bootstrapping.

The second approach is also analytical and recasts the estimation of an HCx as an inverse, non-linear calibration problem and uses the `minpack.lm` and `investr` packages in `R` to obtain both point and interval estimates for a model averaged HCx without the need for bootstrapping. This results in dramatically reduced computational times.

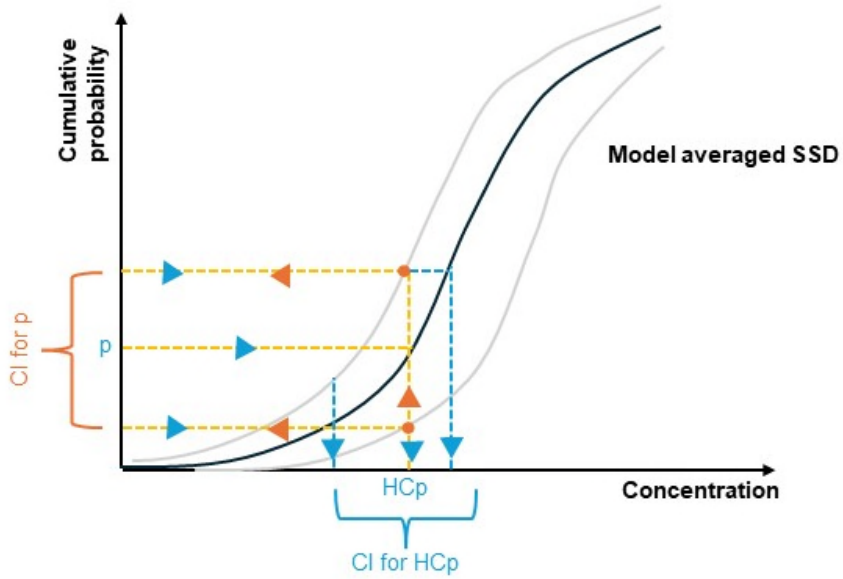



**Figure 15. Conceptualisation of determination of approximate confidence intervals for HCp. Red arrows indicate forward use of the SSD while the cyan arrow indicates the inverse use of the SSD.**

As with `ssdtools`, the proposed methodology also allows weighted SSD models to be fit to assign increased importance to the fit of the SSD in the lower-left tail region. This suggestion has been contemplated in the literature for a very long time although, given the selection of weights is subjective, it remains a contentious procedure. An example of a weighted SSD applied to the empirical *cdf* of Figure 9 is shown in Figure 16. We can use this approach for the computation of point and interval HCx estimates. The appeal of this type of approach is the computational speed – typically 500 times faster than bootstrapping using 1,000 samples (Table 2). The 'cost' of this dramatic increase in speed is a potential increase in bias. The analytical methods rely on approximations and asymptotic statistical results with the quality of such approximations critically dependent on *sample size (n)* of the input toxicity data set.

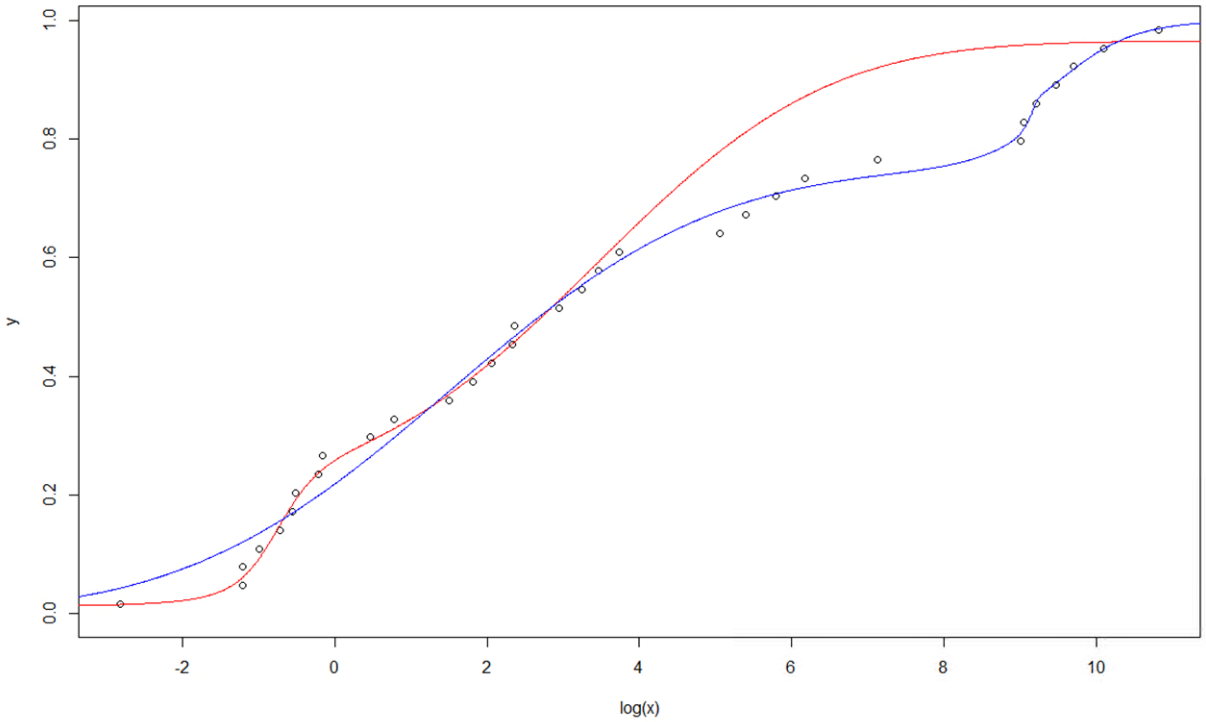



**Figure 16. Comparison of weighted SSD using new approach (red line) and unweighted SSD from `ssdtools` (blue line) applied to empirical cdf of Figure 14 (open circles).**

**Table 2 Comparison of time taken for obtaining the initial fit and estimating confidence intervals (CIs) between `ssdtools` and the analytical approach described above and shown in Figure 15.**

| Type | Method | Time |
|---|---|---|
| **Initial fit** | ssdtools (6 parameters) | 0.11 sec |
| | Analytic method (6 parameters) | 0.02 sec |
| **CIs** | ssdtools (HC1) | 15.72 sec; nboot = 500 |
| | | 42.12 sec; nboot = 1,000 |
| | | 251.52 se ; nboot = 5,000 |
| | | 434.29 sec; nboot = 10,000 |
| | Analytic method (HC1) | 0.03 sec |

While computational time is important, it is nevertheless a second-order consideration. More detailed simulation work is required to ascertain whether the procedure gives results comparable with bootstrapping in terms of: HCx bias; HCx precision; confidence interval coverage; and confidence interval width. A numerical summary for data generated from the Burr III distribution is shown in Table 3. Given this work is in its infancy and not a core activity of the current project, further investigations and analyses are required before any meaningful assessments can be made. However, based on the limited evidence provided in Table 3, the following observations can be made:

- Irrespective of dataset or sample size, there is a monotonic decreasein HCx CI coverage with increasing x. The reasons for this are not fully understood, although this is no doubt a function of: decreasing CI width with increasing x; and decreasing CI width with increasing x.

- HCx coverage is relatively unaffected by sample size.
- Bias is generally larger than the bootstrapping methods
- Bias is predominantly *positive* (i.e. HCx is *overestimated)* for $N \leq 32$ and *negative* (under-estimation) for $N = 40$.

Although this investigation was not a core requirement within the present scope of works, we mention it here as a 'placeholder' for future exploration should the issue of compute time become a limiting factor – particularly when using paid services such as shinyapps.io. Further investigations and evaluation of bias and performance would need to be undertaken before recommending this as an approved alternative estimation strategy.

**Table 3. Summary of analytical method of CI determination (see section 3 of this report). All results based on same 200 datasets generated from the BurrIII distribution as used in the bootstrap simulations of this report. Columns labelled p_HCx are the proportions of two hundred data sets for which the true HCx value was 'captured' by the CI; columns labelled w_x are average confidence interval widths; and bias_x are average bias values.**

| | dataset | p_HC1 | p_HC5 | p_HC10 | p_HC20 | w_1 | w_5 | w_10 | w_20 | bias_1 | bias_5 | bias_10 | bias_20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N=16 | aims_aluminium_marine | 100.0% | 88.6% | 71.6% | 54.5% | 119.1 | 150.7 | 153.3 | 174.3 | 33.0 | 75.7 | 121.9 | 161.4 |
| | anon_b | 98.8% | 89.0% | 72.0% | 51.2% | 2.8 | 3.3 | 3.4 | 10.8 | 1.1 | 1.8 | -0.6 | -46.4 |
| | anon_e | 97.5% | 92.5% | 87.5% | 82.5% | 111.3 | 179.7 | 281.5 | 621.9 | 5.9 | 21.5 | 51.0 | 133.0 |
| | anzg_metolachlor_fresh | 97.6% | 88.1% | 57.1% | 47.6% | 12.6 | 19.3 | 25.7 | 40.3 | 3.3 | 6.3 | 8.4 | -20.1 |
| | ccme_cadmium | 98.0% | 95.1% | 81.4% | 63.7% | 53254.1 | 54760.2 | 38.8 | 36.2 | 0.1 | 0.0 | -3.1 | -50.3 |
| | ccme_chloride | 90.8% | 93.8% | 66.2% | 52.3% | 222.5 | 275.4 | 264.0 | 276.4 | 76.4 | 149.4 | 222.2 | 318.0 |
| | ccme_uranium | 97.5% | 91.3% | 80.0% | 60.0% | 71.7 | 100.4 | 121.9 | 156.0 | 16.0 | 43.3 | 72.6 | 102.5 |
| | csiro_chlorine_marine | 98.6% | 92.9% | 65.7% | 54.3% | 23.4 | 28.2 | 25.6 | 26.4 | 8.7 | 16.5 | 21.3 | -10.3 |
| | **Average** | **97.3%** | **91.4%** | **72.7%** | **58.3%** | **6727.2** | **6939.7** | **114.3** | **167.8** | **18.1** | **39.3** | **61.7** | **73.5** |
| | | | | | | | | | | | | | |
| | **dataset** | **p_HC1** | **p_HC5** | **p_HC10** | **p_HC20** | **w_1** | **w_5** | **w_10** | **w_20** | **bias_1** | **bias_5** | **bias_10** | **bias_20** |
| N=24 | aims_aluminium_marine | 96.7% | 86.9% | 70.5% | 54.9% | 98.5 | 124.0 | 123.1 | 140.7 | 33.9 | 74.9 | 119.4 | 212.1 |
| | anon_b | 98.9% | 93.1% | 73.6% | 54.0% | 2.3 | 2.8 | 2.5 | 2.9 | 0.9 | 1.4 | 0.2 | -9.8 |
| | anon_e | 99.0% | 100.0% | 92.9% | 82.7% | 40.8 | 82.3 | 165.0 | 453.6 | 0.3 | 5.8 | 24.8 | 132.7 |
| | anzg_metolachlor_fresh | 100.0% | 92.5% | 68.8% | 60.0% | 3.2 | 6.0 | 11.1 | 29.4 | 0.4 | 1.4 | 2.1 | 4.4 |
| | ccme_cadmium | 97.6% | 77.4% | 68.5% | 57.3% | 98161.5 | 101295.8 | 21.1 | 25.6 | 0.1 | -0.3 | -2.3 | -14.0 |
| | ccme_chloride | 97.1% | 83.8% | 67.6% | 48.5% | 139.0 | 188.0 | 194.7 | 196.7 | 42.7 | 108.4 | 184.6 | 339.9 |
| | ccme_uranium | 98.9% | 81.8% | 59.1% | 47.7% | 45.4 | 63.5 | 75.6 | 96.1 | 14.0 | 34.3 | 58.9 | 121.2 |
| | csiro_chlorine_marine | 100.0% | 91.9% | 75.7% | 63.5% | 20.4 | 24.1 | 20.5 | 19.5 | 8.6 | 15.8 | 22.0 | 25.0 |
| | **Average** | **98.5%** | **88.4%** | **72.1%** | **58.6%** | **12313.9** | **12723.3** | **76.7** | **120.6** | **12.6** | **30.2** | **51.2** | **101.4** |
| | | | | | | | | | | | | | |
| | **dataset** | **p_HC1** | **p_HC5** | **p_HC10** | **p_HC20** | **w_1** | **w_5** | **w_10** | **w_20** | **bias_1** | **bias_5** | **bias_10** | **bias_20** |
| N=32 | aims_aluminium_marine | 100.0% | 89.1% | 75.2% | 60.6% | 78.3 | 100.4 | 98.8 | 114.5 | 27.6 | 67.0 | 113.5 | 214.2 |
| | anon_b | 98.0% | 83.8% | 68.7% | 55.6% | 1.7 | 2.2 | 2.0 | 2.2 | 0.5 | 1.3 | 2.2 | 3.8 |
| | anon_e | 100.0% | 98.7% | 93.6% | 73.7% | 25.1 | 53.3 | 111.1 | 315.6 | 0.2 | 4.9 | 22.8 | 127.4 |
| | anzg_metolachlor_fresh | 97.2% | 83.1% | 70.4% | 56.3% | 3.1 | 6.3 | 11.2 | 23.3 | 0.3 | 1.9 | 5.5 | 20.2 |
| | ccme_cadmium | 98.1% | 88.9% | 75.0% | 64.8% | 0.2 | 0.3 | 0.3 | 0.5 | 0.0 | 0.0 | 0.0 | 0.1 |
| | ccme_chloride | 98.8% | 83.8% | 70.0% | 56.3% | 124.1 | 164.0 | 173.4 | 177.2 | 35.1 | 98.4 | 175.7 | 334.4 |
| | ccme_uranium | 99.1% | 77.4% | 66.1% | 46.1% | 35.5 | 50.1 | 62.7 | 89.7 | 9.9 | 28.1 | 54.3 | 125.5 |
| | csiro_chlorine_marine | 98.9% | 77.5% | 58.4% | 47.2% | 16.9 | 19.0 | 16.8 | 15.2 | 6.2 | 14.4 | 22.7 | 37.0 |
| | **Average** | **98.8%** | **85.3%** | **72.2%** | **57.6%** | **35.6** | **49.4** | **59.5** | **92.3** | **10.0** | **27.0** | **49.6** | **107.8** |
| | | | | | | | | | | | | | |
| | **dataset** | **p_HC1** | **p_HC5** | **p_HC10** | **p_HC20** | **w_1** | **w_5** | **w_10** | **w_20** | **bias_1** | **bias_5** | **bias_10** | **bias_20** |
| N=40 | aims_aluminium_marine | 99.2% | 83.5% | 69.4% | 54.5% | 73.0 | 89.7 | 89.6 | 103.0 | 9.5 | -12.5 | -41.9 | -96.6 |
| | anon_b | 99.1% | 86.1% | 67.6% | 50.9% | 1.6 | 2.0 | 1.9 | 2.0 | -15.9 | -76.9 | -151.8 | -304.9 |
| | anon_e | 100.0% | 98.9% | 86.0% | 69.7% | 14.6 | 36.5 | 84.1 | 255.9 | -16.3 | -73.9 | -133.2 | -185.7 |
| | anzg_metolachlor_fresh | 99.0% | 83.7% | 71.4% | 56.1% | 2.0 | 4.4 | 8.1 | 19.2 | -16.1 | -76.8 | -149.6 | -291.9 |
| | ccme_cadmium | 98.0% | 82.2% | 75.2% | 61.4% | 0.2 | 0.2 | 0.3 | 0.4 | -16.4 | -78.1 | -153.9 | -308.8 |
| | ccme_chloride | 100.0% | 80.0% | 66.7% | 58.9% | 102.6 | 135.7 | 145.9 | 155.0 | 15.7 | 10.9 | 6.0 | 10.1 |
| | ccme_uranium | 99.2% | 87.0% | 62.6% | 54.5% | 29.9 | 49.7 | 62.6 | 84.8 | -10.6 | -53.7 | -99.0 | -178.2 |
| | csiro_chlorine_marine | 99.0% | 91.8% | 76.5% | 56.1% | 14.9 | 17.0 | 16.2 | 15.4 | -10.9 | -64.8 | -132.1 | -270.6 |
| | **Average** | **99.2%** | **86.6%** | **71.9%** | **57.8%** | **29.8** | **41.9** | **51.1** | **79.5** | **-7.6** | **-53.2** | **-106.9** | **-203.3** |

### 4.1.3 Task S2: Output customisation

The team has worked through the list of customisations that were identified and circulated last year amongs the project team. All have been implemented in a development version of the Shiny App hosted on the Poisson consulting website ( https://poissonconsulting.shinyapps.io/shinyssdtools-dev/) as well as a newly-created Australian prototype website (https://shinyssd.tools).

These customisations include:

- Ensuring greater consistency in the plots on both the Fit and Predict tabs
- Allowing edits to the x and y labels
- Adding drop down lists for selection of protection values in line with the ANZG (99, 95, 90, 80%) and autopopulating the corresponding fraction effected
- Resolving an issue that was preventing the dotted protection values not appearing for ssd_hp()
- Allowing adjustment of the x axis limits, as well as adding custom tick mark labels to the x axis
- Allowing adustment of axis title size

As mentioned above, the url https://shinyssd.tools has been registered by Environmetrics Australia and a prototype web-page site created. Further customisation and additional DECCEW/ANZG branding can be added as hosting issues are resolved. Screenshots of the updated development version of the Shiny App hosted on the Poisson consulting website for the Data, Fit and Predict tabs are shown in figures Figure 17, Figure 18 and Figure 19, respectively.

We have also included an additional button allowing the user to request a report to be generated from the customised shiny output (Figure 20). Once generated, the report can be saved as either an html, pdf or raw .Rmd file. An example pdf of the report is in Appendix C, using the Boron data and based on the defaults settings used. The format and content of the report mimics the original Burrlioz report, and outputs the default protection ANZG (99, 95, 90, 80%) protection values and a plot of the model averaged SSD. It also includes an additional plot showing the fit of all fitted distributions, as well as a table of their relative goodness of fit statistics. Further customisation options have been provided in line with feedback on earlier versions.

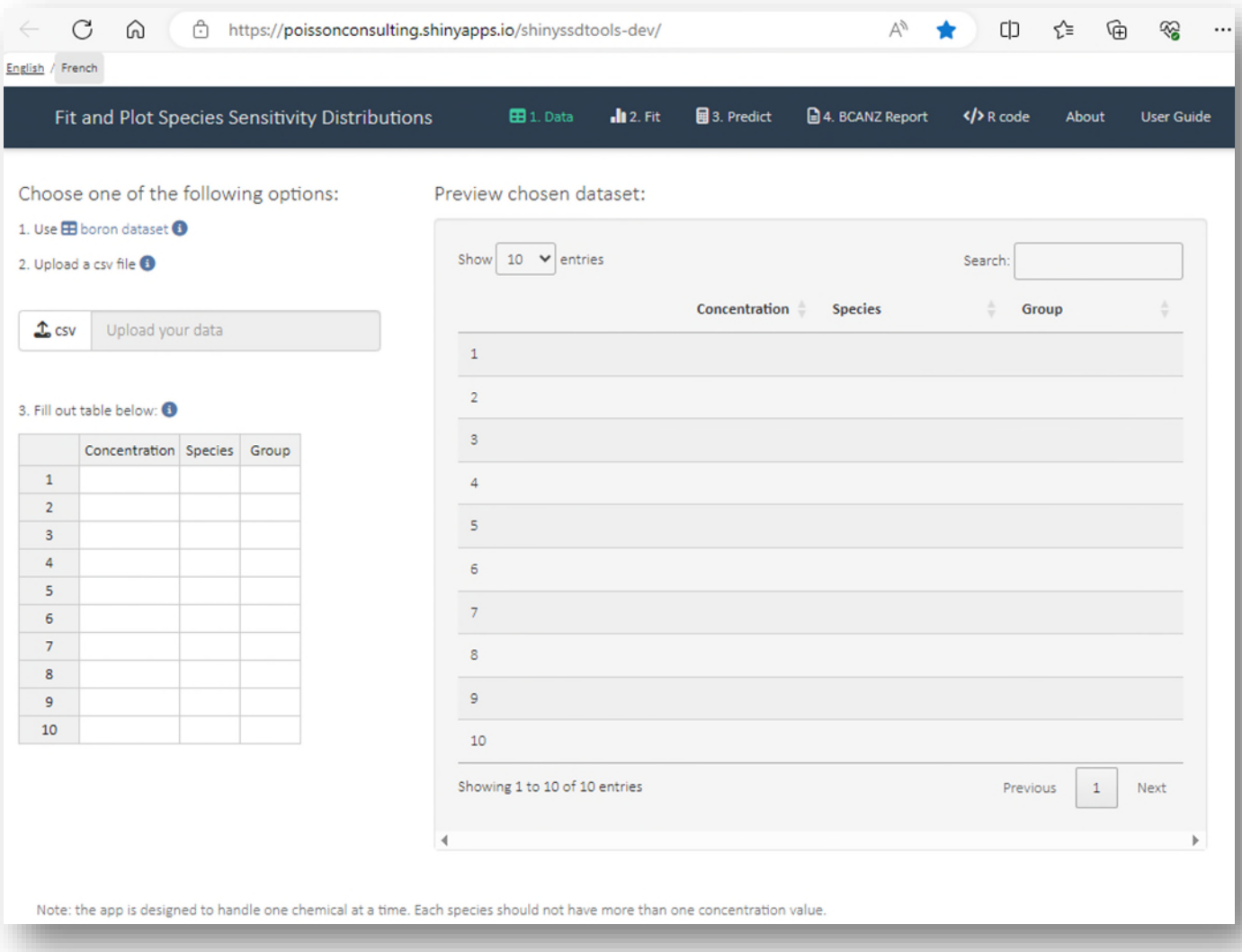



**Figure 17. Screenshot of the Data tab in the updated development version of Shiny App hosted on the Poisson consulting website.**

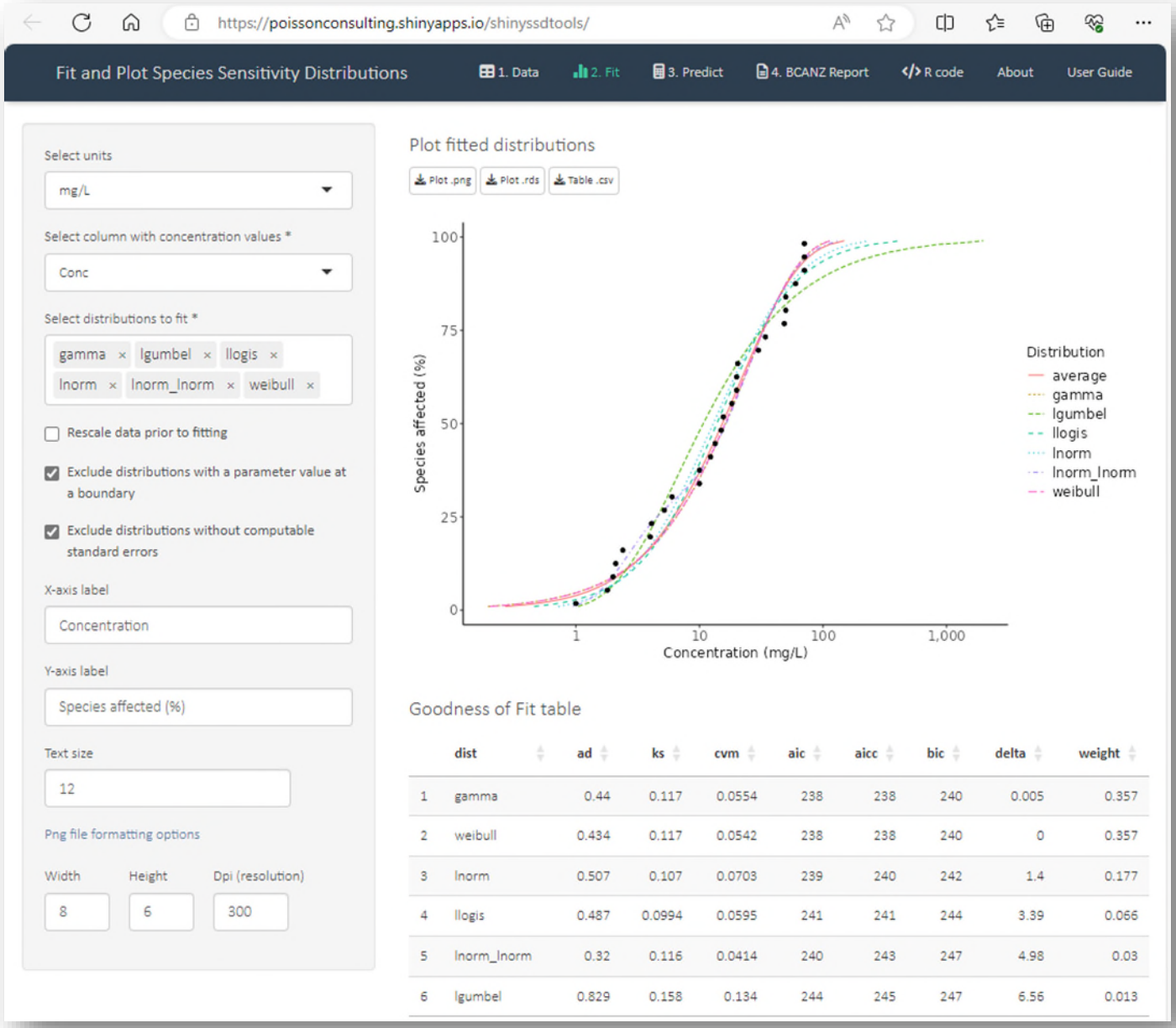



**Figure 18. Screenshot of the Fit tab in the updated development version of Shiny App hosted on the Poisson consulting website.**

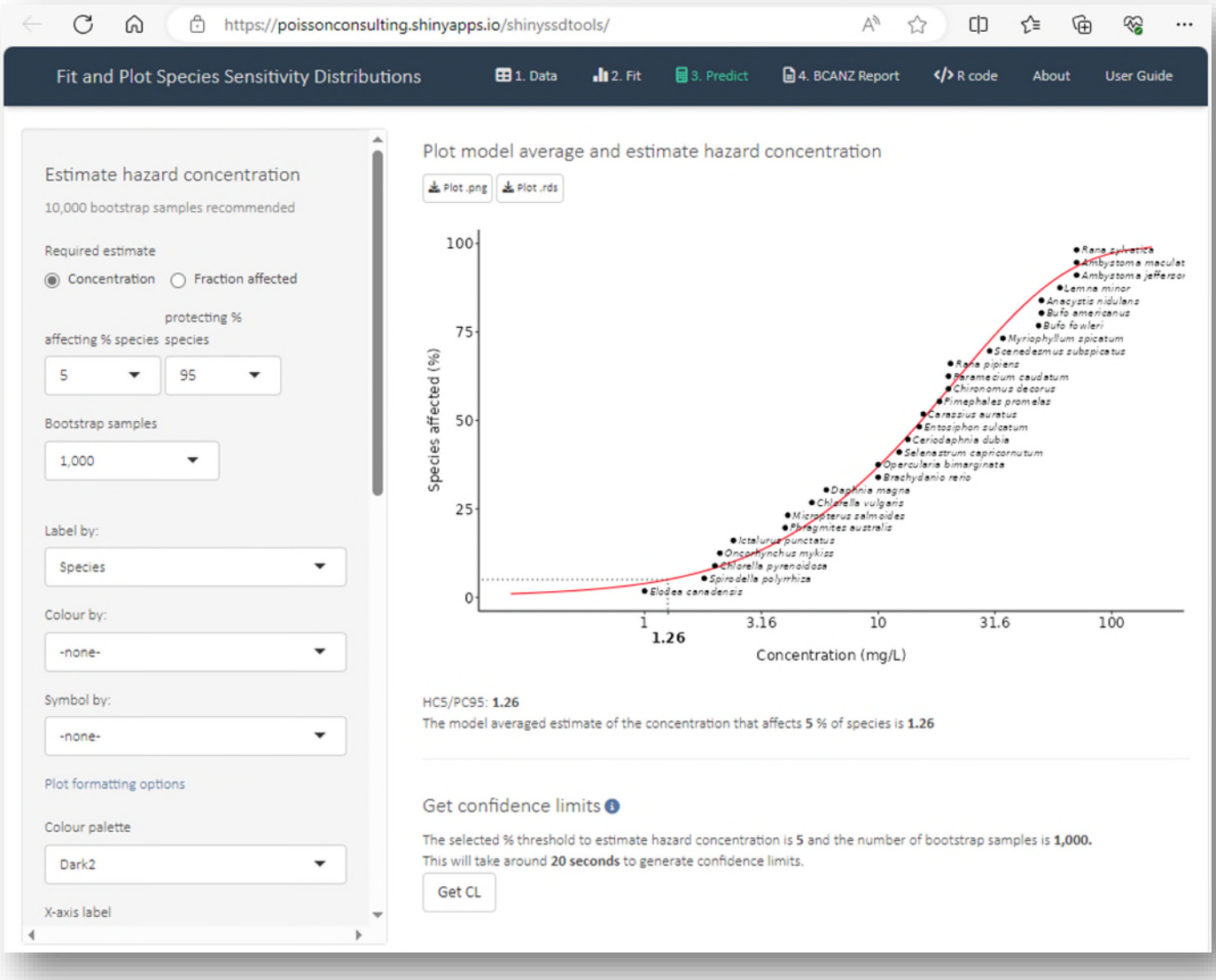



**Figure 19. Screenshot of the Predict tab in the updated development version of Shiny App hosted on the Poisson consulting website.**

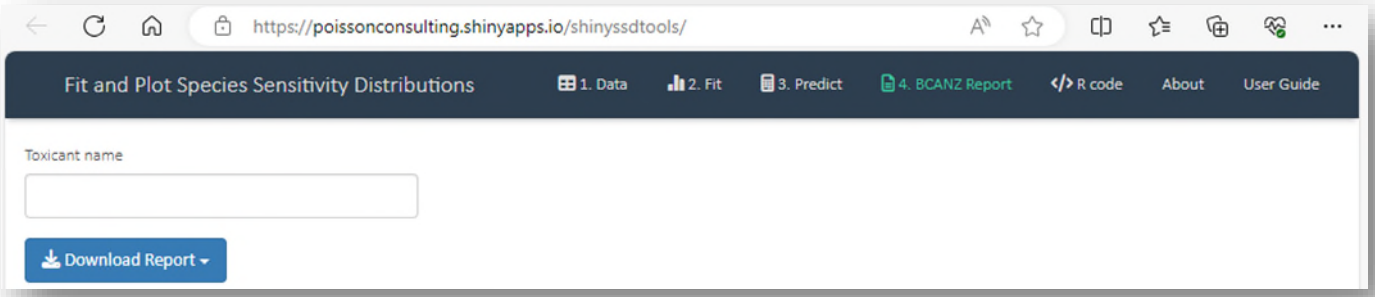



**Figure 20. Screenshot of the BCANZ Report tab in the updated development version of Shiny App hosted on the Poisson consulting website.**

### 4.1.4 Task S4: Update documentation

A detailed inventory of the existing documentation of the `ssdtools` `github` website (found at https://bcgov.github.io/ssdtools/index.html) was undertaken, and it was decided to re-work and rationalise the content to form a more coherent, and logically linked set of vignettes, and this has now been implemented in the development version of `ssdtools`.

The documentation in the development version now includes the following vignettes (see Appendices H-M):

1. **'Getting Started with ssdtools'** – This is a basic overview of the functionality of the `ssdtools` package, with simple examples of usage. This includes a basic introduction and the philosophy behind SSD modelling; instructions on installing, getting help, inputting data and fitting distributions; instructions for obtaining coefficients, plotting, selecting a distribution and obtaining model averaged outputs, estimating the fit and obtaining hazard and protection values; and finally, a description of the censoring capabilities in `ssdtools`.

2. **'Model averaging'** – This includes a digestible explanation/background of what model averaging does and how the revised version of `ssdtools` implements model averaging to estimate HC and HP by treating the model set as a mixture and using the uniroot function to derive HC/PP values from the model-averaged *cdf*. Much of the content in section 4.1.1 overlaps with this vignette.

3. **'Distributions in ssdtools'** – This includes some background on the distributions for SSD modelling, including the distributions adopted in the original version of `ssdtools`, the introduction of the Burr Type III distribution adopted in Burrlioz, as well as an information on the value of using mixture distributions for bi-modal datasets. The set of recommended default distributions is explained (see Figure 21) and there is also a detailed account of all the distributions that are available in `ssdtools` which includes information on stability (if relevant) and discussion around their use in SSD modelling.

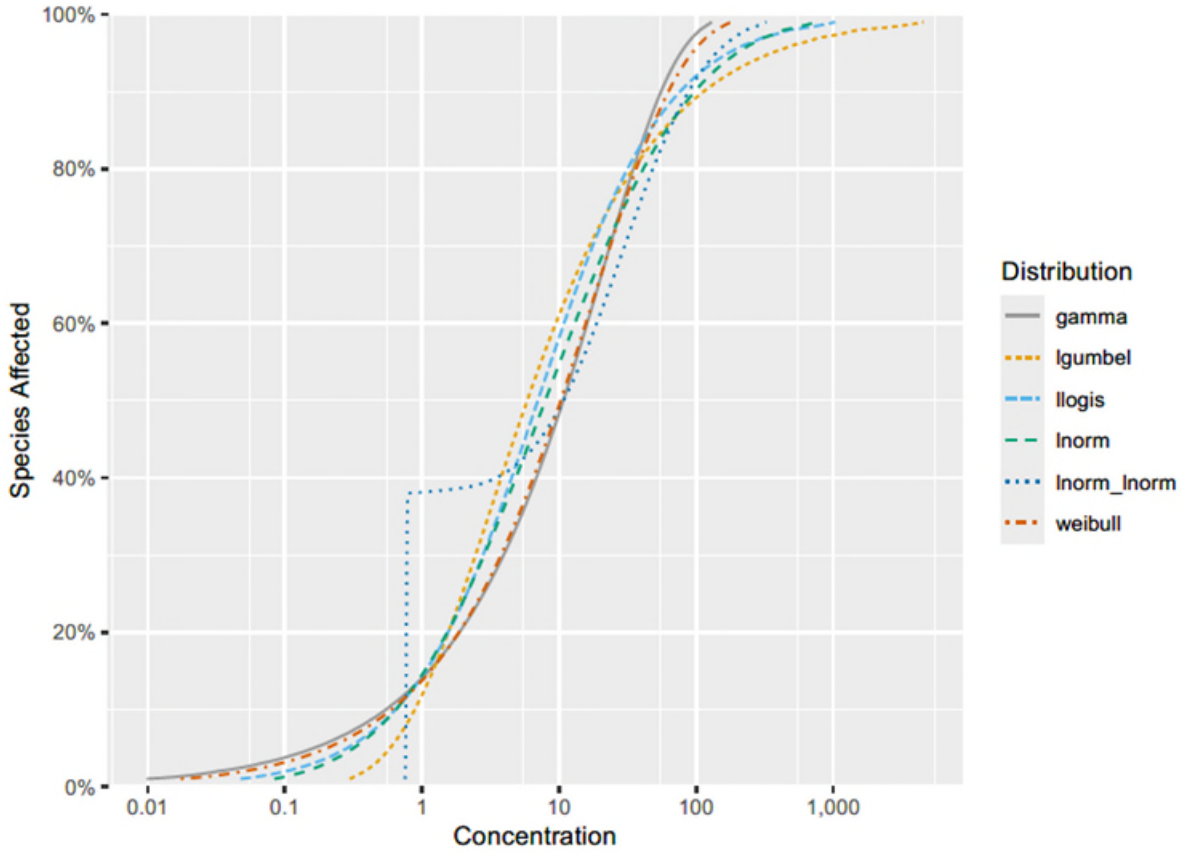



**Figure 21. The default list of candidate distributions in `ssdtools`, comprised of the following: log-normal; log-logistic; gamma; inverse Weibull (log-Gumbel); Weibull; and mixture of two log-normal distributions. Shown are these default distributions plotted with a mean of 2 and standard deviation of 2 on the (natural) log concentration scale or around 7.4 on the concentration scale.**

4. '**Confidence intervals in ssdtools'** – This includes some basic information on bootstrapping and recommendations of the number of bootstrap samples that should be used. A description of parametric versus non-parametric bootstrapping (this is a fundamental difference between the default behaviour of `ssdtools` and the original Burrlioz software) and a justification for the default parametric bootstrapping. In addition, there is a description of the three parametric bootstrapping methods now implemented in the revised version of `ssdtools` following investigations S1 & S3, with an explanation of how these differ and information on which methods should be used, with justification.

5. '**Embellishing plots'** – This explains how to make a simple plot of the cumulative distribution, along with options for customisation, including how to add the model averaged fit, confidence interval ribbons, and data ranges for censored data (see Figure 22).

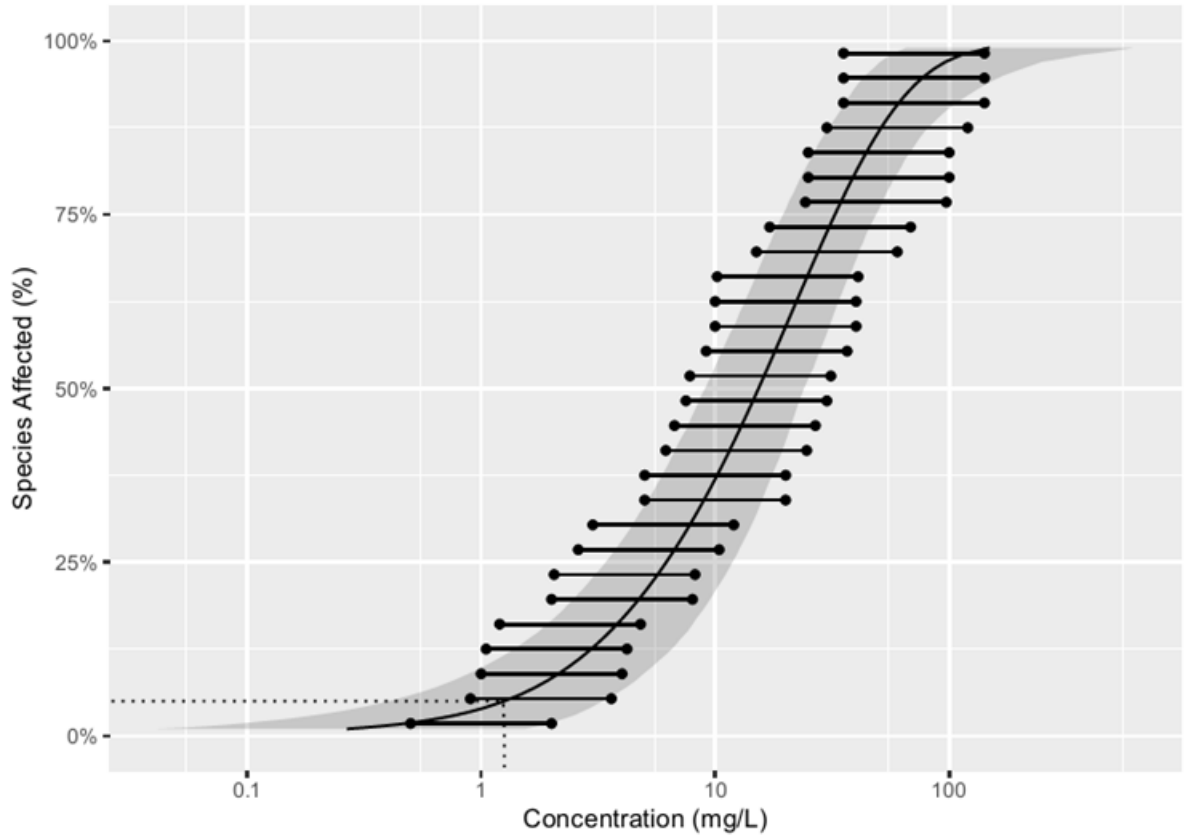



**Figure 22. Example of possible plot embellishments available through ssdtools and the ggplot2 packages in R. The plot shows the range of censored species sensitivity data, with 95% confidence interval ribbons; the fitted model averaged ssd and interpolated HC5.**

Examples from the existing vignette have been retained, including how to plot multiple SSD's together, and embellish a plot with an exposure distribution.

6. '**Additional technical details'** – This vignette is a home for important technical details that are useful to include, but that are too technical for the other vignettes. This vignette will provide important technical details for more advanced users seeking additional mathematical and statistical detail about `ssdtools`. The vignette currently includes a short summary of small sample bias in maximum likelihood estimation with a link to a detailed pdf, as well as technical information on the inverse Pareto and inverse Weibull distributions as limiting distributions of the Burr III distribution.

### 4.1.5 Task S5: Review 'stable" and 'unstable' distributions

*Unstable distributions*

In our previous project (Fox et al 2021) we identified a range of stability issues with various distributions currently implemented in `ssdtools` (

Figure 23).

| sample_size | source/parent | gamma | invpareto | lgumbel | llogis | lnorm | burrIII3 | gompertz | llogis_llogis | lnorm_lnorm | weibull |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | A) Simulation study 1 (log-logistic, inverse Weibull, log-normal) | | | | | | | | | | |
| 8 | inverse.weibull | 0.98 | 0.98 | 1.00 | 1.00 | 1.00 | 0.02 | 0.61 | 0.64 | 0.52 | 0.65 |
| 8 | log.logistic | 0.99 | 0.99 | 1.00 | 1.00 | 1.00 | 0.07 | 0.63 | 0.66 | 0.60 | 0.66 |
| 8 | lognormal | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.38 | 0.25 | 0.81 | 0.79 | 0.47 |
| 16 | inverse.weibull | 0.98 | 0.98 | 1.00 | 1.00 | 1.00 | 0.04 | 0.62 | 0.56 | 0.47 | 0.61 |
| 16 | log.logistic | 0.99 | 0.99 | 1.00 | 1.00 | 1.00 | 0.13 | 0.64 | 0.53 | 0.47 | 0.67 |
| 16 | lognormal | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.72 | 0.14 | 0.70 | 0.64 | 0.34 |
| 32 | inverse.weibull | 0.97 | 0.99 | 1.00 | 1.00 | 1.00 | 0.04 | 0.63 | 0.56 | 0.46 | 0.63 |
| 32 | log.logistic | 0.97 | 0.99 | 1.00 | 1.00 | 1.00 | 0.18 | 0.65 | 0.49 | 0.48 | 0.66 |
| 32 | lognormal | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.90 | 0.07 | 0.59 | 0.53 | 0.22 |
| 64 | inverse.weibull | 0.99 | 0.98 | 1.00 | 1.00 | 1.00 | 0.04 | 0.62 | 0.61 | 0.47 | 0.65 |
| 64 | log.logistic | 0.99 | 0.99 | 1.00 | 1.00 | 1.00 | 0.20 | 0.65 | 0.41 | 0.41 | 0.68 |
| 64 | lognormal | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.98 | 0.02 | 0.53 | 0.42 | 0.16 |
| 128 | inverse.weibull | 0.99 | 0.97 | 1.00 | 0.99 | 1.00 | 0.05 | 0.62 | 0.72 | 0.47 | 0.65 |
| 128 | log.logistic | 1.00 | 0.95 | 1.00 | 1.00 | 1.00 | 0.21 | 0.64 | 0.38 | 0.40 | 0.69 |
| 128 | lognormal | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.01 | 0.50 | 0.36 | 0.11 |
| 1000 | inverse.weibull | 0.96 | 0.84 | 1.00 | 0.99 | 0.89 | 0.02 | 0.60 | 0.95 | 0.68 | 0.67 |
| 1000 | log.logistic | 0.96 | 0.81 | 1.00 | 1.00 | 0.90 | 0.21 | 0.63 | 0.35 | 0.53 | 0.68 |
| 1000 | lognormal | 1.00 | 1.00 | 1.00 | 1.00 | 0.86 | 1.00 | 0.00 | 0.78 | 0.33 | 0.05 |
| | B) Simulation study 2 (Johnson family) | | | | | | | | | | |
| 100 | anon_a | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.90 | 0.00 | 0.80 | 0.82 | 0.00 |
| 100 | anon_b | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.04 | 0.80 | 0.60 | 0.04 |
| 100 | ccme_boron | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.14 | 0.88 | 1.00 | 0.98 | 1.00 |
| 100 | ccme_glyphosate | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.00 | 0.88 | 0.88 | 0.00 |
| 100 | ccme_silver | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.04 | 0.90 | 0.92 | 1.00 |
| 100 | ccme_uranium | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.16 | 0.00 | 0.64 | 0.56 | 0.00 |
| 16 | anon_a | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.54 | 0.00 | 0.90 | 0.82 | 0.00 |
| 16 | anon_b | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.62 | 0.16 | 0.64 | 0.58 | 0.38 |
| 16 | ccme_boron | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.02 | 0.68 | 0.90 | 0.92 | 1.00 |
| 16 | ccme_glyphosate | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.04 | 0.00 | 0.84 | 0.88 | 0.00 |
| 16 | ccme_silver | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.18 | 0.26 | 0.80 | 0.82 | 1.00 |
| 16 | ccme_uranium | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.42 | 0.00 | 0.74 | 0.68 | 0.00 |
| 49 | anon_a | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.76 | 0.00 | 0.82 | 0.82 | 0.00 |
| 49 | anon_b | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.94 | 0.06 | 0.64 | 0.56 | 0.22 |
| 49 | ccme_boron | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.26 | 0.84 | 0.92 | 0.92 | 1.00 |
| 49 | ccme_glyphosate | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.02 | 0.00 | 0.88 | 0.92 | 0.00 |
| 49 | ccme_silver | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.04 | 0.04 | 0.86 | 0.90 | 1.00 |
| 49 | ccme_uranium | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.30 | 0.00 | 0.58 | 0.60 | 0.00 |
| 8 | anon_a | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.24 | 0.00 | 0.84 | 0.82 | 0.00 |
| 8 | anon_b | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.50 | 0.46 | 0.84 | 0.86 | 0.66 |
| 8 | ccme_boron | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.14 | 0.64 | 0.96 | 0.96 | 1.00 |
| 8 | ccme_glyphosate | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.10 | 0.00 | 0.88 | 0.92 | 0.00 |
| 8 | ccme_silver | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.16 | 0.38 | 0.82 | 0.82 | 1.00 |
| 8 | ccme_uranium | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.24 | 0.00 | 0.80 | 0.80 | 0.00 |

**Figure 23. The proportion of simulated datasets/iterations for which the ssdtools -fitted distributions were able successfully converge. Note that for the Burr III and both mixture distributions, this proportion includes distributions that may have returned a result, but this was at one of the bounds of the parameter set (i.e. shape1 or shape2 = 20 or 0.05 in the case of Burr III, or p=<0.2 in the case of mixtures). Reproduced from Figure 32, Fox et al 2022.**

These included occasional issues with the Inverse Pareto and Burr III (Figure 23). There were also surprisingly sometimes convergence issues with the lognormal distribution for very high samples sizes. Here we Investigate and/or discuss 'all' known convergence issues with the distributions in `ssdtools`.

### *Burr III*

Numerical instability in the Burr III distribution is well known and has been investigated thoroughly in previous work by the team. The instability is due to the high degree of collinearity between parameter estimates and/or relatively flat likelihood profiles (Fox et al. 2021), and is one of the motivations behind the logic coded into Burrlioz to revert to either of the two limiting distributions. The Burr III distribution is not currently one of the recommended distributions in the default model set. This is because of 1) unresolvable convergence issues associated with reverting to the limiting forms of the Burr III distribution; 2) the fact that reverting to a limiting two parameter distribution does not fit easily within a model averaging framework; and 3) that one of the two limiting distributions (the inverse Pareto, see below) also has estimation and convergence issues.

### *Inverse Pareto*

While the inverse Pareto distribution is implemented in the Burrlioz 2.0 software, it is important to understand that it is done so only as one of the limiting Burr distributions (see above for details). The inverse Pareto is not offered as a stand-alone option in the Burrlioz 2.0 software. We have spent considerable time and effort exploring the properties of the inverse Pareto distribution, including deriving bias correction equations and alternative methods for deriving confidence intervals (Fox et al. 2021). This work has substantial value for improving the current Burrlioz 2.0 method, and our bias corrections should be adopted when deriving HCx estimates from the inverse Pareto where parameters have been estimated using maximum likelihood.

As is the case with the Burrlioz 2.0 software, we have decided not to include the inverse Pareto distribution in the default candidate set in `ssdtools` although it is offered as a user-selectable distribution to use in the model-fitting process. This decision is based largely on the fact that the distribution is mathematically bounded by the maximum value in the input data and is thus inappropriate for use as a stand-alone SSD.

### *Gompertz*

Prior testing has shown that the Gompertz distribution can have poor convergence behaviour (see Figure 32 in Fox et al 2021). Further, even with the same dataset, `ssdtools` can return different parameter estimates based on different seeds, which suggests a very high degree of instability, even where the distribution has successfully fitted (see https://github.com/bcgov/ssdtools/issues/223).

Investigations into some of these failed fits suggests that there are inherent difficulties with fitting the gomptertz distribution to some datasets that occur within `ssdtools` as well as alternative packages, such as `fitdistrplus` (see Figure 24. Example of a dataset showing instability for the Gompertz distribution using (a) `ssdtools` and (b) `fitdistrplus`. The `ssdtools` fit did not achieve convergence and although the `fitdistrplus` package did achieve convergence, the fit it is identical (apart from the rescaling of the horizontal axis) to the fit from `ssdtools`. In both cases the fit is poor.

Figure 25. `fitdistrplus` profile likelihood plot as a function of the log(location) and log(scale) parameters of the fitted Gompertz distribution in Figure 24(b). The ridge (indicated by the white region) suggests an infinite number of plausible fits. The likelihood for the current fit is indicated by the 'X'.We believe this is most likely due to very flat likelihood profile, suggesting there are an infinite number of equally plausible estimates (Figure 25). This issue appears to associated with the fact that the testing data do not fit within plausible parameter combinations of the gompterz distribution (see more detailed information in Appendix D). In such cases, it may not be possible to ever resolve convergence issues. Failure to converge in the initial model set would simply suggest it is a poor model

for that data. Where the gompterz fits well, it is possible it might be considered, providing pboot values are high in confidence interval estimation.

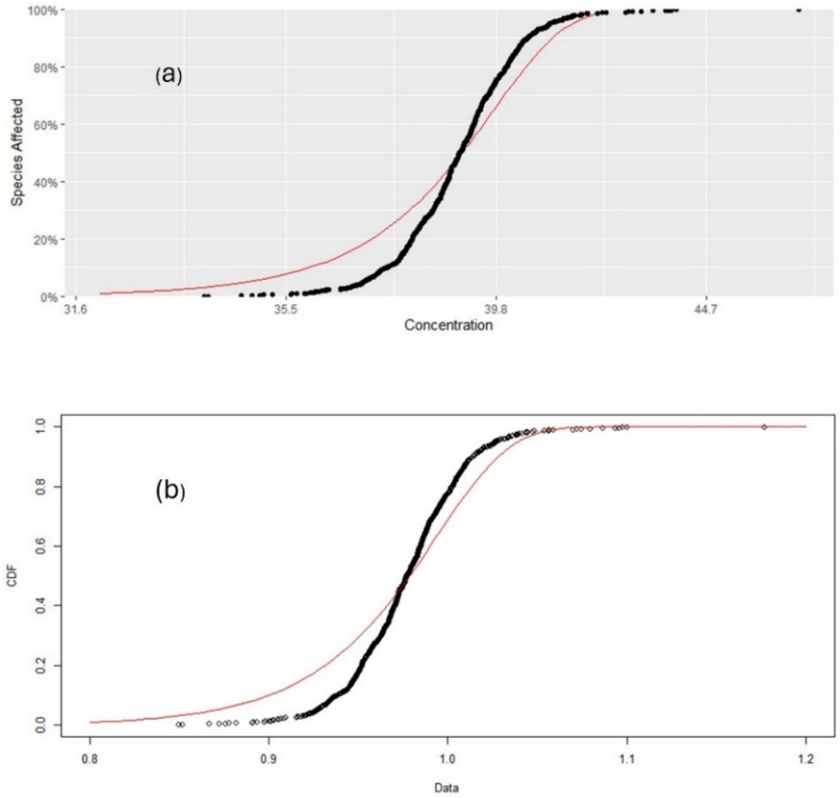



**Figure 24. Example of a dataset showing instability for the Gompertz distribution using (a) `ssdtools` and (b) `fitdistrplus`. The `ssdtools` fit did not achieve convergence and although the `fitdistrplus` package did achieve convergence, the fit it is identical (apart from the rescaling of the horizontal axis) to the fit from `ssdtools`. In both cases the fit is poor.**

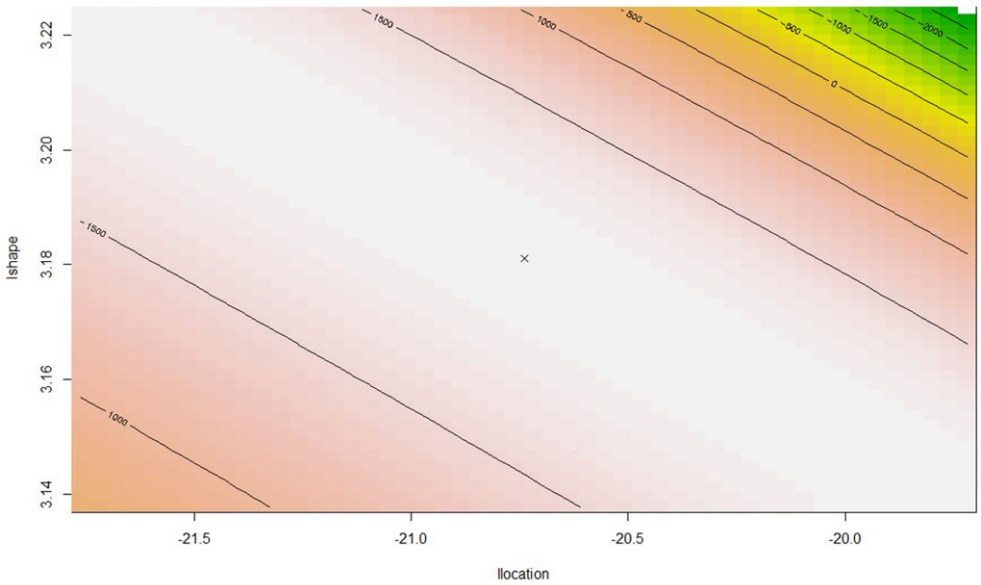



**Figure 25. `fitdistrplus` profile likelihood plot as a function of the log(location) and log(scale) parameters of the fitted Gompertz distribution in Figure 24(b). The ridge (indicated by the white region) suggests an infinite number of plausible fits. The likelihood for the current fit is indicated by the 'X'.**

To examine if convergence is reliable once a Gompertz distribution has been fit, we extracted datasets from Simulation study 1 and examined pboot values for all fits that were able to return a successful fit to the Gompertz distribution. We found that there were highly variable outcomes for pboot for successfully fit Gompertz models (Figure 26). Values of pboot for extremely large sample sizes (N=1,000) were always below 0.95. For all sample sizes <128 median pboot was > 0.95 (Figure 26). In all cases however, there were examples where pboot values were extremely low, suggesting there are pervasive convergence issues when bootstrapping the Gompertz, even when data are generated using a Gompertz distribution.

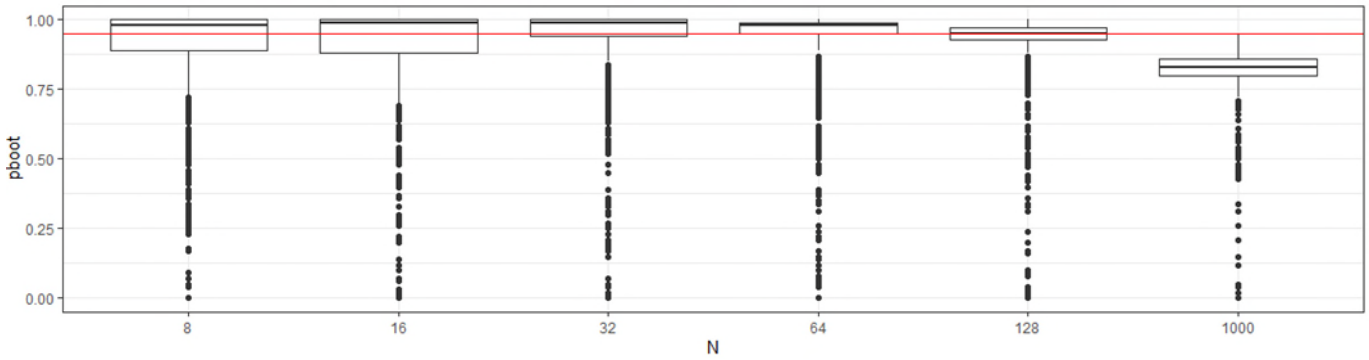



**Figure 26. Convergence propabilities (pboot) as a function of sample size (N) for all datasets from Simulation study 1 from Fox et al. 2021 for which the Gompertz distribution was able to successfully fit.**

In addition to the observed instabilities in bootstrapping, the Gompertz can also fail to converge to the same dataset when a different seed is used to call the optimisation algorithm.

To illustrate, we refited the example data from the posted github issue (https://github.com/bcgov/ssdtools/issues/223) using a range of different seeds, using the following R code:

```
x <- c(3.15284072848962, 1.77947821504531, 0.507778085984185, 1.650387414067,
1.00725113964435, 7.04244885481452, 1.32336941144339, 1.51533791792454)

test_issues_dat_seed <- lapply(1:100, FUN = function(s){

            set.seed(s)

fit <- try(ssd_fit_dists(data, left = 'Conc', dists = "gompertz", rescale =
FALSE), silent = TRUE)

})
```

Of these 100 seeds 31 were able to return a valid fit, with the remaining 69 failing. We extracted HC estimates from the successful fits via:

```
HC_vals <- sapply(test_issues_dat_seed, FUN = function(g){

  if(class(g)=="fitdists"){return(ssd_hc(g)$est)}

})
```

The estimated HC5 values ranged from 0.115 to 0.376 across the 31 successful fits, with a median of 0.134. This represents a 1.9 fold difference in the HC estimate, from a single dataset based entirely on differences among seeds. Such behaviour is clearly problematic in the context of scientific

reproducibility, and suggests that if the gompertz were implemented it would be necessary as a minimum to set a seed to obtain consistent results.

### *Weibull*

The team revisited the data from simulation study 1 using the most revent development version of `ssdtools` and were unable to reproduce the originally observed instability reported in Table 4 of Fox et al. (2021). We therefore conclude that other edits to the `ssdtools` codebase since the 2021 report have also resolved this instability issue, and the Weibull can be considered a stable distribution and should be included in the default model set.

**Table 4. Convergence probabilities for Simulation study 1 data, for the current working version of ssdtools**

| N | gamma | lgumbel | llogis | lnorm | lnorm_lnorm | weibull |
|---|---|---|---|---|---|---|
| 8 | 0.996581 | 1 | 0.99962 | 1 | 0.694149 | 1 |
| 16 | 0.990881 | 1 | 1 | 1 | 0.783435 | 1 |
| 32 | 0.992781 | 0.99924 | 0.99962 | 1 | 0.824468 | 1 |
| 64 | 0.990881 | 0.99962 | 0.99886 | 0.992021 | 0.842705 | 1 |
| 128 | 0.995821 | 0.99886 | 0.99962 | 0.988982 | 0.87538 | 0.99886 |
| 1000 | 0.985182 | 1 | 0.9981 | 0.977584 | 0.885638 | 0.99848 |

### *lognormal*

Somewhat paradoxically, prior testing has shown that the lognormal distribution can exhibit convergence issues when sample sizes are very high (see Figure 32, Fox et al. 2021). For example, from Simulation study 1 the lognormal distribution failed to fit up to 14% of the time when N=1,000 for data simulated from an Inverse Weibull, log logistic and lognormal distributions (see Figure 32, Fox et al. 2021).

Investigation into this issue revealed that the non-convergence of the `lnorm` for large samples sizes (e.g. 1,000) is because the initial values which are estimated from the input data were set equal to the MLEs (see more details in Appendix E). This causes issues with the specific optimization algorithm utilized in `ssdtools` via the R package `TMB`. The source of the problem appears to come from some deep-seated issue in the old (Fortran 77) solvers being used by TMB. The current solution is to adjust all initial values prior to passing to the optimization engine, to ensure they are not the exact MLE solutions. This approach successfully resulted in much higher convergence rates for the Study 1 simulation data from Fox et al. 2021 (Table 5). From a practical perspective this issue is minor as it impacts only very large sample size SSDs  is highly unlikely to be an issue for the sample sizes typically encountered for SSD modelling in ecotoxicology (e.g. <100).

**Table 5. Convergence probabilities for the Simulation study 1 datasets from Fox et al. 2021, for the lognormal distribution, across a range of sample sizes.**

| 8 | 16 | 32 | 64 | 128 | 1000 |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0.990502 | 0.985562 | 0.97758 |

This solution is a temporary fix as it does nothing to address the underlying issue. In the case where closed form solutions for ML estimation exist it would be possible to implement these for the relevant distributions using binary logic that by-passes the solver and just returns the exact values. Such a solution would require significant re-working of the `ssdtools` code base and is beyond the scope of the present work. The temporary work-around that has been implemented is technically unsatisfactory, but nevertheless ensures reliable fits with accurate MLE estimates and can be adopted safely in the interim.

*Lognormal-lognormal mixture*

*See Appendix F and Task M2 below.*

### 4.1.6 Task S6: Fix documentation for ssd_hc() and ssd_hp()

The documentation for `ssd_hc()` and `ssd_hp()` in the `ssdtools` helpfiles has been updated (see Appendix G). The detail contained in the text now reads:

*'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.'*

### 4.1.7 Task M1: Integrity checks

*Decision rules for minimum data requirements with and without mixtures.*

The lognormal-lognormal has a total of 5 parameters – a $\mu$ and $\sigma$ value for each of the 2 component distributions and a mixing proportion $p_{mix}$. As a rule-of-thumb[1], the absolute minimum number of data points required to fit a 5 parameter distribution should be n ≥ 3k+1, where k is the number of estimated parameters, which in the case of the lognormal-lognormal mixture suggests a minimum of sample size of 16 uncensored data points. Preferablly there would be n ≥ 5k+1 [1], which for the lognormal-lognormal mixture suggests a preferred samples size of at least 26. These guidelines have

[1] Prof. David Fox, *pers comm*.

been included in the lognormal-lognormal mixture section of the updated distributions vignette in ssdtools.

These guidelines have have ramifications in the context of consistency across distributions. For the two parameter distributions this would result in a minimum of 7 which is slightly higher than the current values across both jurisdictions. Burriloz currently requires only 5, and the CRAN 1.0.6 version of ssdtools requires 6. For a three parameter distribution distribution (ie the Burr III) the required minimum sample size would be 10, which is also greater than is currently required by Burrlioz to fit a Burr III distribution (N=9).

While we have made recommendations on required and preferred sample sizes, these should form the basis of further discussions. A final decision on minimum and preferred samples sizes is a matter that should be referred to the technical committee, preferably prior to the revision of Warne et al. 2018.

*Inclusion/exclusion rules based on data attributes (e.g. left and right censoring).*

Interval and left censoring is currently implemented in the current CRAN version 1.0.6 of ssdtools. Censoring can be specified by providing a data set with one or more rows that have:

- a finite value for the left column that is smaller than the finite value in the right column (interval censored)

- a zero or missing value for the left column and a finite value for the right column (left censored)

It is currently not possible to fit distributions to data sets that have an infinite or missing value for the right column and a finite value for the left column (right censored)

Rows that have a zero or missing value for the left column and an infinite or missing value for the right column (fully censored) will result in an error. Right censored data may occur when a CR experiment fails to return a valid toxicity estimate because there was no response at the highest treatment, and can occur. We have an open issue to implement ssdtools with right censored data (not just interval and left censored (see https://github.com/bcgov/ssdtools/issues/207)) but it is not a deliverable under the current contract.

For uncensored data, Akaike Weights are calculated using AICc (which corrects for sample size).

In the case of censored data, Akaike Weights are calculated using AIC (as the sample size cannot be estimated) but only if all the distributions have the same number of parameters (to ensure the weights are valid). This is a challenge for the mixture distributions because the number of rows is not the number of datapoints when there are censored data. A solution is to impose restrictions on the distribution sets for which censored data are allowed to ensure all are of the same parameter count, and/or warnings when AIC is used instead of AICc.

Australia has not historically accommodated censoring through the Burrlioz software, and for now there is no reason for this to be adopted prior to implementation in Australia. Any final decisions on this topic should be deferred to the technical committee for discussion.

### 4.1.8 Task M2: Convergence issues with the lnorm-lnorm distribution

Prior testing has shown that the lognormal-lognromal mixture distribution can have poor convergence behaviour in `ssdtools` (see Figure 32, Fox et al 2021). This behaviour can result in low `pboot` values being returned during bootstrapping of some distributions (see the related issue at https://github.com/bcgov/ssdtools/issues/295). To ensure reasonable behaviour, `ssdtools` has lowered the `pboot` tolerance for the bcanz function to ensure the lognormal-lognormal mixture distirbution to ensure its inclusion. The `pboot` argument was introduced when the backend of ssdtools was converted to TMB. This parameter tracks the number of successfullly converging bootstrap samples, and was initially set at 99, meaning the number of non-convergent samples that would be tolerated was no more than 1%.. In hindsight, was overly conservative, and we note that neither the original implentation of `ssdtools`. Nor `Burrlioz` 2.0 track bootstrap convergence. This default cut-off value has now been lowered to 95% (5%). non-convergence rate However, lowering `pboot` has the potential to intoduce bias in the HC and HP confidence interval estimation. We examined the stability of the lnorm-lnorm mixture distribution using a range of testing data, and evaluated different strategies for improving convergence reliability so this mixture distribution can be included with confidence in the default set when the minimum sample size requirements are met (see M1, section 4.1.7).

Note that the primary purpose of improving stability of the lognormal-lognormal mixture distribution is to ensure sucessful bootstrap sampling and maximize the returned value of `pboot` to minimise the possibility of potential bias in HC and HP confidence interval estimates. Failure of the lognormal-lognormal mixture distribution to fit initially to the original data is of less concern, because the mixture will simply be left out of the model set. In that case, failure to converge simply suggests that the mixture distribution is inappropriate for these data.

Investigations suggested that in general poor convergence is usually related to issues with the `optim()` solver used by `ssdtools` when attempting to fit a lognormal mixture model to data that display little or no bi-modality. In such instances, we would expect the estimated `pmix` to be close to zero or one, however ssdtools fails with an error message. Importantly, there are a range of other solvers outside of ssdtools that are able to obtain valid parameter estimates for lognormal-lognormal mixtures distributions even when the `pmix` parameter is near the boundary (see Table 6, p5).

**Table 6. Paramater estimates (p1-p5) and convergence statistics for a lognormal-lognormal mixture distirbution fitted to an example dataset that ssdtools fails to fit (see Appendix F) obtained across all available methods available in the R package optimx.**

| | p1 | p2 | p3 | p4 | p5 | value | fevals | gevals | niter | convcode | kkt1 | kkt2 | xtime |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| L-BFGS-B | 2.44 | 0.02 | 1.05 | 1.03 | 1.00 | -98.60 | 77 | 77 | NA | 52 | FALSE | NA | 0.11 |
| nlminb | 2.44 | 0.02 | 2.09 | 1.14 | 1.00 | -99.55 | 166 | 1154 | 150 | 1 | FALSE | NA | 0.09 |
| spg | 2.54 | 0.00 | 2.44 | 0.02 | 0.00 | -94.88 | 396 | NA | 363 | 0 | NA | NA | 0.24 |
| Rcgmin | 2.17 | 1.58 | 2.61 | 0.16 | 0.00 | 2067.25 | 41 | 17 | NA | 0 | FALSE | FALSE | 0 |
| Rvmmin | 2.15 | 3.74 | 2.44 | 0.03 | 0.00 | -12.93 | 41 | 21 | NA | 21 | FALSE | FALSE | 0 |
| bobyqa | 2.51 | 0.18 | 1.10 | 1.02 | 0.92 | 1796.39 | 28 | NA | NA | 0 | FALSE | FALSE | 0 |
| nmkb | 2.44 | 0.02 | 2.68 | 0.34 | 1.00 | -100.40 | 796 | NA | NA | 0 | FALSE | NA | 0.19 |
| hjkb | 2.44 | 0.04 | 2.44 | 0.02 | 0.04 | -103.31 | 2928 | NA | 19 | 0 | FALSE | TRUE | 0.33 |

We undertook a series of tests to evaluate the effectiveness of different strategies for improving the convergence of the lognormal-lognormal mixture distribution. Full details are reported in Appendix F. We started by re-factoring the ssdtools code such that the `pmix` paramater of the lognormal-lognormal mixture distribution is explicitly bounded at 0 and 1, instead of being modelled on the logit scale as is done in ssdtools version 1.0.6. This resulted in a 20% improvement in convergence success (see red bars, Figure 27).

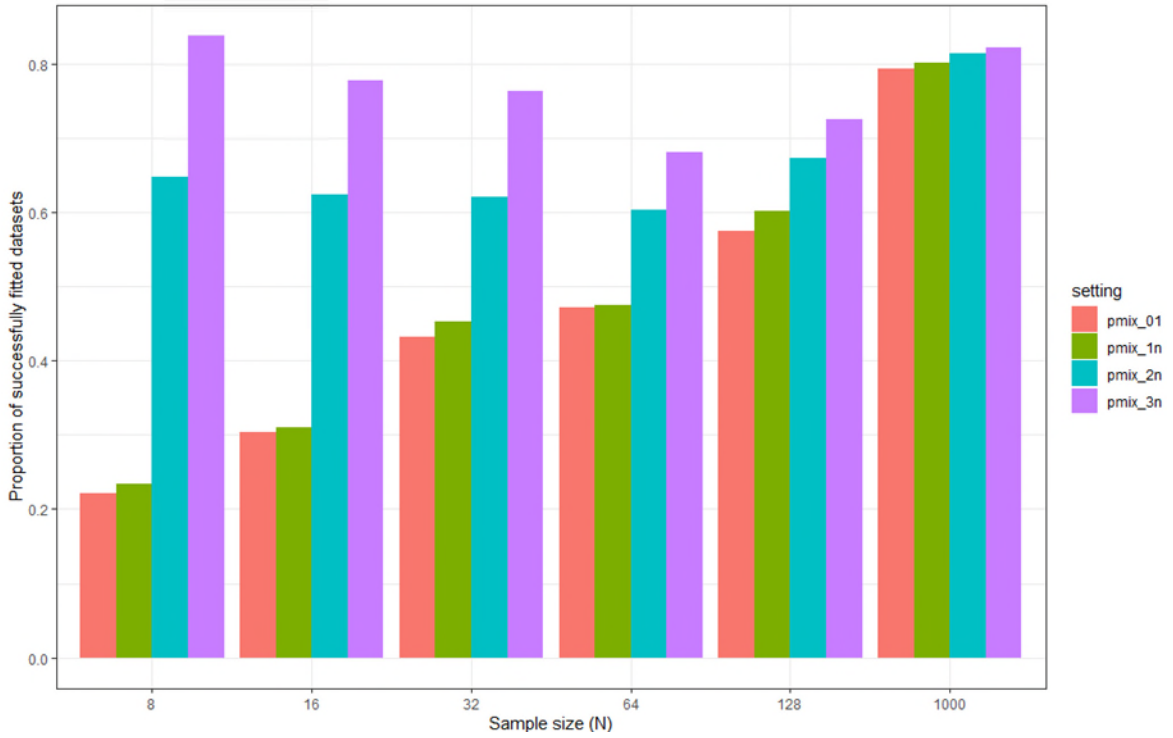



**Figure 27. Proportion of successfully fitted data sets based on Simulation study 1 that originally failed to fit the lognormal lognormal mixture distribution using the ssdtools version 1.0.6 implementation of pmix on the logit scale with no bounds. Four different pmix bounds were examined including 0-1 bounded, and lower bounded at 1/N, 2/N and 3/N, where N is the test data size.**

We also tested a range of possible lower bounds for the mixing proportion, and found that setting the lower bound of the mixing proportion relative to the sample size of the input data can result in substantial improvements to convergence reliability (see Figure 27, and further details in Appendix F). Various lower bounds on pmix were investigated and, for the time being, we have settled on restricting min_pmix to 3/N. Under these conditions, we achieved > 90% convergence of the univariate distributions that originally failed to fit a lognormal-lognormal mixture from Simulation study 1 (Figure 27) and 99% for a large sample of previously failed boostrap datasets based on the boron example from ssddata (see Appendix F). This modification also results in near perfect convergence for true mixture distributions, providing valid fits include those with estimated pmix values at the bounds (at_boundary_ok, see Appendix F).

Based on these results, as a pragmatic solution to the convergence issues in ssdtool we are recommending that a min_pmin value of 3/N be used as an interim default, and that convergence at the bounds is allowed for the lognormal-lognormal mixture distribution during bootstrap sampling.

# 5. CONCLUSIONS AND RECOMMENDATIONS

## 5.1 Obtaining model averaged HC/P estimates and CIs

This report documents the main outcomes and findings of many mathematical, statistical, and computational analyses associated with Tasks S1 (methods for obtaining HCx confidence intervals) and S3 (reconciliation of the `ssdtools` `hc()`– `hp()` inconsistency ).

The current version of `ssdtools` now correctly computes $\widehat{hc_p}$ for a model-averaged SSD such that $ssd\_hp\left[\widehat{hc_p}\right] = p$.

With respect to point estimation of an HCx, we demonstrated that the use of new `ssdtools` function, `ssd_qmulti()` has uniformly smaller bias than any of the other methods examined. `ssd_qmulti()` applies the R `uniroot` function to the single, model-averaged SSD rather than applying the MA weights to the quantiles estimated from the component distributions.

We investigated five methods for computing a confidence interval for the HCx and found that:

1. the `weighted_sample` method provides superior coverage for the estimation of $HC_1$ and $HC_5$ with relatively small differences in coverage among all methods for the estimation of $HC_{10}$ and $HC_{20}$;
2. the CI *width* was invariant to differences in: parent distribution for random data generation; sample size; characteristics of the dataset; and HCx value. Furthermore;
3. there is an almost perfect 1-to-1 correspondence between the CI limits obtained using the `weighted_sample` method and those derived from the `ssd_qmulti()`function.
4. the `weighted_sample` method is approximately 10x faster than the methods utilizing the `ssd_qmulti()`, `ssd_pmulti()`, and `ssd_rmulti()`functions.

Our recommendations follow.

*Recommendation #1*

The CRAN-version of `ssdtools` be updated to incorporate the new functions and associated `uniroot` procedures for HCx estimation: `ssd_qmulti()`, `ssd_pmulti()`,`ssd_rmulti()`; `ssd_dmulti()`thereby resolving the `hc()-hp()` inconsistency identified in Task S3.

*Recommendation #2*

Subsequent releases of `ssdtools` use the `weighted_average` method to determine confidence intervals for the HCx estimated by the procedures specified in Recommendation #1.

## 5.2 Lognormal-lognormal stability

Investigations into lognormal-lognormal convergence reliability revealed there are issues with the `optim` solver in `ssdtools` when estimating the mixing parameter at values near 0 and 1. These issues may not be able to be resolved with the currently available solvers within `ssdtools`. After testing several possible criteria as bounds for the mixing proportion, we found substantial improvements to convergence for lower bounds set of 3/N.

*Recommendation #3*

The minimum bound for pmix (mixing proportion) should be set at 3/N

## 5.3 Final set of 'stable' distributions

We carried out a thorough investigation into the distributions currently implemented in `ssdtools` showing unstable behaviour. Some of these instabilities are well understood and cannot be resolved. For other distributions, there are theoretical considerations for not considering them in the default set, regardless of stability. The team were able to implement fixes that improved the convergence reliability of the lognormal-lognormal distributions, such that this should now reliably converge.

*Recommendation #4*

The final set of 'default' distributions recommended have not changed from our original recommendations, and include:

- Gamma
- Log-Gumbel
- Log-logistic
- Lognormal
- Lnorm_lnorm
- Weibull

## 5.4 Decision rules for minimum data requirements

Reasonable decision rules around minimum data requirements were discussed. As a rule-of-thumb, the absolute minimum number of data points required to fit a 5 parameter distribution should be $n \geq 3k+1$, where k is the number of estimated parameters. Perferablly there would be $n \geq 5k+1$. Based on these criteria we recommend:

*Recommendation #5*

The minimum sample size for two parameter distributions should be 7 (Gamma, Log-Gumbel, Lognormal and Weibull), for three parameter distributions should be 10 (Burr III, not a default distribution), and for five parameter distributions this should by 16 (lognormal-lognormal). To enforce these recommendations in the `ssd_fit_bcanz()` function requires approval by the technical committee.

*Recommendation #6*

The preferred minimum sample size for two parameter distributions should be 11 (Gamma, Log-Gumbel, Lognormal and Weibull), for three parameter distributions should be 16 (Burr III, not a default distribution), and for five parameter distributions this should by 26 (lognormal-lognormal). This requires approval by the technical committee.

## 5.5 Censoring

While ssdtools allows left censoring, the package does not support right censoring (see https://github.com/bcgov/ssdtools/issues/207) and this would require additional support to resolve.

Finally, there are unresolved issues related to the sample size being undefined with censoring that need further examination. The current ANZG method has not historically accommodated censoring as this was not available through the Burrlioz software, and is is therefore not critical that issues with censoring be resolved prior to adoption of ssdtools as a relacement to Burrlioz. Any final decisions on censoring should be deferred to the technical committee for discussion and recommended actions (see also Further work below).

*Recommendation #7*

Issues around allowing censoring in ssdtools be referred to the technical committee for further consideration.

# 6. COMPLETE LIST OF ADDITIONS AND MODIFICATIONS TO SSDTOOLS

## 6.1 Additions

- Added David Fox and Rebecca Fisher as co-authors.
- Added to `ssd_hc()` and `ssd_hp()`.
- `multi_est = TRUE` argument to calculate model averaged estimates treating the distributions as constituting a single mixture distribution.
- `- \`method_ci = "weighted_samples"\` to specify whether to use \`"weighted_samples"\`, \`"weighted_arithmetic"\`, \`"multi_free"\` or \`"multi_fixed"\` methods to generate confidence intervals. samples` argument to include bootstrap samples as list of numeric vector(s).
- `save_to` argument to specify a directory in which to save the bootstrap datasets as csv files and parameter estimates as `rds` files. The files are named `data_000000001_xx.csv` and `estimates_000000001_xx.rds` etc where `xx` is the distribution. The parent data set and estimates are named `boot_000000000_xx.csv` and `estimates_000000000_xx.csv`.
- Added `ssd_pmulti()`, `ssd_qmulti()` and `ssd_rmulti()` for combined mixture distributions.
- Added `ssd_exx()` functions to get default parameter estimates for distributions.
- Added `ssd_hp.fitburrlioz()` function to get hazard proportion.
- Add trans = `"log10"` and `add_x = 0` arguments to `ssd_plot()` and `ssd_plot_data()`.

## 6.2 Modifications

- Changed to `min_pboot = 0.95` for all functions.
- `estimates.fitdists()`now includes weights in returned parameters as well as an `all_estimates = FALSE` argument to allow parameter values for all implemented distributions to be included.

- `ssd_fit_bcanz()`, `ssd_wqg_bc()` and `ssd_wqg_burrlioz()` no longer rescale data by default.
- `rescale = TRUE` now divides by the geometric mean of the minimum and maximum positive finite values as opposed to dividing by the geometric mean of the maximum finite value.
- Replaced column percentage between 0 and 100 with proportion between 0 and 1 in output of `ssd_hc()`
- Changed `delta = 7` to `delta = 9.21` to weight of included models no more than 0.01.
- seeds now allocated to bootstrap samples as opposed to distributions (this results in a speed gain when more cores than the number of distributions).
- Exported `dists = ssd_dists_bcanz()` argument to `ssd_fit_bcanz()` to allow other packages to modify.
- Check ... unused where appropriate.
- `ssd_plot_cdf()` now includes average with other distributions if `average = NA`
- switched from `logit_pmix` to `pmix` in mixture distributions
- `lnorm` no longer initializes optimization with maximum likelihood estimates
- Offset starting values for `gompertz` distribution.

## 6.3 Fixes

- `ssd_hc()` and `ssd_hp()` now include parametric column.
- `ssd_hp()` now includes `wt` column.

## 6.4 Deprecation

- Soft-deprecated argument percent = 5 for proportion = 0.05 for `ssd_hc()` and `predict()` and `is_censored()`, `ssd_plot_cf()` and `comma_signif(...)` now warn deprecated unconditionally.
- `plot.fitdists()` now defunct.
- Removed defunct `ssd_cfplot()`.
- Removed `ccme_data` and `ccme_boron` data set.

# 7. FURTHER WORK

## 7.1 Decisions to finalise through the Technical Advisory Group (TAG)

### 7.1.1 Default settings for `ssd_fit_bcanz()`

The following default values for `ssd_fit_bcanz()` have been recommended/adopted by the project team and need to be agreed across jurisdictions through the TAG.

- `min_pboot (0.95)`.
- `min_pmix` for the lognormal-lognormal mixture distribution (3/N).
- minimum sample sizes (7, 10 and 16 for distributions with 2, 3 and 5 parameters respectively).
- the `weighted_sample` method to be used as the default CI estimation method.

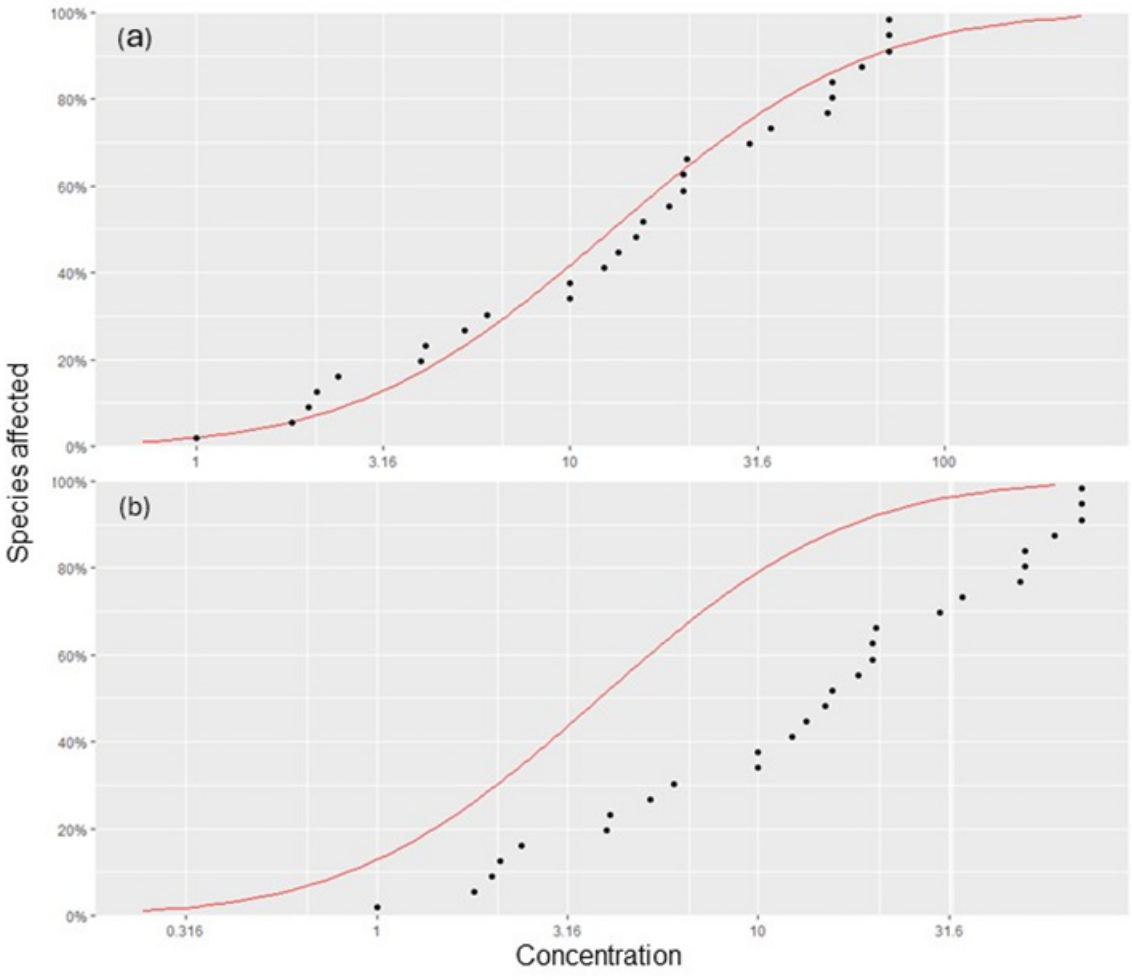
### 7.1.2 Defaults and level of restrictions for the ANZG Shiny App

There needs to be a final decision on if the `ssd_fit_bcanz()` method should underpin the customised ANZG Shiny App. Doing so will ensure the recommended methods are used but will also remove functionality from the Shiny App if the default settings are enforced. In addition, it will be necessary to decide if the default settings for ssd_fit_bcanz() should be consistent with ssd_fit_dists().

### 7.1.3 Weighting

An additional issue was raised during the project that suggested that using the weighting functionality in `ssdtools` results in unexpected behaviour (https://github.com/bcgov/ssdtools/issues/344) whereby attaching increased weight to the left tail of the distribution results in a fitted SSD which is shifted to the left of the empirical *cdf* (Figure 28b).

Investigations confirmed that the introduction of weights in the log-likelihood function in `ssdtools` has been done correctly, and that the *cdf* presented in Figure 28(b) is technically correct. However, the outcome (including estimated HCx values) is substantially different to what would be obtained if weights were implemented using least squares estimation. Least squares estimation results in a more



intuitive outcome that more closely aligns with an ecotoxicologist's expectations with respect to SSD weighting.

**Figure 28.** Illustration of the SSD weighting issue showing unweighted (a) and weighted (b) fits from `ssdtools`.

In general, the desired weighting behaviour would be akin to a weighted regression, that is a 'best' fit to the data where not all discrepancies between model and data are regarded as having equal importance (i.e. the left tail is given greater importance). The likelihood approach adopted for model fitting in `ssdtools` does not do this – it essentially ends up finding the 'best' fit to a modified data set. The likelihood is essentially the *joint probability* for the sample, and the idea behind maximum likelihood estimation is to estimate the model parameters that maximise this 'probability'. So, in this sense, the likelihood is not providing a measure of discrepancy between the empirical and theoretical *cdf's* in the same way OLS does, and therefore weighting has different behaviour in these two methods.

Thus, the weighting method as implemented is technically correct, and there may be a case where this is in fact desired. Weighting the data in a tail end of a distribution will likely increase the length of the tails and could be useful if the species in the tails are of great importance and if there are few data points to represent the class of taxa that are found to be sensitive. A decision needs to be made as to whether the weighting functionality in `ssdtools` is retained as an option, is retained but a least-squares solution rather than a maximum likelihood solution is returned or is removed entirely.

## 7.2 Unresolved technical issues

### 7.2.1 Instability for large sample sizes

Testing has shown that the lognormal and some other distribution can exhibit convergence issues when sample sizes are very high. Investigations indicated this is because the initial values which are estimated from the input data are almost exactly equal to the MLEs, which causes issues with the optimization algorithm utilized in `ssdtools` via TMB that cannot be resolved. The current solution is to adjust all initial values slightly prior to passing to the optimization engine, to ensure they are not the exact MLE solutions. However, it is important to note that this solution is a temporary fix as it does nothing to address the underlying issue. In the case where closed form solutions for ML estimation exist it would be possible to implement these for the relevant distributions. This would require significant re-working of the `ssdtools` code base but would result in a more technically correct and faster fitting algorithm for relevant distributions. This issue needs to be discussed, along with an evaluation of the consequences and time to undertake the task to determine if the required work should be funded.

### 7.2.2 Convergence reliability when min_pmix is 0 (statistical mixture distributions)

As mentioned in sections 4.1.8 and 5.2, investigations into lognormal-lognormal convergence reliability revealed there are issues with the `optim` solver in `ssdtools` when estimating the mixing parameter (`pmix`) at values near 0 and 1. As an interim measure we have recommended the lower bound be set at 3/N, which has been shown to result in reasonably reliable convergence (Appendix F). Setting the lower bound of `pmix` has implications for the type of statistical mixture that can be

modelled. Such restrictions may be appropriate and relevant based on philosophical grounds and the topic should be discussed in more detail by the TAG. Theoretically it should be possible to reliably obtain convergence even at a bound of 0 and ideally `ssdtools` should support a `min_pmix` of 0. Better convergence behaviour has been obtained using other solvers outside of `ssdtools` (Appendix F). It may be possible to utilise solvers other than `optim` within the `ssdtools` framework, although this may be a substantial undertaking and requires extensive validation and testing.

## 7.3 Additional features

The focus of all three phases of this project has been to ensure that `ssdtools` employs robust, efficient, and technically correct methods for SSD modelling. We believe that the many hours of detailed technical work accompanied by hundreds of hours of computer modelling, simulation, and analysis by several experts over 4+ years has achieved this objective. These efforts will be reflected in the forthcoming release of `ssdtools 2.0`.

While we are confident that `ssdtools 2.0` represents the most comprehensive SSD modelling tool available, we have identified several potentially useful enhancements as detailed below.

### 7.3.1 Additional distributions

- A significant advantage of `ssdtools` is its ability to fit (statistical) mixture models as well as its intrinsic use of model-averaging. At the present time, the available distributions in `ssdtools` are (except the Burr III) all two-parameter distributions or mixtures of two, 2-parameter distributions with a mixing proportion, resulting in 5 parameters. There is an obvious omission in this offering – namely the inclusion of three and four parameter distributions. Experience to date suggests that because of the invariably small sample sizes used in ecotoxicology, mixture models will be heavily penalised due to the high parameter count relative to the sample size even when the fit is superior to any single model fit. The inclusion of three and four parameter models would provide for greater modelling flexibility than two parameter models while reducing the penalty associated with mixture models. Possibilities include three-parameter versions of the Weibull, lognormal, log-logisticand gamma distributions. Other distributions (available in the `ExtDist` package) include:
- `Beta_ab` – a four parameter beta distribution
- `JohnsonSB` – the Johnson SB distribution
- `JohnsonSU` – the Johnson SU distribution
- `Normal_sym_trunc_ab` – The symmetric truncated normal distribution
- `Normal_trunc_ab` – The truncated normal distribution

Finally, a four-parameter lognormal or log-logistic mixture model could be contemplated in situations where it is known (or can reasonably be assumed) for example, that either the `meanlog` or `sdlog` parameters of the component lognormal distributions are equal. A common `meanlog` parameter would be appropriate where it was thought that the bimodality in the empirical *cdf* was due only to differences in *variation* while a common `sdlog` parameter would assume the bimodality was a consequence of different locations (i.e. means) only.

### 7.3.2 Bounded distributions

Currently, all distributions in `ssdtools` are defined on the positive real line – that is for *concentrations > 0*. When analysing toxicity data obtained from C-R models in which *concentration* is reported as a percentage or fraction between 0 and 1 (for example in whole effluent toxicity tests (WET) where concentrations are expressed as a fractional dilution), the use of probability models which are unbounded on the right may be sub-optimal. In these cases, a distribution bounded between 0 and 1, such as a beta distribution is more consistent with the nature of the data.

A second possibility where a bounded distribution may be appropriate is where there is an elevated background concentration of a contaminant. In this case, an SSD defined for *concentrations > b* (where *b* is the background concentration) may be preferred although this is probably more conveniently handled by simply subtracting *b* from the concentration data and using an existing two-parameter model (assuming *b* is known or can be otherwise estimated).

### 7.3.3 Censoring

The issue of censoring has been discussed elsewhere in this report. Interval and left censoring is implemented in the current CRAN version 1.0.6 of ssdtools whereas right censoring is not. The desirability of including right censoring is a matter best left for the technical advisirory group to decide.

# 8. NEXT STEPS

While this report signifies the completion of the set of tasks that were identified as being necessary to correct known issues in `ssdtools` as well as undertaking further mathematical and statistical investigations into methodological and performance issues, we have flagged several additional refinements and matters requiring decisions by the newly established Technical Advisory Group (TAG).

In terms of the release of the major `ssdtools` 2.0 update, the only outstanding matters are: (i) TAG review of default and fixed values for `ssd_fit_bcanz()` and `ssd_hc_bcanz()` including the use of the weighted bootstrap and minimum sample size for log-normal log-normal mixture; and (ii) updating of `shinyssdtools` and user guide and reports associated with the bilingual (English/French) interface.

**REFERENCES**

Fisher R, van Dam R, Batley G, Fox D, Harford A, Humphrey C, King C, Menendez P, Negri A, Proctor A, Shao Q, Stauber J, van Dam J and Warne M. (2019). Key issues in the derivation of water quality guideline values: a workshop report. *Australian Institute of Marine Science Report, Crawley, WA, Australia. (47 pp).*

Fox D.R., Fisher R., Thorley J.L., Schwarz C. (2021) Joint Investigation into statistical methodologies underpinning the derivation of toxicant guidelines values in Australia and New Zealand. Report prepared for the Department of Agriculture, Water and the Environment. Environmetrics Australia, Beaumaris, Vic and the Australian Institute of Marine Science, Perth, WA. (148 pp).

**References available from** https://www.environmetrics.net.au/resources/documents-and-reports/

# Appendices – Detailed Analyses and Results

# Appendix A - BIAS by dataset for all simulation studies

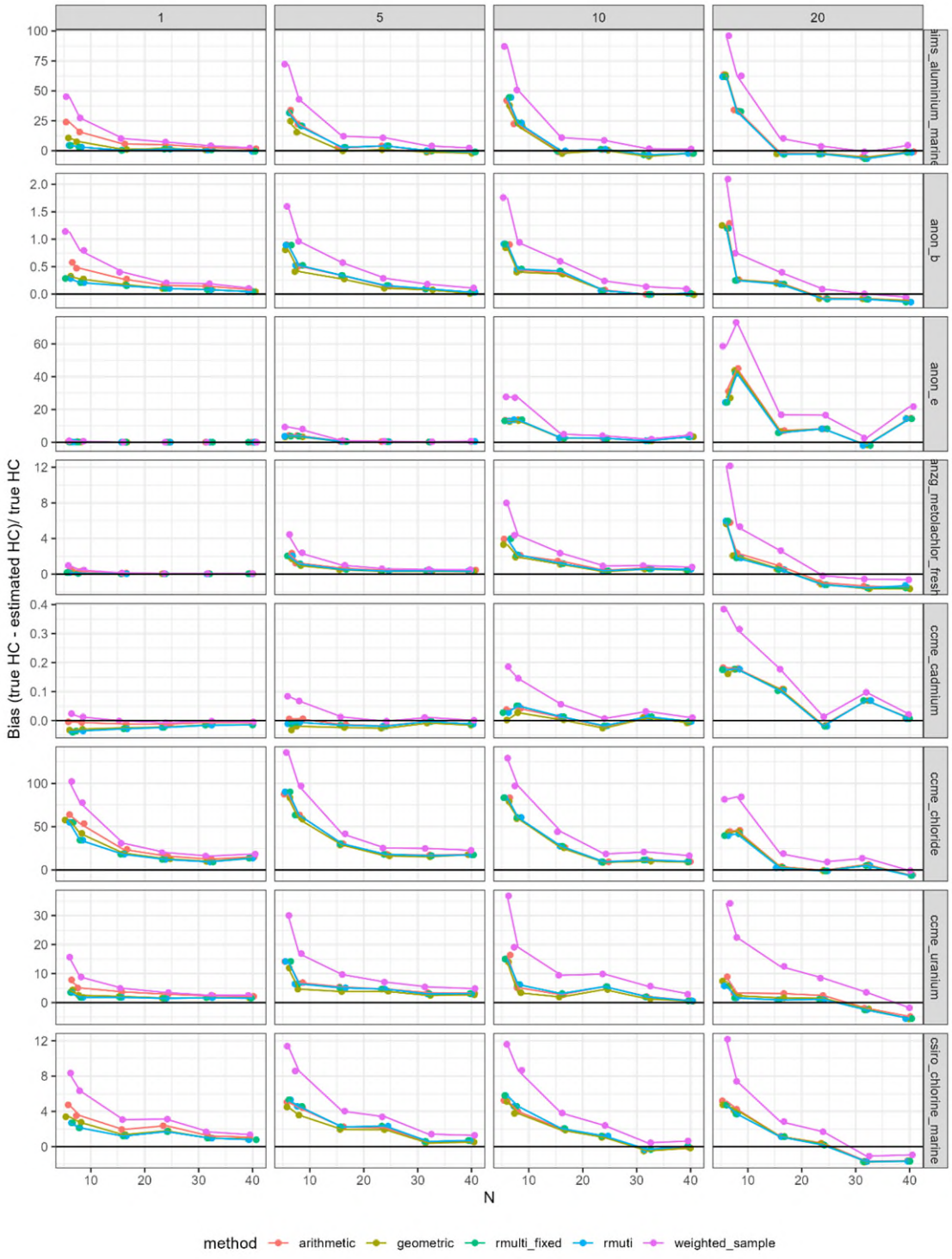



**Figure A- 1. Bias (estimated hazard concentration – true hazard concentration) for 8 simulation datasets based on a Burr III distribution (plot rows) across a range of sample sizes (6, 8, 16, 24, 32, and 40), for hazard concentration values of 1, 5, 10 and 20 (plot columns) for a range of different methods (see text). Each dataset was simulated 200 times, for each dataset, and a minimum of 1000 bootstrap samples were used to estimate confidence intervals. Fitted candidate distributions included 'gamma', 'lgumbel', 'llogis', and 'lnorm.**

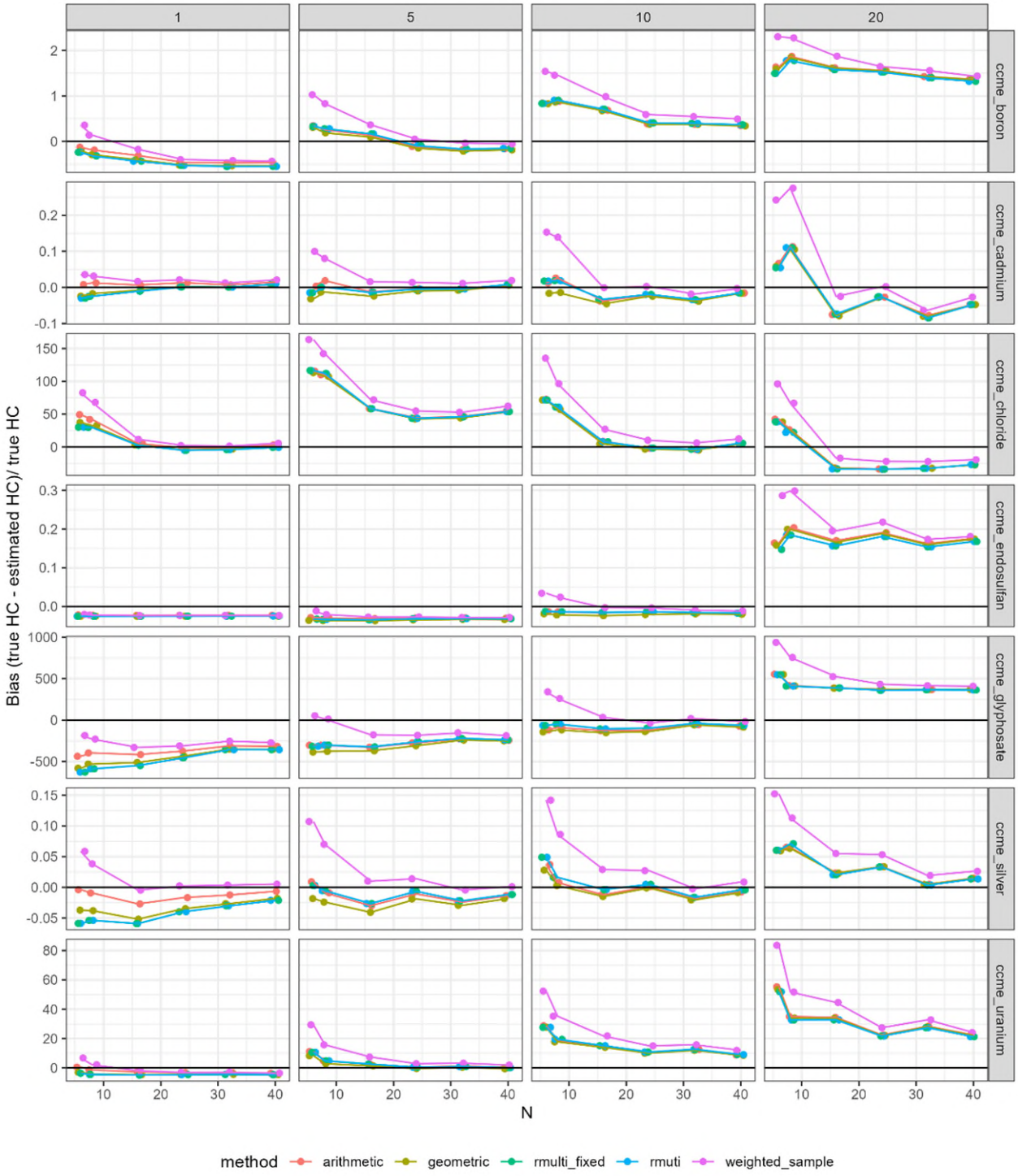



**Figure A- 2. Bias (estimated hazard concentration – true hazard concentration) for 7 simulation datasets based on a log-normal log-normal mixture distribution (plot rows) across a range of sample sizes (6, 8, 16, 24, 32, and 40), for hazard concentration values of 1, 5, 10 and 20 (plot columns) for a range of different methods (see text). Each dataset was simulated 200 times, for each dataset, and a minimum of 100 bootstrap samples were used to estimate confidence intervals. Fitted candidate distributions included 'gamma', 'lgumbel', 'llogis', and 'lnorm.**

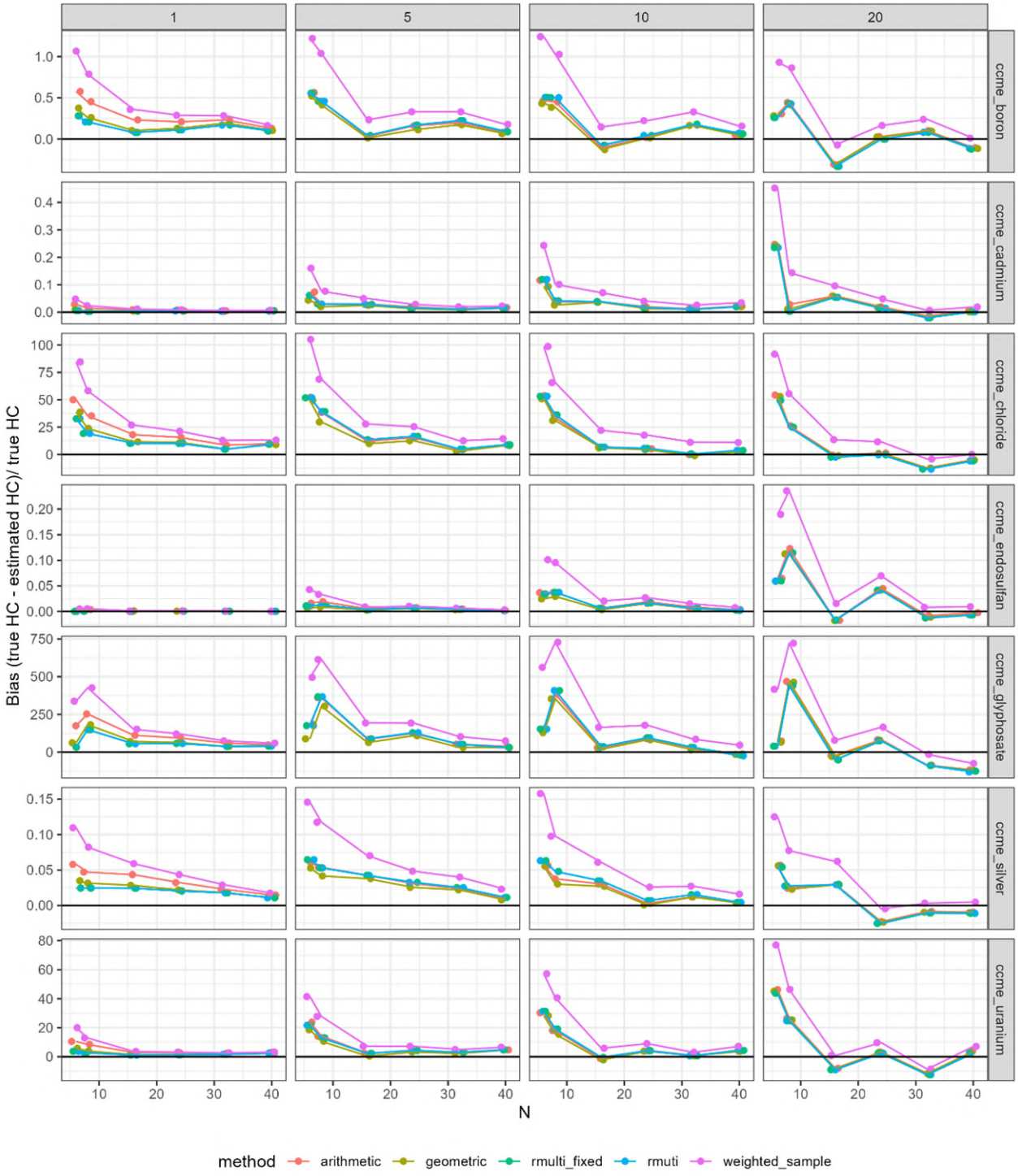



**Figure A- 3Bias (estimated hazard concentration – true hazard concentration) for 7 simulation datasets based on a log-logistic distribution (plot rows) across a range of sample sizes (6, 8, 16, 24, 32, and 40), for hazard concentration values of 1, 5, 10 and 20 (plot columns) for a range of different methods (see text). Each dataset was simulated 200 times, for each dataset, and a minimum of 1000 bootstrap samples were used to estimate confidence intervals. Fitted candidate distributions included 'gamma', 'lgumbel', 'llogis', and 'lnorm.**

# Appendix B - COVERAGE by dataset for all simulation studies.

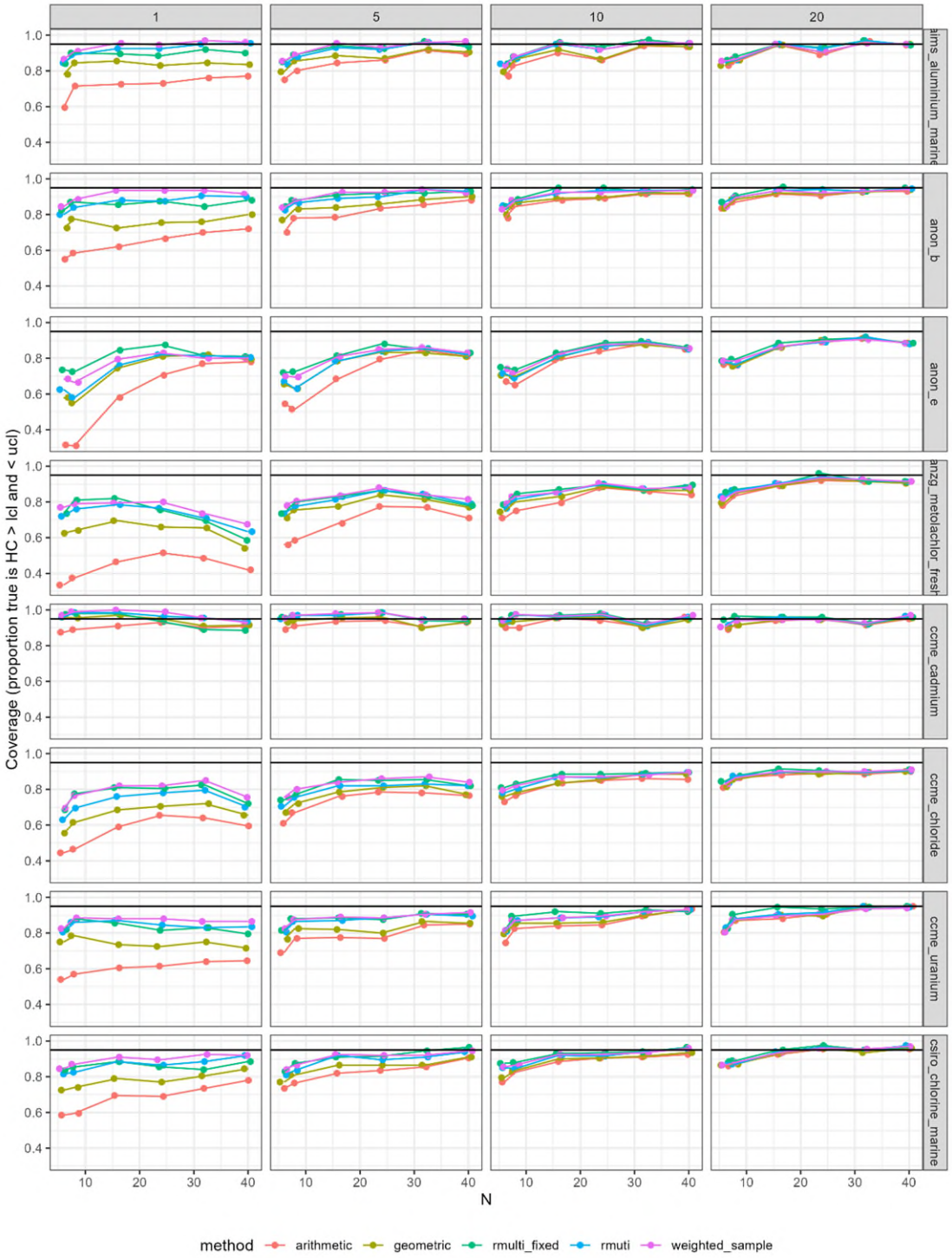



**Figure B- 1. Coverage (the proportion of times the true HC fell within the 95% confidence interval) for 8 simulation datasets based on a Burr III distribution (plot rows) across a range of sample sizes (6, 8, 16, 24, 32, and 40), for hazard concentration values of 1, 5, 10 and 20 (plot columns) for a range of different methods (see text). Each dataset was simulated 200 times, for each dataset, and a minimum of 1000 bootstrap samples were used to estimate confidence intervals. Fitted candidate distributions included 'gamma', 'lgumbel', 'llogis', and 'lnorm**

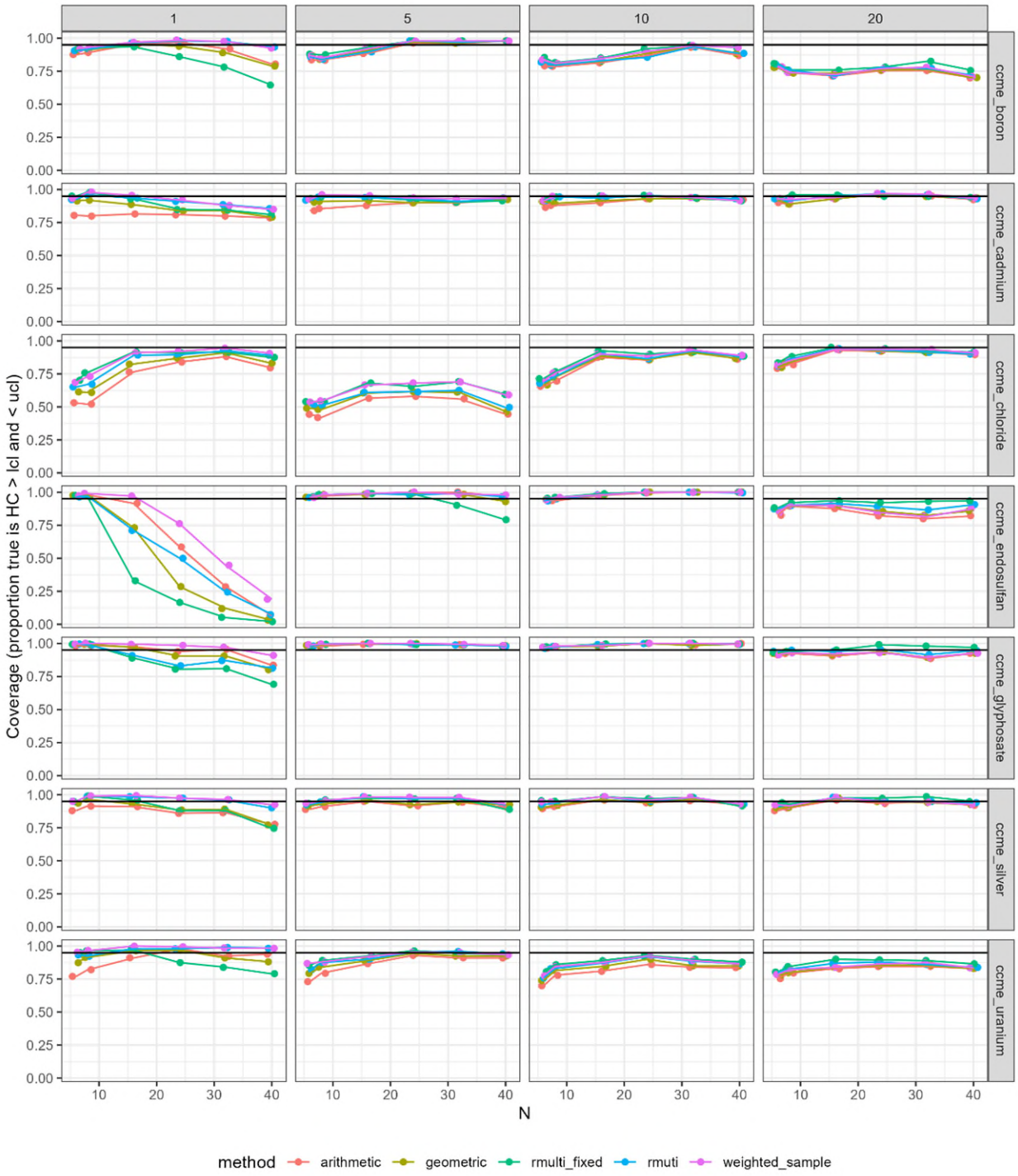



**Figure B- 2. Coverage (the proportion of times the true HC fell within the 95% confidence interval) for 7 simulation datasets based on a log-normal log-normal mixture distribution (plot rows) across a range of sample sizes (6, 8, 16, 24, 32, and 40), for hazard concentration values of 1, 5, 10 and 20 (plot columns) for a range of different methods (see text). Each dataset was simulated 200 times, for each dataset, and a minimum of 1000 bootstrap samples were used to estimate confidence intervals. Fitted candidate distributions included 'gamma', 'lgumbel', 'llogis', and 'lnorm.**

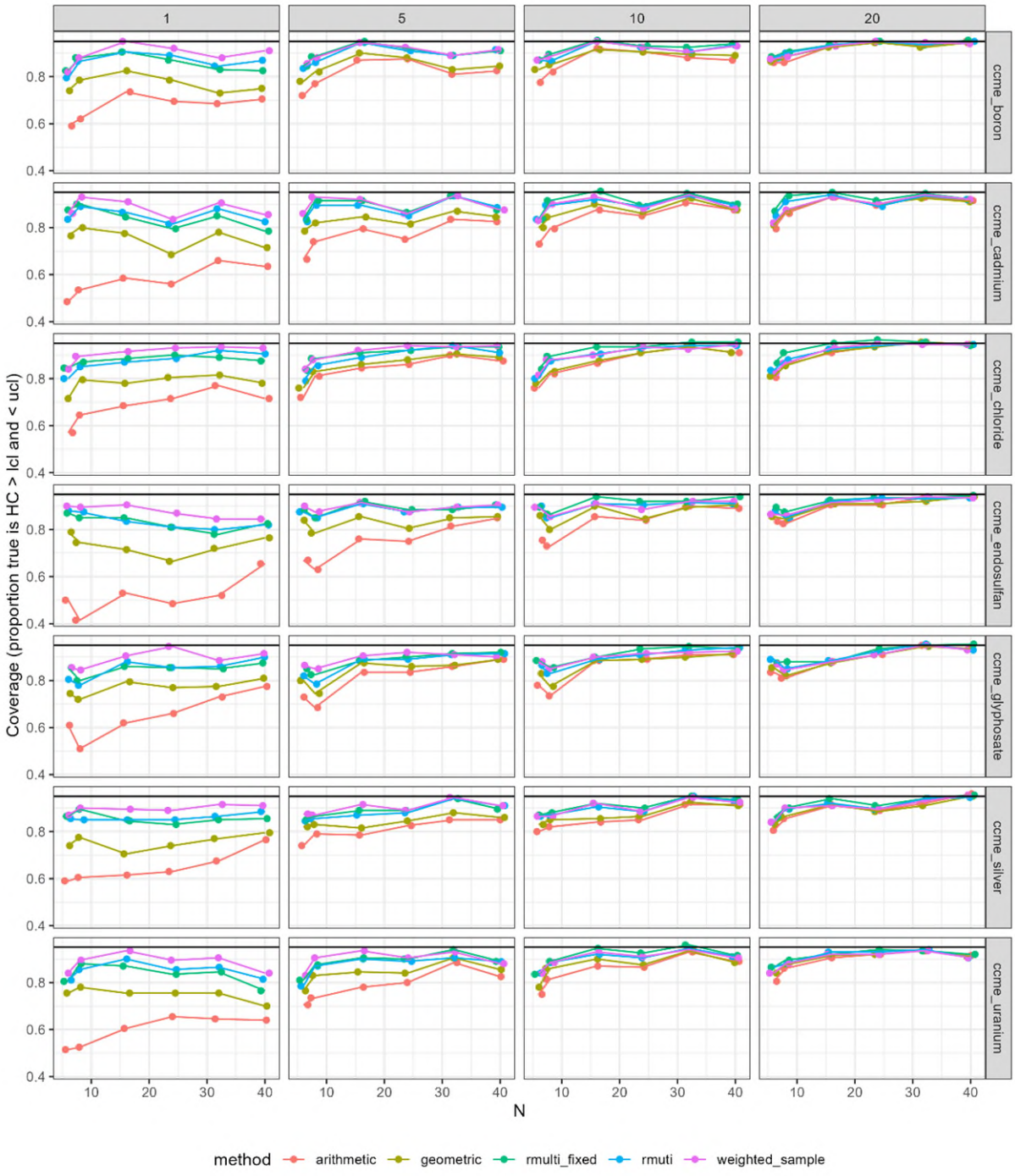



**Figure B- 3. Coverage (the proportion of times the true HC fell within the 95% confidence interval) for 7 simulation datasets based on log-logistic distribution (plot rows) across a range of sample sizes (6, 8, 16, 24, 32, and 40), for hazard concentration values of 1, 5, 10 and 20 (plot columns) for a range of different methods (see text). Each dataset was simulated 200 times, for each dataset, and a minimum of 1000 bootstrap samples were used to estimate confidence intervals. Fitted candidate distributions included 'gamma', 'lgumbel', 'llogis', and 'lnorm.**

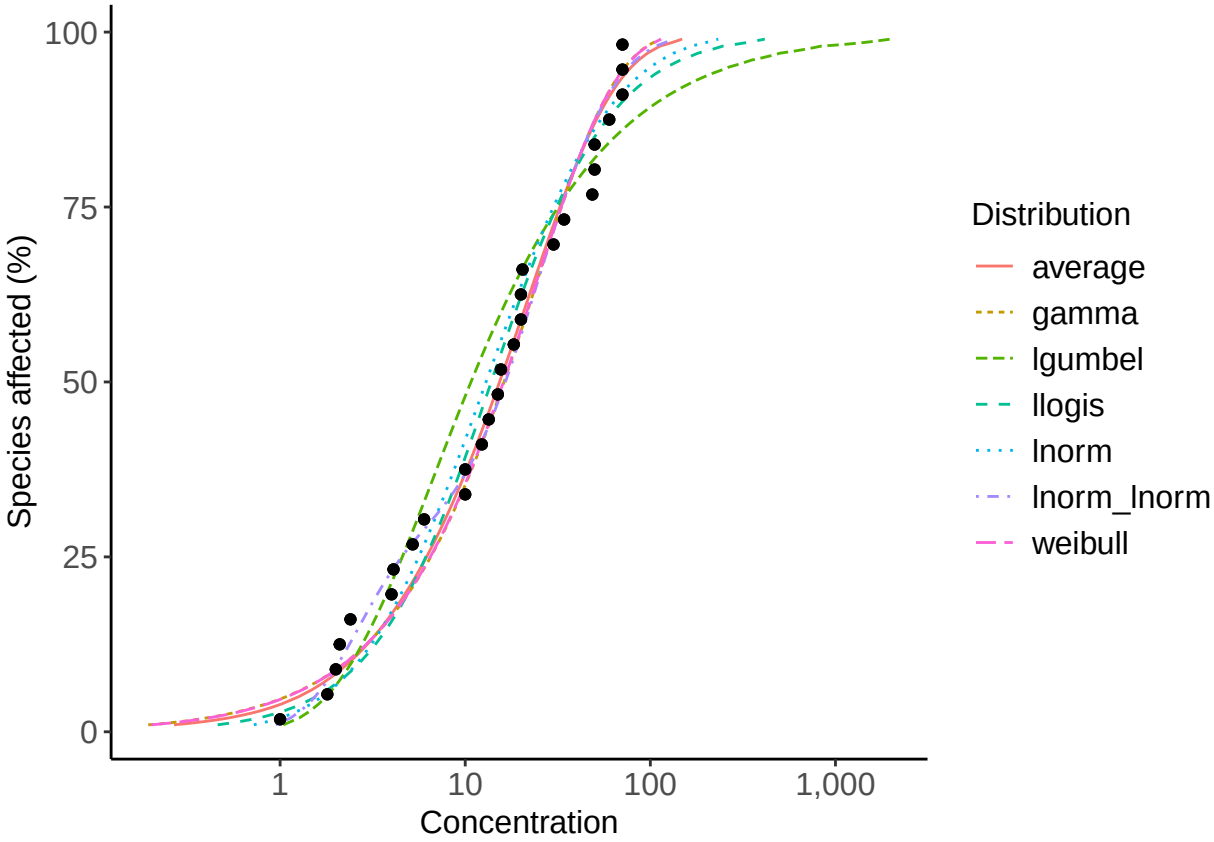
# Appendix C: Example of the pdf generated through the updated Shinny App.

# ssdtools BCANZ Report

This report was generated through the ssdtools Shiny app which fits species sensitivity distributions to concentration data. The app is built from the R package ssdtools, and shares the same functionality.

**Toxicant:** Boron
**Report created:** 04/04/2024
**Input distributions:** gamma, lgumbel, llogis, lnorm, lnorm_lnorm, weibull
**ssdtools version:** 1.0.6.9010
**R version:** 4.3.2 (2023-10-31)

## Fit of all distributions

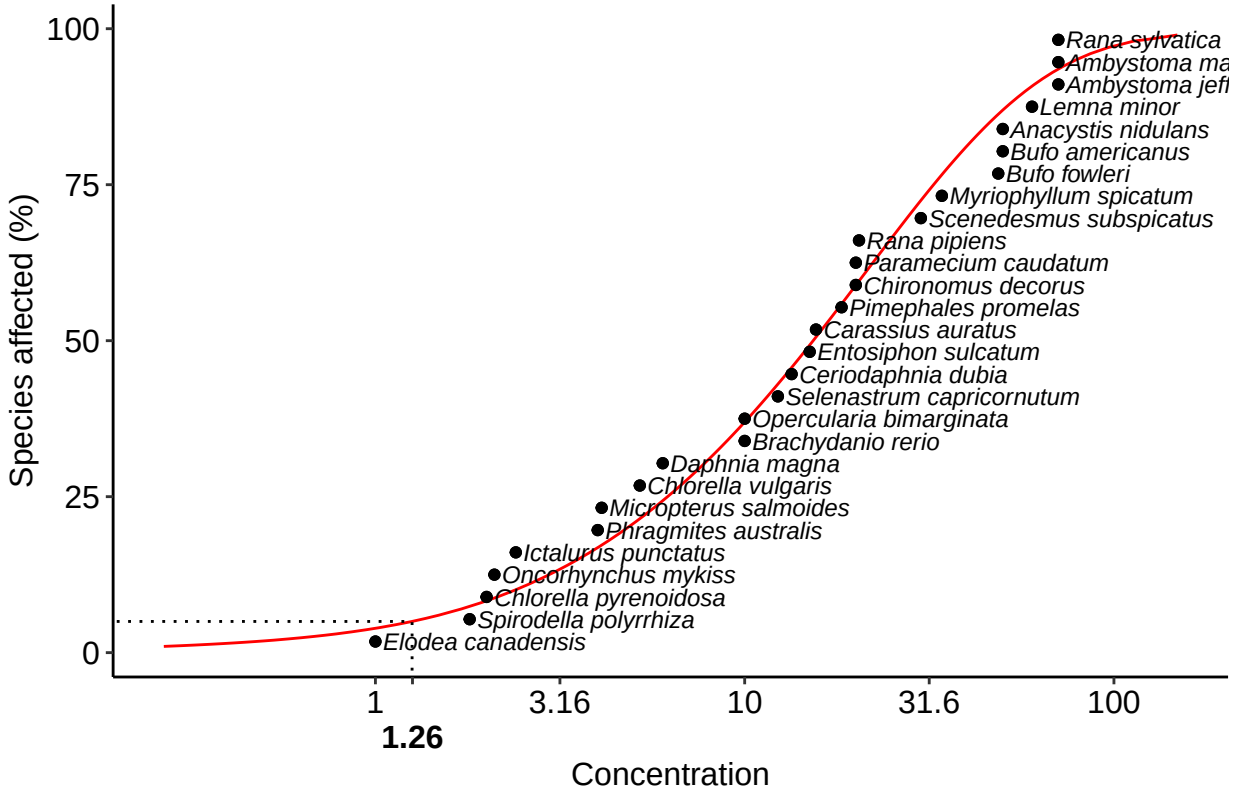



## Goodness of fit table

| dist | ad | ks | cvm | aic | aicc | bic | delta | weight |
|---|---|---|---|---|---|---|---|---|
| gamma | 0.440 | 0.1170 | 0.0554 | 238 | 238 | 240 | 0.005 | 0.357 |
| weibull | 0.434 | 0.1170 | 0.0542 | 238 | 238 | 240 | 0.000 | 0.357 |
| lnorm | 0.507 | 0.1070 | 0.0703 | 239 | 240 | 242 | 1.400 | 0.177 |
| llogis | 0.487 | 0.0994 | 0.0595 | 241 | 241 | 244 | 3.390 | 0.066 |
| lnorm_lnorm | 0.320 | 0.1160 | 0.0414 | 240 | 243 | 247 | 4.980 | 0.030 |
| lgumbel | 0.829 | 0.1580 | 0.1340 | 244 | 245 | 247 | 6.560 | 0.013 |

Table 1. The goodness of fit statistics. 'dist' is the distribution name; 'ad' is the Anderson-Darling statistic; 'ks' is the Kolmogorov-Smirnov statistic; 'cvm' is the Cramer-von Mises statistic; 'aic' is the Akaike's Information Criterion; 'aicc' is the Akaike's Information Criterion corrected for sample size; 'bic' is the Bayesian Information Criterion; 'delta' is the Information Criterion differences; 'weight' is the Information Criterion weights. 'delta' and 'weight' are based on 'aic' for censored data and 'aicc' for non-censored data.

## Model averaged fit




## Estimated hazardous/protective concentration

| HCx | PCx | est | se | lcl | ucl | nboot | pboot |
|---|---|---|---|---|---|---|---|
| 1 | 99 | 0.267 | 0.404 | 0.031 | 1.523 | 1000 | 1 |
| 5 | 95 | 1.257 | 0.804 | 0.324 | 3.536 | 1000 | 1 |
| 10 | 90 | 2.382 | 1.167 | 0.878 | 5.576 | 1000 | 1 |
| 20 | 80 | 4.810 | 1.814 | 2.265 | 9.585 | 1000 | 1 |

Table 2. The estimated hazardous/protective concentrations. 'HCx' is the % species affected; 'PCx' is the % species protected; 'est' is the model-averaged estimate of the concentration; 'se' is the bootstrap based standard error of the estimate; 'lcl' and 'ucl' are the bootstrapped-based lower and upper 95% confidence limits; 'nboot' is the number of bootstrap samples; 'pboot' is the proportion of bootstrap samples that converged. The model-averaged estimate(s) are calculated by treating the distributions as a single mixture distribution. Distributions with an absolute AIC difference greater than a delta of by default 9.21 have considerably less support (weight $< 0.01$) and are excluded prior to bootstrapping.

## Input data

| Chemical | Species | Conc | Group | Units |
|---|---|---|---|---|
| Boron | Oncorhynchus mykiss | 2.1 | Fish | mg/L |
| Boron | Ictalurus punctatus | 2.4 | Fish | mg/L |
| Boron | Micropterus salmoides | 4.1 | Fish | mg/L |
| Boron | Brachydanio rerio | 10.0 | Fish | mg/L |
| Boron | Carassius auratus | 15.6 | Fish | mg/L |
| Boron | Pimephales promelas | 18.3 | Fish | mg/L |
| Boron | Daphnia magna | 6.0 | Invertebrate | mg/L |
| Boron | Opercularia bimarginata | 10.0 | Invertebrate | mg/L |
| Boron | Ceriodaphnia dubia | 13.4 | Invertebrate | mg/L |
| Boron | Entosiphon sulcatum | 15.0 | Invertebrate | mg/L |
| Boron | Chironomus decorus | 20.0 | Invertebrate | mg/L |
| Boron | Paramecium caudatum | 20.0 | Invertebrate | mg/L |
| Boron | Rana pipiens | 20.4 | Amphibian | mg/L |
| Boron | Bufo fowleri | 48.6 | Amphibian | mg/L |
| Boron | Bufo americanus | 50.0 | Amphibian | mg/L |
| Boron | Ambystoma jeffersonianum | 70.7 | Amphibian | mg/L |
| Boron | Ambystoma maculatum | 70.7 | Amphibian | mg/L |
| Boron | Rana sylvatica | 70.7 | Amphibian | mg/L |
| Boron | Elodea canadensis | 1.0 | Plant | mg/L |
| Boron | Spirodella polyrrhiza | 1.8 | Plant | mg/L |
| Boron | Chlorella pyrenoidosa | 2.0 | Plant | mg/L |
| Boron | Phragmites australis | 4.0 | Plant | mg/L |
| Boron | Chlorella vulgaris | 5.2 | Plant | mg/L |
| Boron | Selenastrum capricornutum | 12.3 | Plant | mg/L |
| Boron | Scenedesmus subspicatus | 30.0 | Plant | mg/L |
| Boron | Myriophyllum spicatum | 34.2 | Plant | mg/L |
| Boron | Anacystis nidulans | 50.0 | Plant | mg/L |
| Boron | Lemna minor | 60.0 | Plant | mg/L |

# Appendix D : Gompertz stability issues

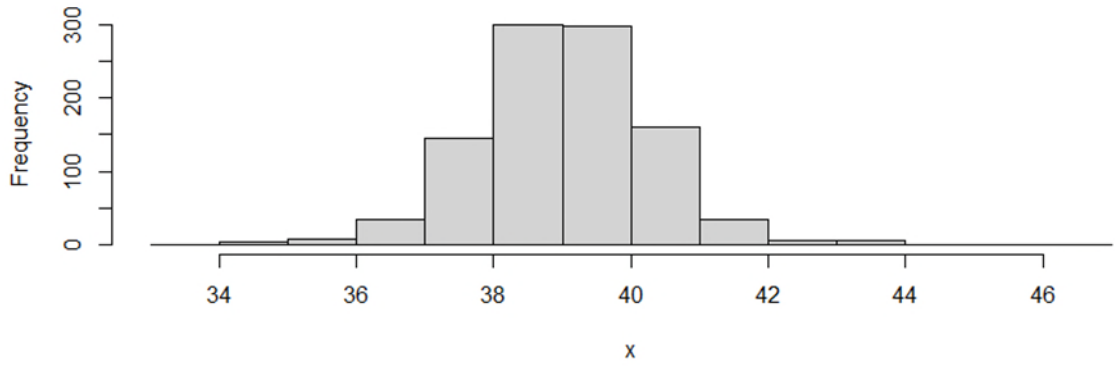
Prior testing has shown that the Gompertz distribution can have poor convergence behaviour (see Figure 32, Fox et al 2022). Further, even with the same dataset, ssdtools can return different parameter estimates based on different seeds, which suggests a very high degree of instability, even where the distribution has successfully fitted (see https://github.com/bcgov/ssdtools/issues/223). Here we investigate issues with the Gompertz distribution, to identify sources of instability.

## Investigation 1:

We extracted datasets from Simulation study 1 and used this to investigate convergence issues and stability in the Gompertz distribution.

Examining the first dataset we get:

```
x <- failed_data_allN[[1]]$failed.gompertz[, 3]

data <- data.frame(Conc = x)

fit <- ssd_fit_dists(data, left = 'Conc', dists = use.dists,
at_boundary_ok=TRUE, rescale = FALSE)
```




**Comment**: Nothing leaps off the page with the histogram above in terms of extremeness, dispersion etc. In fact, looks decidedly 'normal'.

As we already know, ssdtools fails to fit a Gompertz to this data:

```
Unscaled data - ssdtools
```

```
> ssd_fit_dists(data=data,dists = "gompertz")
Error:
! All distributions failed to fit.
Run `rlang::last_trace()` to see where the error occurred.
Warning message:
```

```
Distribution 'gompertz' failed to fit (try rescaling data): Error in optim
(par, fn, gr, method = method, lower = lower, upper = upper,  :
L-BFGS-B needs finite values of 'fn'
```

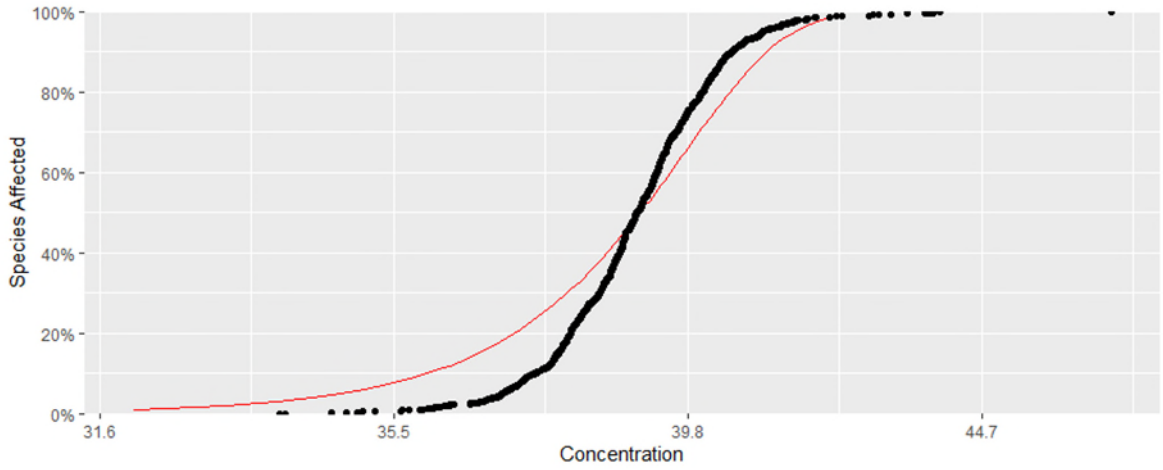
But, does succeed if rescaling used:

Scaled data – ssdtools

```
> m.ssd<-ssd_fit_dists(data=data,dists = "gompertz",rescale=TRUE)
> summary(m.ssd)
Distribution 'gompertz'
location 9.84102e-10
shape 24.071

Parameters estimated from 1000 rows of rescaled (39.93) data.
```

| dist | ad | ks | cvm | aic | aicc | bic | delta | weight |
|---|---|---|---|---|---|---|---|---|
| | <chr> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> |
| gompertz | Inf | 0.144 | 6.88 | -3575. | -3575. | -3565. | 0 | 1 |

**Comment**: The location parameter is essentially zero which is a little curious. The AD test statistic is Inf which I suspect may be due to the evaluation of the Gompertz *cdf*returning Inf values. The plot below shows that the fit is not good:



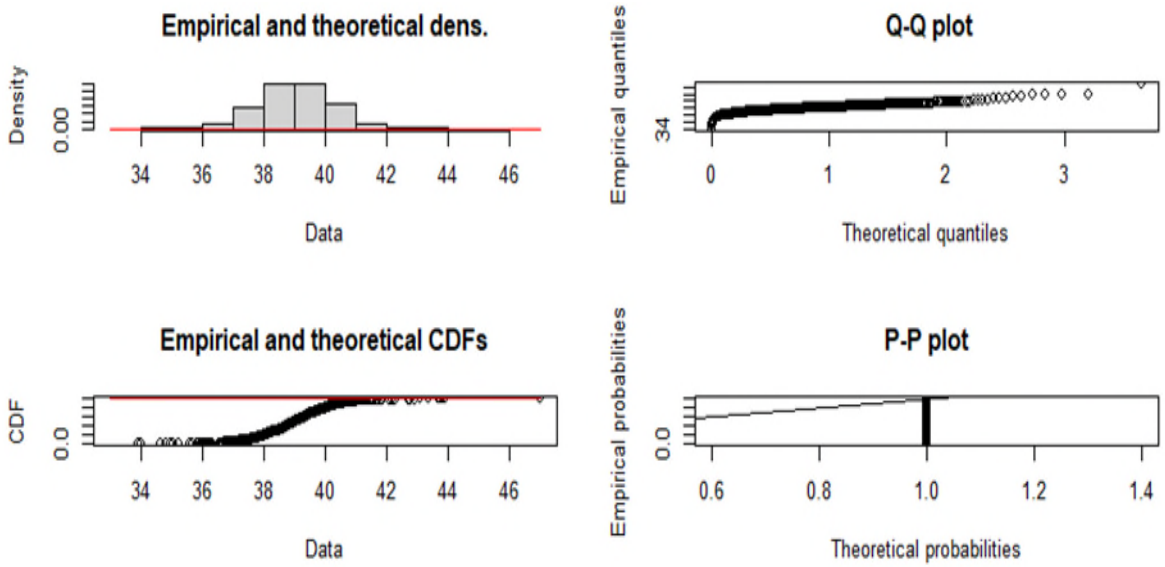

**Comparison with fitdistrplus**

We can use ssdtool's Gompertz distribution with fitdistrplus to make a direct comparison (again using unscaled data):

```
> m.fd<-fitdist(data=data$Conc,distr="gompertz",start = list(llocation=0,l
shape=0))
> summary(m.fd)
Fitting of the distribution ' gompertz ' by maximum likelihood
Parameters :
           estimate   Std. Error
```

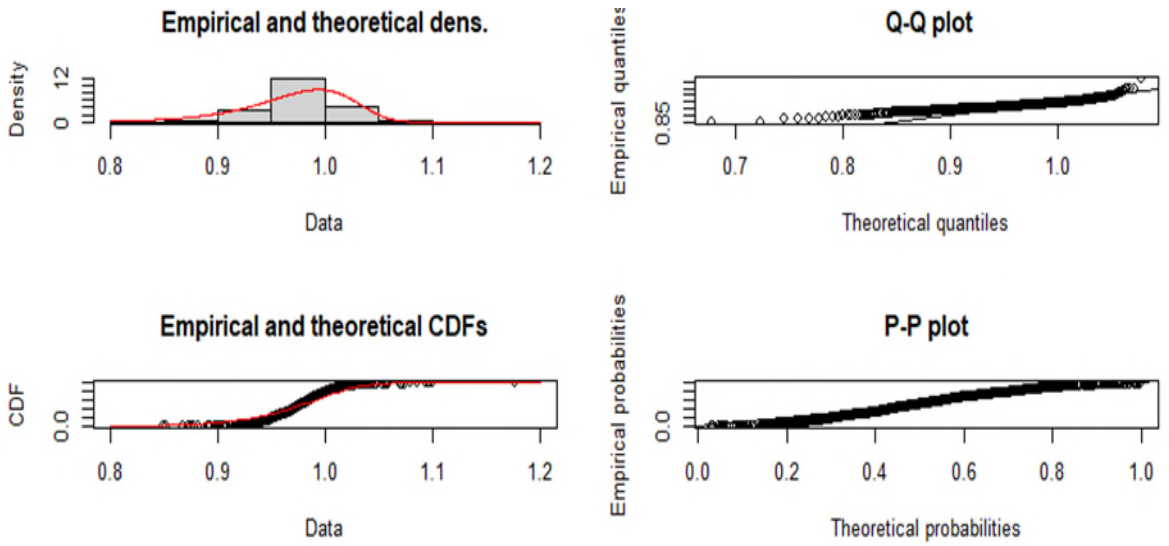
```
llocation  0.540625 0.0019897906
lshape    -2.287500 0.0005497542
Loglikelihood:  -875505.7   AIC:  1751015   BIC:  1751025
Correlation matrix:
           llocation     lshape
llocation  1.0000000 -0.8443869
lshape    -0.8443869  1.0000000
```

**Comment**: This does work without the need to rescale which suggests fitdistrplus may have a better algorithm for fitting and/or selection of initial values. *However*, the fit is absolute rubbish:




Repeating the fitdistrplus calculations with scaled data (and ssdtool's dgompertz distribution and same scaling value)

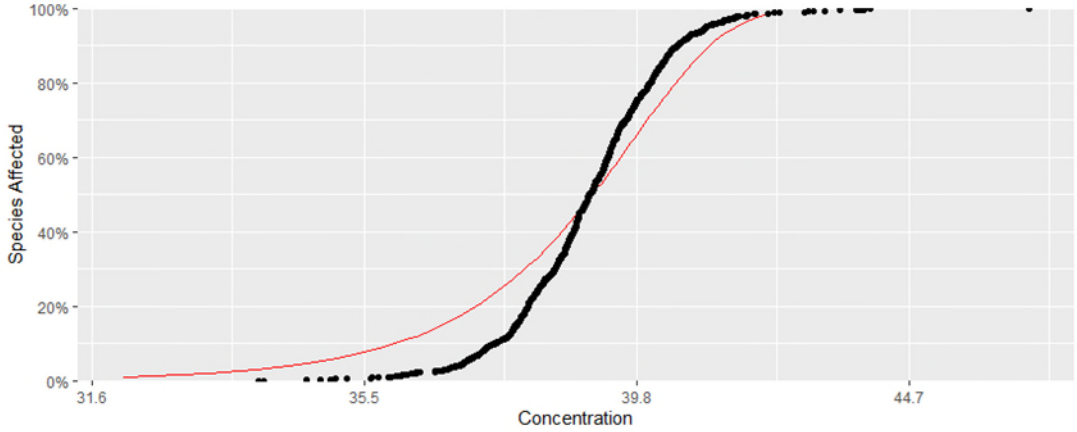
```
> summary(m.fd)
Fitting of the distribution ' gompertz ' by maximum likelihood
Parameters :
                   estimate Std. Error
llocation -20.739903 0.42578237
lshape      3.181094 0.01804378
Loglikelihood:  1789.628   AIC:  -3575.257   BIC:  -3565.441
Correlation matrix:
           llocation     lshape
llocation  1.0000000 -0.9972379
lshape    -0.9972379  1.0000000
```



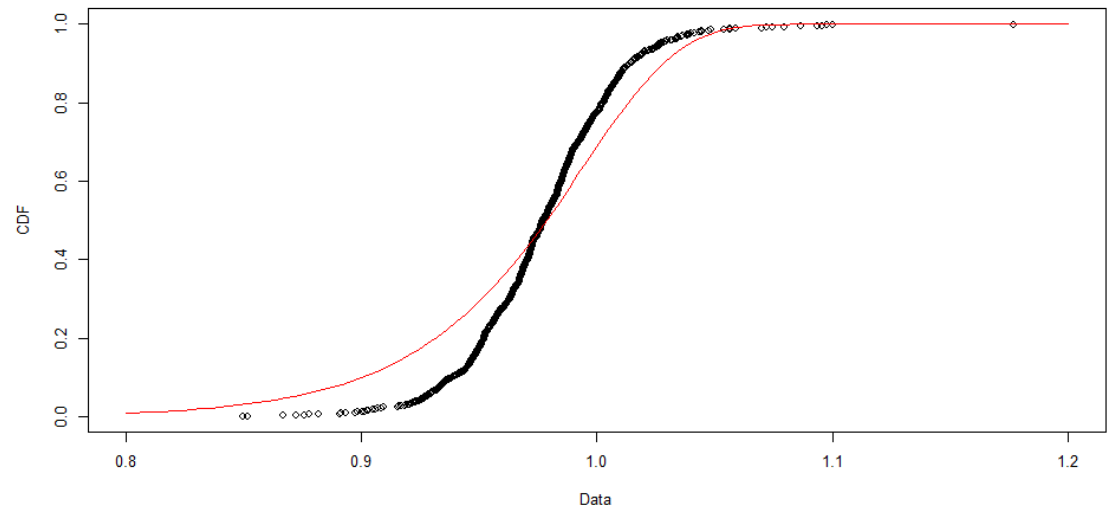

**Comment**: This does work and the fit is much better. But – looking at the estimates above we see llocation = -20.739903 => location = 9.835005e-10 and lshape = 3.181094 => shape =24.07308 which is identical to ssdtools parameter estimates! So why is the fit much better – at least visually? NOTE: AIC from fitdistrplus and ssdtools are identical (= -3575.257) – the plot thickens!

The apparent better fit from fitdistrplus referred to above was just due to the compressed plot. The cdf plot from ssdtools (upper plot) and fitdistrplus (lower plot) are the same for the scaled data case:

A) Fit with ssdtools

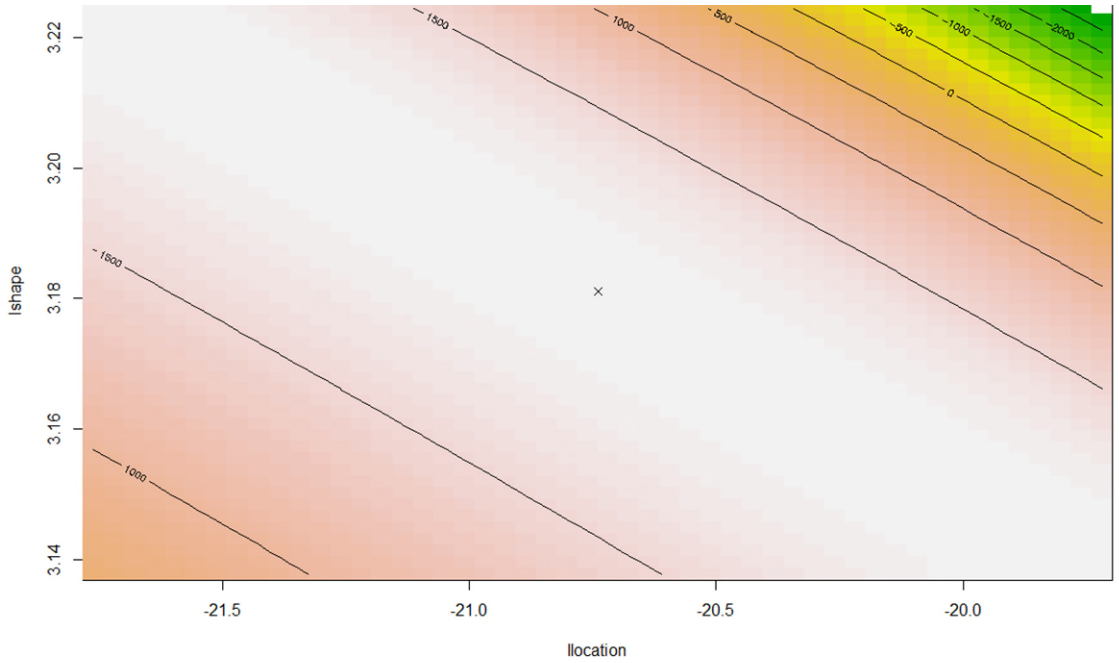



B) Fit with fitdistrplus



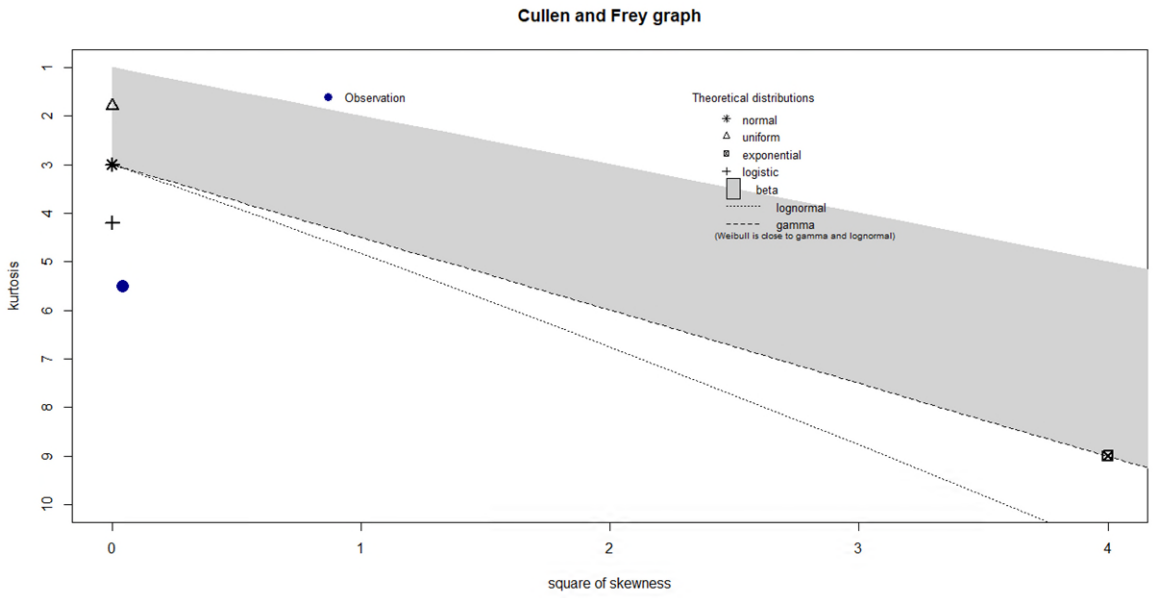

**The log-likelihood surface and Cullen-Frey plots**

The plot below shows the log-likelihood surface from the fitdistrplus fit to the scaled data. The 'x' mark is the MLE.

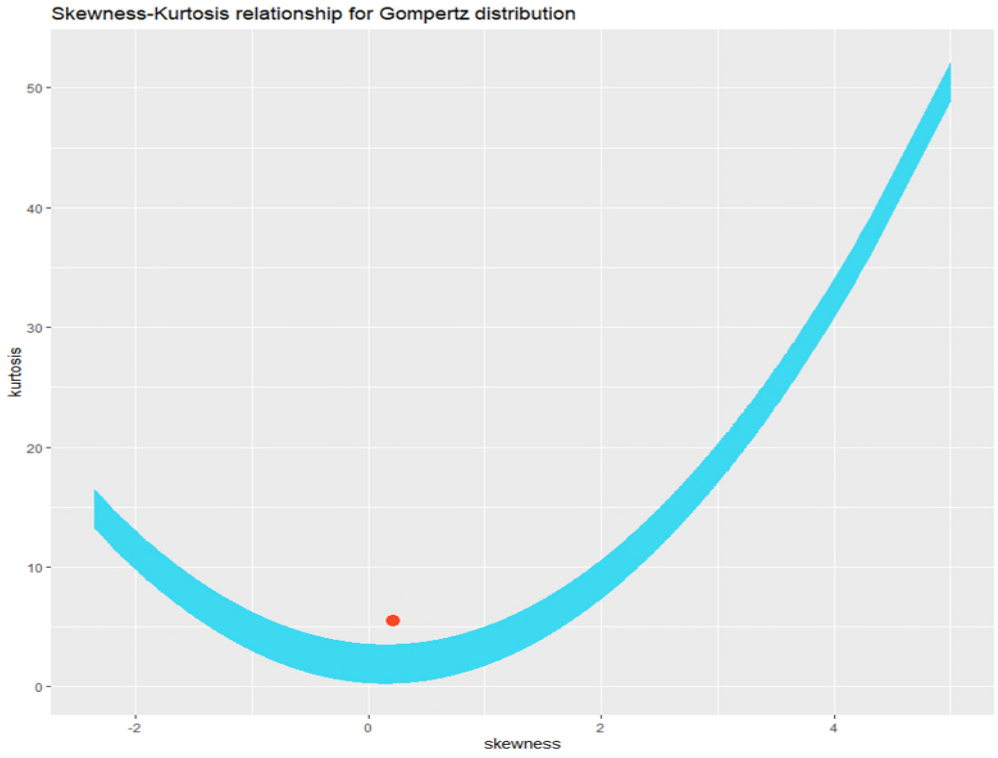



What's immediately apparent from this plot is that the MLE lies on a ridge meaning there is an infinite number of equally plausible estimates of the shape and scale parameters.

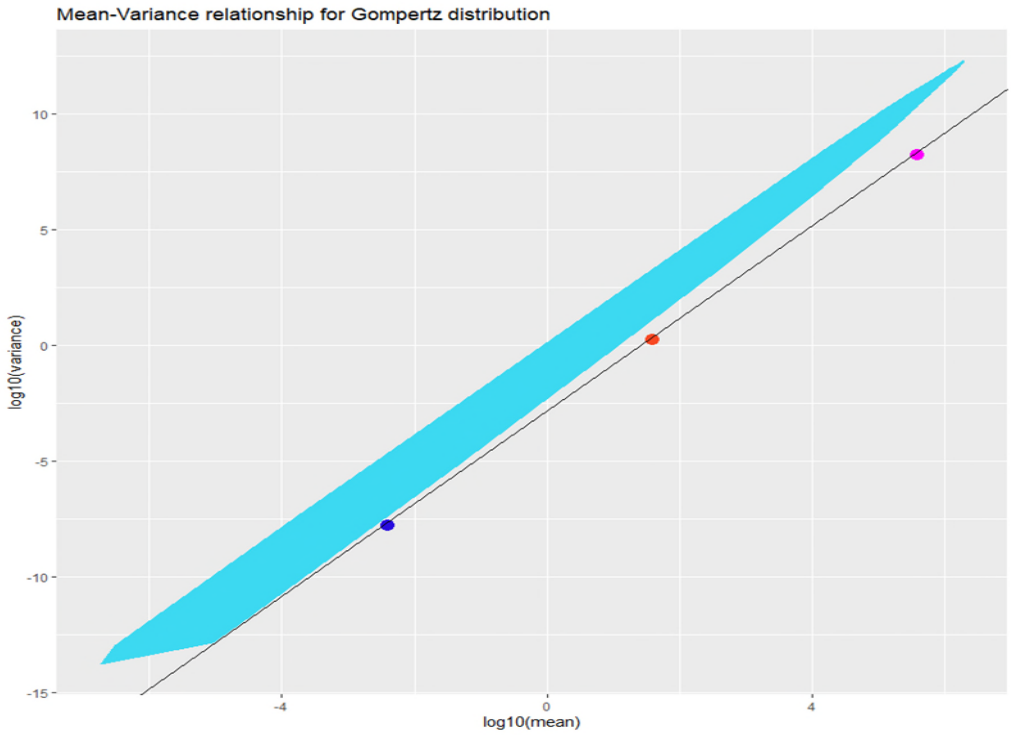
If we look at the Cullen-Frey plot we see that the data we're using falls outside any of the fitdistrplus distributions (indicated by the solid blue point at about (0,5.5).




This prompted an examination of the Cullen-Frey plot for the Gompertz distribution – see below (note, horizontal axis is skewness rather than skewness^2). The blue band represents feasible skewness-kurtosis combinations for the Gompertz distribution. The orange sold point represents our sample (the location of this point is scale-invariant).




So, what the plot above suggests is that it's impossible to match the skewness and kurtosis of the data with *any* Gompertz distribution. Now, that may not be a 'show-stopper' in terms of ML estimation, but it is also instructive to see where our sample lies on a mean-variance plot for the Gompertz family:




In the plot above, the blue band is again all feasible combinations of mean and variance using a Gompertz distribution. The orange point is the unscaled data. Note, that this plot does depend on scale as indicated by the magenta point which is the data multiplied by a big scale factor and the blue point which is the data divided by a big scale factor. The connecting line is the locus of the result of applying *any* scaling factor.

So this is revealing – we know that we can't match the third and fourth moments of our data using a Gompertz distribution and the plot above suggests that we will struggle to match the first two moments. Further, applying a large multiplicative scaling factor takes us further away from a feasible position on the mean-variance plot, while applying a very small (<<1) scale factor brings us closer to a feasible position on this plot.

On the basis of the above, it suggests that this this is not necessary a 'convergence issue'– it's simply an artefact of trying to fit a model that is incapable of representing the characteristics of the particular data set – we're essentially trying to force a square peg into a round hole!

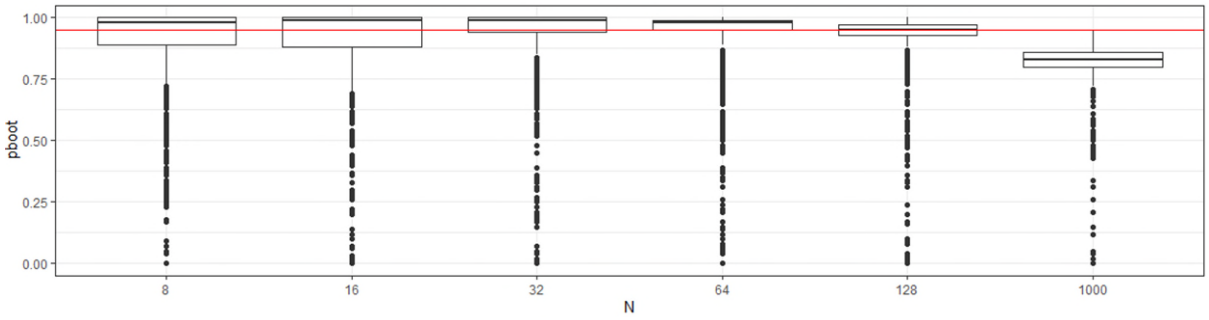
These investigations suggest that there are inherent difficulties with fitting the gomptertz distribution to some datasets that occur within ssdtools as well as alternative packages, such as fitdistrplus. This is related to an extremley flat likelihood profile, meaning there are an infinite number of equally plausible estimates.

This issue appears to associated with the fact that the testing data do not fit within plausible parameter combinations of the gompertz distribution. It may be impossible in such cases to ever resolve convergence issues.

Regardless, the gompertz distribution may remain a useful distribution in some cases, particularly if it represents completely different distribuional shapes relative to the others in the model set.

Failure to converge in the initial model set would simply suggest it is a poor model for that data. Where the gompterz fits well, it may be worth considering providing pboot values are high in confidence interval estimation.

To examine if convergence is reliable once a Gompertz distribution has been fit, we extracted datasets from Simulation study 1 and examined pboot values for all fits that were able to return a successful fit to the Gompertz distribution. We found that there were highly variable outcomes for pboot for successfully fit Gompertz models. Values of pboot for extremely large sample sizes (N=1,000) were always below 0.95. For all sample sizes <128 median pboot was > 0.95.




## 8.1 Conclusions:

However, in all cases there

To further explore the reprex highlighting Gompertz instability provided in the ssdtools github issues page https://github.com/bcgov/ssdtools/issues/223 we refit the example data using a range of different seeds, through the following code:

```
x <- c(3.15284072848962, 1.77947821504531, 0.507778085984185,
1.650387414067, 1.00725113964435, 7.04244885481452,
1.32336941144339, 1.51533791792454)

test_issues_dat_seed <- lapply(1:100, FUN = function(s){
          set.seed(s)
fit <- try(ssd_fit_dists(data, left = 'Conc', dists = "gompertz",
rescale = FALSE), silent = TRUE)
})
```

Of these 100 seeds 31 were able to return a valid fit, with the remaining 69 failing. We extracted HC estimates from the successful fits via:

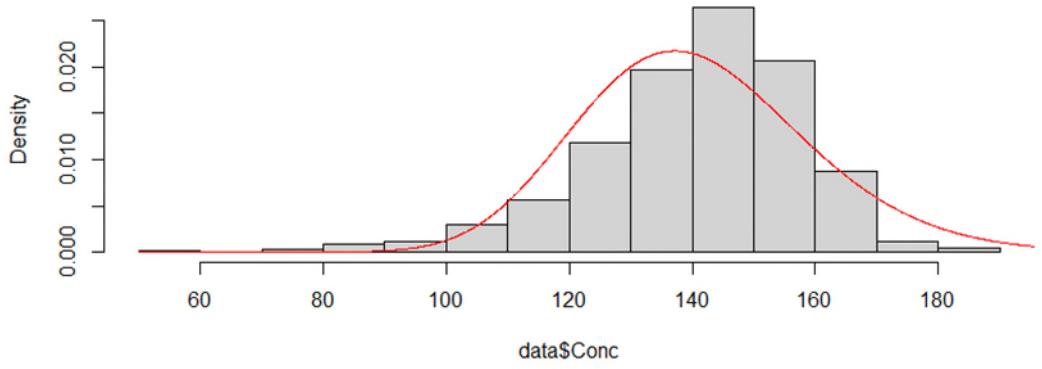
```
HC_vals <- sapply(test_issues_dat_seed, FUN = function(g){
          if(class(g)=="fitdists"){return(ssd_hc(g)$est)}
})
```

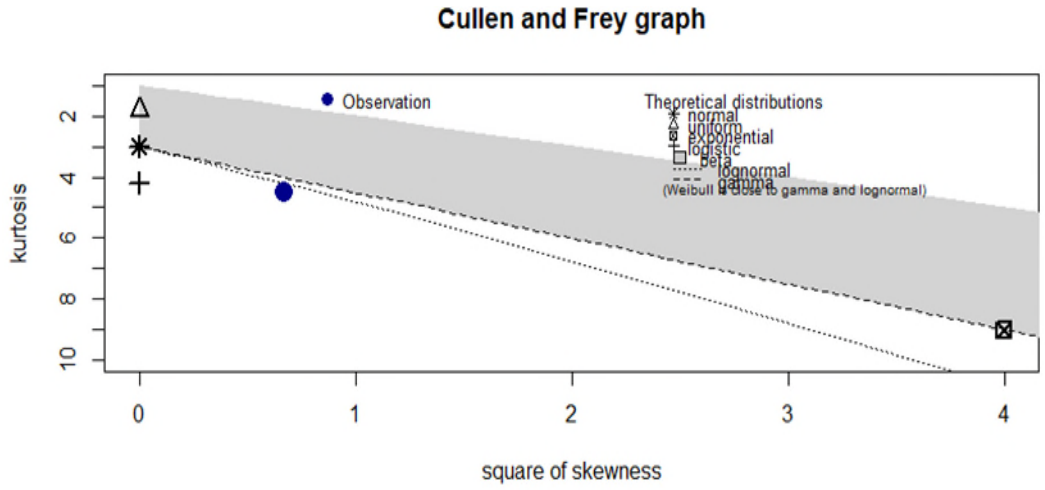
The estimated HC5 values ranged from 0.115 to 0.376 across the 31 successful fits, with a median of 0.134. This represents a 1.9 fold difference in the HC estimate, from a single dataset based entirely on differences among seeds. Such behaviour is clearly problematic in the context of scientific reproducibility, and suggests that if the gompertz were implemented it would be necessary as a minimum to set a seed to obtain consistent results.

# Appendix E : Instability when N is large

Prior testing has shown that the lognormal distribution can surprisingly show convergence issues when sample sizes are very high (see Figure 32, Fox et al 2022). For example, from Simulation study 1 the lognormal distribution failed to fit up to 14% of the time when N=1,000 for data simulated from an Inverse Weibull, log logistic and lognormal distributions (see Figure 32, Fox et al 2022). Here we describe our investigations this instability.

Consider the following data:




The summary statistics for the *n=1,000* data values are:

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
52.31  131.74  143.19  140.76  152.02  187.29
Skewness = -0.8189654;  Kurtosis = 4.50964
```

The mle's for the fitted lognormal distribution are easily obtained as the mean and (biased) standard deviation of the log-transformed data respectively. i.e $\hat{\mu} = 4.938859$ and $\hat{\sigma} = 17.10578$.

However, `ssdtools` fails to fit the lognormal distribution to this data:

```
> str(data)  # confirm that we are using a properly formatted data frame
'data.frame':  1000 obs. of  1 variable:
 $ Conc: num  142 146 146 142 149 ...
> ssd_fit_dists(data=data,dists="lnorm")
Error:
```

```
! All distributions failed to fit.
Run `rlang::last_trace()` to see where the error occurred.

Warning message:
Distribution 'lnorm' failed to converge (try rescaling data): ERROR: ABN
ORMAL_TERMINATION_IN_LNSRCH.
```

BUT, if we use a smaller sample randomly selected from the *n=1,000* data values, ssdtools successfully fits the lognormal distribution:

```
ssd_fit_dists(data=data.frame(Conc=sample(data$Conc,8)),dists="lnorm")
Distribution 'lnorm'
  meanlog 4.91451
  sdlog 0.150053

Parameters estimated from 8 rows of data.

ssd_fit_dists(data=data.frame(Conc=sample(data$Conc,32)),dists="lnorm")
Distribution 'lnorm'
  meanlog 4.93776
  sdlog 0.137802

Parameters estimated from 32 rows of data.

ssd_fit_dists(data=data.frame(Conc=sample(data$Conc,320)),dists="lnorm")
Distribution 'lnorm'
  meanlog 4.93646
  sdlog 0.131214

Parameters estimated from 320 rows of data.

ssd_fit_dists(data=data.frame(Conc=sample(data$Conc,640)),dists="lnorm")
Distribution 'lnorm'
  meanlog 4.93629
  sdlog 0.128506

Parameters estimated from 640 rows of data.
```

But, most bizarrely if we simply *permute* the rows of the data frame, `ssdtools` works:

```
ssd_fit_dists(data=data.frame(Conc=data[permute::shuffle(1000),]),dists=
"lnorm")
Distribution 'lnorm'
  meanlog 4.93886
  sdlog 0.132416

Parameters estimated from 1000 rows of data.
```

**Note:** `fitdistrplus` has no difficulties. For example, we can fit the lognormal distribution using the `optim()` function (rather than computing `meanlog` and `sdlog` as follows:

```
> mledist(data$Conc,"lnorm") # uses the default Nelder-Mead method
$estimate
  meanlog     sdlog
4.9388590 0.1324164

$convergence
[1] 0

$value
[1] 4335.994


$hessian
          meanlog    sdlog
meanlog 57031.69      0.0
```

```
sdlog       0.00 114121.9

$optim.function
[1] "optim"

$optim.method
[1] "Nelder-Mead"

$fix.arg
NULL

$fix.arg.fun
NULL

$weights
NULL

$counts
function gradient
      59       NA

$optim.message
NULL

$loglik
[1] -4335.994
```

Further investigation into this issue revealed that the non-convergence of the lnorm for large samples sizes (e.g. 1,000) is because in this particular case the initial values which are estimated from the input data are almost exactly equal to the MLEs. This causes issues with the specific optimization algorithm utilised in ssdtools via TMB. The source of the problem appears to come from some deep-seated issue in the old (Fortran 77) solvers being used by TMB. The current solution is to adjust all initial values as prior to passing to the optimization engine, to ensure they are not the exact MLE solutions.

This solution should be considered only temporary and does nothing to address the underlying issue. In the case where closed form solutions for ML estimation exist it would be possible to implement these for the relevant distributions using binary logic that by-passes the solver and just returns the exact values. Such a solution would require significant re-working of the ssdtools code base and is beyond the scope of the present work. The temporary work-around that has been implemented is technically unsatisfactory, but nevertheless does ensure reliable fits with accurate MLE estimates and can be adopted safely in the interim.

Re-testing of the Simulation data 1 showed that the interim solution returns much higher convergence proportions for the lognormal distribution at larger samples sizes. Across all 2,632 datasets in the simulation study there is now 97.8% convergence at sample sizes of 1,000, 0.99% convergence for sample sizes of 64, 128 and complete convergence of all smaller sample sizes.

| 8 | 16 | 32 | 64 | 128 | 1000 |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0.990502 | 0.985562 | 0.97758 |

# Appendix F : lnorm-lnorm stability investigations

Prior testing showed that the ssdtools implementation for fitting lognormal-lognromal mixture distributions can have poor convergence behaviour (see Figure 32, Fox et al 2022). This behaviour can result in low pboot values being returned during bootstrapping of some distributions (see the related issue at https://github.com/bcgov/ssdtools/issues/295). To ensure reasonable behaviour ssdtools has lowered the pboot threshold for the bcanz function to ensure the lognormal-lognormal mixture distiribution rreliably eturns confidence intervals. However, lowering pboot has the potential to intoduce bias in the HC confidence interval estimation if the 'failed' bootstrap samples are not representative of the parent distribution. Here we examine the stability of the lnorm-lnorm mixture distribution using a range of testing data, and evaluate different strategies for improving convergence so this mixture distribution can be included with confidence in the default set when the minimum sample size requirements are met.

## Investigating instability issues

```
> x<-failed_data_allN[[1]]$failed.lnorm_lnorm[,1]
```

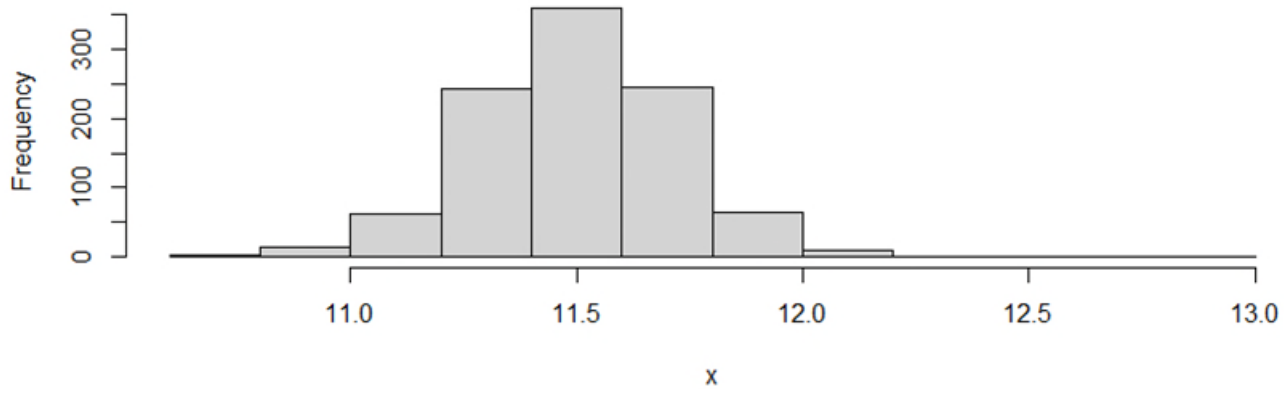



Figure F- 1. Histogram of example dataset

```
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.75   11.36   11.49   11.50   11.64   12.83

> skewness(x)
[1] 0.1429461
> kurtosis(x)
[1] 4.369028
```

**Unscaled data - ssdtools**

```
> fit <- ssd_fit_dists(data=data.frame(Conc=x), left = 'Conc', dists = 'lno
rm_lnorm', at_boundary_ok=TRUE, rescale = FALSE)

Error:
! All distributions failed to fit.
Run `rlang::last_trace()` to see where the error occurred.
Warning message:
Distribution 'lnorm_lnorm' failed to fit (try rescaling data): Error in opt
im(par, fn, gr, method = method, lower = lower, upper = upper,  :
  L-BFGS-B needs finite values of 'fn'
```

## Unscaled data – fitdistrplus

**(hand-coded dlnorm_lnorm)**

```
> fitdist(data=x,distr = "lnln",start=list(lm1=2.5,ls1=-1.5,lm2=2,ls2=-1.5,
p=0.99))
<simpleError in optim(par = vstart, fn = fnobj, fix.arg = fix.arg, obs = da
ta,     gr = gradient, ddistnam = ddistname, hessian = TRUE, method = meth,
lower = lower, upper = upper, ...): function cannot be evaluated at initial
parameters>
Error in fitdist(data = x, distr = "lnln", start = list(lm1 = 2.5, ls1 = -1
.5,  :
  the function mle failed to estimate the parameters,
                with the error code 100

> fitdist(data=x,distr = "lnln",start=list(lm1=2.5,ls1=-1.5,lm2=2,ls2=0.000
1,p=0.99))
<simpleError in optim(par = vstart, fn = fnobj, fix.arg = fix.arg, obs = da
ta,     gr = gradient, ddistnam = ddistname, hessian = TRUE, method = meth,
lower = lower, upper = upper, ...): function cannot be evaluated at initial
parameters>
Error in fitdist(data = x, distr = "lnln", start = list(lm1 = 2.5, ls1 = -1
.5,  :
  the function mle failed to estimate the parameters,
                with the error code 100
```

## Scaled data – ssdtools

```
> ssd_fit_dists(data=data.frame(Conc=scale(x,center = FALSE,scale=TRUE)),d
ists='lnorm_lnorm')
Error:
! All distributions failed to fit.
Run `rlang::last_trace()` to see where the error occurred.
Warning message:
Distribution 'lnorm_lnorm' failed to fit (try rescaling data): Error in op
tim(par, fn, gr, method = method, lower = lower, upper = upper,  :
  L-BFGS-B needs finite values of 'fn'
.
> ssd_fit_dists(data=data.frame(Conc=scale(x,center = FALSE,scale=TRUE)),d
ists='lnorm_lnorm',rescale = TRUE)
Error:
! All distributions failed to fit.
Run `rlang::last_trace()` to see where the error occurred.
Warning message:
Distribution 'lnorm_lnorm' failed to fit: Error in optim(par, fn, gr, meth
od = method, lower = lower, upper = upper,  :
  L-BFGS-B needs finite values of 'fn'
.
```

## Scaled data – fitdistrplus

```
> fitdist(data=as.numeric(scale(x,center = FALSE,scale=TRUE)),distr = "lnl
n",start=list(lm1=2.5,ls1=-1.5,lm2=2,ls2=0.0001,p=0.99))
<simpleError in optim(par = vstart, fn = fnobj, fix.arg = fix.arg, obs = d
ata,     gr = gradient, ddistnam = ddistname, hessian = TRUE, method = met
h,     lower = lower, upper = upper, ...): function cannot be evaluated at
initial parameters>
Error in fitdist(data = as.numeric(scale(x, center = FALSE, scale = TRUE))
,  :
  the function mle failed to estimate the parameters,
                with the error code 100
```

## Unscaled data – alternative solvers

```
> optimx(par=c(2,0.1,1,1,0.9),llnln,lower=c(-Inf,0,-Inf,0,0),upper=c(Inf,Inf,Inf,Inf,1))
               p1        p2       p3       p4        p5     value fevals gevals nit
er convcode  kkt1 kkt2 xtime
L-BFGS-B 2.441941 0.0188725 1.047126 1.025552 0.9987057 -98.60032     77     77
NA       52 FALSE   NA   0.2
There were 23 warnings (use warnings() to see them)
> optimx(par=c(2,0.1,1,1,0.9),llnln,lower=c(-Inf,0,-Inf,0,0),upper=c(Inf,Inf,Inf,Inf,1),control=list(all.methods=TRUE))
```

| | p1 | p2 | p3 | p4 | p5 | value | fevals | gevals | niter | convcode | kkt1 | kkt2 | xtime |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| L-BFGS-B | 2.44 | 0.02 | 1.05 | 1.03 | 1.00 | -98.60 | 77 | 77 | NA | 52 | FALSE | NA | 0.11 |
| nlminb | 2.44 | 0.02 | 2.09 | 1.14 | 1.00 | -99.55 | 166 | 1154 | 150 | 1 | FALSE | NA | 0.09 |
| spg | 2.54 | 0.00 | 2.44 | 0.02 | 0.00 | -94.88 | 396 | NA | 363 | 0 | NA | NA | 0.24 |
| Rcgmin | 2.17 | 1.58 | 2.61 | 0.16 | 0.00 | 2067.25 | 41 | 17 | NA | 0 | FALSE | FALSE | 0 |
| Rvmmin | 2.15 | 3.74 | 2.44 | 0.03 | 0.00 | -12.93 | 41 | 21 | NA | 21 | FALSE | FALSE | 0 |
| bobyqa | 2.51 | 0.18 | 1.10 | 1.02 | 0.92 | 1796.39 | 28 | NA | NA | 0 | FALSE | FALSE | 0 |
| nmkb | 2.44 | 0.02 | 2.68 | 0.34 | 1.00 | -100.40 | 796 | NA | NA | 0 | FALSE | NA | 0.19 |
| hjkb | 2.44 | 0.04 | 2.44 | 0.02 | 0.04 | -103.31 | 2928 | NA | 19 | 0 | FALSE | TRUE | 0.33 |

**Comment:** As can be seen, there are several alternatives to optim() that do successfully converge (including optim with method ="L-BFGS-B"):

```
> optim(par=c(2,0.1,1,1,0.9),llnln,lower=c(-Inf,0,-Inf,0,0),upper=c(Inf,Inf,Inf,Inf,1),method="L-BFGS-B")
$par
[1] 2.4419412 0.0188725 1.0471260 1.0255519 0.9987057

$value
[1] -98.60032

$counts
function gradient
      77       77

$convergence
[1] 52

$message
[1] "ERROR: ABNORMAL_TERMINATION_IN_LNSRCH"
```

**Comment:** There's nothing further about this error message. A search online doesn't provide anything particularly revealing. I suspet it's a numerical issue associated with gradient Computations:

```
convergence  52

indicates an error from the "L-BFGS-B" method; see component message for
further details.
```

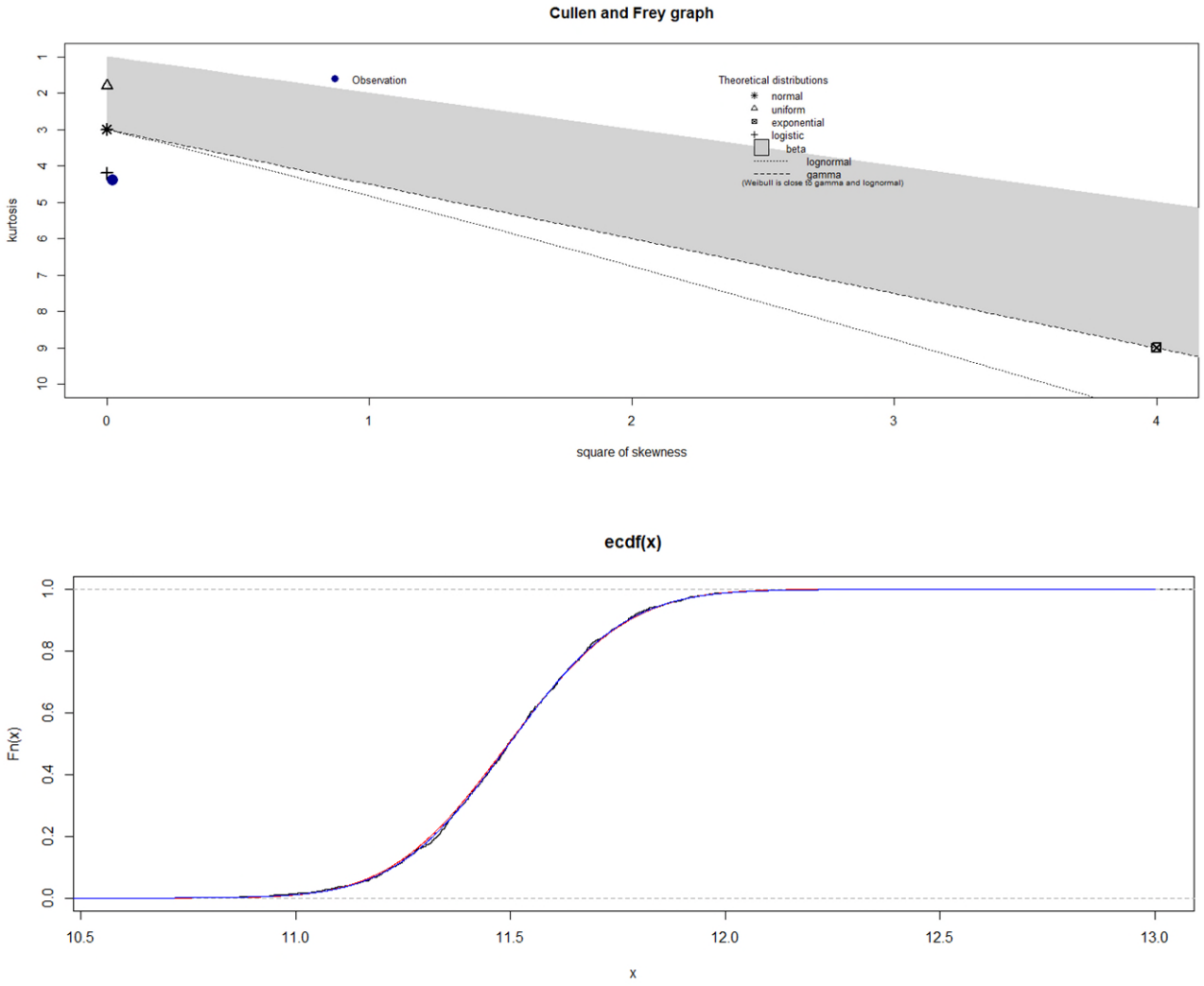



**Figure F- 2. This is a plot of the ecdf (black) with 2 fitted distributions: with parameters estimated from: L-BFGS-B (red); and hjkb (blue). The fit is good because the mixing parameter is ~ 0 => a single log-normal.**

It is clear from this investigation that there are are convergence issues with the log-normal log-normal mixture distribution where the mixing parameter been estimated is near 1 (a unimodal distribution), that are common to both ssdtools and fitdistrplus. These issues are related to the optimisation algorithm being used. While there are alternative solvers that do provide a valid solution these cannot currently be implemented within ssdtools as a means of resolving this issue.

## Comparing bootstrapping methods

Note that the primary purpose of improving stability of the fitting algorithim for the lognormal-lognormal mixture distribution is to ensure sucessful bootstrap sampling and maximize the `pboot` to minimise the possibility of potential bias in HC and HP confidence interval estimates. Failure of the lognormal-lognormal mixture distribution to fit initially to the original data is of less concern, because the mixture will simply be left out of the model set. In that case, failure to converge simply suggests that the mixture distribution is inappropriate for these data.

Low bootstrap convergence was identified as the source of the reason the current Shiny App fails to returnconfidence intervals for the testing dataset, boron (see

https://github.com/bcgov/ssdtools/issues/295). The revised version of ssdtools implements two alternative bootstrapping methods (See Milestone Report 1), the rmulti method, which draws a random sample from the model set as a mixture distribution, and the weighted sample method that draws random samples from the individual distributions and combines these in proportion to their AICc weights.

We examined how pboot compares across the two bootstrapping methods using the CCME boron dataset as a case study. We found that there was perfect convergence (pboot=1) when using the weighted sample method, and a pboot of 0.896 when using the rmulti method. Investigation into the distributions that failed to converge suggest that these are almost entirely based on failure of the lognormal-lognormal mixture to converge. This provides further support for the idea that convergence issues with the lognormal lognormal mixture are associated with failure to fit essential unimodal datasets. For the weighted sample method, bootstrapping of the lognormal-lognormal distribution within the set is done only using data frame from that same mixture distribution, which likely explains the higher convergence rate using that method. Given that the wieghted bootstrap sample method is faster than the rmulti method and appears to return similar confidence intervals, there may be no critical need to resolve convergence issues for the lognormal-lognormal distribution when applied to univariate data.

### *Coefficient of bimodality*

The source of convergence issues associated with the lognormal-lognormal mixture are likely in part associated with attempts to fit a unimodal distiribution. The team discussed the possibility of pre-screening data for evidence of bi-modality using a coefficient of bimodality, as is currently outlined in Warne et al. 2018.

From Wikipedia **(**https://en.wikipedia.org/wiki/Multimodal_distribution **)**

**Bimodality coefficient** [ edit ]

Sarle's bimodality coefficient *b* is[25]

$$\beta = \frac{\gamma^2 + 1}{\kappa}$$

where $\gamma$ is the skewness and $\kappa$ is the kurtosis. The kurtosis is here defined to be the standardised fourth moment around the mean. The value of *b* lies between 0 and 1.[26] The logic behind this coefficient is that a bimodal distribution with light tails will have very low kurtosis, an asymmetric character, or both – all of which increase this coefficient.

The formula for a finite sample is[27]

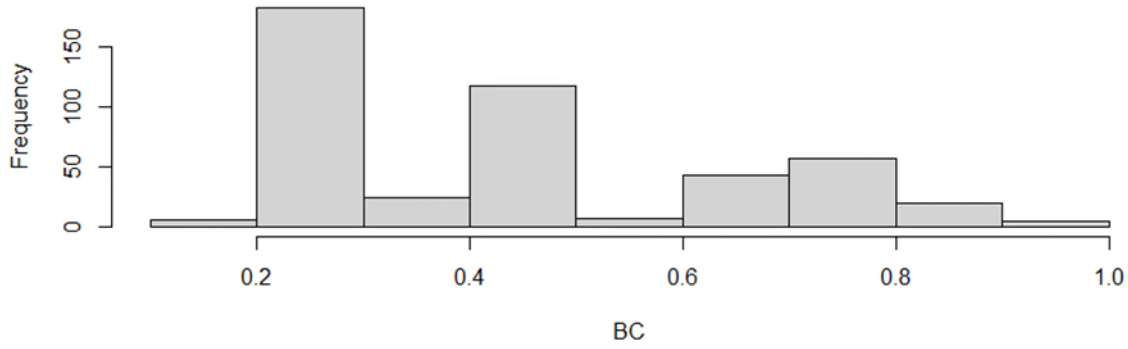
$$b = \frac{g^2 + 1}{k + \frac{3(n-1)^2}{(n-2)(n-3)}}$$

where *n* is the number of items in the sample, *g* is the sample skewness and *k* is the sample excess kurtosis.

The value of *b* for the uniform distribution is 5/9. This is also its value for the exponential distribution. Values greater than 5/9 may indicate a bimodal or multimodal distribution, though corresponding values can also result for heavily skewed unimodal distributions.[28] The maximum value (1.0) is reached only by a Bernoulli distribution with only two distinct values or the sum of two different Dirac delta functions (a bi-delta distribution).

The distribution of this statistic is unknown. It is related to a statistic proposed earlier by Pearson – the difference between the kurtosis and the square of the skewness (*vide infra*).

This is what's recommended in Warne et al.




### *Histogram of all 463 data sets of n=1,000*

```
bc<-function(dat){
  x<-dat
  n<-length(x)
  coef<-(skewness(x)^2+1)/((kurtosis(x)-3)+(3*(n-1)^2)/(n-2)/(n-3))
  return(coef)
}
```

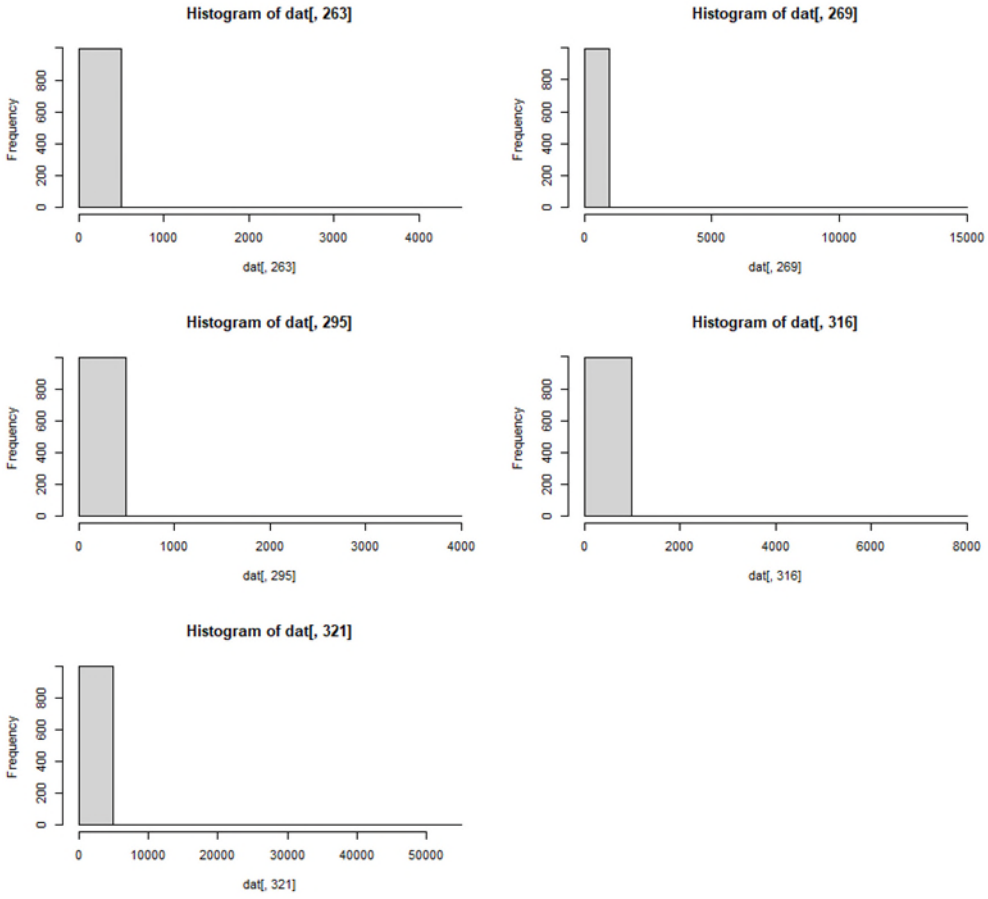



**Figure F- 3. Example data sets for samples to failed to converge for a lognormal-lognormal mixture distribution for which bc > 0.9**

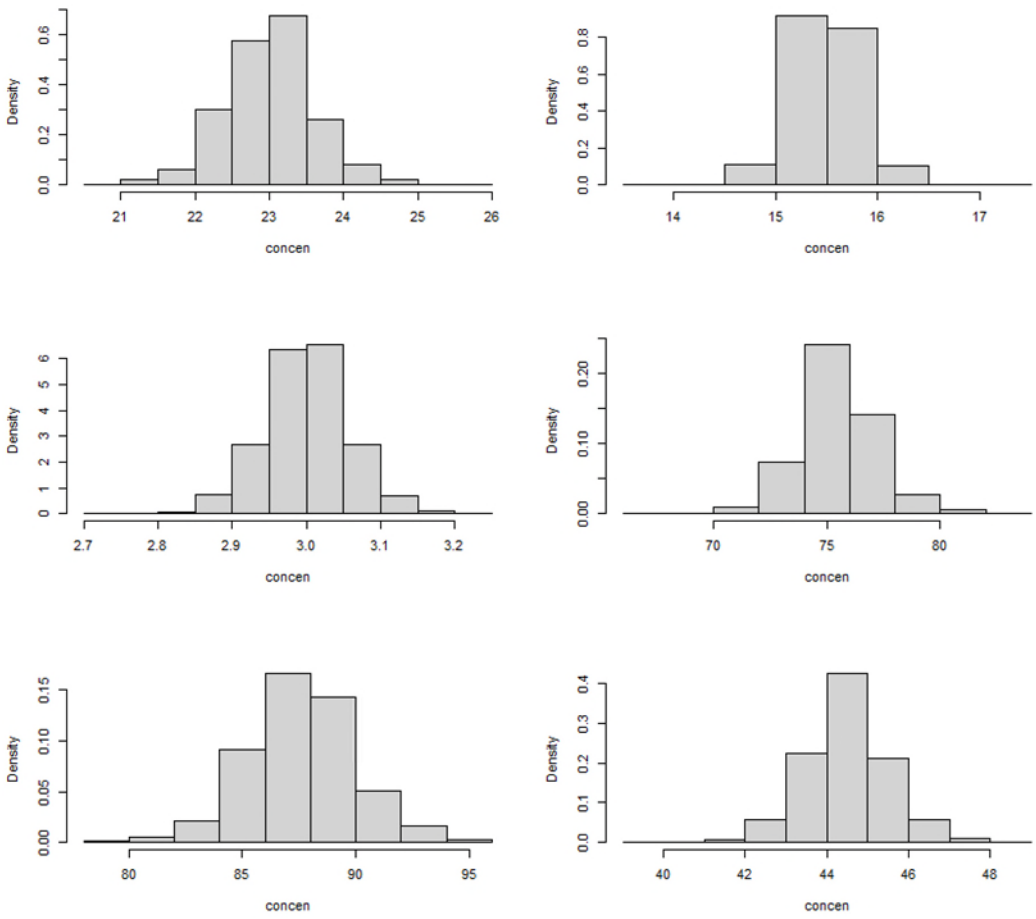



**Figure F- 4. Example data sets for samples to failed to converge for a lognormal-lognormal mixture distribution for which bc 0.4 < bc < 0.5**

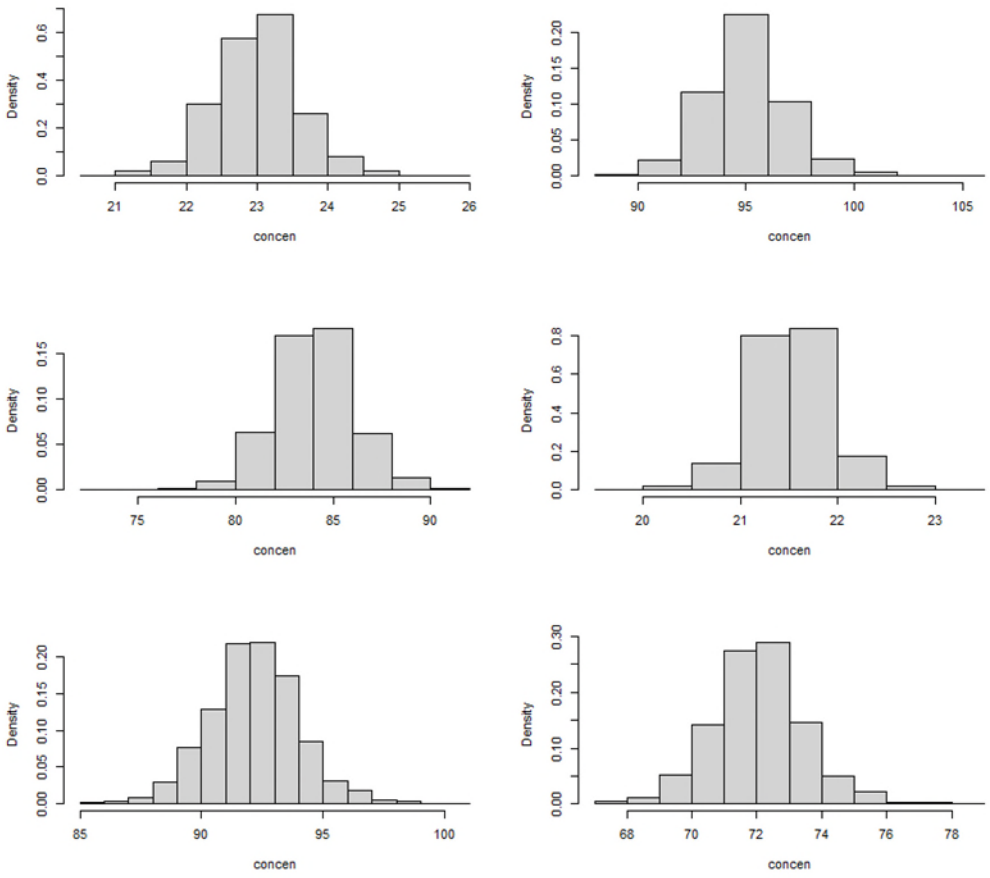



**Figure F- 5. Example data sets for samples to failed to converge for a lognormal-lognormal mixture distribution for which bc > 0.2 < bc < 0.3**

## Constraining pmix (the mixing proportion)

The ssdtools package on CRAN currently models the mixing parameter (`pmix`) on the logit scale, which allows 0-1 bounded parameters to be appropriately constrained without requiring an explicit specification of parameter constraints since values of the logit-transformed proportion are unconstrained.. Where data do not display any bi- or multi-modality, this may result in an attempt to estimate extremely small (near 0) or large (near infinity) values of pmix, because for a unimodal dataset pmix is theoretically 1. As the estimate of pmix tends to 1, the value of logit(pmix) will tend to infinity and this can result in unstable behaviour. We believe this is the reason behind the lnorm-lnorm convergence issues. In the first instance the team has tested an implemntation of ssdtools that estimates pmix on it's natural scale by requiring pmix to satisfy the constraint $0 \leq pmix \leq 1$ . In the bootstrap setting, it is important to allow the lognormal lognormal mixture to return a successfully converged fit, even when the parameter estimate is at these bounds, because 0 or 1 are the true theoretical pmix values for a univariate distribution. The current version of ssdtools allows this by setting at_boundary_ok=TRUE.

In addition to fixing the bounds of pmix to 0 and 1 on the natural scale, we also examined setting varying bounds for min_pmix, based on the proportion that can theoretically be supported by the total sample size of the input data (N). The logic here is that if the mixture distribution is to be

supported, this must necessarily mean that at least 1, or some, of the input data must be from each of the component distributions. In that case, it may be reasonable to set bounds based on the minimun proportions of the input sample data. In our testing, we set the min_pmix bound based on 1/N, 2/N and 3/N to examine convergence reliability across a range of minimum required data for each component distribution. Note that setting the lower bound as a propotion in ssdtools also sets an equivalent assumed upper bound (ie (N-1)/N, (N-2)/N and (N-3)/N).

**Testing lognormal-lognormal convergence using data from Simulation study 1**

We explored the effect of bounding min_pmix on convergence reliability for the lognormal mixture by attempting to re-fit all datasets that failed to fit the datasets in sumulation study 1 (see further details in Fox et al. 2022). We found that re-factoring the pmix parameter so it was no longer on the logit scale and instead bounded to 0-1 did slightly improve convergence of the lognormal-lognormal mixture distribution (Figure F-6). This improvement was greatest for larger samples sizes and represents a 30-40% higher rate of convergence  for the datasets from simulation study 1 (N = 16 and 32,Figure F-6).

In addition to the refactoring such that min_pmix is bounded to 0 and 1, there was also substantial improvements in convergence when the min_pmix is bounded proportional to minimum representations of data within each distribution (Figure F-6). This gain was very minor for a min_pmix of 1/N (at least 1 datapoint in each component distribution) but jumped substantially for 2/N (at least 2 datapoints in each component distribution) and even further for 3/N (at least 3 datapoint3 in each component distribution, Figure F-6). For a min_pmix bound of 3/N there was 70-80% convergence for previoualy failed datasets, even at relatively small sample sizes (Figure F-6). When applied to all the original datasets in Simulation study 1, convergence rates in the case where min_pmix is bounded to 3/N exceeded 90% even for relatively small sample sizes, which is quite high given these simulated data are all from known unimodal distributions.

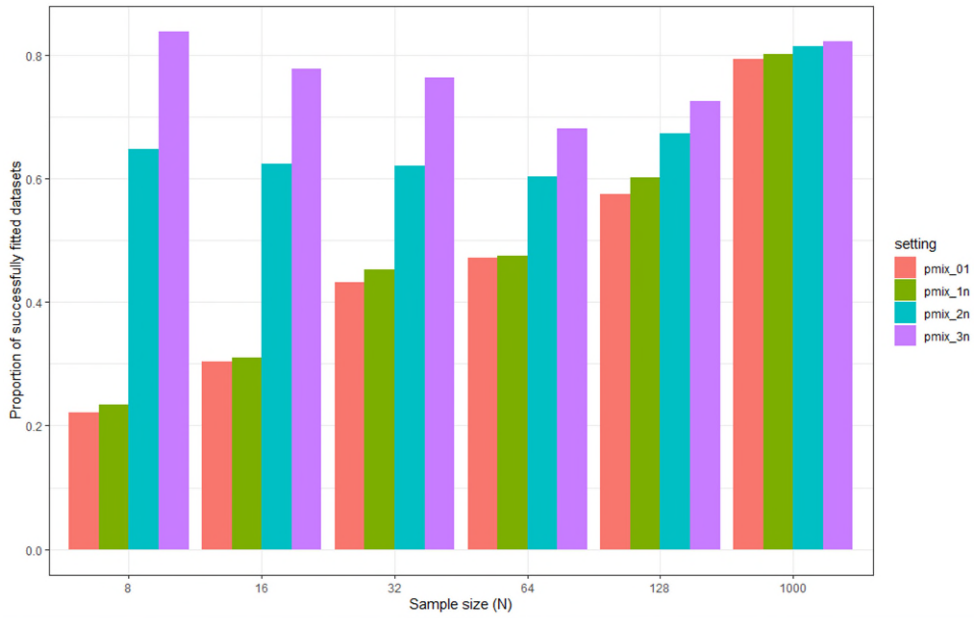



**Figure F- 6. Proportion of successfully fitted data sets based on Simulation study 1 that failed to fit the lognormal lognormal mixture using the current implementation of pmix on the logit scale with no bounds. Four different pmix bounds were examined including 0-1 bounded, and lower bounded at 1/N, 2/N and 3/N, where N is the test data size.**

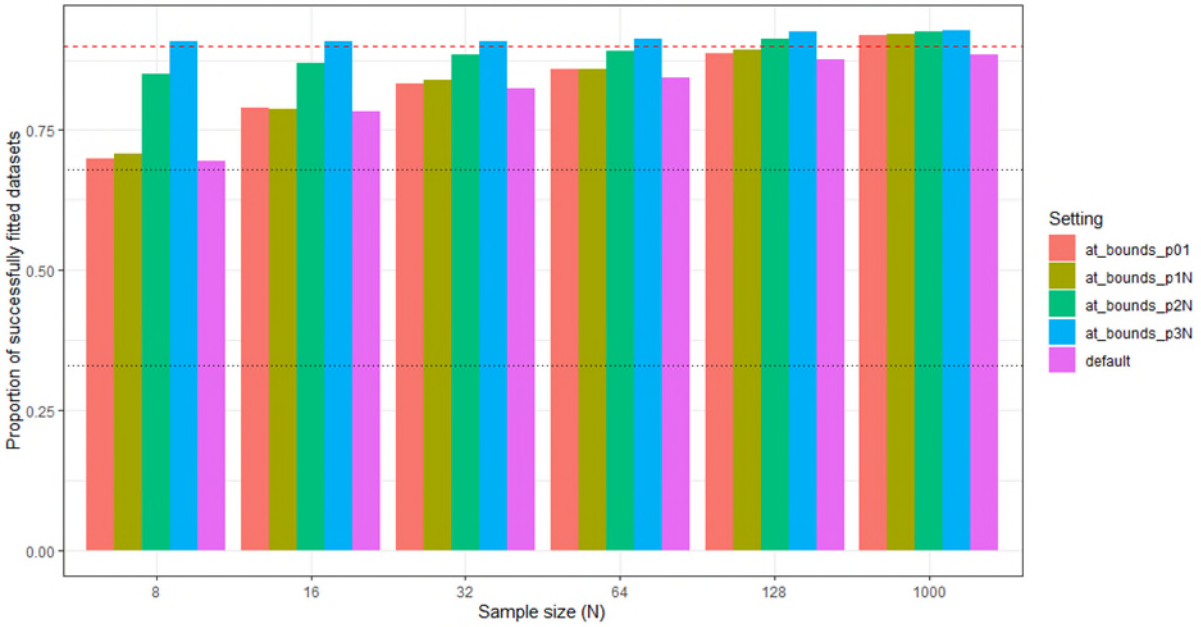



**Figure F- 7. Proportion of successfully fitted data sets based on all Simulation study 1 data sets (See Fox et al. 2022 for more details). Four different pmix bounds were examined including 0-1 bounded, and lower bounded at 1/N, 2/N and 3/N, where N is the test data size and successful convergence included bounded distributions (computable = FALSE, at_boundary_ok = TRUE). Also shown are results for the default settings (computable = TRUE, at_boundary_ok = FALSE, min_pmix = 0). Horizonal dotted lines indicate the range of convergence observed in the original study (see Figure 32, Fox et al 2021) and the horizontal dashed red line indicates 90% convergence.**

**Testing lognormal-lognormal mixture convergence the using failed bootstrap samples.**

One of the primary motivations for improving convergence in the lognormal-lognormal distribution is to return a high pboot estimate for the bootstrapping procedure to ensure the resulting confidence intervals remain unbiased. To test the effectiveness of different min_pmix bounds on convergence success in this case, we examined the effect of different bounding scenarios on convergence probabilities for a sample of 11,046 datasets that failed to result in a valid lognormal-lognormal fit using the default 0-1 bounds. The bounding scenarios incuded re-fitting using the default 0-1 bounds and min_pmix settings of 1/N, 2/N and 3/N.

The results of this test clearly indicate that there is a substantial gain in convergence probability when the minimum allowed mixing proportion corresponds to at least 2 data points (proportion converged = 0.61, Table 7). There was an even great gain, representing near perfect convergence in this example (proportion converged = 0.99) for a mixing proportion corresponding to at least 3 data points (Table 7).

**Table 7. The proportion of successfully converged datasets for a range of min_pmix settings.**

| min_pmix | Proportion |
|---|---|
| 0 | 0.01 |
| 1/N | 0.09 |
| 2/N | 0.61 |
| 3/N | 0.99 |

**Lognormal-lognormal convergence for true mixtures**

All of the tests above have looked at convergence failure for data that come largely from unimodal distributions. We simulated data from a true lognormal-lognormal mixture distribution to verify convergence rates and explore the effect of bounding min_pmix at different proportions of the total sample size.

This was achieved by randomly generating data using the rlorm function in R. Data were generated from two distributions and subsequentely pooled. Once distribution had a meanlog of 0 and sdlog of 1, and the other a meanlog of 5 and sdlog of 1. We generated 16 data points (N) from each distirbution with different known proportions (actual pmix values), ranging from 1/16 (only 1 data point from distribution one) to 15/16 (only 1 data point from distribution 2) and including all mixing proportions in between. These data were fitted using ssdtools allowing valid "convergence" to include fits where the pmix estimate is at the bounds (at_bounds_ok = TRUE), as well as excluding bounded fits (at_bounds_ok = FALSE).

Four different bounding scenarios were considered, including the default 0-1 bounds, and min_pmix settings of 1/N, 2/N and 3/N.

The exercise was repeated for a sample size (N) of 32, although in this case the simulated mixture data range fom a minimum of 2 data points from distribution one up to 2 data points from distribution 2.

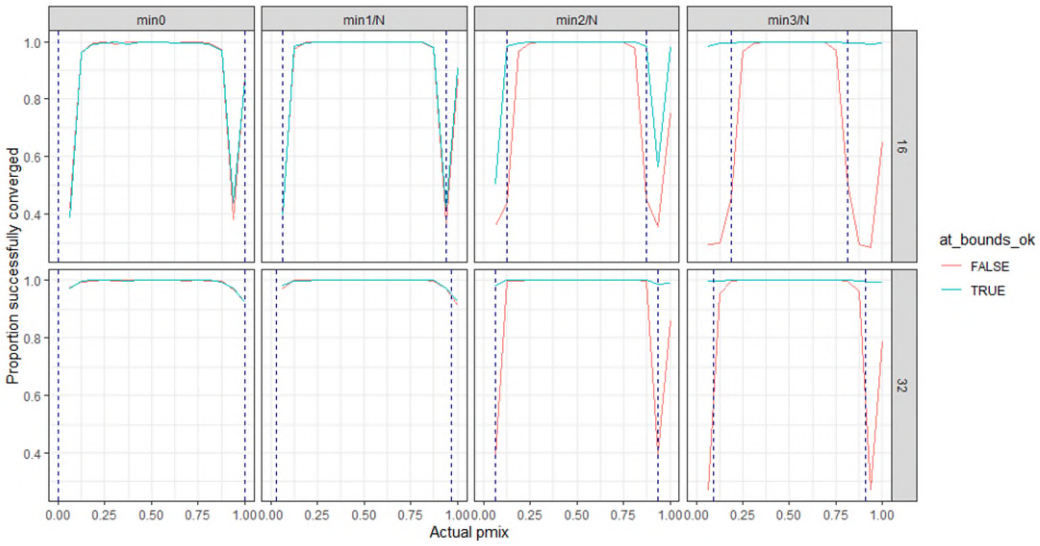



**Figure F- 8. Proportion of successfully converged lognormal-lognormal mixture fits as a function of the actual mixing proportion for a true lognormal mixture. Data were generated from two lognormal distributions (meanlog = 0 and sdlog = 1 meanlog = 5 and sdlog = 1), with actual true mixing proportions ranging from 1/16 to 15/16. Plot rows show data for two different sample sizes (N), and plot columns indicate the applied min_pmix boundary (the default 0-1 bounds, and min_pmix settings of 1/N, 2/N and 3/N). Line colours indicate the outcome where the pmix estimate is allowed at the bounds (at_bounds_ok = TRUE), as well as excluding bounded fits (at_bounds_ok = FALSE). Blue vertical lines show the position of the upper and lower bounds imposed by min_pmix.**

This simulation study shows that for a true mixture distribution with actual pmix values within the range of 0.2-0.8, and where convergence is considered successful when the estimated pmix values are allowed at the bounds (at_bounds_ok = TRUE), convergence rates are near 1 (Figure F-8). This is

true regardless of of the min_pmix bounds that are imposed (Figure F-8). Outside this range of actual mixing values convergence of a true mixure can be quite low when no lower and upper bounds are imposed (min_pmix = 0, m0 Figure F-8). Convergence for distributions at the extremes improves as min_pmix is increase from 1/N to 3/N, providing convergence is considered successful if the estimated pmix is at those boundaries (blue line, all plots, Figure F-8)). Convergence actually declines with increasing imposed min_pmix values if fits where the estimated pmix is at the boundaries are excluded (red line, all plots, Figure F-8)).

**Discussion and recommendations**

Re-factoring the ssdtools code such that the pmix paramater of the lognormal-lognormal mixture distribution resulted in a 20% improvement in convergence success and should definitely be adopted in the revision of ssdtools before submission to CRAN.

Setting the lower bound of the mixing proportion relative to the sample size of the input data can also result in substantial improvements to convergence reliability. There is little gain in setting the bound at 1/N, but significant gains with 2/N and the best performance is where min_pmix is set to 3/N. Convergence of the univariate distirbutions from Simulation study 1 was >90%, and ths was 99% for a large sample of previously failed boostrap datasets based on the boron example from ssddata.  The setting also results in near perfect convergence for true mixture distributions, providing valid fits include those with extimated pmix values at the bounds (at_boundary_ok).

# Appendix G – Updated ssdtools help documentation

## ssd_hc {ssdtools}

R Documentation

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 = 0.99,
  multi_est = TRUE,
  multi_ci = TRUE,
  weighted = TRUE,
  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 = 0.99,
  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. |
| `multi_ci` | A flag specifying whether to treat the distributions as constituting a single distribution which is now the recommended approach (as opposed to taking the mean) when calculating model averaged confidence intervals. |
| `weighted` | A flag which specifies whether to use the original model weights (as opposed to re-estimating for each bootstrap sample) unless `multi_ci = FALSE` in which case it specifies whether to 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**

Run 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_hp {ssdtools}

R Documentation

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 = 0.99,
  multi_est = TRUE,
  multi_ci = TRUE,
  weighted = TRUE,
  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 = 0.99,
  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. |
| `multi_ci` | A flag specifying whether to treat the distributions as constituting a single distribution which is now the recommended approach (as opposed to taking the mean) when calculating model averaged confidence intervals. |
| `weighted` | A flag which specifies whether to use the original model weights (as opposed to re-estimating for each bootstrap sample) unless `multi_ci = FALSE` in which case it specifies whether to 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

Run examples

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

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

---

# Appendix H – Getting Started Vignette

# Getting Started with ssdtools

ssdtools Team

2024-05-17

## Introduction

`ssdtools` is an R package to fit Species Sensitivity Distributions (SSDs) using Maximum Likelihood and model averaging.

SSDs are cumulative probability distributions that are used to estimate the percent of species that are affected and/or protected by a given concentration of a chemical. The concentration that affects 5% of the species is referred to as the 5% Hazard Concentration ($HC5$). This is equivalent to a 95% protection value ($PC_{95}$). For more information on SSDs the reader is referred to Posthuma et al. (2001).

In order to use `ssdtools` you need to install R (see below) or use the Shiny app. The shiny app includes a user guide. This vignette is a user manual for the R package.

## Philosophy

`ssdtools` provides the key functionality required to fit SSDs using Maximum Likelihood and model averaging in R. It is intended to be used in conjunction with tidyverse packages such as `readr` to input data, `tidyr` and `dplyr` to group and manipulate data and `ggplot2` (Wickham 2016) to plot data. As such it endeavors to fulfill the tidyverse manifesto.

## Installing

In order to install R (R Core Team 2018) the appropriate binary for the users operating system should be downloaded from CRAN and then installed.

Once R is installed, the `ssdtools` package can be installed (together with the tidyverse) by executing the following code at the R console

```
install.packages(c("ssdtools", "tidyverse"))
```

The `ssdtools` package (and ggplot2 package) can then be loaded into the current session using

```
library(ssdtools)
library(ggplot2)
```

## Getting Help

To get additional information on a particular function just type `?` followed by the name of the function at the R console. For example `?ssd_gof` brings up the R documentation for the `ssdtools` goodness of fit function.

For more information on using R the reader is referred to R for Data Science (Wickham and Grolemund 2016).

If you discover a bug in `ssdtools` please file an issue with a reprex (repeatable example) at https://github.com/bcgov/ssdtools/issues.

## Inputting Data

Once the `ssdtools` package has been loaded the next task is to input some data. An easy way to do this is to save the concentration data for a *single* chemical as a column called `Conc` in a comma separated file (`.csv`). Each row should be the sensitivity concentration for a separate species. If species and/or group information is available then this can be saved as `Species` and `Group` columns. The `.csv` file can then be read into R using the following

```
data <- read_csv(file = "path/to/file.csv")
```

For the purposes of this manual we use the CCME dataset for boron.

```
ccme_boron <- ssddata::ccme_boron
print(ccme_boron)
#> # A tibble: 28 x 5
#>    Chemical Species                 Conc Group        Units
#>    <chr>    <chr>                  <dbl> <fct>        <chr>
#>  1 Boron    Oncorhynchus mykiss      2.1 Fish         mg/L
#>  2 Boron    Ictalurus punctatus      2.4 Fish         mg/L
#>  3 Boron    Micropterus salmoides    4.1 Fish         mg/L
#>  4 Boron    Brachydanio rerio       10   Fish         mg/L
#>  5 Boron    Carassius auratus       15.6 Fish         mg/L
#>  6 Boron    Pimephales promelas     18.3 Fish         mg/L
#>  7 Boron    Daphnia magna            6   Invertebrate mg/L
#>  8 Boron    Opercularia bimarginata 10   Invertebrate mg/L
#>  9 Boron    Ceriodaphnia dubia      13.4 Invertebrate mg/L
#> 10 Boron    Entosiphon sulcatum     15   Invertebrate mg/L
#> # i 18 more rows
```

## Fitting Distributions

The function `ssd_fit_dists()` inputs a data frame and fits one or more distributions. The user can specify a subset of the following 10 distributions. Please see the Distributions and Model averaging vignettes for more information appropriate use of distributions and the use of model-averaged SSDs.

```
ssd_dists_all()
#>  [1] "burrIII3"      "gamma"         "gompertz"      "invpareto"
#>  [5] "lgumbel"       "llogis"        "llogis_llogis" "lnorm"
#>  [9] "lnorm_lnorm"   "weibull"
```

using the `dists` argument.

```
fits <- ssd_fit_dists(ccme_boron, dists = c("llogis", "lnorm", "gamma"))
```

## Coefficients

The estimates for the various terms can be extracted using the tidyverse generic `tidy` function (or the base R generic `coef` function).

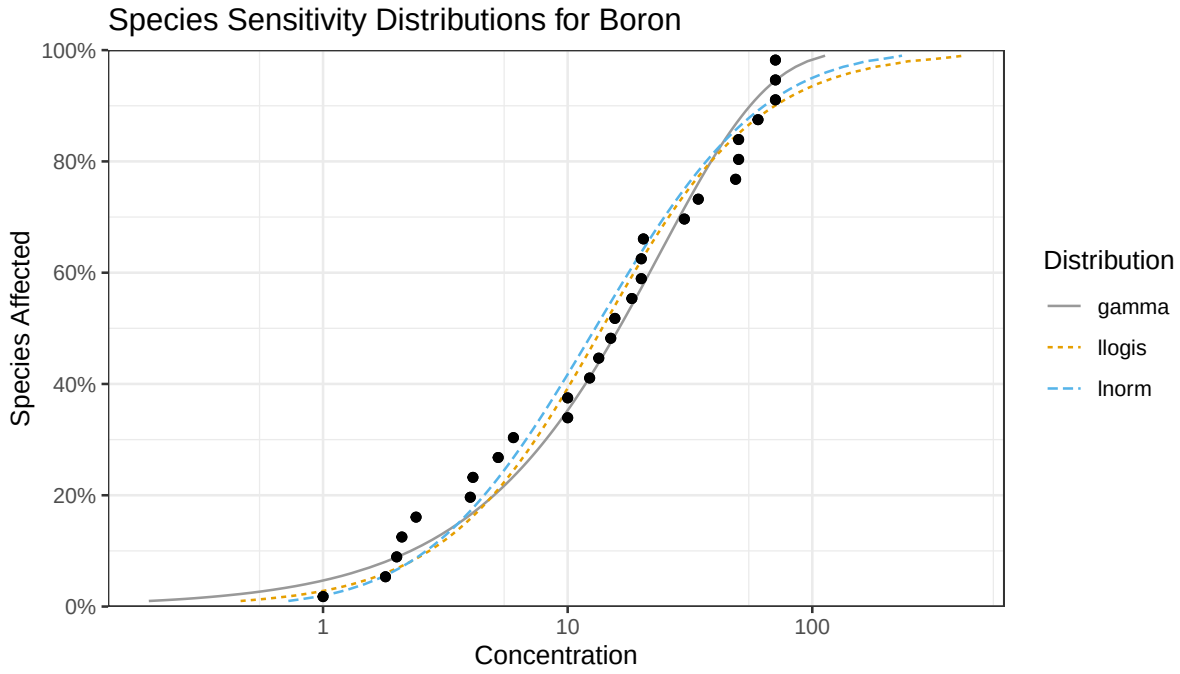
```
tidy(fits)
#> # A tibble: 6 x 4
#>   dist   term           est    se
#>   <chr>  <chr>        <dbl> <dbl>
#> 1 llogis locationlog  2.63  0.248
#> 2 llogis scalelog     0.740 0.114
```

```
#> 3 lnorm  meanlog      2.56  0.235
#> 4 lnorm  sdlog        1.24  0.166
#> 5 gamma  scale       25.1   7.64
#> 6 gamma  shape        0.950 0.223
```

## Plots

It is generally more informative to plot the fits using the `autoplot` generic function (a wrapper on `ssd_plot_cdf()`). As `autoplot` returns a `ggplot` object it can be modified prior to plotting.

```
theme_set(theme_bw()) # set plot theme

autoplot(fits) +
  ggtitle("Species Sensitivity Distributions for Boron") +
  scale_colour_ssd()
```




## Selecting One Distribution

Given multiple distributions the user is faced with choosing the "best" distribution (or as discussed below averaging the results weighted by the fit).

```
ssd_gof(fits)
#> # A tibble: 3 x 9
#>   dist      ad     ks     cvm   aic  aicc   bic delta weight
#>   <chr>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
#> 1 llogis 0.487 0.0994 0.0595  241.  241.  244.  3.38  0.11
#> 2 lnorm  0.507 0.107  0.0703  239.  240.  242.  1.40  0.296
#> 3 gamma  0.440 0.117  0.0554  238.  238.  240.  0     0.595
```

The `ssd_gof()` function returns three test statistics that can be used to evaluate the fit of the various distributions to the data.

- Anderson-Darling (`ad`) statistic,

- Kolmogorov-Smirnov (`ks`) statistic and
- Cramer-von Mises (`cvm`) statistic

and three information criteria

- Akaike's Information Criterion (`AIC`),
- Akaike's Information Criterion corrected for sample size (`AICc`) and
- Bayesian Information Criterion (`BIC`)

Note if `ssd_gof()` is called with `pvalue = TRUE` then the p-values rather than the statistics are returned for the ad, ks and cvm tests.

Following Burnham and Anderson (2002) we recommend the `AICc` for model selection. The best predictive model is that with the lowest `AICc` (indicated by the model with a `delta` value of 0.000 in the goodness of fit table). In the current example the best predictive model is the gamma distribution but the lnorm distribution has some support.

For further information on the advantages of an information theoretic approach in the context of selecting SSDs the reader is referred to Fox et al. (2021).

## Averaging Multiple Distributions

Often other distributions will fit the data almost as well as the best distribution as evidenced by `delta` values $< 2$ (Burnham and Anderson 2002). In this situation the recommended approach is to estimate the average fit based on the relative weights of the distributions (Burnham and Anderson 2002). The `AICc` based weights are indicated by the `weight` column in the goodness of fit table. In the current example, the gamma and log-normal distributions have `delta` values $< 2$. A detailed introduction to model averaging can be found in the Model averaging vignette. A discussion on the recommended set of default distributions can be found in the Distributions vignette.

## Estimating the Fit

The `predict` function can be used to generate model-averaged (or if `average = FALSE` individual) estimates by parametric bootstrapping. Model averaging is based on `AICc` unless the data censored is which case `AICc` in undefined. In this situation model averaging is only possible if the distributions have the same number of parameters. Parametric bootstrapping is computationally intensive. To bootstrap for each distribution in parallel register the future back-end and then select the evaluation strategy.

```
doFuture::registerDoFuture()
future::plan(future::multisession)

set.seed(99)
boron_pred <- predict(fits, ci = TRUE)
```

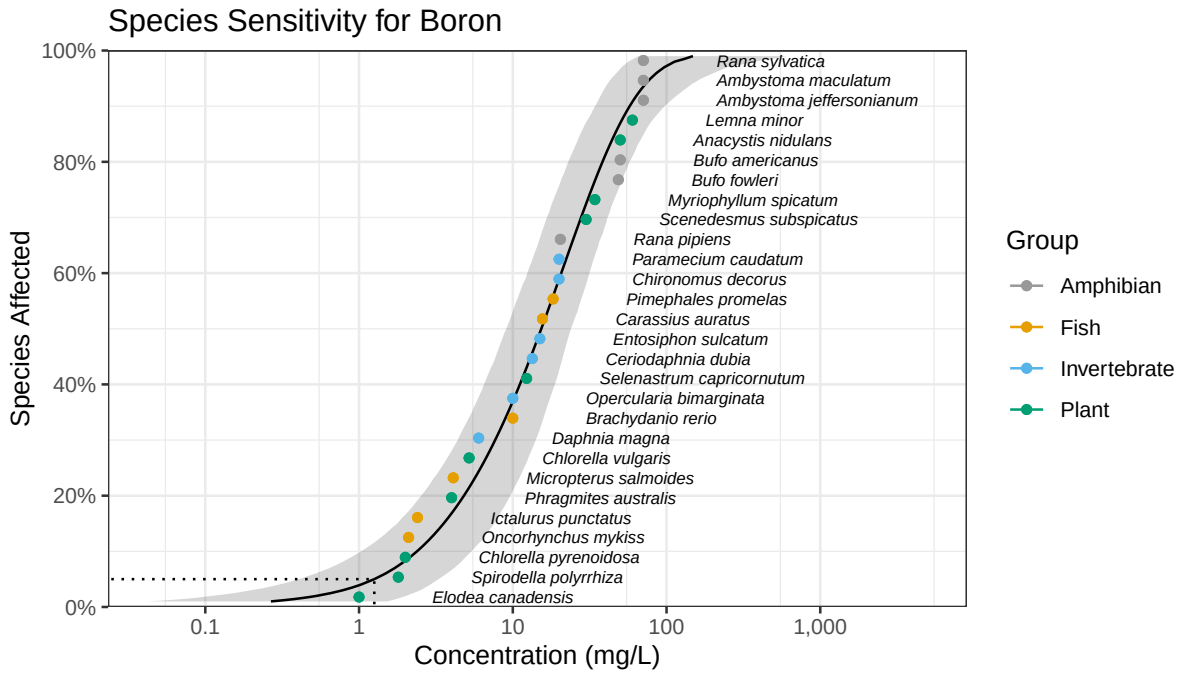
The resultant object is a data frame of the estimated concentration (`est`) with standard error (`se`) and lower (`lcl`) and upper (`ucl`) 95% confidence limits (CLs) by percent of species affected (`percent`). The object includes the number of bootstraps (`nboot`) data sets generated as well as the proportion of the data sets that successfully fitted (`pboot`). There is no requirement for the bootstrap samples to converge.

```
boron_pred
#> # A tibble: 99 x 11
#>    dist    proportion   est    se    lcl   ucl    wt method  nboot pboot samples
#>    <chr>        <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <chr>   <dbl> <dbl> <I<lis>
#>  1 average       0.01 0.267 0.401 0.0418  1.53     1 parame~  1000 0.999 <dbl>
#>  2 average       0.02 0.531 0.517 0.110   2.03     1 parame~  1000 0.999 <dbl>
#>  3 average       0.03 0.783 0.614 0.198   2.50     1 parame~  1000 0.999 <dbl>
```

```
#>  4 average       0.04 1.02  0.700 0.300   2.90     1 parame~  1000 0.999 <dbl>
#>  5 average       0.05 1.26  0.781 0.407   3.29     1 parame~  1000 0.999 <dbl>
#>  6 average       0.06 1.48  0.859 0.520   3.72     1 parame~  1000 0.999 <dbl>
#>  7 average       0.07 1.71  0.933 0.645   4.16     1 parame~  1000 0.999 <dbl>
#>  8 average       0.08 1.93  1.01  0.768   4.58     1 parame~  1000 0.999 <dbl>
#>  9 average       0.09 2.16  1.08  0.896   4.95     1 parame~  1000 0.999 <dbl>
#> 10 average       0.1  2.38  1.15  1.03    5.39     1 parame~  1000 0.999 <dbl>
#> # i 89 more rows
```

The data frame of the estimates can then be plotted together with the original data using the `ssd_plot()` function to summarize an analysis. Once again the returned object is a `ggplot` object which can be customized prior to plotting.

```
ssd_plot(ccme_boron, boron_pred,
  color = "Group", label = "Species",
  xlab = "Concentration (mg/L)", ribbon = TRUE
) +
  expand_limits(x = 5000) + # to ensure the species labels fit
  ggtitle("Species Sensitivity for Boron") +
  scale_colour_ssd()
```




In the above plot the model-averaged 95% confidence interval is indicated by the shaded band and the model-averaged 5%/95% Hazard/Protection Concentration ($HC5$/ $PC_{95}$) by the dotted line. Hazard/Protection concentrations are discussed below.

## Hazard/Protection Concentrations

The 5% hazard concentration ($HC5$) is the concentration that affects 5% of the species tested. This is equivalent to the 95% protection concentration which protects 95% of species ($PC_{95}$). The hazard and protection concentrations are directly interchangeable, and terminology depends simply on user preference.

The hazard/protection concentrations can be obtained using the ssd_hc function, which can be used to obtain any desired percentage value. The fitted SSD can also be used to determine the percentage of species protected at a given concentration using ssd_hp.

```
set.seed(99)
boron_hc5 <- ssd_hc(fits, proportion =  0.05, ci = TRUE)
print(boron_hc5)
#> # A tibble: 1 x 11
#>   dist    proportion   est    se   lcl   ucl    wt method    nboot pboot samples
#>   <chr>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>     <dbl> <dbl> <I<lis>
#> 1 average       0.05  1.32 0.849 0.370  3.67     1 parametr~  1000     1 <dbl>
boron_pc <- ssd_hp(fits, conc = boron_hc5$est, ci = TRUE)
print(boron_pc)
#> # A tibble: 1 x 11
#>   dist     conc   est    se   lcl   ucl    wt method     nboot pboot samples
#>   <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>      <dbl> <dbl> <I<list>>
#> 1 average  1.32     5  3.23 0.586  12.8     1 parametric  1000     1 <dbl [0]>
```

### Censored Data

Censored data is that for which only a lower and/or upper limit is known for a particular species. If the `right` argument in `ssd_fit_dists()` is different to the `left` argument then the data are considered to be censored. Let's make some example censored data.

```
example_dat <- ssddata::ccme_boron |>
  dplyr::mutate(left=Conc, right=Conc)

left_censored_example <- example_dat
left_censored_example$left[c(3,6,8)] <- NA
```

There are less goodness-of-fit statistics available for fits to censored data (currently just `AIC` and `BIC`). The `delta` values are calculated using AIC'.

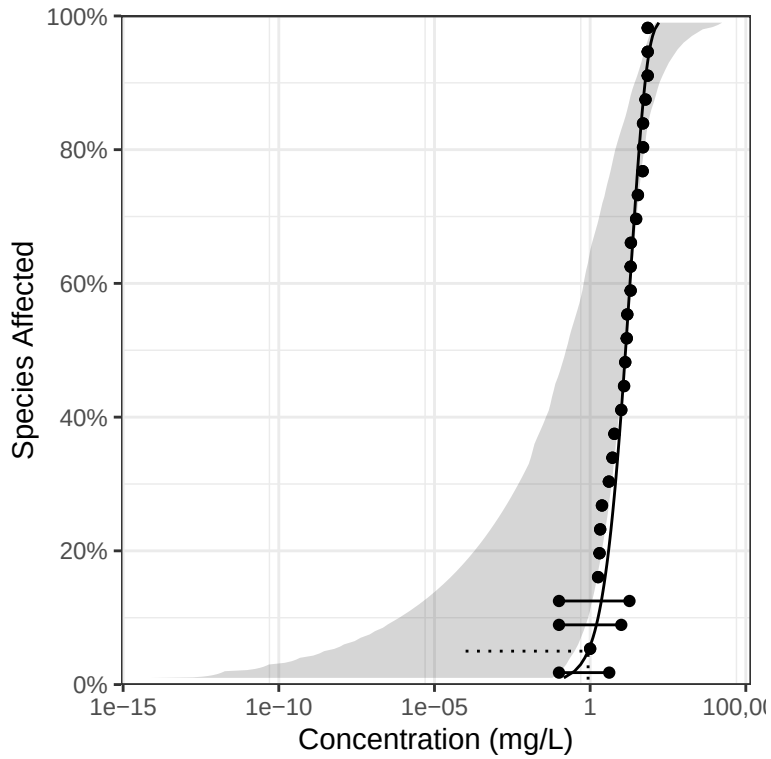
As the sample size `n` is undefined for censored data, `AICc` cannot be calculated. However, if all the models have the same number of parameters, the `AIC` `delta` values are identical to those for `AICc`. For this reason, `ssdtools` only permits the analysis of censored data using two-parameter models. We can call only the default two parameter models using `ssd_dists_bcanz(n = 2)`.

```
left_censored_dists <- ssd_fit_dists(left_censored_example,
                                     dists = ssd_dists_bcanz(n = 2),
                                     left = "left", right = "right")
ssd_hc(left_censored_dists, average = FALSE)
#> # A tibble: 5 x 11
#>   dist    proportion   est    se   lcl   ucl     wt method   nboot pboot samples
#>   <chr>        <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl> <chr>    <int> <dbl> <I<lis>
#> 1 gamma         0.05 0.674    NA    NA    NA 0.376  paramet~     0    NA <dbl>
#> 2 lgumbel       0.05 1.51     NA    NA    NA 0.0221 paramet~     0    NA <dbl>
#> 3 llogis        0.05 1.15     NA    NA    NA 0.0590 paramet~     0    NA <dbl>
#> 4 lnorm         0.05 1.32     NA    NA    NA 0.176  paramet~     0    NA <dbl>
#> 5 weibull       0.05 0.752    NA    NA    NA 0.367  paramet~     0    NA <dbl>
ssd_hc(left_censored_dists)
#> # A tibble: 1 x 11
#>   dist    proportion   est    se   lcl   ucl    wt method    nboot pboot samples
#>   <chr>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>     <int> <dbl> <I<lis>
#> 1 average       0.05 0.859    NA    NA    NA     1 parametr~     0   NaN <dbl>
ssd_gof(left_censored_dists)
#> # A tibble: 5 x 9
#>   dist       ad    ks   cvm   aic  aicc   bic delta weight
#>   <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
#> 1 gamma      NA    NA    NA  222.    NA    NA 0      0.376
#> 2 lgumbel    NA    NA    NA  228.    NA    NA 5.67   0.022
```

```
#> 3 llogis     NA    NA    NA  226.    NA    NA 3.70   0.059
#> 4 lnorm      NA    NA    NA  224.    NA    NA 1.52   0.176
#> 5 weibull    NA    NA    NA  222.    NA    NA 0.046  0.367
```

The model-averaged predictions (and hazard concentrations complete with 95% confidence limits) can be calculated using `AIC` and the results plotted complete with arrows indicating the censorship.

```
set.seed(99)
left_censored_pred <- predict(left_censored_dists, ci = TRUE)

ssd_plot(left_censored_example, left_censored_pred,
  left = "left", right = "right",
  xlab = "Concentration (mg/L)"
)
```




Note that `ssdtools` doesn't currently support right censored data:

```
right_censored_example <- example_dat
right_censored_example$right[c(3,6,8)] <- NA
right_censored_dists <- try(ssd_fit_dists(right_censored_example,
                                          dists = ssd_dists_bcanz(n = 2),
                                          left = "left", right = "right"))
#> Error in eval(expr, envir, enclos) :
#>   Distributions cannot currently be fitted to right censored data.
```

## References


Burnham KP, Anderson DR (eds) (2002) Model Selection and Multimodel Inference. Springer New York, New York, NY

Fox DR, Dam RA, Fisher R, et al (2021) Recent Developments in Species Sensitivity Distribution Modeling. Environmental Toxicology and Chemistry 40:293–308. https://doi.org/10.1002/etc.4925

Posthuma L, Suter II GW, Traas TP (2001) Species sensitivity distributions in ecotoxicology. CRC press

R Core Team (2018) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria
Wickham H (2016) ggplot2: Elegant graphics for data analysis. Springer-Verlag New York
Wickham H, Grolemund G (2016) R for data science: Import, tidy, transform, visualize, and model data, First edition. O'Reilly, Sebastopol, CA

## Licensing

# Appendix I – Model Averaging Vignette

# Model Averaged SSDs

ssdtools Team

2024-05-17

## Background

*"Many authors have noted that there is no guiding theory in ecotoxicology to justify any particular distributional form for the SSD other than that its domain be restricted to the positive real line (Newman et al. 2000), (Zajdlik 2005), (Chapman et al. 2007), (Fox 2016). Indeed, (Chapman et al. 2007) described the identification of a suitable probability model as one of the most important and difficult choices in the use of SSDs. Compounding this lack of clarity about the functional form of the SSD is the omnipresent, and equally vexatious issue of small sample size, meaning that any plausible candidate model is unlikely to be rejected (Fox et al. 2021a). The ssdtools R package uses a model averaging procedure to avoid the need to a-priori select a candidate distribution and instead uses a measure of 'fit' for each model to compute weights to be applied to an initial set of candidate distributions. The method, as applied in the SSD context is described in detail in (Fox et al. 2021a), and potentially provides a level of flexibility and parsimony that is difficult to achieve with a single SSD distribution".* (Fox et al. 2021b)

## Preliminaries

Before we jump into model averaging and in particular, SSD Model Averaging, let's backup a little and consider why we average and the advantages and disadvantages of averaging.

The pros and cons of averaging

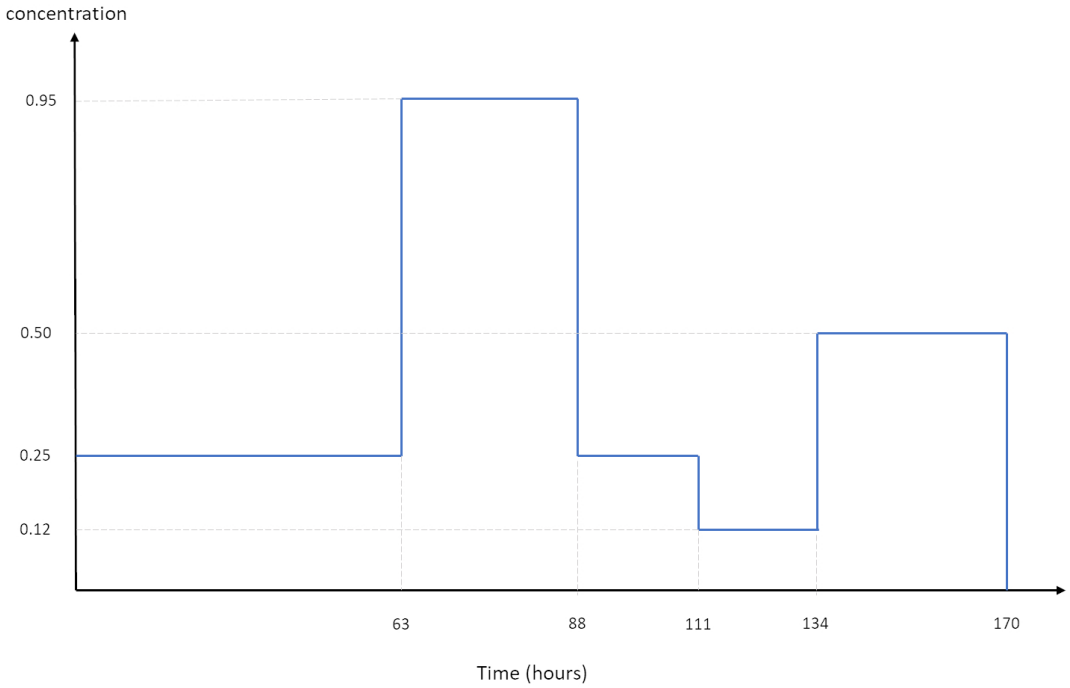
We're all familiar with the process of averaging. Indeed, *averages* are pervasive in everyday life - we talk of average income; mean sea level; average global temperature; average height, weight, age etc. etc. So what's the obsession with *averaging*? It's simple really - it's what statisticians call data reduction which is just a fancy name to describe the process of summarising a lot of *raw data* using a small number of (hopefully) representative summary statistics such as the mean and the standard deviation. Clearly, it's a lot easier to work with just a single mean than all the individual data values. That's the upside. The downside is that the process of data reduction decimates your original data - you lose information in the process. Nevertheless, the benefits tend to outweigh this information loss. Indeed, much of 'conventional' statistical theory and practice is focused on the mean. Examples include T-tests, ANOVA, regression, and clustering. When we talk of an 'average' we are usually referring to the simple, *arithmetic mean*:

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

although we recognize there are other types of mean including the geometric mean, the harmonic mean and the weighted mean. The last of these is particularly pertinent to model averaging.

### Weighted Averages

For the simple arithmetic mean, all of the individual values receive the same weighting - they each contribute $\frac{1}{n}$ to the summation. While this is appropriate in many cases, it's not useful when the components contribute to varying degrees. An example familiar to ecotoxicologists is that of a *time-varying* concentration as shown in the figure below.




From the figure we see there are 5 concentrations going from left to to right: $\{0.25, 0.95, 0.25, 0.12, 0.5\}$. If we were to take the simple arithmetic mean of these concentrations we get $\bar{X} = 0.414$. But this ignores the different *durations* of these 5 concentrations. Of the 170 hours, 63 were at concentration 0.25, 25 at concentration 0.95, 23 at concentration 0.25, 23 at concentration 0.12, and 36 at concentration 0.50. So if we were to *weight* these concentrations by time have:

$$\bar{X}_{TW} = \frac{(63 \cdot 0.25 + 25 \cdot 0.95 + 23 \cdot 0.25 + 23 \cdot 0.12 + 36 \cdot 0.50)}{(63 + 25 + 23 + 23 + 36)} = \frac{56.01}{170} = 0.33$$

So, our formula for a *weighted average* is:

$$\bar{X} = \sum_{i=1}^{n} w_i X_i$$

with $0 \leq w_i \leq 1$ and $\sum_{i=1}^{n} w_i = 1$.Note, the simple arithmetic mean is just a special case of the weighted mean with $\sum_{i=1}^{n} w_i = \frac{1}{n}$ ; $\forall i = 1, \ldots, n$

## Model Averaging

The *weighted average* acknowledges that the elements in the computation are not of equal 'importance'. In the example above, this importance was based on the *proportion of time* that the concentration was at a particular level. Bayesians are well-versed in this concept - the elicitation of *prior distributions* for model parameters provides a mechanism for weighting the degree to which the analysis is informed by existing knowledge versus using a purely data-driven approach. Model averaging is usually used in the context of estimating model parameters or quantities derived from a fitted model - for example an EC50 derived from a C-R model. Let's motivate the discussion using the following small dataset of toxicity estimates for some chemical.

```
#>  [1] 1.73 0.57 0.33 0.28 0.30 0.29 2.15 0.80 0.76 0.54 0.42 0.83 0.21 0.18 0.59
```

Now, suppose we have only two possibilities for fitting an SSD - both lognormal distributions. Model 1 is the LN(-1.067,0.414) distribution while Model 2 is the LN(-0.387,0.617) distribution. A plot of the empirical *cdf* and Models 1 and 2 is shown below.

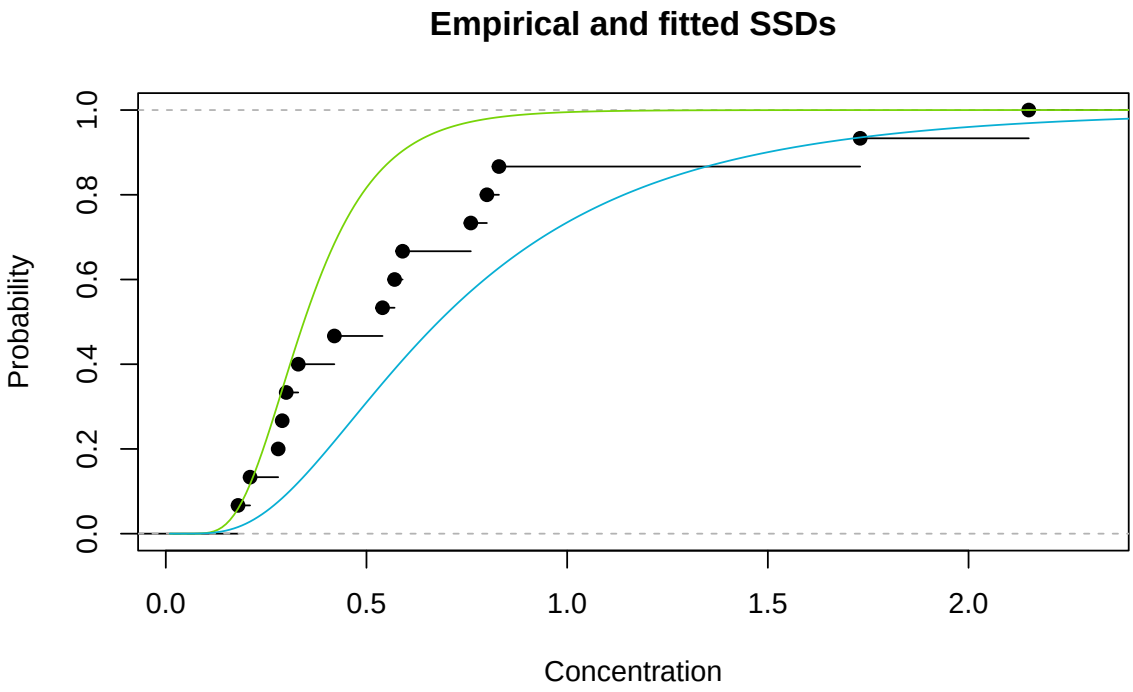



Figure 1: Emprirical cdf (black); Model 1(green); and Model 2 (blue)

We see that Model 1 fits well in the lower, left region and poorly in the upper region, while the reverse is true for Model 2. So using *either* Model 1 **or** Model 2 is going to result in a poor fit overall. However, the obvious thing to do is to **combine** both models. We could just try using 50% of Model 1 and 50% of Model 2, but that may be sub-optimal. It turns out that the best fit is obtained by using 44% of Model 1 and 56% of Model 2. Redrawing the plot and adding the *weighted average* of Models 1 and 2 is shown below.

Clearly the strategy has worked - we now have an excellent fitting SSD.What about estimation of an *HC20*? It's a simple matter to work out the *individual HC20* values for Models 1&2 using the appropriate `qlnorm()` function in `R`. Thus we have:

```
# Model 1 HC20
cat("Model 1 HC20 =",qlnorm(0.2,-1.067,0.414))
#> Model 1 HC20 = 0.2428209

# Model 2 HC20
cat("Model 2 HC20 =",qlnorm(0.2,-0.387,0.617))
#> Model 2 HC20 = 0.4040243
```

What about the averaged distribution? An intuitively appealing approach would be to apply the same weights to the individual *HC20* values as was applied to the respective models. That is `0.44*0.2428209 + 0.56*0.4040243 = 0.33`.

So our model-averaged *HC20* estimate is 0.33. As a check, we can determine the *fraction affected* at concentration = 0.33 - it should of course be 20%. Let's take a look at the plot.

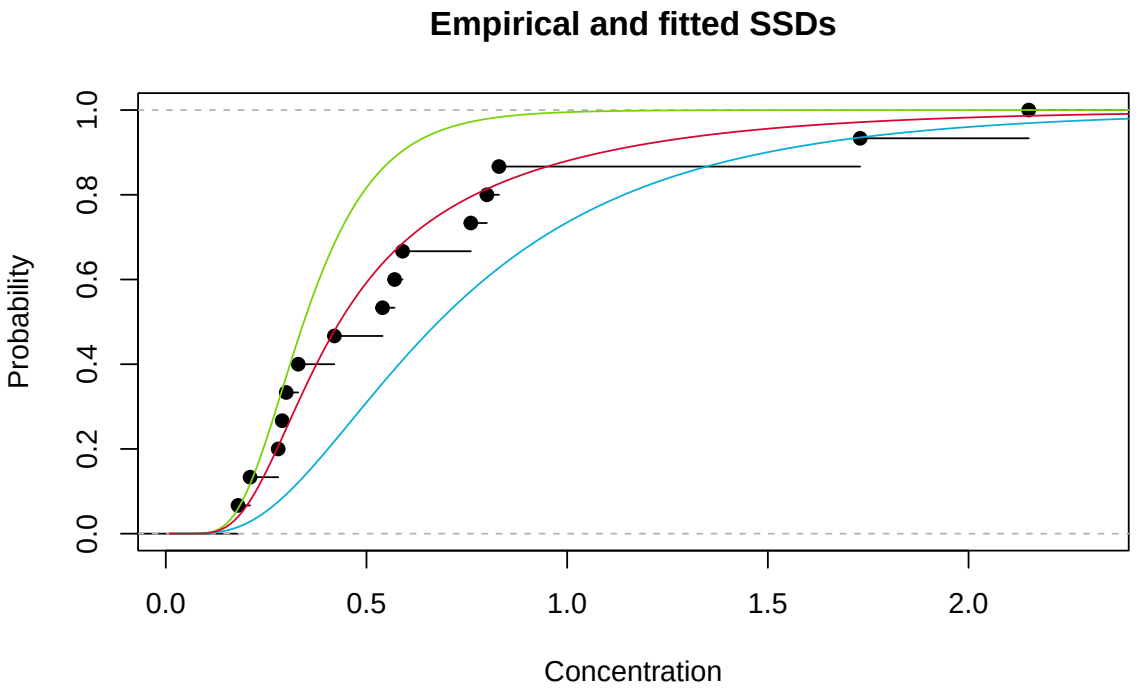
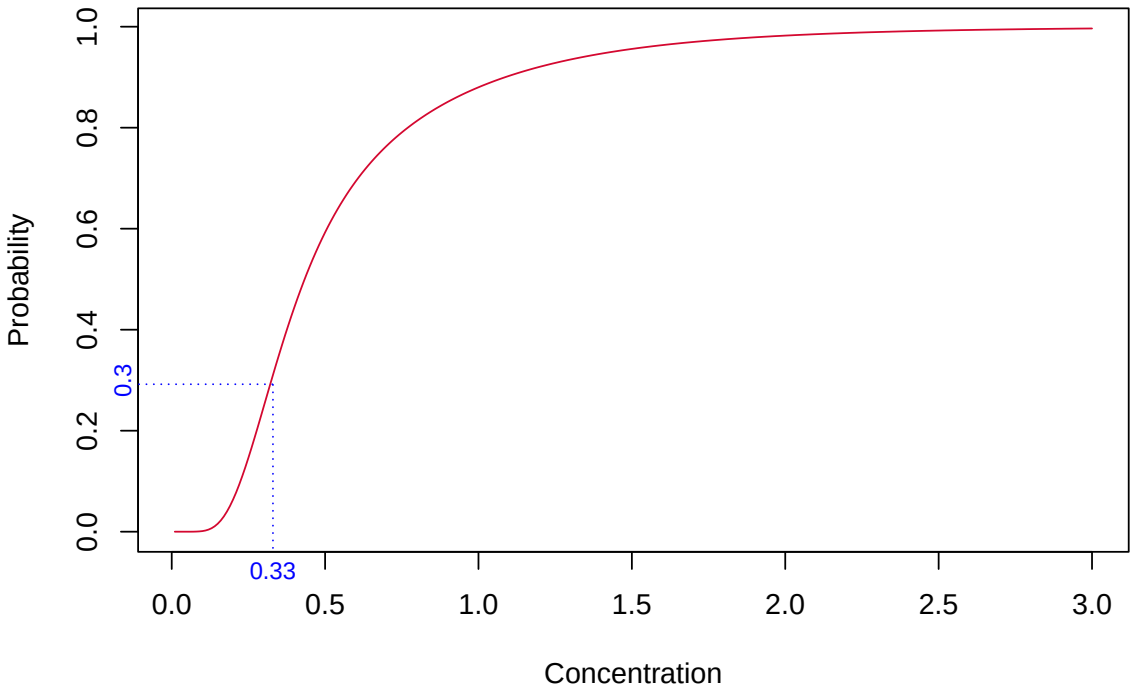



Figure 2: Empirical cdf (black); Model 1(green); Model 2 (blue); and averaged Model (red)




Something's wrong - the fraction affected at concentration 0.33 is 30% - **not the required 20%**. This issue is taken up in the next section

## Model Averaged SSDs

As we've just seen, applying the model weights to component *HCx* values and summing does **not** produce the correct result. The reason for this can be explained mathematically as follows (*if your not interested in the mathematical explanation - skip ahead to the next section*).

The fallacy of weighting individual *HCx* values

The correct expression for a model-averaged SSD is:

$$G\left(x\right)=\sum_{i=1}^{k}w_{i}F_{i}\left(x\right)$$

where $F_i\left(\cdot\right)$ is the $i^{th}$ component SSD (i.e. *cdf*) and $w_i$ is the weight assigned to $F_i\left(\cdot\right)$. Notice that the function $G\left(x\right)$ is a proper *cumulative distribution function* (*cdf*) which means for a given quantile, $x$, $G\left(x\right)$ returns the *cumulative probability*:

$$P\left[X\leqslant x\right]$$

Now, the *incorrect* approach takes a weighted sum of the component *inverse cdf's*, that is:

$$H\left(p\right)=\sum_{i=1}^{k}w_{i}{F_{i}}^{-1}\left(p\right)$$

where ${F_i}^{-1}\left(\cdot\right)$ is the $i^{th}$ *inverse cdf*. Notice that $G_i\left(\cdot\right)$ is a function of a *quantile* and returns a **probability** while $H_i\left(\cdot\right)$ is a function of a *probability* and returns an **quantile**.

Now, the correct method of determining the *HCx* is to work with the proper model-averaged *cdf* $G\left(x\right)$. This means finding the **inverse** function $G^{-1}\left(p\right)$. We'll address how we do this in a moment.

The reason why $H\left(p\right)$ does **not** return the correct result is because of the implicit assumption that the inverse of $G\left(x\right)$ is equivalent to $H\left(p\right)$. This is akin to stating the inverse of a *sum* is equal to the sum of the inverses i.e.

$$\sum_{i=1}^{n}\frac{1}{X_{i}}=\frac{1}{\sum_{i=1}^{n}X_{i}}\ ???$$

For the mathematical nerds: There are some very special cases where the above identity does in fact hold, but for that you need to use **complex numbers**.

For example, consider two complex numbers

$$\mathrm{a}=\frac{(5-i)}{2}\text{ and }b=-1.683-1.915i$$

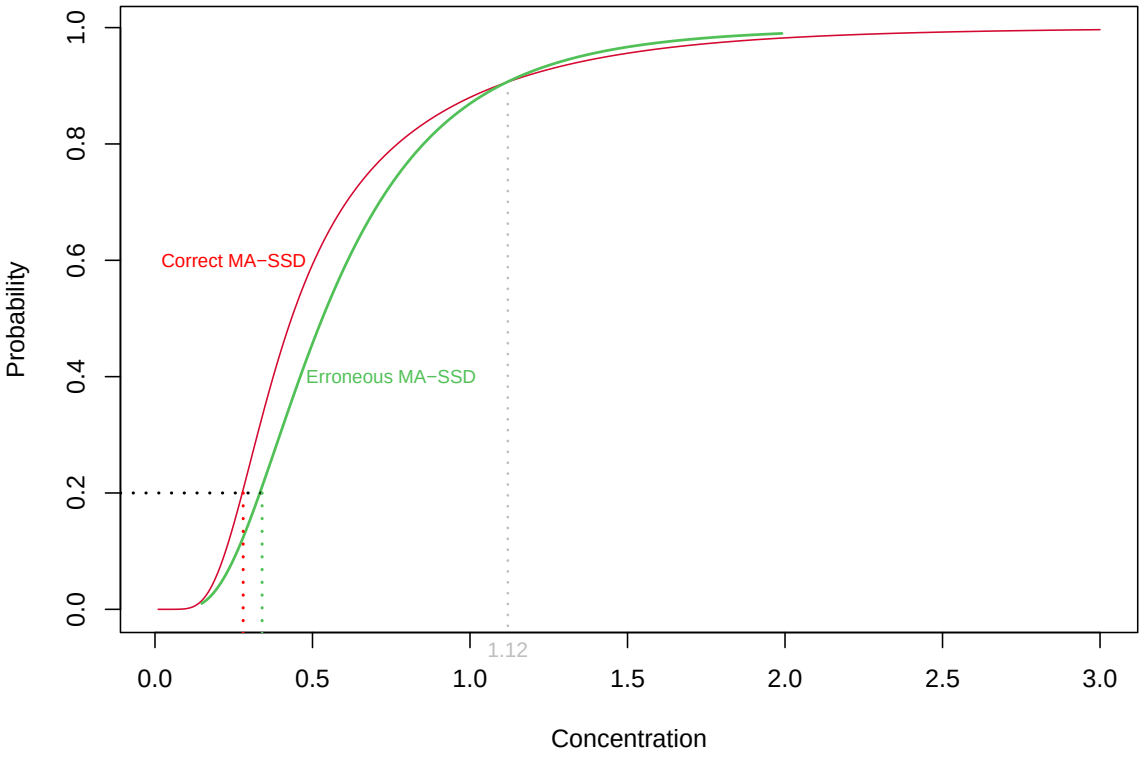
It can be shown that

$$\frac{1}{a+b}=\frac{1}{a}+\frac{1}{b}=0.126+0.372i$$

Back to the issue at hand, and since we're not dealing with complex numbers, it's safe to say:

$$G^{-1}\left(p\right)\neq H\left(p\right)$$

If you need a visual demonstration, we can plot $G\left(x\right)$ and the *inverse* of $H\left(p\right)$ both as functions of $x$ (a quantile) for our two-component lognormal distribution above.




Clearly, the two functions are **not** the same and thus *HCx* values derived from each will nearly always be different (as indicated by the positions of the vertical red and green dashed lines in the Figure above corresponding to the 2 values of the *HC20*). (Note: The two curves do cross over at a concentration of about 1.12 corresponding to the $90^{\text{th}}$ percentile, but in the region of ecotoxicological interest, there is no such cross-over and so the two approaches will **always** yield different *HCx* values with this difference $\rightarrow$ 0 as x $\rightarrow$ 0).

WE next discuss the use of a model-averaged SSD to obtain the *correct* model-averaged *HCx*.

## Computing a model-averaged *HCx*

A proper *HCx* needs to satisfy what David Fox refers to as **the inversion principle**.

More formally, the inversion principle states that an *HCx* (denoted as $\varphi_x$) must satisfy the following:

$$df\left(\varphi_x\right) = x \qquad and \qquad qf\left(x\right) = \varphi_x$$

where $df\left(\cdot\right)$ is a model-averaged *distribution function* (i.e. SSD) and $qf\left(\cdot\right)$ is a model-averaged *quantile function*. For this equality to hold, it is necessary that $qf\left(p\right) = df^{-1}\left(p\right)$.

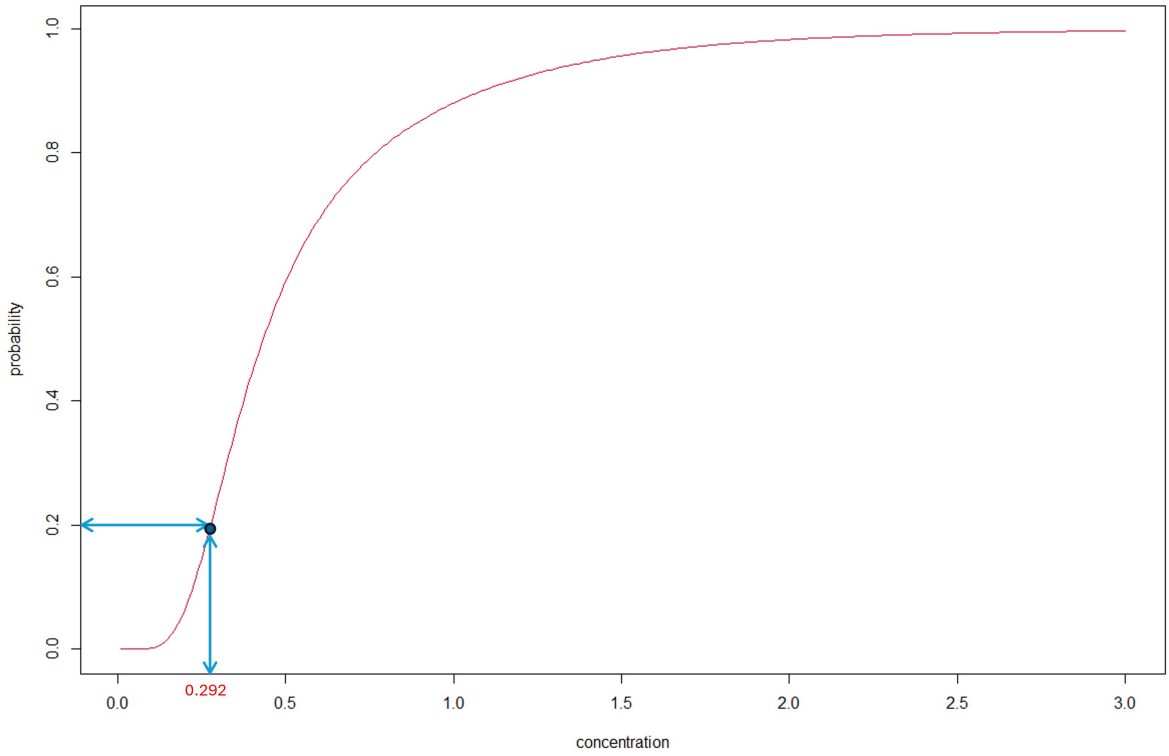
So, in our example above, the green curve was taken to be $qf\left(x\right)$ and this was used to derive $\varphi_x$ but the *fraction affected* $\{= df\left(\varphi_x\right)\}$ at $\varphi_x$ is computed using the red curve.

In `ssdtools` the following is a check that the inversion principle holds:

```
# Obtain a model-averaged HCx using the ssd_hc() function
hcp<-ssd_hc(x, p = p)
# Check that the inversion principle holds
ssd_hp(x, hcp, multi_est = TRUE) == p   # this should result in logical ‘TRUE‘
```

*Note: if the* `multi_est` *argument is set to* `FALSE` *the test will fail.*

The *inversion principle* ensures that we only use a **single** distribution function to compute both the *HCx and* the fraction affected. Referring to the figure below, the *HCx* is obtained from the MA-SSD (red curve) by following the $\rightarrow$ arrows while the fraction affected is obtained by following the $\leftarrow$ arrows.




Finally, we'll briefly discuss how the *HCx* is computed in `R` using the same method as has been implemented in `ssdtools`.

### Computing the *HCx* in `R/ssdtools`

Recall, our MA-SSD was given as

$$G(x) = \sum_{i=1}^{k} w_i F_i(x)$$

and an *HCx* is obtained from the MA-SSD by essentially working 'in reverse' by starting at a value of $x$ on the **vertical** scale in the Figure above and following the $\rightarrow$ arrows and reading off the corresponding value on the horizontal scale.

Obviously, we need to be able to 'codify' this process in `R` (or any other computer language).Mathematically this is equivalent to seeking a solution to the following equation:

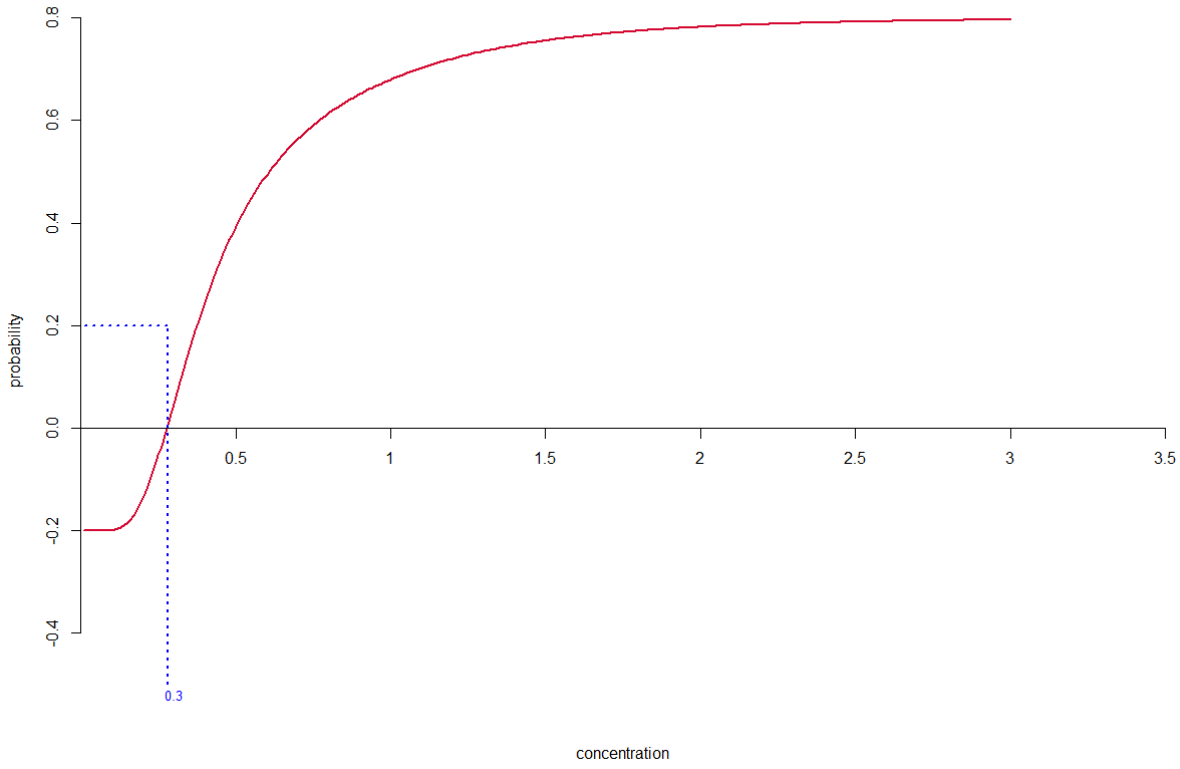
$$x : G(x) = p$$

or, equivalently:

$$x : G(x) - p = 0$$

for some fraction affected, $p$.

Finding the solution to this last equation is referred to as *finding the root(s)* of the function $G(x)$ or, as is made clear in the figure below, *finding the zero-crossing* of the function $G(x)$ for the case $p = 0.2$.




In R finding the roots of $x : G(x) - p = 0$ is achieved using the `uniroot()` function.

Help on the `uniroot` function can be found here

## Where do the model-averaged weights come from?

This is a little more complex, although we'll try to provide a non-mathematical explanation. For those interested in going deeper, a more comprehensive treatment can be found in (Burnham and Anderson 2002) and (Fletcher 2018) as well as this on-line course .

This time, we'll look at fitting a gamma, lognormal, and pareto distribution to our sample data:

```
#>  [1] 1.73 0.57 0.33 0.28 0.30 0.29 2.15 0.80 0.76 0.54 0.42 0.83 0.21 0.18 0.59
```

The adequacy (or otherwise) of a fitted model can be assessed using a variety of numerical measures known as **goodness-of-fit** or GoF statistics. These are invariably based on a measure of discrepancy between the emprical data and the hypothesized model. Common GoF statistics used to test whether the hypothesis of some specified theoretical probability distribution is plausible for a given data set include: *Kolmogorov-Smirnov test; Anderson-Darling test; Shapiro-Wilk test;and Cramer-von Mises test.* The Cramer-von Mises test is a good choice and is readily performed using the `cvm.test()` function in the `goftest` package in R as follows:

```
dat<-data.frame(Conc=c(1.73,0.57,0.33,0.28,0.3,0.29,2.15,0.8,0.76,0.54,0.42,0.83,0.21,0.18,0.59))
library(goftest)
library(EnvStats)  # this is required for the Pareto cdf (ppareto)

# Examine the fit for the gamma distribution (NB: parameters estimated from the data)
cvm.test(dat$Conc,null = "pgamma",shape = 2.0591977,scale = 0.3231032,estimated = TRUE)

# Examine the fit for the lognormal distribution (NB: parameters estimated from the data)
cvm.test(dat$Conc,null = "plnorm",meanlog=-0.6695120,sd=0.7199573,estimated = TRUE)
```

```r
# Examine the fit for the Pareto distribution (NB: parameters estimated from the data)
cvm.test(dat$Conc,null = "ppareto",location = 0.1800000,shape    = 0.9566756,estimated = TRUE)
```

```
    Cramer-von Mises test of goodness-of-fit
    Braun's adjustment using 4 groups
    Null hypothesis: Gamma distribution
    with parameters shape = 2.0591977, scale = 0.3231032
    Parameters assumed to have been estimated from data

data:  dat$Conc
omega2max = 0.34389, p-value = 0.3404


    Cramer-von Mises test of goodness-of-fit
    Braun's adjustment using 4 groups
    Null hypothesis: log-normal distribution
    with parameter meanlog = -0.669512
    Parameters assumed to have been estimated from data

data:  dat$Conc
omega2max = 0.32845, p-value = 0.3719


    Cramer-von Mises test of goodness-of-fit
    Braun's adjustment using 4 groups
    Null hypothesis: distribution 'ppareto'
    with parameters location = 0.18, shape = 0.9566756
    Parameters assumed to have been estimated from data

data:  dat$Conc
omega2max = 0.31391, p-value = 0.4015
```

From this output and using a level of significance of $p = 0.05$, we see that none of the distributions is implausible. However, if *forced* to choose just one distribution, we would choose the *Pareto* distribution (smaller values of the `omega2max` statistic are better). However, this does not mean that the gamma and lognormal distributions are of no value in describing the data. We can see from the plot below, that in fact both the gamma and lognormal distributions do a reasonable job over the range of toxicity values. The use of the Pareto may be a questionable choice given it is truncated at 0.18 (which is the minimum value of our toxicity data).

As in the earlier example, we might expect to find a better fitting distribution by combining *all three distributions* using a *weighted SSD*. The issue we face now is *how do we choose the weights* to reflect the relative fits of the three distributions? Like all tests of statistical significance, a *p-value* is computed from the value of the relevant *test statistic* - in this case, the value of the `omega2max` test statistic. For this particular test, it's a case of the smaller the better. From the output above we see that the `omega2max` values are 0.344 for the gamma distribution, 0.328 for the lognormal distribution, and 0.0.314 for the Pareto distribution.

We might somewhat naively compute the relative weights as: $w_1 = \frac{0.344^{-1}}{(0.344^{-1}+0.328^{-1}+0.314^{-1})} = 0.318$ $w_2 = \frac{0.328^{-1}}{(0.344^{-1}+0.328^{-1}+0.314^{-1})} = 0.333$ and $w_3 = \frac{0.314^{-1}}{(0.344^{-1}+0.328^{-1}+0.314^{-1})} = 0.349$ (we use *reciprocals* since smaller values of `omega2max` represent better fits). As will be seen shortly - these are incorrect.

However, being based on a simplistic measure of discrepancy between the *observed* and *hypothesized* distributions, the `omega2max` statistic is a fairly 'blunt instrument' and has no grounding in information theory which *is* the basis for determining the weights that we seek.

A discussion of *information theoretic* methods for assessing goodness-of-fit is beyond the scope of this vignette. Interested readers should consult (Burnham and Anderson 2002) or the on-line course. A

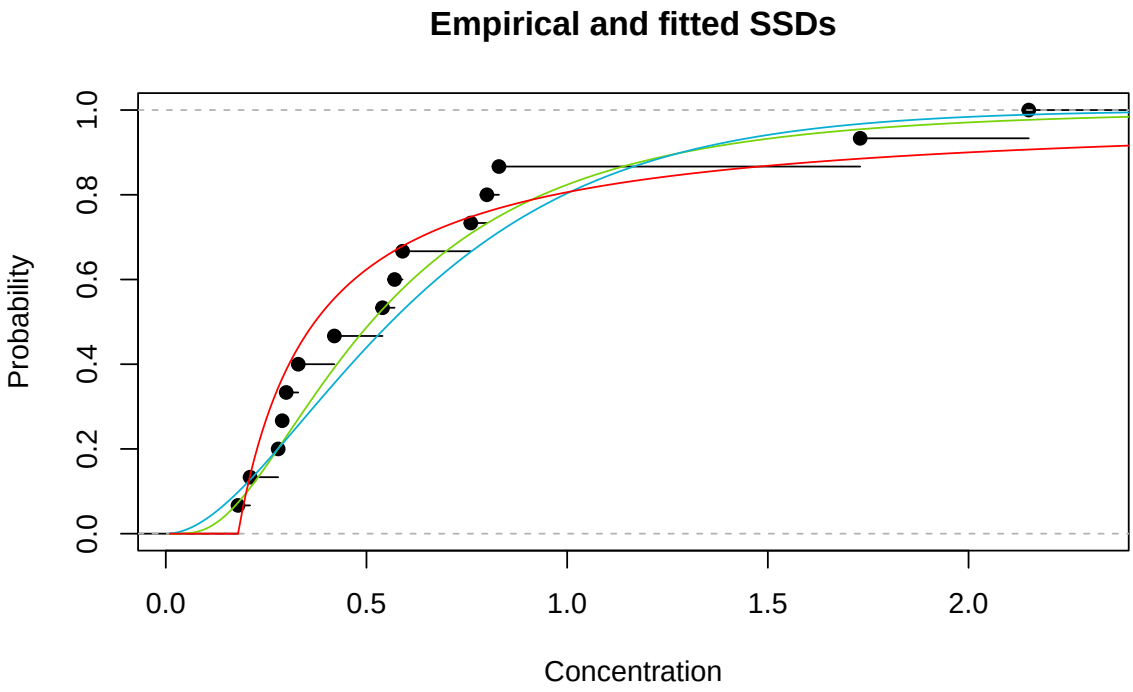



Figure 3: Emprirical cdf (black); lognormal (green); gamma (blue); and Pareeto (red)

commonly used metric to determine the model-average weights is the **Akaike Information Criterion** or AIC. The formula for the $AIC$ is:

$$AIC = 2k - 2\ln(\ell)$$

where $k$ is the number of model parameters and $\ell$ is the *likelihood* for that model. Again, a full discussion of statistical likelihood is beyond the present scope. A relatively gentle introduction can be found here.

The likelihood for our three distributions can be computed in `R` as follows:

```
sum(log(EnvStats::dpareto(dat$Conc,location = 0.1800000, shape=0.9566756)))
#> [1] -5.621683
sum(log(dgamma(dat$Conc,shape = 2.0591977,scale = 0.3231032)))
#> [1] -7.020597
sum(log(dlnorm(dat$Conc, meanlog = -0.6695120,sdlog   =  0.7199573)))
#> [1] -5.812947
```

From which the $AIC$ values readily follow:

```
#> AIC for gamma distribution = 18.04119
#> AIC for lognormal distribution = 15.62589
#> AIC for Pareto distribution = 15.24337
```

As with the `omega2max` statistic, **smaller** values of $AIC$ are better. Thus, a comparison of the AIC values above gives the ranking of distributional fits (best to worst) as: *Pareto* > *lognormal* > *gamma*

### Computing model weights from the `AIC`

We will simply provide a formula for computing the model weights from the `AIC` values. More detailed information can be found here.

The $AIC$ for the $i^{th}$ distribution fitted to the data is

$$AIC_i = 2k_i - 2\ln(L_i)$$

where $L_i$ is the $i^{th}$ *likelihood* and $k_i$ is the *number of parameters* for the $i^{th}$ *distribution*. Next, we form the differences:

$$\Delta_i = AIC_i - AIC_0$$

where $AIC_0$ is the $AIC$ for the **best-fitting** model (i.e.$AIC_0 = \min_i \{AIC_i\}$ ). The *model-averaged weights* $w_i$ are then computed as:

The model-averaged weights for the gamma, lognormal, and Pareto distributions used in the previous example can be computed 'manually' in R as follows:

```
dat<-c(1.73,0.57,0.33,0.28,0.3,0.29,2.15,0.8,0.76,0.54,0.42,0.83,0.21,0.18,0.59)
aic<-NULL
k<-2  # number of parameters for each of the distributions


aic[1]<-2*k-2*sum(log(dgamma(dat,shape = 2.0591977,scale = 0.3231032))) # Gamma distribution

aic[2]<-2*k-2*sum(log(dlnorm(dat, meanlog = -0.6695120,sdlog   =  0.7199573)))  # lognormal distribu

aic[3]<-2*k-2*sum(log(EnvStats::dpareto(dat,location = 0.1800000, shape=0.9566756))) # Pareto distri

delta<-aic-min(aic)  #  compute the delta values

aic.w<-exp(-0.5*delta); aic.w<-round(aic.w/sum(aic.w),4)

cat(" AIC weight for gamma distribution =",aic.w[1],"\n",
    "AIC weight for lognormal distribution =",aic.w[2],"\n",
    "AIC weight for pareto distribution =",aic.w[3],"\n")
```

```
 AIC weight for gamma distribution = 0.1191
 AIC weight for lognormal distribution = 0.3985
 AIC weight for pareto distribution = 0.4824
```

Finally, let's look at the fitted *model-averaged SSD*:

As can be seen from the figure above, the model-averaged fit provides a very good fit to the empirical data.

### Correcting for distributions having differing numbers of parameters

In deriving the AIC, Akaike had to make certain, strong assumptions. In addition, the bias factor (the $2k$ term) was derived from theoretical considerations (such as mathematical *expectation*) that relate to *infinite* sample sizes. For small sample sizes, the AIC is likely to select models having too many parameters (i.e models which *over-fit*) In 1978, Sugiura proposed a modification to the AIC to address this problem, although it too relied on a number of assumptions. This 'correction' to the AIC for small samples (referred to as $AIC_c$) is:

It is clear from the formula for $AIC_c$ that for $n \gg k$, $AIC_c \simeq AIC$. The issue of sample size is ubiquitous in statistics, but even more so in ecotoxicology where logistical and practical limitations invariably mean we are dealing with (pathologically) small sample sizes. There are no hard and fast rules as to what constitutes an *appropriate* sample size for SSD modelling. However, Professor David Fox's personal rule of thumb which works quite well is:

Since most of the common SSD models are 2-parameter, we should be aiming to have a sample size of at least 11. For 3-parameter models (like the Burr III), the minimum sample size is 16 and if we wanted to fit a mixture of two, 2-parameter models (eg. *logNormal-logNormal* or *logLogistic-logLogistic*) the sample size should be *at least* 26. Sadly, this is rarely the case in practice!

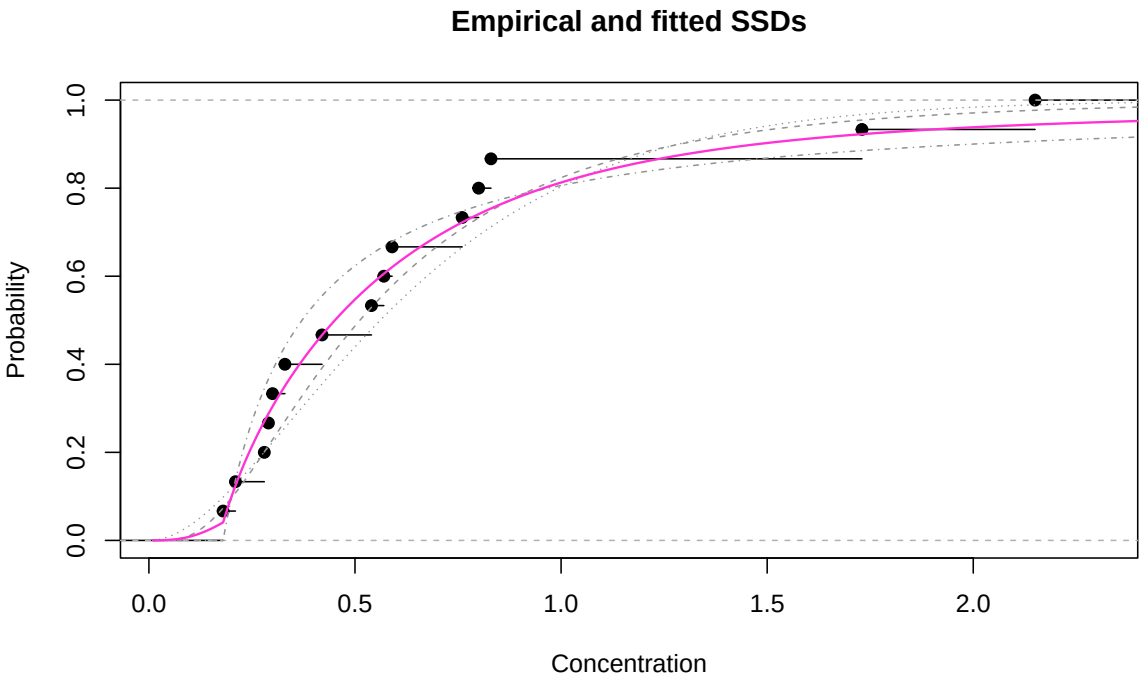



Figure 4: Empirical cdf (black) and model-averaged fit (magenta)

### Model-Averaging in `ssdtools`

Please see the Getting started with ssdtools vignette for examples of obtaining model-averaged *HCx* values and predictions using `ssdtools`.

## References


Burnham K, Anderson D (2002) Model selection and multimodel inference - a practical information-theoretic approach. Springer

Chapman PR, Hart A, Roelofs W, et al (2007) Methods of uncertainty analysis. In: A H (ed) EUFRAM Concerted Action to Develop a European Framework for Probabilistic Risk Assessment of the Environmental Impacts of Pesticides, Vol 2, Detailed Reports on Role, Methods, Reporting and Validation

Fletcher D (2018) Model averaging. Springer

Fox DR (2016) Contemporary methods for statistical design and analysis. In: Blasco J, Chapman PM, Campana O, Hampel M (eds) Marine Ecotoxicology

Fox DR, Dam RA, Fisher R, et al (2021a) Recent Developments in Species Sensitivity Distribution Modeling. Environmental Toxicology and Chemistry 40:293–308. https://doi.org/10.1002/etc.4925

Fox DR, Fisher R, Thorley JL, Schwarz C (2021b) Joint investigation into statistical methodologies underpinning the derivation of toxicant guideline values in Australia and New Zealand. Environmetrics Australia; Australian Institute of Marine Science

Newman MC, Ownby DR, Mézin LCA, et al (2000) Applying species-sensitivity distributions in ecological risk assessment: Assumptions of distribution type and sufficient numbers of species. Environmental Toxicology and Chemistry 19:508–515. https://doi.org/10.1002/etc.5620190233

Zajdlik B (2005) Statistical analysis of the SSD approach for development of canadian water quality guidelines. Zajdlik; Associates

## Licensing

# Appendix J – Distributions in ssdtools Vignette

# Distributions in ssdtools

ssdtools Team

2024-05-17

## Distributions for SSD modelling

Many authors have noted that there is no guiding theory in ecotoxicology to justify any particular distributional form for the SSD other than that its domain be restricted to the positive real line (Newman et al. 2000; Zajdlik 2005; Fox 2016).

Distributions selected to use in model averaging of SSDs must be bounded by zero given that effect concentrations cannot be negative. They must also be continuous, and generally unbounded on the right.

Furthermore, the selected distributions within the candidate model set should provide a variety of shapes to capture the diversity of shapes in empirical species sensitivity distributions.

There is a wide range of distributions that have been implemented in `ssdtools`, although not all distributions appear in the default set. Here we provide a detailed account of the distributions available in `ssdtools`, and guidance on their use.

### Original `ssdtools` distributions

The log-normal, log-logistic and Gamma distributions have been widely used in SSD modelling, and were part of the original distribution set for early releases of `ssdtools` as developed by Thorley and Schwarz 2018. They were adopted as the default set of three distributions in early updates of `ssdtools` and the associated ShinyApp (Dalgarno 2021). All three distributions show good convergence properties and are retained as part of the default model set in version 2.0 of `ssdtools`.

In addition to the log-normal, log-logistic and Gamma distributions, the original version of `ssdtools` as developed by Thorley and Schwarz 2018 also included three additional distributions in the candidate model set, including the log-gumbel, Gompertz and Weibull distributions. Of these, the log-Gumbel (otherwise known as the inverse Weibull, see below) shows relatively good convergence (see Figure 32, Fox et al. 2021b), and is also one of the limiting distributions of the Burrr Type 3 distribution implemented in `ssdtools`, and has been retained in the default model set. The Gompertz and Weibull distributions, however can exhibit unstable behaviour, sometimes showing poor convergence, and therefore been excluded from the default set (see Figure 32, Fox et al. 2021b)

### Burr III distribution

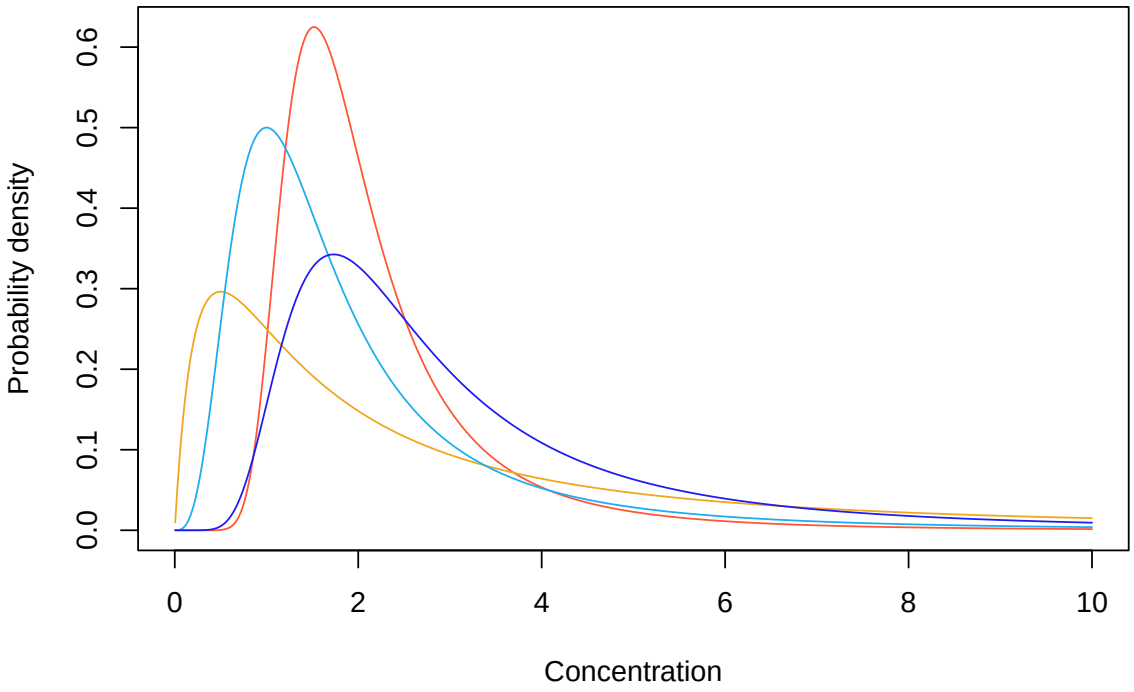
A history of Burrlioz and the primary distributions it used were recently summmarized by Fox et al. (2021a).

*"In 2000, Australia and New Zealand (Australian and New Zealand Environment and Conservation Council/Agriculture and Resource Management Council of Australia and New Zealand 2000) adopted an SSD-based method for deriving WQBs, following a critical review of multiple WQB derivation methods (Warne 1998). A distinct feature of the method was the use of a 3-parameter Burr distribution to model the empirical SSD, which was implemented in the Burrlioz software tool (Campbell et al. 2000). This represented a generalization of the methods previously employed by Aldenberg and Slob (1993) because the log–logistic distribution was shown to be a specific case of the Burr family (Tadikamalla 1980). Recent revision of the derivation method recognized that using the 3-parameter Burr distributions for small*

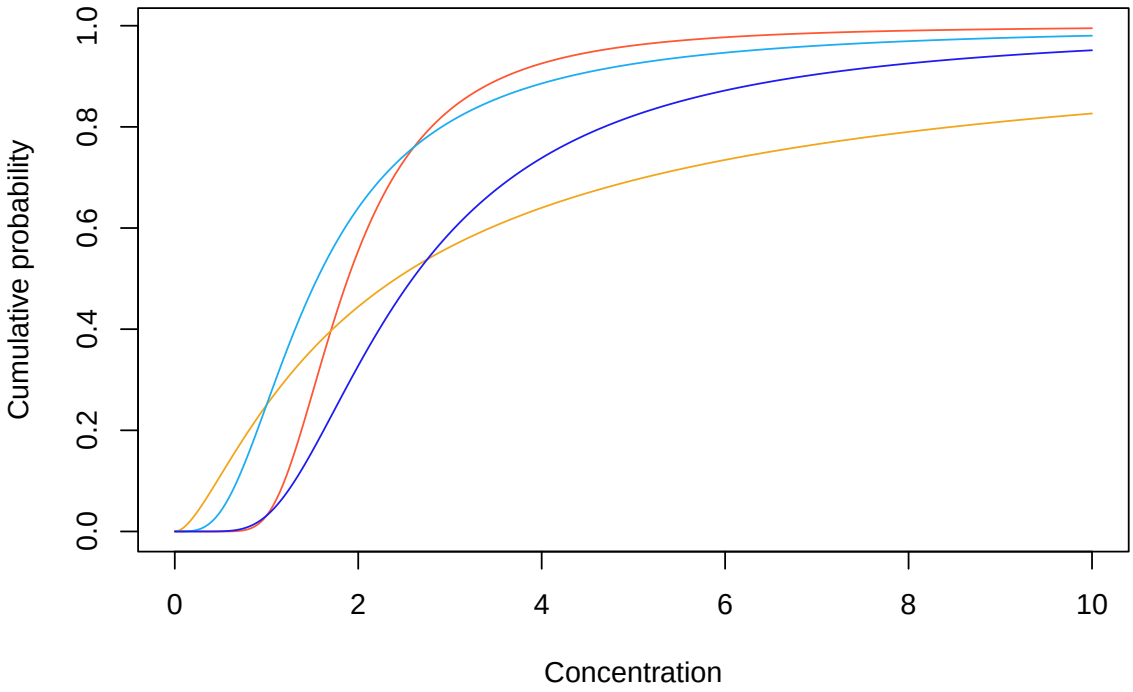
*sample sizes (<8 species) created additional uncertainty by estimating more parameters than could be justified, essentially overfitting the data (Batley et al. 2018). Consequently, the method, and the updated software (Burrlioz Ver 2.0), now uses a 2-parameter log–logistic distribution for these small data sets, whereas the Burr type III distribution is used for data sets of 8 species or more (Batley et al. 2018; Australian and New Zealand Guidelines 2018).”* (Fox et al. 2021a)

The probability density function, $f_X(x; b, c, k)$ and cumulative distribution function, $F_X(x; b, c, k)$ for the Burr III distribution are:

Sample Burr probability density functions




Sample Burr cumulative distribution functions




While the Burr type III distribution was adopted as the default distribution in Burrlioz, it is well known (e.g., Tadikamalla (1980)) that the Burr III distribution is related to several other theoretical distributions, some of which only exist as limiting cases of the Burr III, i.e., as one or more of the Burr III parameters approaches either zero or infinity. The Burrlioz software incorporates logic that aims to identify situations where parameter estimates are tending towards either very large or very small values. In such cases, fitting a Burr III distribution is abandoned and one of the limiting distributions is fitted instead.

Specifically:

- As c tends to infinity the Burr III distribution tends to the inverse (North American) Pareto distribution (see technical details)
- As k tends to infinity the Burr III distribution tends to the inverse Weibull (log-Gumbel) distribution (see technical details)

In practical terms, if the Burr III distribution is fitted and k is estimated to be greater than 100, the estimation procedure is carried out again using an inverse Weibull distribution. Similarly, if c is greater than 80 an (American) Pareto distribution is fitted. This is necessary to ensure numerical stability.

Since the Burr type III, inverse Pareto and inverse Weibull (log Gumbel) distributions are used by the Burrlioz software, these have been implemented in `ssdtools`. However, we have found there are stability issues with both the Burr type III, as well as the inverse Pareto distributions, which currently precludes their inclusion in the default model set (see Fox et al. (2021b), and below for more details).

### Bimodal distributions

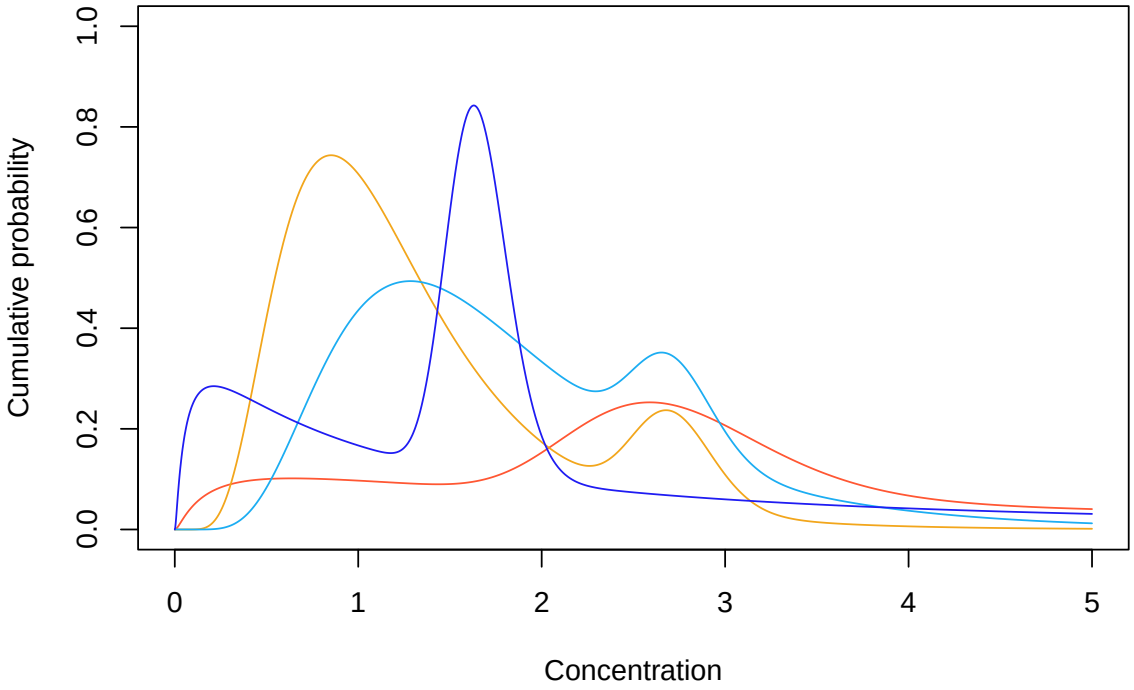
The use of statistical mixture-models was promoted by Fox as a convenient and more realistic way of modelling bimodal toxicity data (Fisher et al. 2019). Although parameter heavy, statistical mixture models provide a better conceptual match to the inherent underlying data generating process since

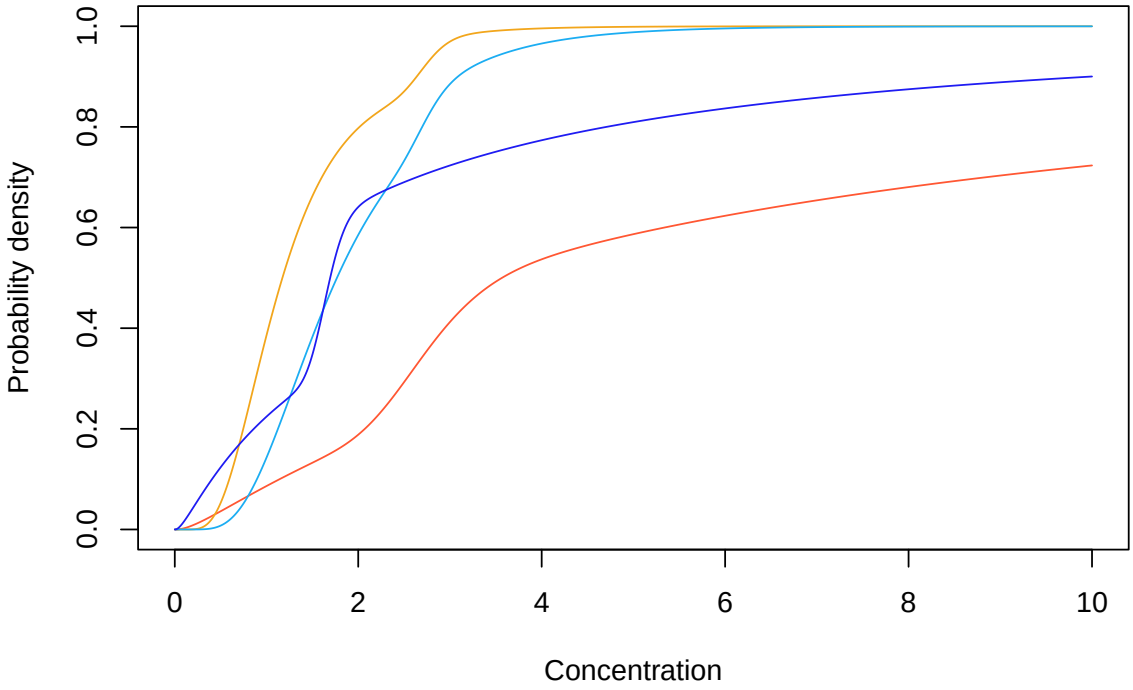
they directly model bimodality as a mixture of 2 underlying univariate distributions that represent, for example, different modes of action (Fox et al. 2021a). It has been postulated that a mixture-model would only be selected in a model-averaging context when the fit afforded by the mixture is demonstrably better than the fit afforded by any single distribution. This is a consequence of the high penalty in AICc associated with the increased number of parameters (p in Equation 7 of (Fox et al. 2021a)) and will be most pronounced for relatively small sample sizes.

The TMB version of `ssdtools` now includes the option of fitting two mixture distributions, individually or as part of a model average set. These can be fitted using `ssdtools` by supplying the strings "llogis_llogis" and/or "lnorm_lnorm" to the *dists* argument in the *ssd_fit_dists* call.

The underlying code for these mixtures has three components: the likelihood function required for TMB; exported R functions to allow the usual methods for a distribution to be called (p, q and r); and a set of supporting R functions (see Fox et al. (2021b) Appendix D for more details). Both mixtures have five parameters - two parameters for each of the component distributions and a mixing parameter (pmix) that defines the weighting of the two distributions in the 'mixture.'

Sample lognormal mixture distributions






AS can be see from the plot above, the mixture distributions provide a highly flexible means of modelling *bimodality* in an emprical SSD. This happens, for example, when the toxicity data for some toxicant include both animal and plant species, or there are different modes of action operating. Unfortunately, this increased flexibilty comes with a high penalty in the model-averaging process. The combination of small sample sizes and a high parameter count (typically 5 or more) means that mixture distributions will be down-weighted - even when they do a good job at describing the data. For this reason, when attempting to model bimodal data, we suggest looking at the fit using the default set of distributions and then examining the fit with just one of either the log-normal mixture or the log-logistic mixture. Keep in mind that this should only be done if the sample size is not pathologically small. As a guide, Prof. David Fox recommends as an *absolute minimum* $n \geq 3k + 1$ but preferably $n \geq 5k + 1$ where $k$ is the number of model parameters.

## Default Distributions

While there is a variety of distributions available in `ssdtools`, the inclusion of all of them for estimating a model-averaged SSD is not recommended.

By default, `ssdtools` uses the (corrected) Akaike Information Criterion for small sample size (AICc) as a measure of relative quality of fit for different distributions and as the basis for calculating the model-averaged weights. However, the choice of distributions used to fit a model-averaged SSD can have a profound effect on the estimated *HCx* values.

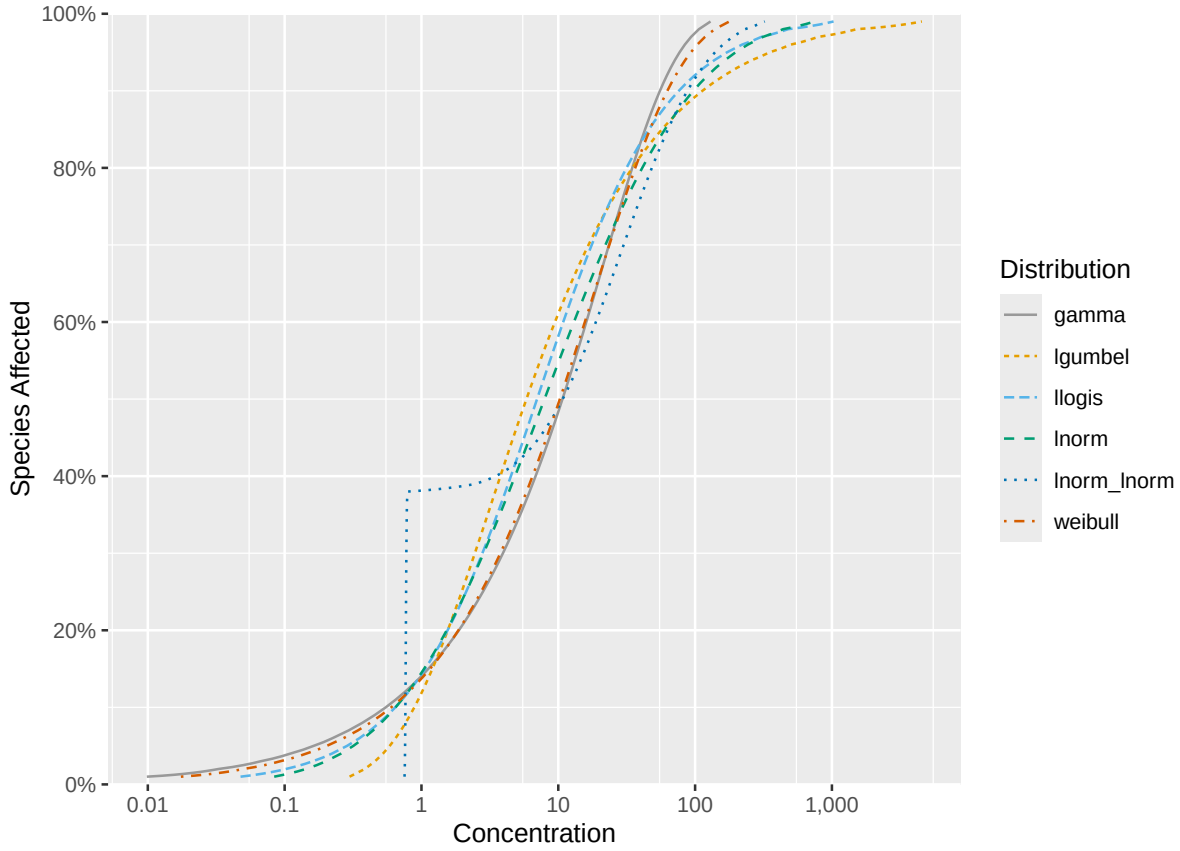
Deciding on a final default set of distributions to adopt using the model averaging approach is non-trivial, and we acknowledge that there is probably no definitive 'solution' to this issue. However, the default set should be underpinned by a guiding principle of parsimony, i.e., the set should be as large as is necessary to cover a wide variety of distributional shapes and contingencies but no bigger. Further, the default set should result in model-averaged estimates of *HCx* values that: 1) minimise bias; 2) have actual coverages of confidence intervals that are close to the nominal level of confidence; 3) estimated *HCx* and confidence intervals of *HCx* are robust to small changes in the data; and 4) represent a positively continuous distribution that has both right and left tails.

The `ssdtools` development team has undertaken extensive simulation studies, as well as some detailed technical examinations of the various candidate distributions to examine issues of bias, coverage and numerical stability. A detailed account of our findings can be found in our report (Fox et al. 2021b) and are not repeated in detail here, although some of the issues associated with individual distributions are outlined below.

### Currently recommended default distributions

The default list of candidate distributions in `ssdtools` is comprised of the following: log-normal; log-logistic; gamma; inverse Weibull (log-Gumbel); Weibull; mixture of two log-normal distributions

The default distributions are plotted below with a mean of 2 and standard deviation of 2 on the (natural) log concentration scale or around 7.4 on the concentration scale.




## Distributions currently implemented in `ssdtools`

**Burr Type III distribution** The Burr Type 3 is a flexible three parameter distribution can be fitted using `ssdtools` by supplying the string `burrIII3` to the `dists` argument in the `ssd_fit_dists` call.

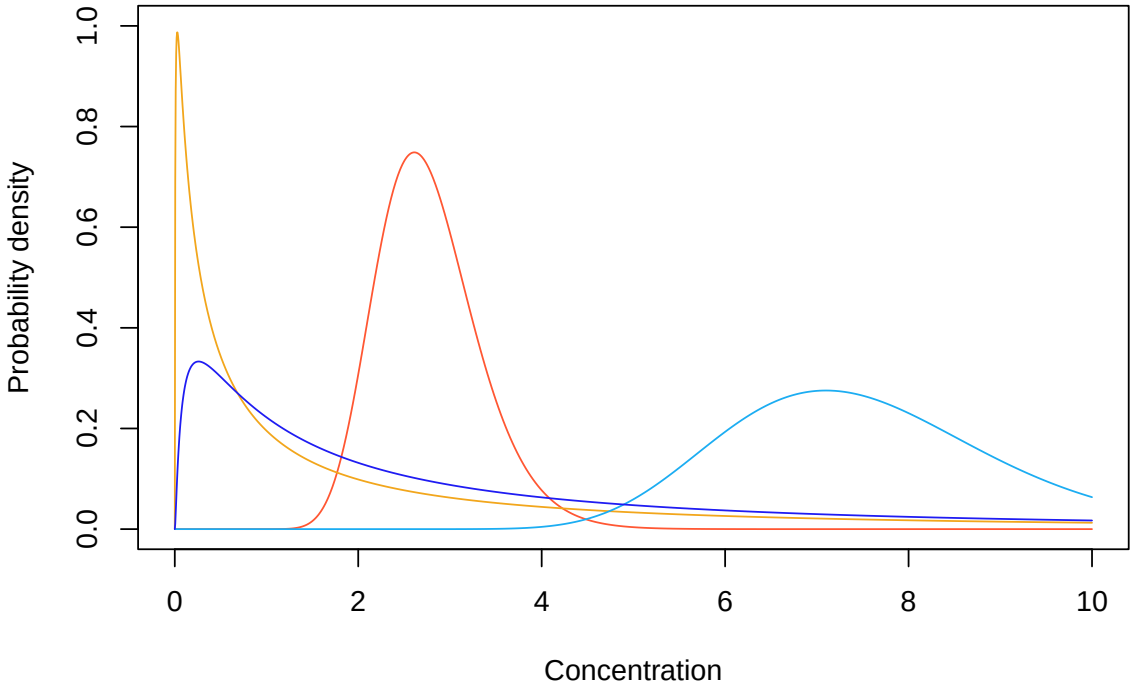
The Burr family of distributions has been central to the derivation of guideline values in Australia and New Zealand for over 20 yr (Fox et al. 2021a). While offering a high degree of flexibility, experience with these distributions during that time has repeatedly highlighted numerical stability and convergence issues when parameters are estimated using maximum likelihood (Fox et al. 2021a). This is thought to be due to the high degree of collinearity between parameter estimates and/or relatively flat likelihood profiles (Fox et al. 2021a), and is one of the motivations behind the logic coded into Burrlioz to revert to either of the two limiting distributions. Burr Type 3 distribution is not currently one of the recommended distributionsin the default model set. This is because of 1) the convergence issues associated with the Burr Type 3 distribution, 2) the fact that reverting to a limiting two parameter distribution does not fit easily within a model averaging framework, and 3) that one of the two limiting distributions (the inverse Pareto, see below) also has estimation and convergence issues.

**Log-normal** The log-normal distribution is a commonly used distribution in the natural sciences - particularly as a probability model to describe right (positive)-skewed pehnomena such as *concentration* data.

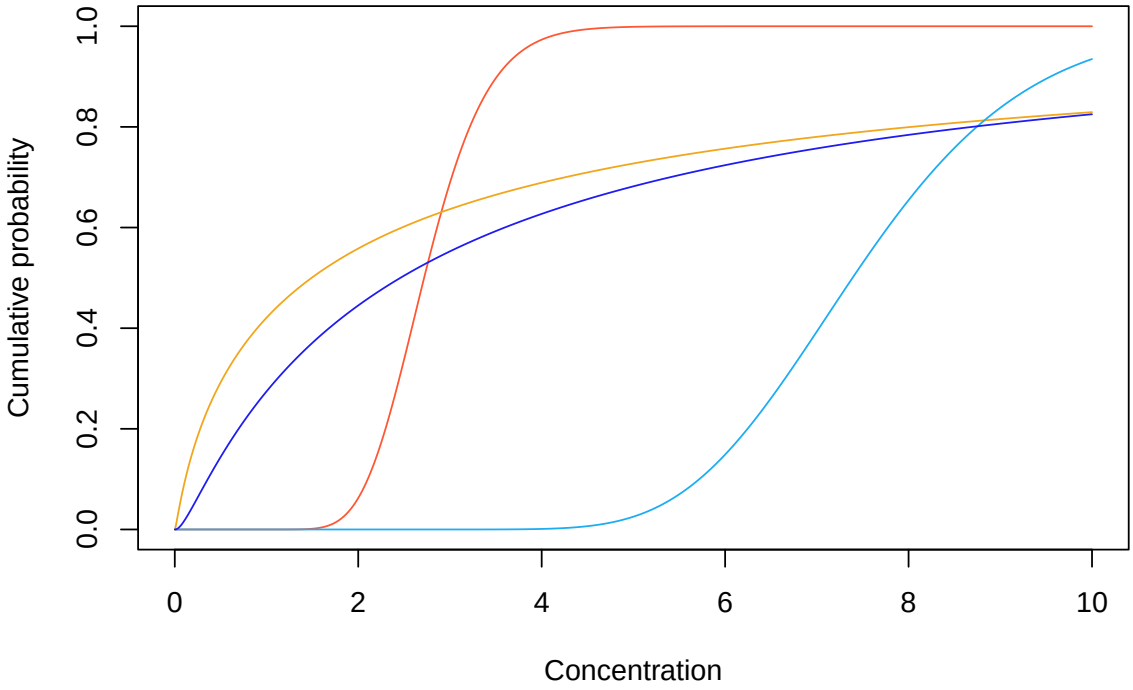
A random variable, $X$ is lognormally distributed if the *logarithm* of $X$ is normally distributed. The *pdf* of $X$ is given by

The lognormal distribution was selected as the starting distribution given the data are for effect concentrations. The log-normal distribution can be fitted using `ssdtools` supplying the string `lnorm` to the `dists` argument in the `ssd_fit_dists` call.

Sample lognormal probability density functions




Sample lognormal cumulative distribution functions

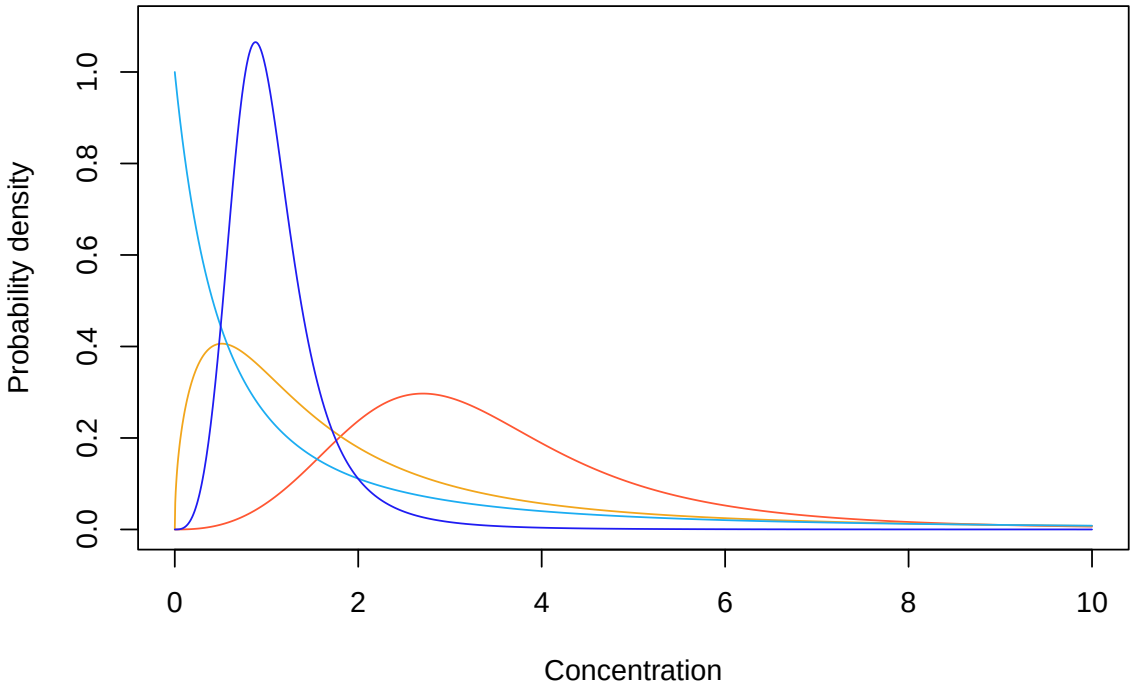



**Log-logistic distribution** Like the *lognormal* distribution, the log-logistic is similarly defined, that is: if $X$ has a log-logistic distribution, then $Y = \ln(X)$ has a *logistic* distribution.

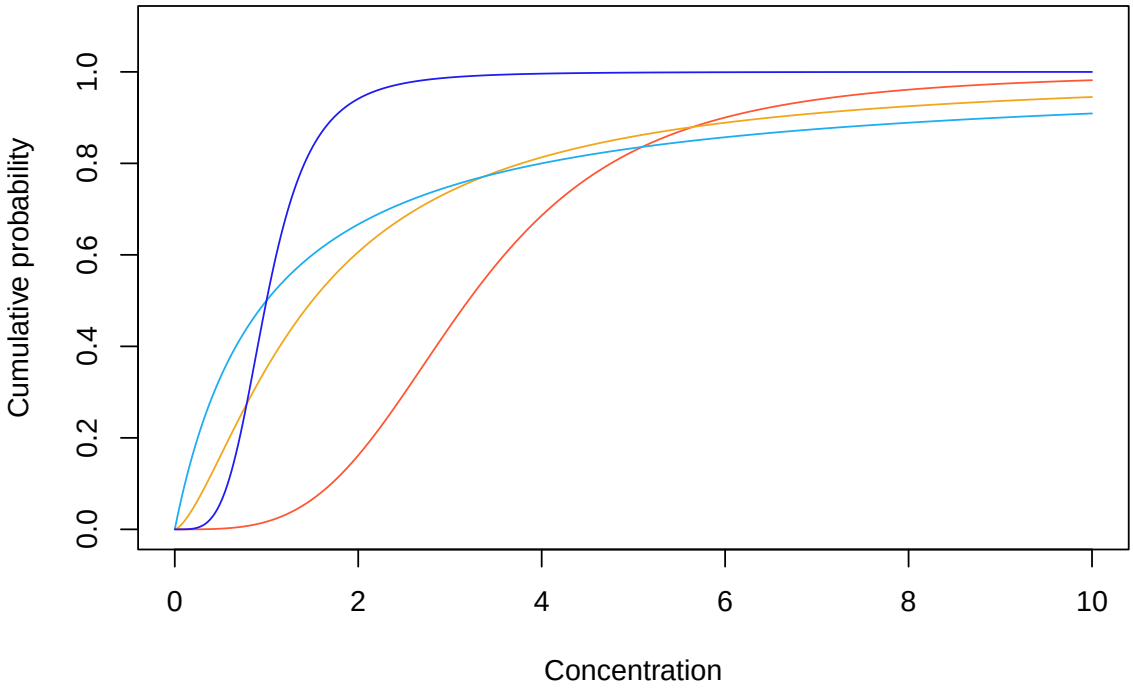
letting $\mu = \ln(\alpha)$ and $s = \frac{1}{\beta}$ we have:

The log-logistic distribution is often used as a candidate SSD primarily because of its analytic tractability (Aldenberg and Slob 1993). We included it because it has wider tails than the log-normal and because it is a specific case of the more general Burr family of distributions Burr (1942). The log-logistic distribution can be fitted using `ssdtools` by supplying the string `lnorm` to the `dists` argument in the `ssd_fit_dists` call.

Sample Log-logistic probability density functions




Sample Log-logistic cumulative distribution functions



**Gamma distribution** The two-parameter gamma distribution has the following *pdf* and *cdf*.

where $\Gamma(\cdot)$ is the *gamma* function (in `R` this is simply `gamma(x)`) and $\gamma(\cdot)$ is the (lower) *incomplete gamma* function

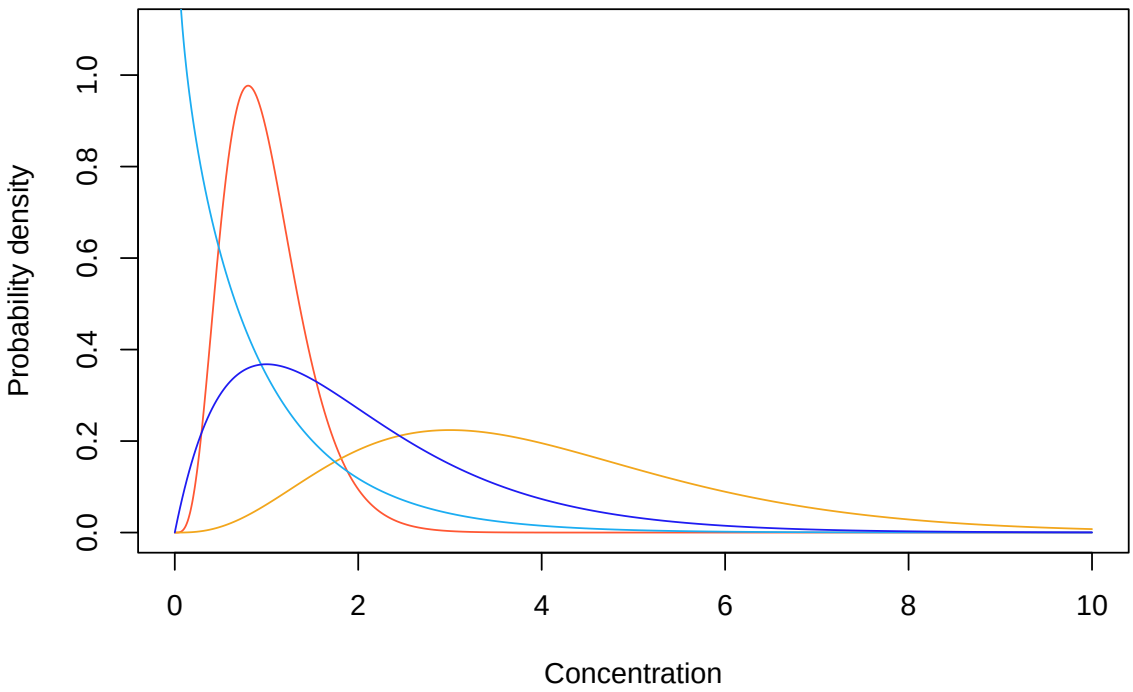
$$\gamma(x, a) = \int_0^x t^{a-1} e^{-t} dt$$

(this can be computed using the `gammainc` function from the `pracma` package in `R`).

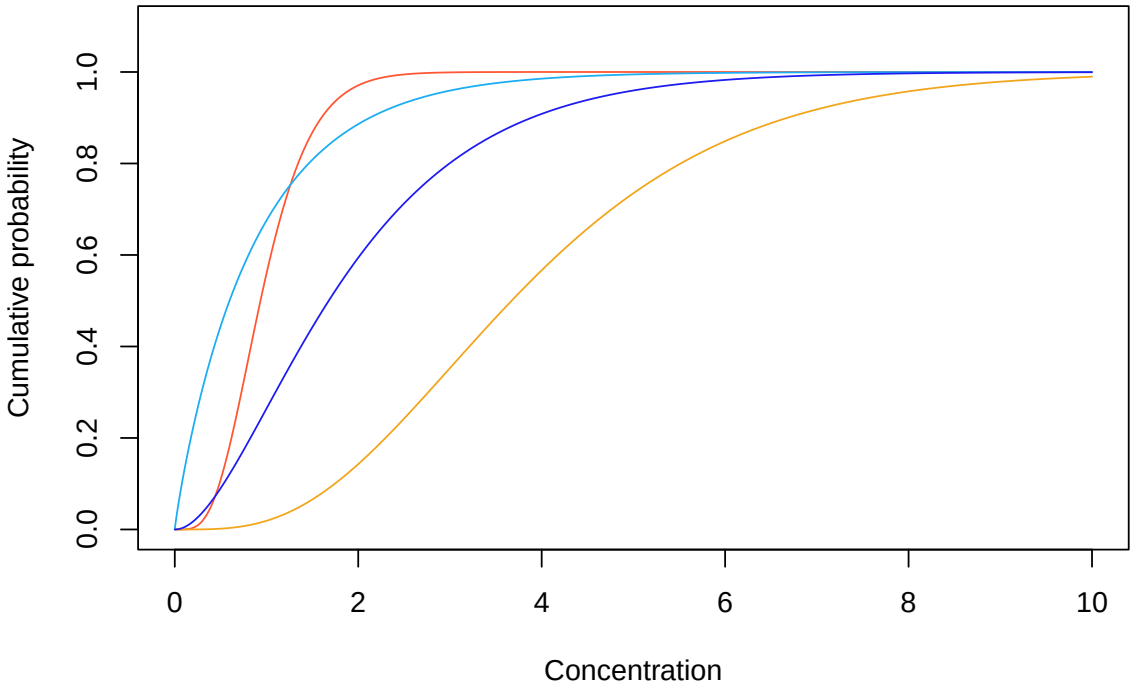
For use in modeling species sensitivity data, the gamma distribution has two key features that provide additional flexibility relative to the log-normal distribution: 1) it is asymmetrical on the logarithmic scale; and 2) it has wider tails.

The gamma distribution can be fitted using `ssdtools` by supplying the string "gamma" to the *dists* argument in the *ssd_fit_dists* call.

Sample gamma probability density functions




Sample gamma cumulative distribution functions

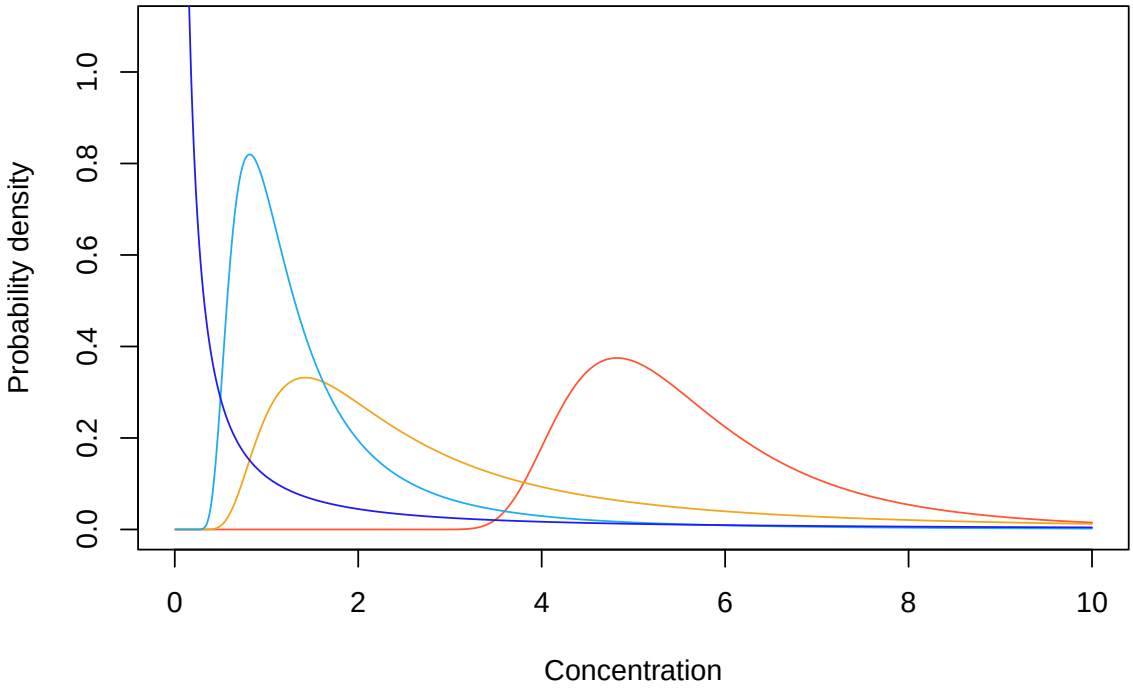



**Log-gumbel (inverse Weibull) distribution** The log-gumbel distribution is a two-parameter distribution commonly used to model extreme values.

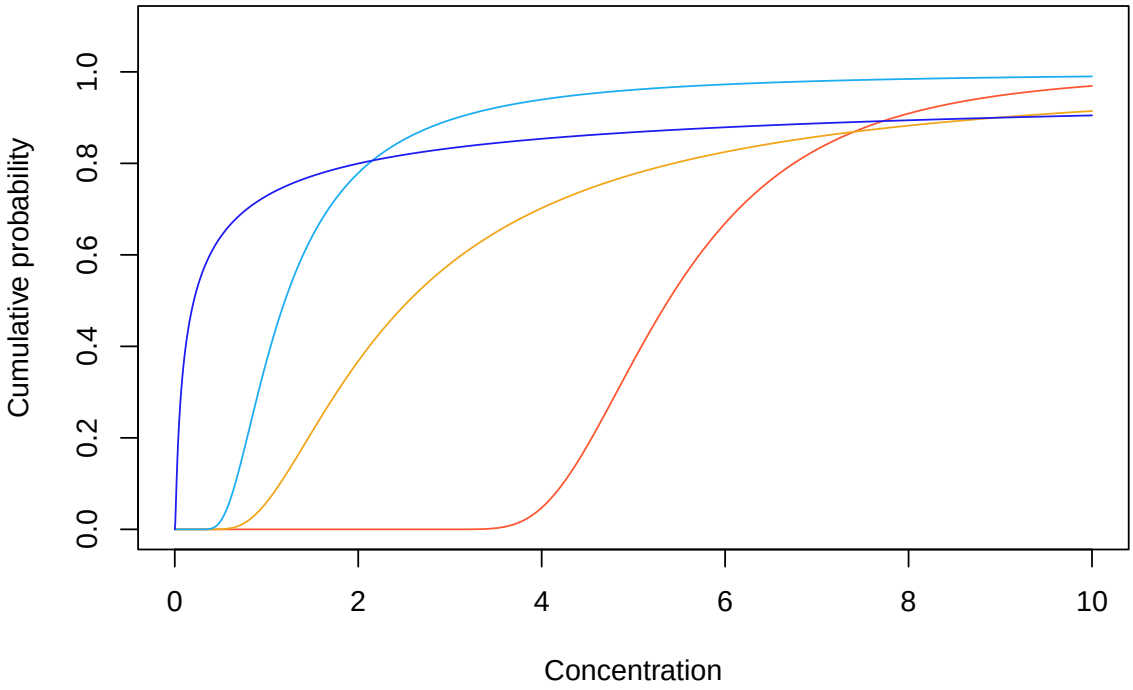
The log-gumbel distribution can be fitted using `ssdtools` by supplying the string `lgumbel` to the `dists` argument in the `ssd_fit_dists` call.

The two-parameter log-gumbel distribution has the following *pdf* and *cdf*:

Sample Log-Gumbel probability density functions




Sample Log-Gumbel cumulative distribution functions



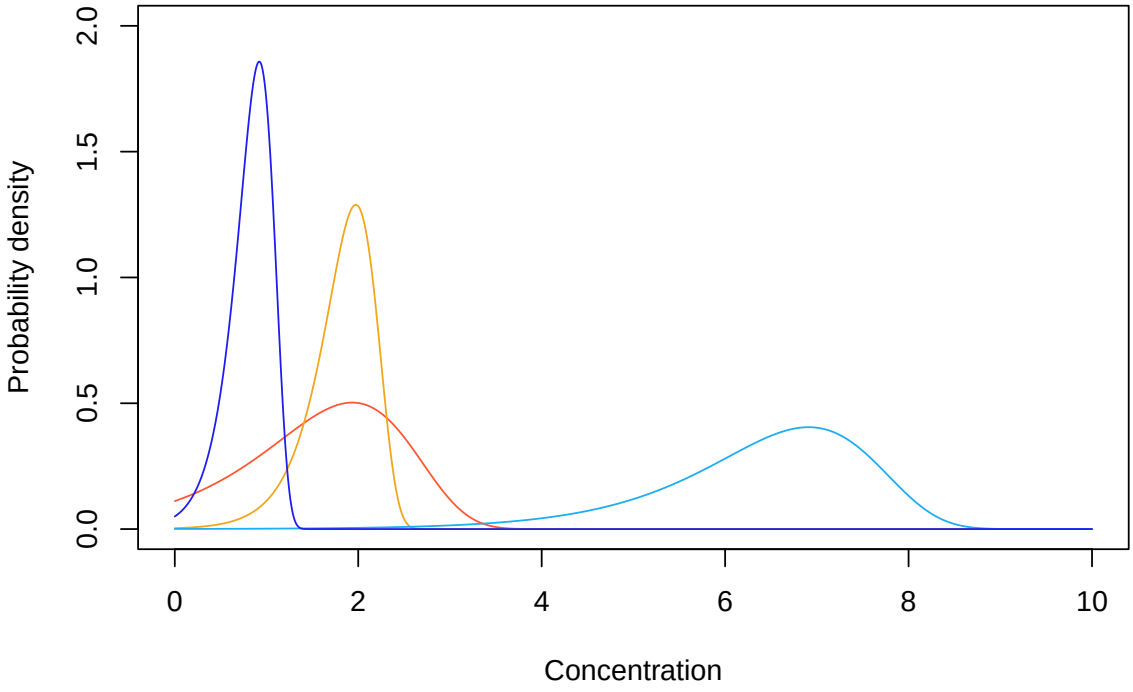
**Gompertz distribution** The Gompertz distribution is a flexible distribution that exhibits both positive and negative skewness.

The Gompertz distribution can be fitted using `ssdtools` by supplying the string `gompertz` to the `dists` argument in the `ssd_fit_dists` call.

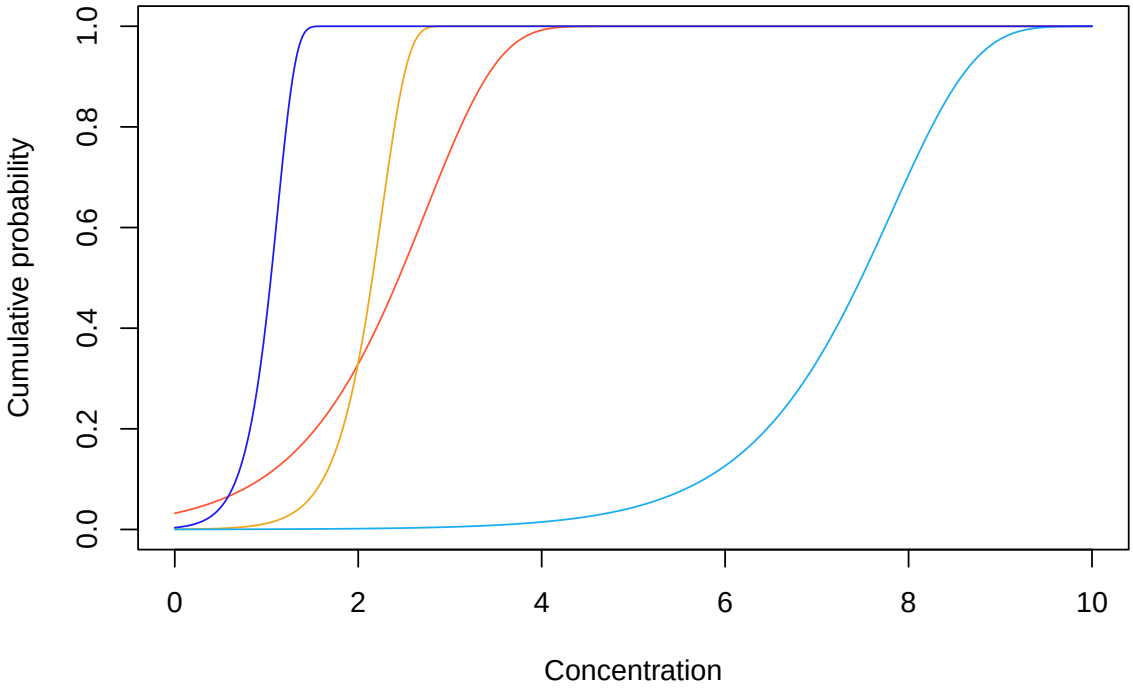
We condiser two parameterisations of the Gompertz distribution.The first, as given in Wikipedia and also used in `ssdtools` [Gompertz] has the following *pdf* and *cdf*:

The second parameterisation in which the *product* $b\eta$ in the formulae above is replaced by the parameter $a$ giving:

Sample Gompertz probability density functions




Sample Gompertz cumulative distribution functions



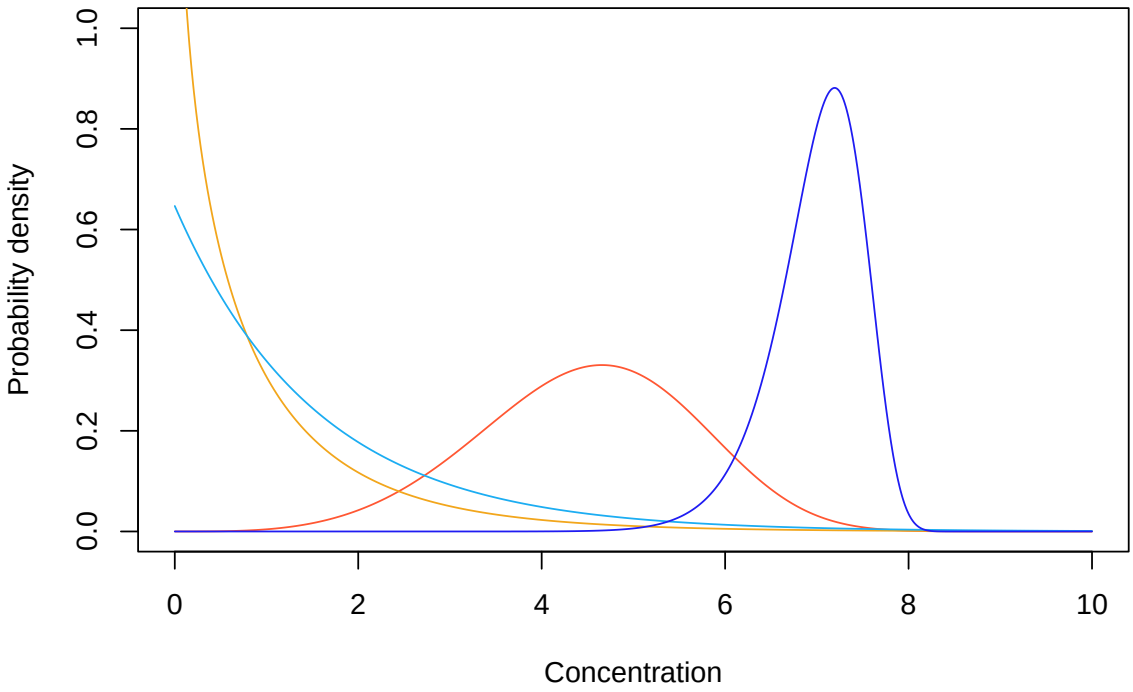

The Gompertz distribution is available in `ssdtools`, however parameter estimation can be somewhat unstable (Fox et al. 2021b), and for this reason it is not currently included in the default set.

**Weibull distribution** The inclusion of the Weibull distribution and inverse Pareto distribution (see next) in `ssdtools` was primarily necessitated by the need to maintain consistency with the calculations undertaken in `Burrlioz`. As mentioned earlier, both the Weibull and inverse Pareto distributions arise as *limiting distributions* when the Burr parameters $c$ and $k$ tend to either zero and/or infinity in specific ways.

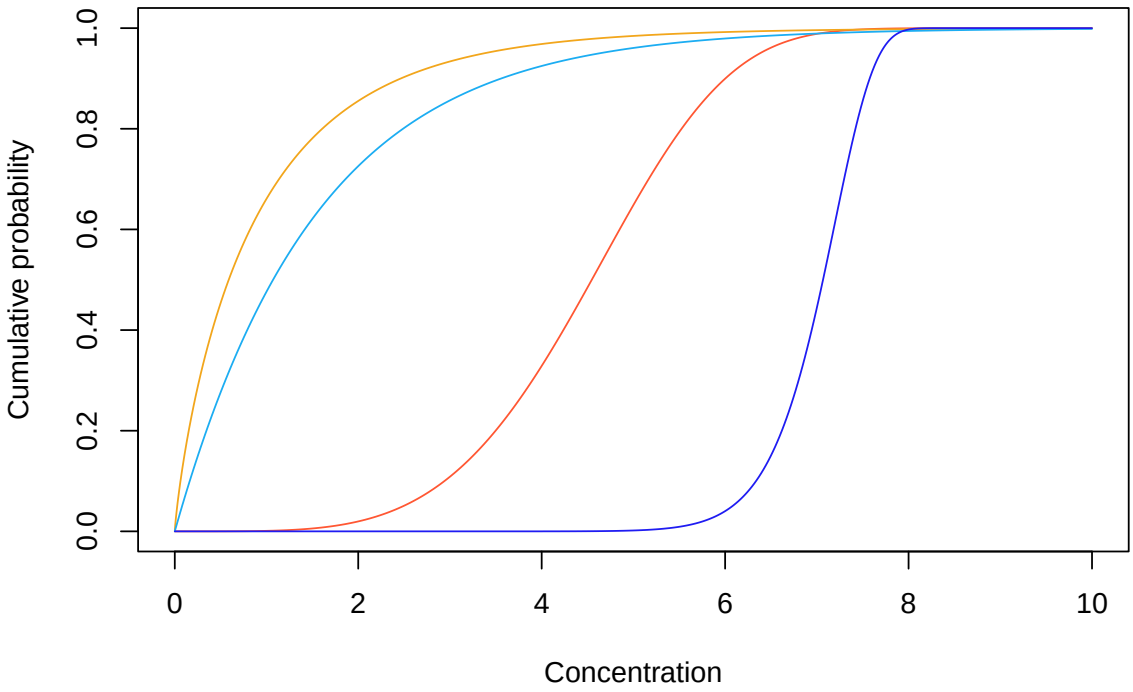
The two-parameter Weibull distribution has the following *pdf* and *cdf*:

The Weibull distribution can be fitted in `ssdtools` by supplying the string `weibull` to the `dists` argument in the `ssd_fit_dists` call.

Sample Weibull probability density functions




Sample Weibull cumulative distribution functions



**Inverse Pareto distribution** The inverse Pareto distribution can be fitted using `ssdtools` by supplying the string `invpareto` to the `dists` argument in the `ssd_fit_dists` call.

While the inverse Pareto distribution is implemented in the `Burrlioz 2.0` software, it is important to understand that it is done so only as one of the limiting Burr distributions (see technical details). The inverse Pareto is not offered as a stand-alone option in the `Burrlioz 2.0` software. We have spent considerable time and effort exploring the properties of the inverse Pareto distribution, including deriving bias correction equations and alternative methods for deriving confidence intervals (Fox et al. 2021b). This work has substantial value for improving the current `Burrlioz 2.0` method, and our bias corrections should be adopted when deriving *HCx* estimates from the inverse Pareto where parameters have been estimated using maximum likelihood.

As is the case with the `Burrlioz 2.0` software, we have decided not to include the inverse Pareto distribution in the default candidate set in `ssdtools` although it is offered ass a user-selectable distribution to use in the model-fitting process.

As with many statistical distributions, different 'variants' exist. These 'variants' are not so much different distributions as they are simple re-parameterisations. For example, many distributions have a *scale* parameter, $\beta$ and some authors and texts will use $\beta$ while others use $\frac{1}{\beta}$. An example of this re-paramterisation was given above for the Gompertz distribution. While the choice of mathematical representation may be purely preferential, it is sometimes done for mathematical convenience. For example, Parameterisation I of the Gompertz distribution above was obtained by letting $a = b\eta$ in Parameterisation II. This re-expression involving parameters $b$ and $\eta$ would be particularly useful when trying to fit a distribution for which one of $\{b, \eta\}$ was very small and the other was very large.

It has already been noted that the particular parameterisation of the (Inverse)Pareto distribution used in both `Burrlioz 2.0` and `ssdtools` was **not** a matter of preference, but rather was dictated by mathematical considerations which demonstrated convergence of the Burr distribution to one specific version of the (Inverse)Pareto distribution. While the mathematics provides an elegant solution to an otherwise problematic situation, this version of the (Inverse)Pareto distribution is not particularly use as a stand-alone distribution for fitting an SSD (other than as a special, limiting case of the Burr distribution).

The two versions of the (Inverse)Pareto distribution are known as the *European* and *North American* versions. Their *pdfs* and *cdfs* are given below.

Importantly, we see that the *North American* versions of these distributions are bounded with the Pareto distribution bounded **below** by $\beta$ and the inverse Pareto distribution bounded *above* by $\frac{1}{\beta}$.As an aside, the *mle* of $\beta$ in the Pareto distribution is

$$\hat{\beta} = \min\{X_1, \ldots, X_n\}$$

and the *mle* of $\frac{1}{\beta}$ in the inverse Pareto is

$$\begin{aligned}\tilde{\beta} &= \max\{Y_1, \ldots, Y_n\} \\ &= \max\left\{\tfrac{1}{X_1}, \ldots, \tfrac{1}{X_n}\right\} = \tfrac{1}{\min\{X_1, \ldots, X_n\}} \\ &= \tfrac{1}{\hat{\beta}}\end{aligned}$$

.

and the *mle* of $\alpha$ is:

$$\hat{\alpha} = \left[\ln\left(\frac{g}{\hat{\beta}}\right)\right]^{-1}$$

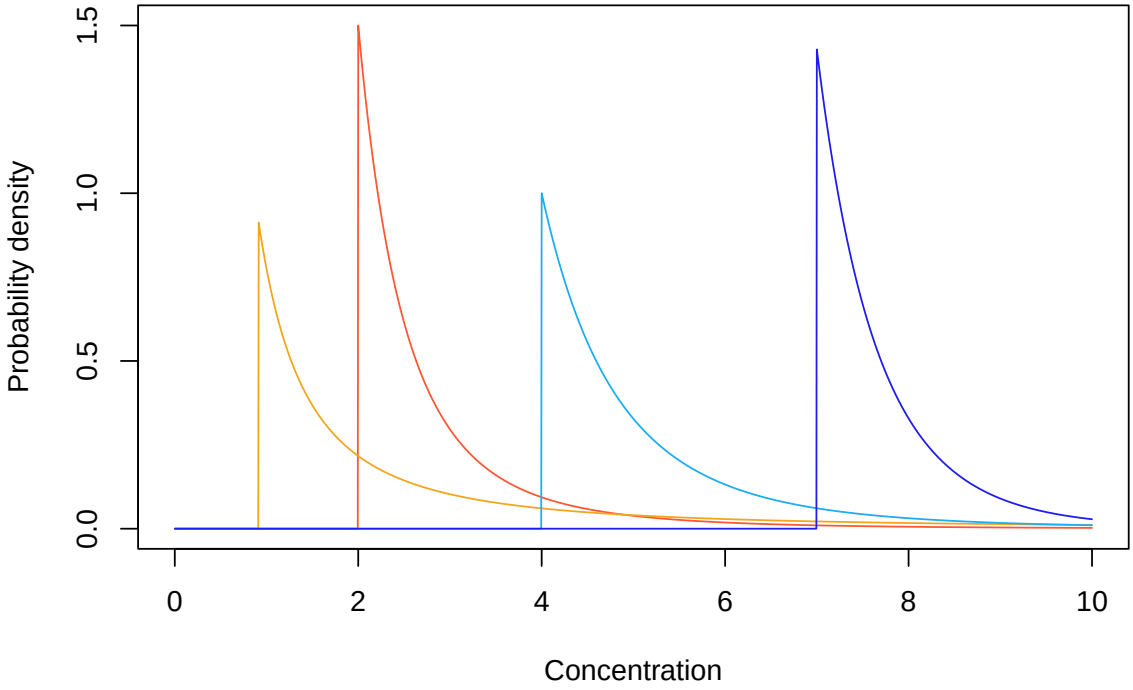
where $g$ is the *geometric mean*:

$$g = \left[\prod_{i=1}^{n} X_i\right]^{\frac{1}{n}}$$

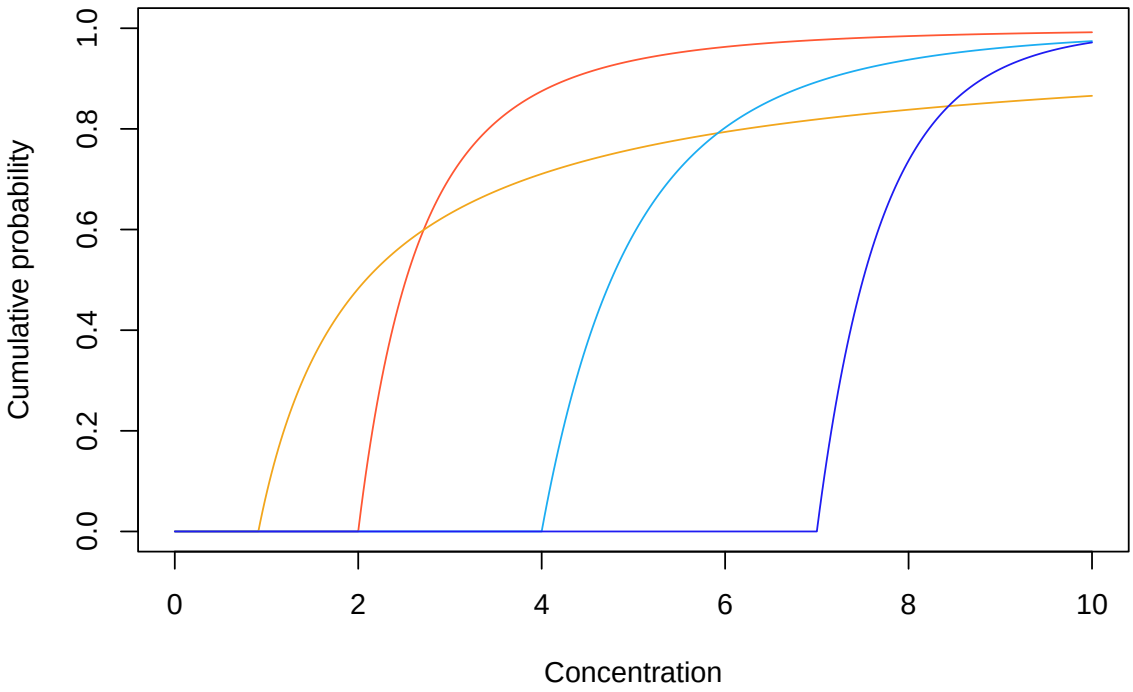
Thus, it doesn't matter whether you're fitting a Pareto or inverse Pareto distribution to your data - the parameter estimates are the same.

Because it is *bounded*, the North American version of the (Inverse)Pareto distribution is not useful as a stand-alone SSD - more so for the *inverse Pareto* distribution since it is bounded from *above*.

Sample North American probability density functions : Pareto distribution

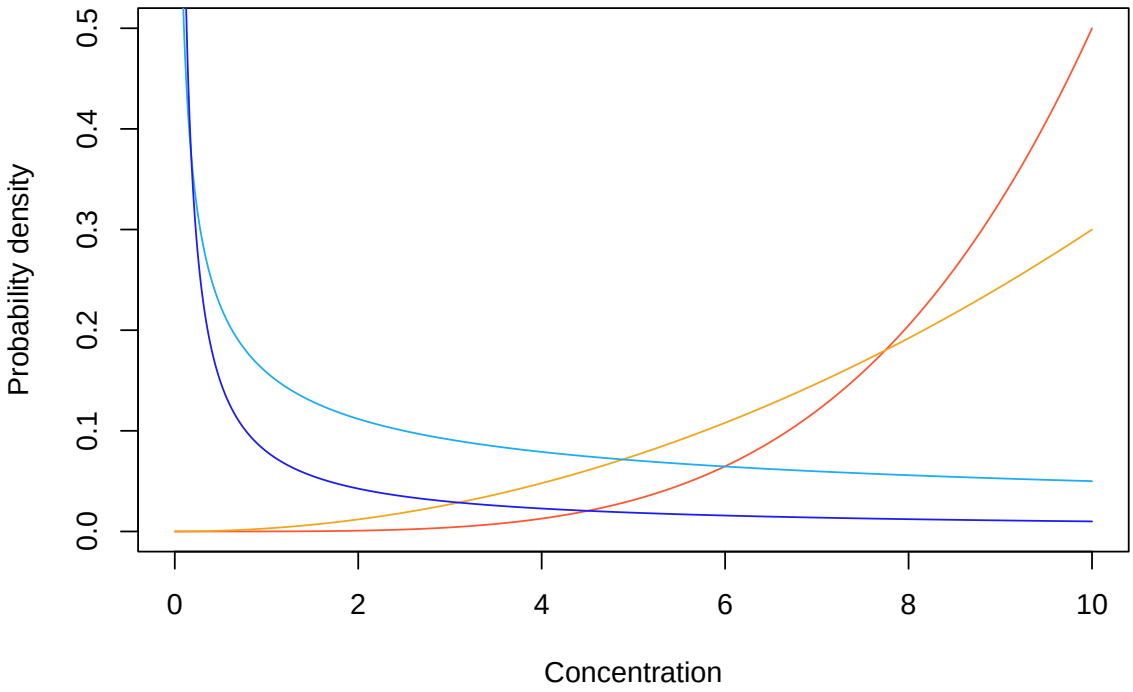



Sample North American cumulative distribution functions : Pareto distribution

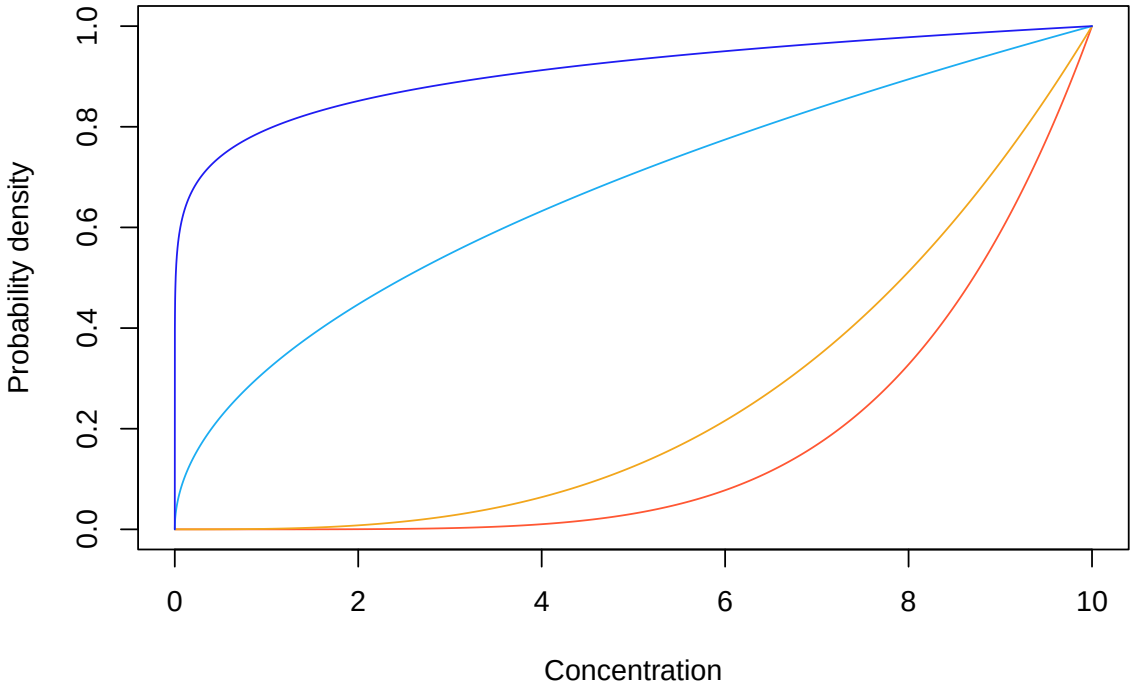



We see from the *pdf* plots that the

Sample North American probability density functions : inverse Pareto distribution



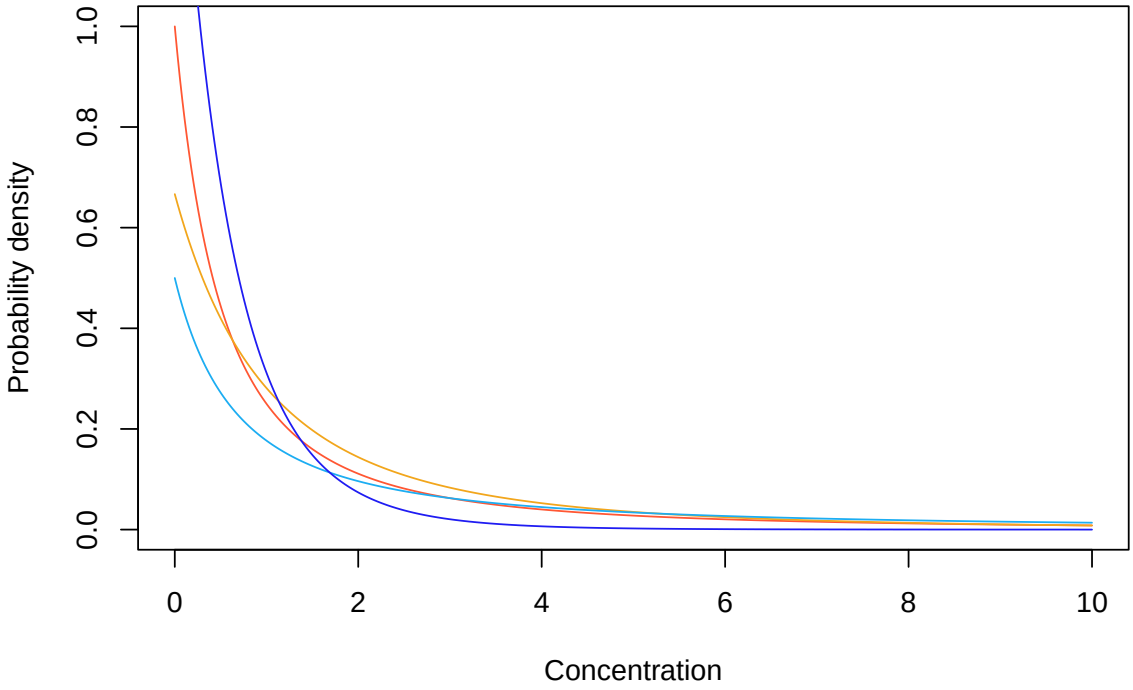

Sample North American cumulative distribution functions : inverse Pareto distribution



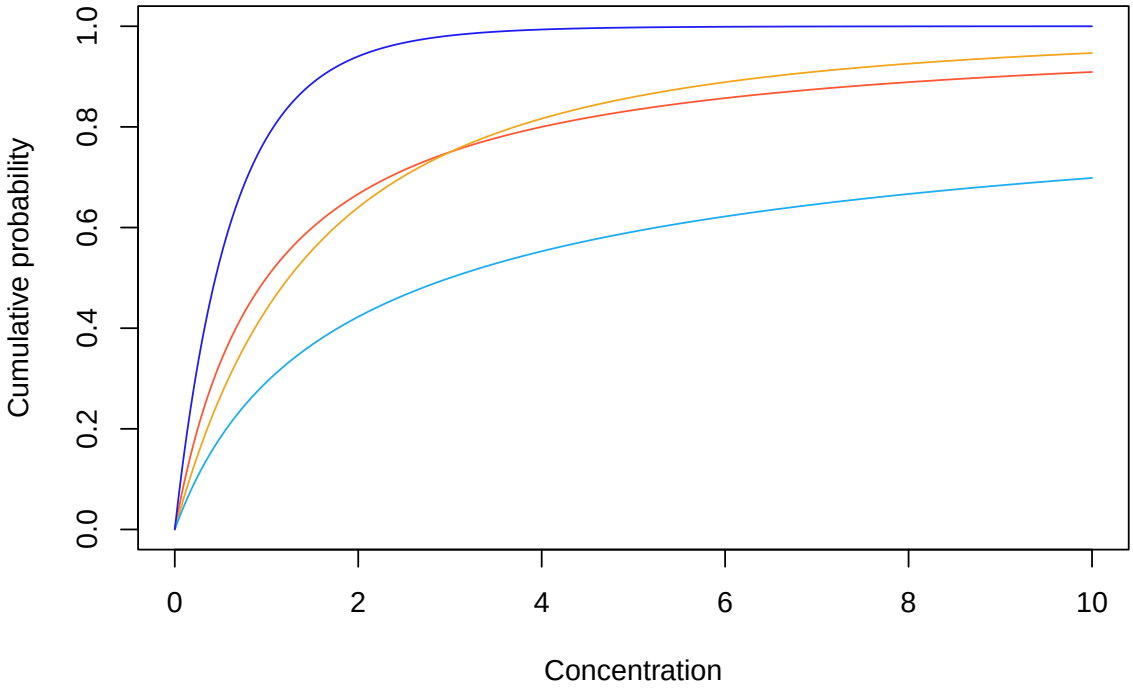
The alternative, *European* version of the inverse Pareto distribution is a more realistic candidate.

We note in passing that *both* versions of these Pareto and inverse Pareto distrbutions are availabale in `R`. For example, the `R`package `extraDistr` has North American versions, while the `actuar` package has European versions.

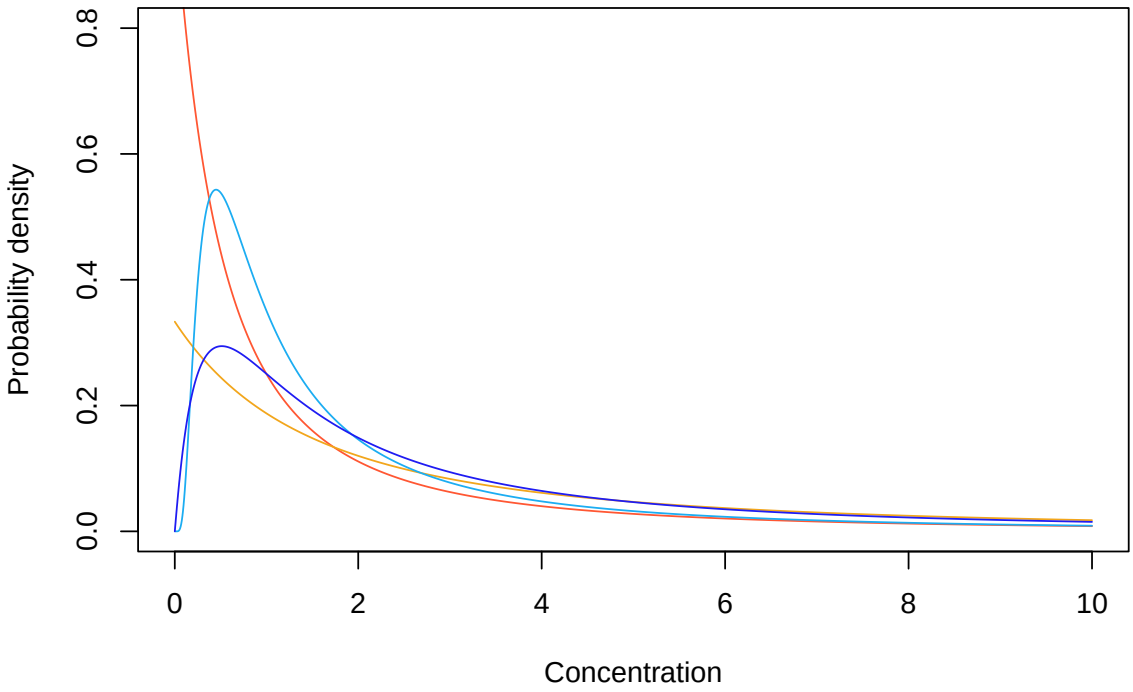
Sample European probability density functions : Pareto distribution



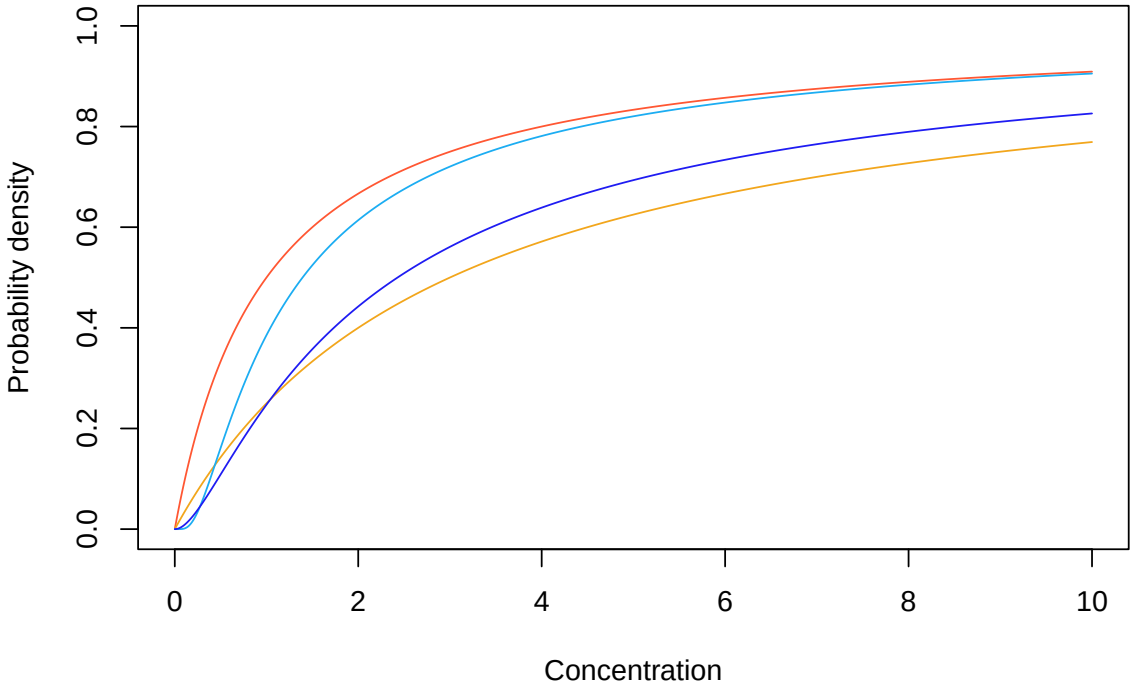

Sample European cumulative distribution functions : Pareto distribution




Sample European probability density functions : inverse Pareto distribution



Sample European cumulative distribution functions : inverse Pareto distribution



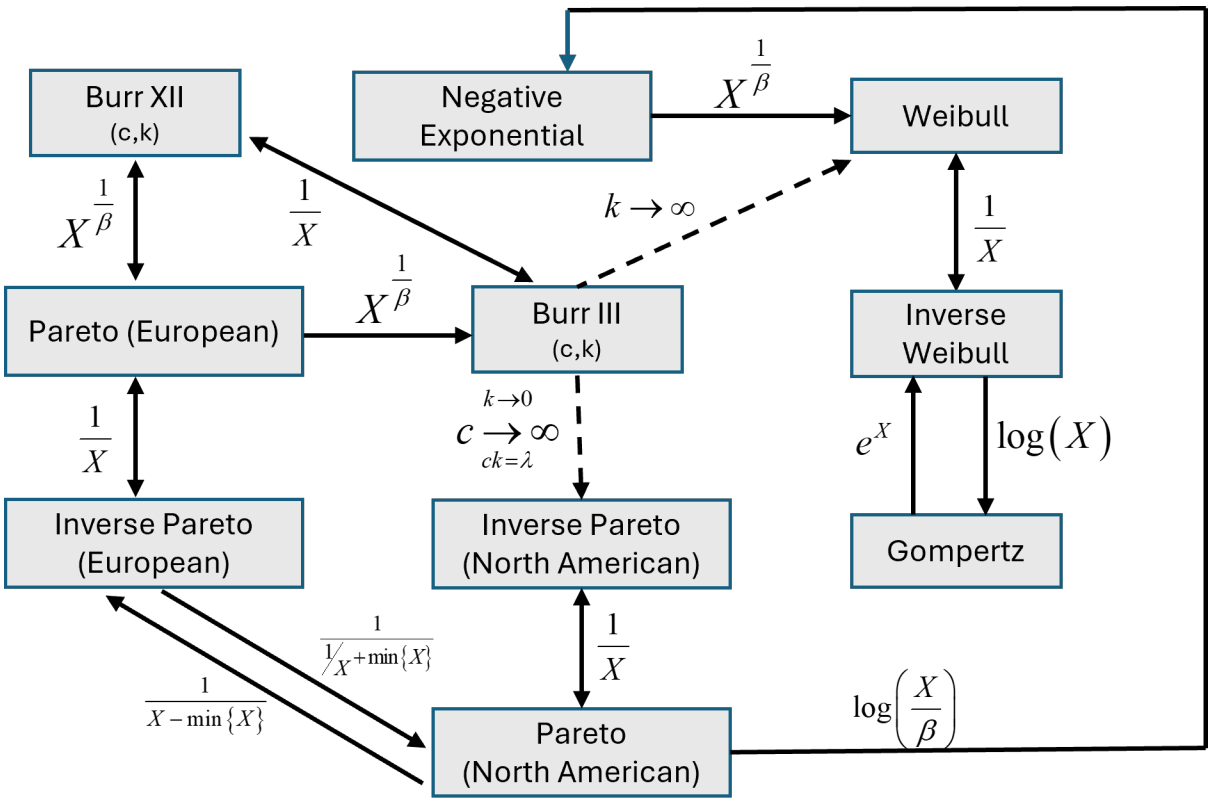

**Inverse Weibull distribution (see log-Gumbel, above)** The inverse Weibull is mathematically equivalent to the log-Gumbel distribution described above. While which is also a limiting distribution of the Burr Type 3, this distribution does not show the same instability issues, and is unbounded to the right. It therefore represents a valid SSD distribution and is included in the default model set as a distribution in its own right.

The inverse Weibull (log-Gumbel) distribution can be fitted in `ssdtools` by supplying the string `lgumbel` to the `dists` argument in the `ssd_fit_dists` call.

## Relationships among distributions in `ssdtools`

NOTES

1. In the diagram below, $X$ denotes the random variable in the box at the *beginning* of the arrow and the expression beside the arrow indicates the mathematical transformation of $X$ such that the resultant *transformed* data has the distribution identified in the box at the *end* of the arrow.
2. *Reciprocal* transformations ($\frac{1}{X}$) are **bi-directional** ($\leftrightarrow$).
3. Although the *negative exponential* distribution is **not** explicitly included in `ssdtools`, it is a special case of the *gamma* distribution with $c = 1$. It is included in this figure as it is related to other distributions that *are* included in `ssdtools`.
4. The *European* versions of the Pareto and inverse Pareto distributions are **unbounded**; the *North American* versions are **bounded**.




## References


Aldenberg T, Slob W (1993) Confidence Limits for Hazardous Concentrations Based on Logistically Distributed NOEC Toxicity Data. Ecotoxicology and Environmental Safety 25:48–63. https://doi.org/10.1006/eesa.1993.1006

Burr IW (1942) Cumulative Frequency Functions. The Annals of Mathematical Statistics 13:215–232

Dalgarno S (2021) Shinyssdtools: A web application for fitting Species Sensitivity Distributions (SSDs). Journal of Open Source Software 6:2848. https://doi.org/10.21105/joss.02848

Fisher R, Dam R van, Batley G, et al (2019) KEY ISSUES IN THE DERIVATION OF WATER QUALITY GUIDELINE VALUES: A WORKSHOP REPORT

Fox DR (2016) Contemporary methods for statistical design and analysis. In: Blasco J, Chapman PM, Campana O, Hampel M (eds) Marine Ecotoxicology

Fox DR, Dam RA, Fisher R, et al (2021a) Recent Developments in Species Sensitivity Distribution Modeling. Environmental Toxicology and Chemistry 40:293–308. https://doi.org/10.1002/etc.4925

Fox DR, Fisher R, Thorley JL, Schwarz C (2021b) Joint investigation into statistical methodologies underpinning the derivation of toxicant guideline values in Australia and New Zealand. Environmetrics Australia; Australian Institute of Marine Science

Newman MC, Ownby DR, Mézin LCA, et al (2000) Applying species-sensitivity distributions in ecological risk assessment: Assumptions of distribution type and sufficient numbers of species. Environmental Toxicology and Chemistry 19:508–515. https://doi.org/10.1002/etc.5620190233

Shao Q (2000) Estimation for hazardous concentrations based on NOEC toxicity data: An alternative approach. 13

Zajdlik B (2005) Statistical analysis of the SSD approach for development of canadian water quality guidelines. Zajdlik; Associates


## Licensing

# Appendix K – Confidence Intervals Vignette

# Confidence Intervals for Hazard Concentrations

ssdtools Team

2024-05-17

## Bootstrap confidence intervals

Bootstrapping is a resampling technique used to obtain estimates of summary statistics. The team have explored the use of alternative methods for obtaining the confidence interval of *HCx* estimates. This included using the closed-form expression for the variance-covariance matrix of the parameters of the Burr III distribution, coupled with the delta-method, as well as an alternative bootstrap method for the inverse Pareto distribution based on statistical properties of the parameters (Fox et al. 2021). In both cases, it appeared that these methods can give results similar to other traditional bootstrapping approaches in much less time, and are therefore potentially worth further investigation. However, implementation of such methods across all the distributions now available in ssdtools would be a substantial undertaking.

The revised version of ssdtools retains the computationally intensive bootstrapping method to obtain confidence intervals and an estimate of standard errors. We recommend a minimum bootstrap sample of 1,000 (the current default - see argument nboot in *?ssd_hc()*). However, more reliable results can be obtained using samples of 5,000 or 10,000. We recommend larger bootstrap samples for final reporting.

## Parametric versus non-parametric bootstrapping

Burrlioz 2.0 uses a non-parametric bootstrap method to obtain confidence intervals on the *HCx* estimate. Non-parametric bootstrapping is carried out by repeatedly resampling the raw data with replacement, and refitting the distribution many times. The 95% confidence limits are then obtained by calculating the lower 0.025th and upper 0.975th quantiles of the resulting *HCx* estimates across all56 the bootstrap samples (typically >1000). This type of bootstrap takes into account uncertainty in the distribution fit based on uncertainty in the data.

The ssdtools package by default uses a parametric bootstrap. Instead of resampling the data, parametric bootstrapping draws a random a set of new data (of the same sample size as the original) from the fitted distribution to repeatedly refit the distribution. Upper and lower 95% bounds are again calculated as the lower 0.025th and upper 0.975th quantiles of the resulting *HCx* estimates across all the bootstrap samples (again, typically >1000). This will capture the possible uncertainty that may occur for a sample size from a given distribution, but it assumes no uncertainty in that original fit, so it is not accounting for uncertainty in the input data.

The new TMB version of ssdtools has the capacity to do bootstrapping either using the Burrlioz non-parametric method, or the original parametric method of ssdtools (based on fitdistrplus (Delignette-Muller and Dutang 2015)).

Using simulation studies the ssdtools team examined bias and compared the resulting coverage of the parametric and non-parametric bootstrapping methods (Fox et al. 2021). They found that coverage was better using the parametric bootstrapping method, and this has been retained as the default bootstrapping method in the update to ssdtools.

## Bootstrapping model-averaged SSDs

Bootstrapping to obtain confidence intervals for individual fitted distributions is relatively straightforward. However, obtaining bootstrap confidence intervals for model-averaged SSDs requires careful

consideration, as the procedure is subject to the same pitfalls evident when obtaining model-averaged *HCx* estimates. The Model Average SSDs vignette contains a detailed explanation of the fallacy of using the summed weighting of individual *HCx* values (as weighted arithmetic average), and how this can lead to spurious results. 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()*, and this applies for model-averaged confidence intervals.

The revised version of ssdtools supports three weighting methods for obtaining bootstrap confidence intervals and an estimate of the standard error, and these are discussed in detail below.

### Weighted arithmetic mean

The early versions of ssdtools provided model-averaged confidence intervals (cis) and standard errors (se) that were calculated as weighted arithmetic means of the upper and lower cis and se values obtained via bootstrap simulation from each of the individual candidate distributions independently. This method is incorrect and may lead to spurious results (as described above) and has been shown via simulations studies to result in confidence intervals with very low coverage. The current version of ssdtools retains the functionality to reproduce the original behavior of ssdtools.

```
fit <- ssd_fit_dists(data = ssddata::ccme_silver)
set.seed = 99

# Using the original ssdtools weighted arithmetic mean
hc1 <- ssd_hc(fit, ci = TRUE, multi_est = FALSE, multi_ci = FALSE, weighted = FALSE)
```

```
hc1
#> # A tibble: 1 x 11
#>   dist    proportion   est    se    lcl   ucl    wt method   nboot pboot samples
#>   <chr>        <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <chr>    <dbl> <dbl> <I<lis>
#> 1 average       0.05 0.192 0.216 0.0679 0.861     1 paramet~  1000 0.998 <dbl>
```

Use of this method for obtaining ci and se values is not recommended and only retained for legacy comparison purposes. It is both technically incorrect, and computationally inefficient.

### Weighted mixture distribution

A more theoretically correct way of obtaining ci and se values is to consider the model average set as a mixture distribution (see above, and the Model Average SSDs vignette). When we consider the model set as a mixture distribution, bootstrapping is achieved by resampling from the model set according to the AICc based model weights. A method for sampling from mixture distributions has been implemented in ssdtools, via the function *ssd_rmulti()*, which will generate random samples from a mixture of any combination of distributions currently implemented in `ssdtools`. Setting "multi_ci = TRUE" in the *ssd_hc()* call will ensure that bootstrap samples are drawn from a mixture distribution, instead of individual candidate distributions.

When bootstrapping from the mixture distribution, a question arises whether the model weights should be re-estimated for every bootstrap sample, or fixed at the values estimated from the models fitted to the original sample of toxicity data? This is an interesting question that may warrant further investigation, however our current view is that they should be fixed at their nominal values in the same way that the component distributions to be used in bootstrapping are informed by the fit to the sample toxicity data. Using simulation studies we explored the coverage and bias of ci values obtained without and without fixing the distribution weights, and results indicate little difference.

If treating the distributions as a single mixture distribution when calculating model average confidence intervals (i.e. with "multi_ci = TRUE"), then setting "weighted = FALSE" specifies to use the original

model weights. Setting "weighted = TRUE" will result in bootstrapping that will re-estimate weights for each bootstrap sample.

The following code can be used to obtain confidence intervals for *HCx* estimates via bootstrapping from the weighted mixture distribution (using *ssd_rmutli()*), with and without fixed weight values respectively.

```
# Using the rmulti boostrapping method with fixed weights
hc2 <- ssd_hc(fit, ci = TRUE, multi_est = TRUE, multi_ci = TRUE, weighted = FALSE)
```

```
hc2
#> # A tibble: 1 x 11
#>   dist    proportion   est    se    lcl   ucl    wt method   nboot pboot samples
#>   <chr>        <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <chr>    <dbl> <dbl> <I<lis>
#> 1 average       0.05 0.190 0.212 0.0216 0.878     1 paramet~  1000     1 <dbl>
```

```
# Using the rmulti boostrapping method with fixed weights
hc3 <- ssd_hc(fit, ci = TRUE, multi_est = TRUE, multi_ci = TRUE, weighted = TRUE)
```

```
hc3
#> # A tibble: 1 x 11
#>   dist    proportion   est    se    lcl   ucl    wt method   nboot pboot samples
#>   <chr>        <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <chr>    <dbl> <dbl> <I<lis>
#> 1 average       0.05 0.190 0.208 0.0216 0.813     1 paramet~  1000     1 <dbl>
```

Use of this method (without or without fixed weights) is theoretically correct, but is computationally very inefficient.

### Weighted bootstrap sample

The developers of `ssdtools` investigated a third method for obtaining confidence intervals for the model-averaged SSD. This method bootstraps from each of the distributions individually, taking a weighted sample from each, and then combining these into a pooled bootstrap sample for estimation of te ci and se values. Psuedo code for this method is as follows:

- For each distribution in the `fitdists` object, the proportional number of bootstrap samples to draw (`nboot_vals`) is found using `round(nboot * weight)`, where `nboot` is the total number of bootstrap samples and weight is the AICc based model weights for each distribution based on the original `ssd_fitdist` fit.
- For each of the `nboot_vals` for each distribution, a random sample of size N is drawn (the total number of original data points included in the original SSD fit) based on the estimated parameters from the original data for that distribution.
- The random sample is re-fitting using that distribution.
- *HCx* is estimated from the re-fitted bootstrap fit.
- The *HCx* estimates for all `nboot_vals` for each distribution are then pooled across all distributions, and *quantile()* is used to determine the lower and upper confidence bounds for this pooled weighted bootstrap sample of *HCx* values.

This method does not draw random samples from the mixture distribution using *ssd_rmulti* (thus "multi_ci = FALSE"). While mathematically the method shares some properties with obtaining *HCx* estimates via summing the weighted values (weighted arithmetic mean), simulation studies have shown that, as a method for obtaining confidence intervals, this pooled weighted sample method yields similar ci values and coverage the *ssd_rmulti()* method, and is computationally much faster.

This method is currently the default method in ssdtools, and can be implemented by setting "multi_ci = FALSE" and "weighted = TRUE" in the ssd_hc() call.

```
# Using a weighted pooled bootstrap sample
hc4 <- ssd_hc(fit, ci = TRUE, multi_est = FALSE, multi_ci = FALSE, weighted = TRUE)
```

```
hc4
#> # A tibble: 1 x 11
#>   dist    proportion   est    se    lcl   ucl    wt method   nboot pboot samples
#>   <chr>        <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <chr>    <dbl> <dbl> <I<lis>
#> 1 average       0.05 0.192 0.214 0.0181 0.816     1 paramet~  1000 0.999 <dbl>
```

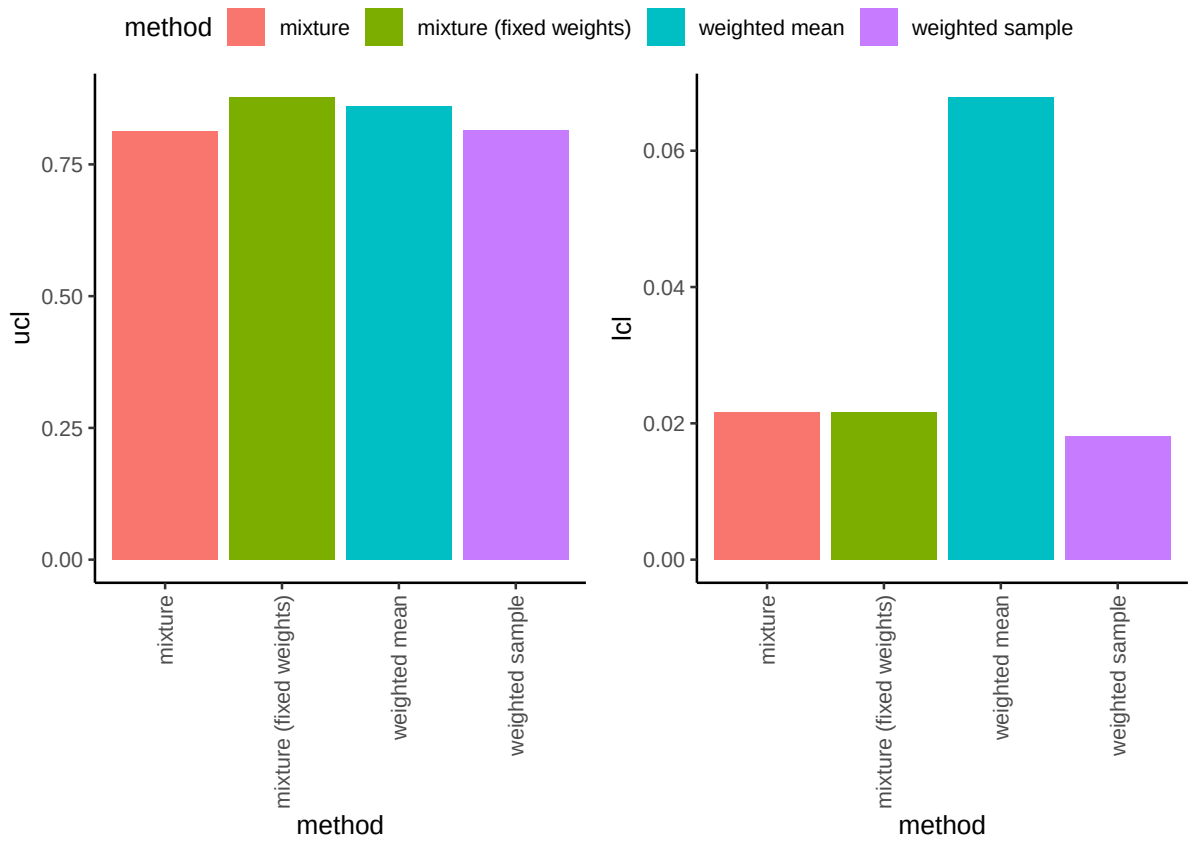
Here, the argument "weighted = TRUE" specifies to take bootstrap samples from each distribution proportional to its weight (so that they sum to nboot).

## Comparing bootrapping methods

We have undertaken extensive simulation studies comparing the implemented methods, and the results of these are reported elsewhere. For illustrative purposes, here we compare upper and lower confidence intervals using only a single example data set, the Silver data set from the Canadian Council of Ministers of the Environment (ccme).

Using the default settings for ssdtools, we compared the upper and lower confidence intervals for the four bootstrapping methods described above. Estimate upper confidence limits are relatively similar among the four methods. However, the lower confidence interval obtained using the weighted arithmetic mean (the method implemented in earlier versions of ssdtools) is much higher than the other three methods, potentially accounting for the relatively poor coverage of this method in our simulation studies.

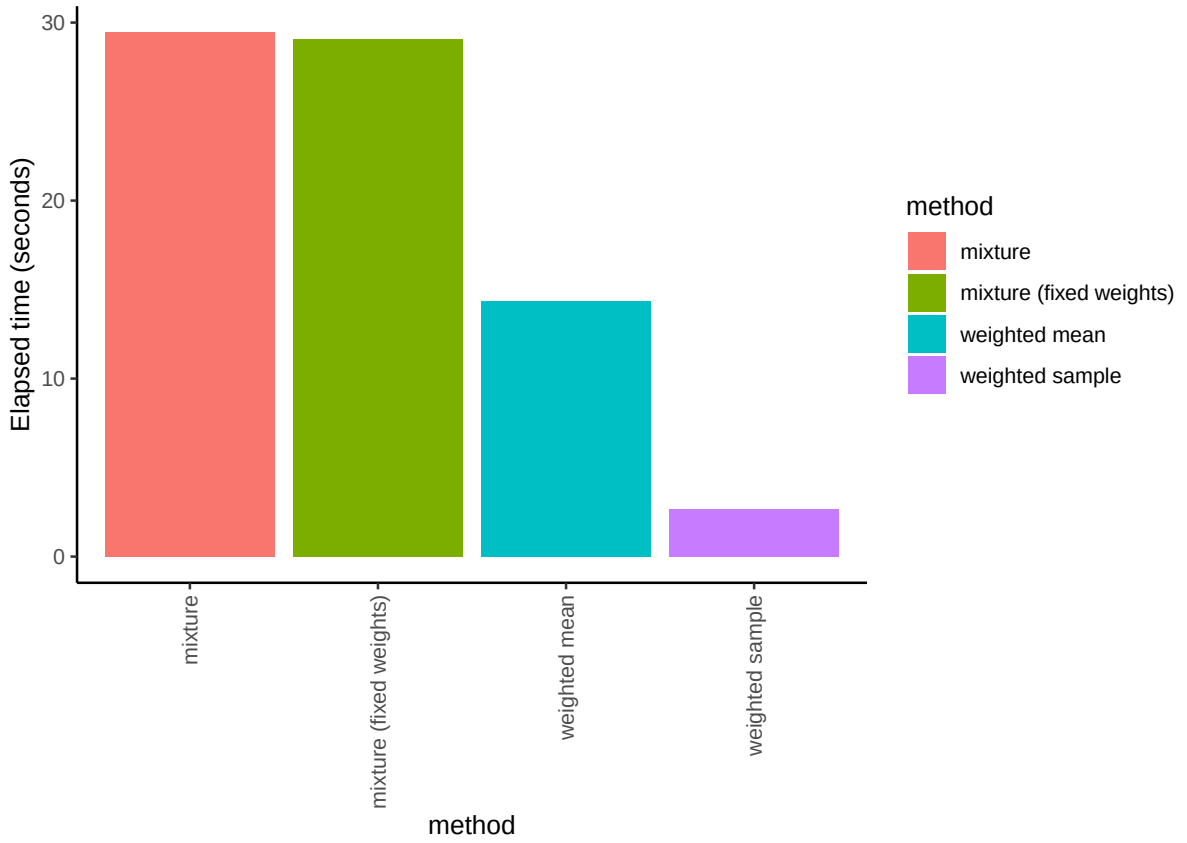
```
library(ggplot2)
library(ggpubr)
p1 <- ggplot(compare_dat, aes(method, ucl, fill = method)) +
    geom_bar(stat="identity", position=position_dodge()) +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
p2 <- ggplot(compare_dat, aes(method, lcl, fill = method)) +
    geom_bar(stat="identity", position=position_dodge()) +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

ggarrange(p1, p2,common.legend = TRUE)
```




Given the similarity of upper and lower confidence intervals of the weighted bootstrap sample method compared to the potentially more theoretically correct, but computationally more intensive weighted mixture method (via *ssd_rmulti()*), we also compared the time taken to undertake bootstrapping across the methods.

Using the default 1,000 bootstrap samples, the elapsed time to undertake bootstrapping for the mixture method was 29.07 seconds, compared to 2.66 seconds for the weighted bootstrap sample. This means that the weighted bootstrap method is ~ 11 times faster, representing a considerable computational saving across many SSDs. For this reason, this method is currently set as the default method for confidence interval estimation in ssdtools.

```
p3 <- ggplot(compare_dat, aes(method, time, fill = method)) +
    geom_bar(stat="identity", position=position_dodge()) +
    ylab("Elapsed time (seconds)") +
    theme_classic() +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
p3
```




## References

Delignette-Muller ML, Dutang C (2015) fitdistrplus: An R package for fitting distributions. Journal of Statistical Software 64:1–34

Fox DR, Fisher R, Thorley JL, Schwarz C (2021) Joint investigation into statistical methodologies underpinning the derivation of toxicant guideline values in Australia and New Zealand. Environmetrics Australia; Australian Institute of Marine Science

## Licensing

# Appendix L – Embellishing plots Vignette

# Customising Plots

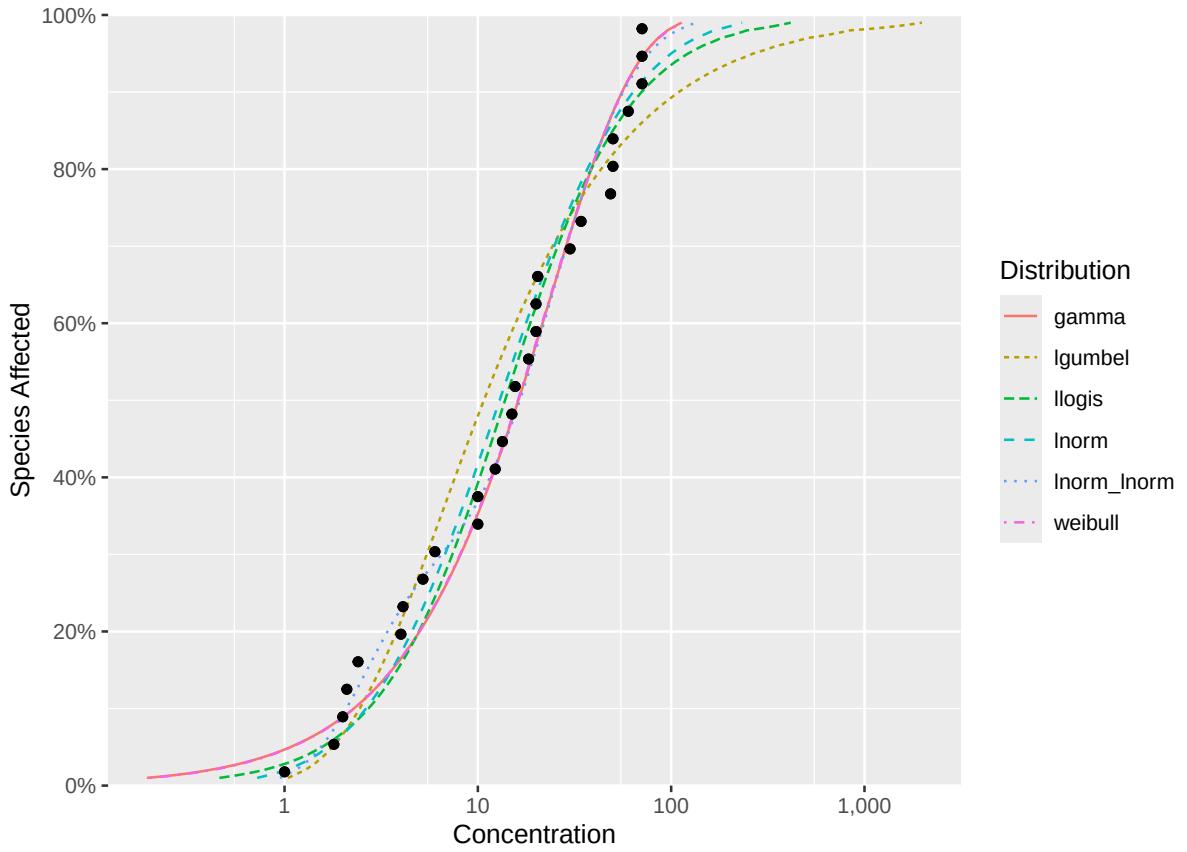
ssdtools Team

2024-05-17

## Plotting the cumulative distribution

The `ssdtools` package produces a plot of the cumulative distribution functions for the multiple input distributions through the use of the `ssd_plot_cdf()` function. For example, consider the boron data. We can fit, and then plot the cdf using:

```
library(ggplot2)
library(ssdtools)

fits <- ssd_fit_dists(ssddata::ccme_boron)
gp <- ssd_plot_cdf(fits)

print(gp)
```



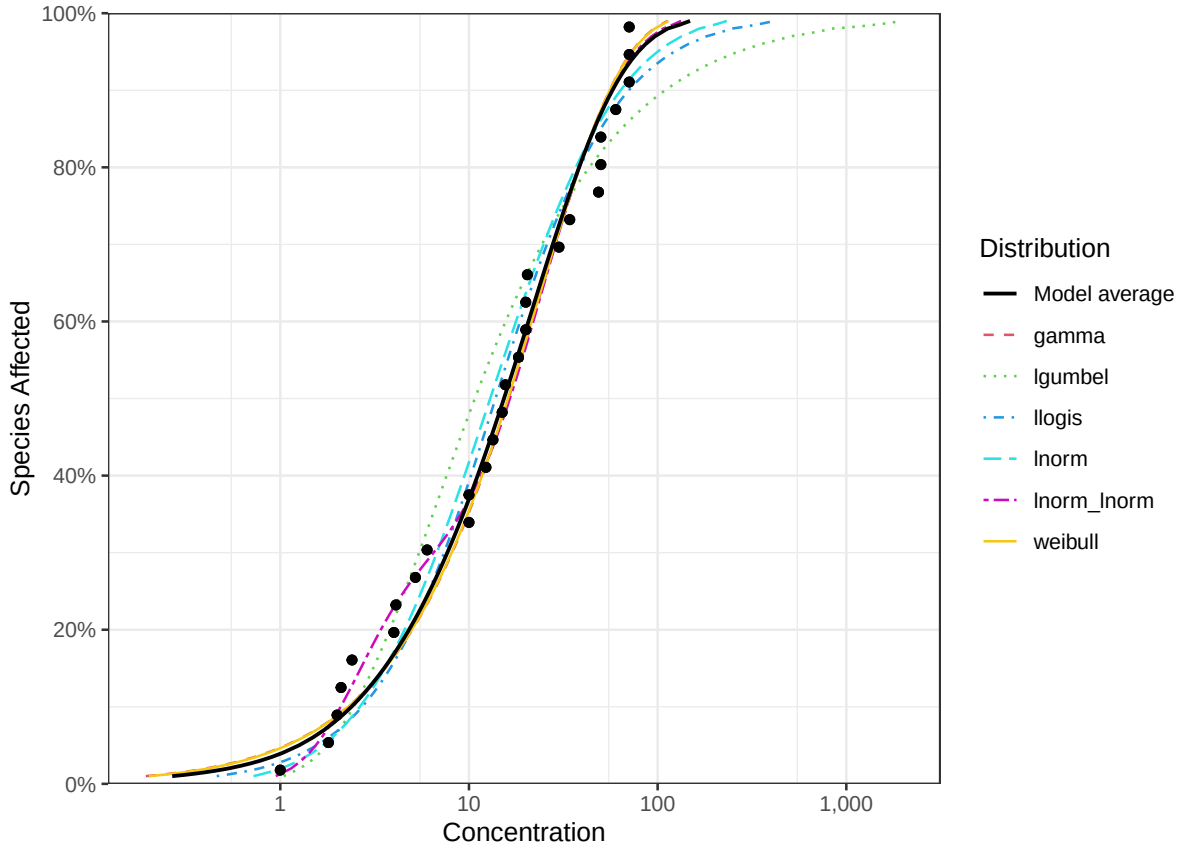

This graphic is a ggplot object and so can be saved and embellished in the usual way.

## Customising the cumulative distribution plot

### Plot the model-averaged fit with individual fits

We can add the model-averaged cdf by first obtaining predicted values, and extending the default ssdtools ggplot in the usual way using geom_line:

```
library(ssddata)
library(ssdtools)
library(ggplot2)

dist <- ssdtools::ssd_fit_dists(ssddata::ccme_boron)
pred <- predict(dist, ci = FALSE)
boron_hc5 <- ssd_hc(dist)
ssdtools::ssd_plot_cdf(dist) +
  geom_line(data = pred, aes(x = est, y = proportion, colour = "Model average", lty = "Model average
  scale_linetype_manual(name = "Distribution", breaks = c("Model average", names(dist)), values = 1:
  scale_color_manual(name = "Distribution", breaks = c("Model average", names(dist)), values = 1:7)
  theme_bw()
```

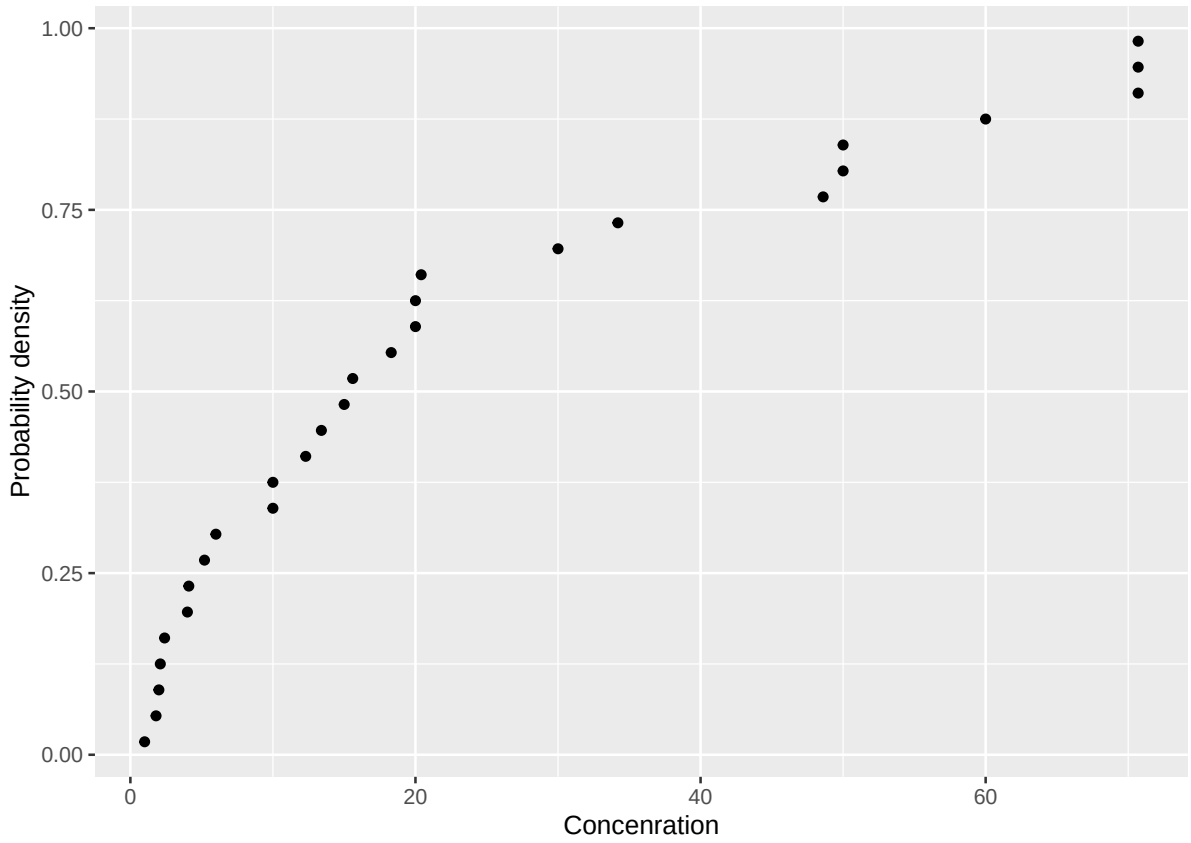



### Other customisations

The `ssdtools` package provides four ggplot geoms to allow you construct your own plots.

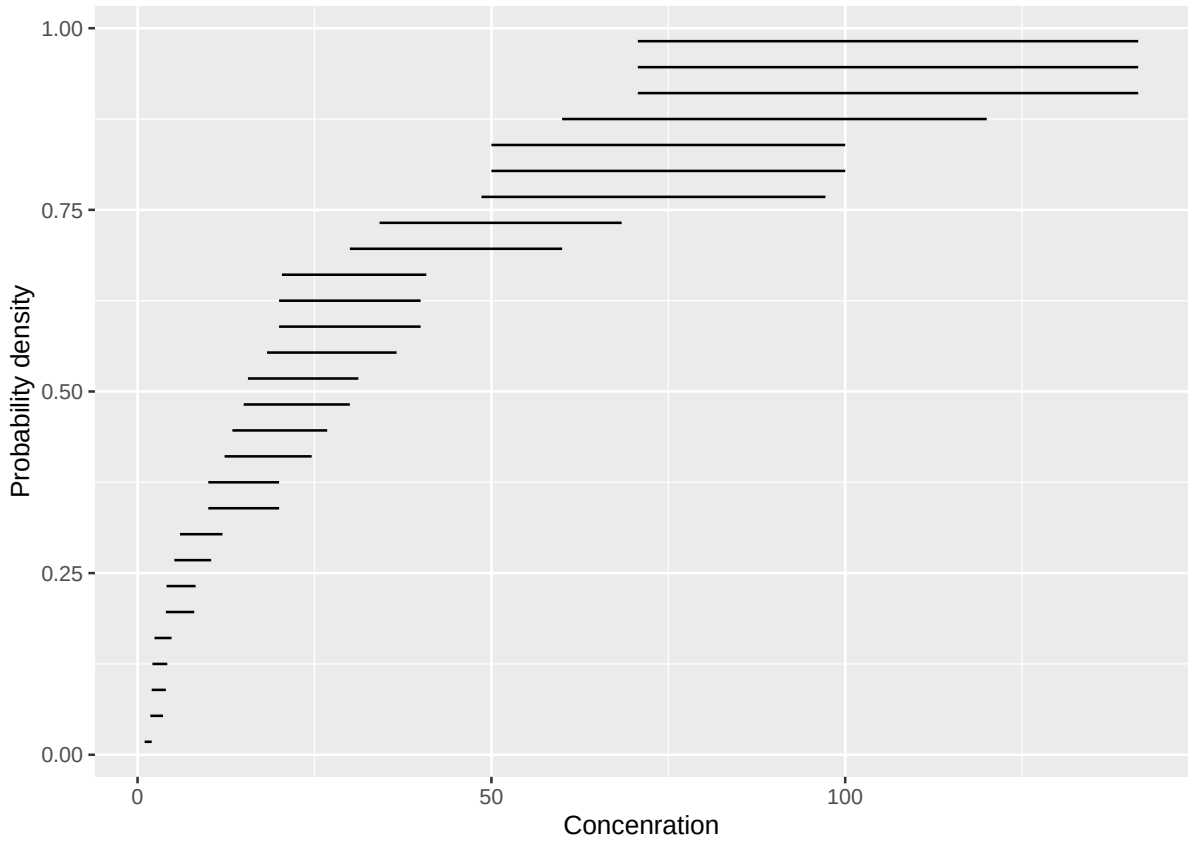
The first is `geom_ssdpoint()` which plots species sensitivity data

```
ggplot(ccme_boron) +
  geom_ssdpoint(aes(x = Conc)) +
  ylab("Probability density") +
  xlab("Concenration")
```



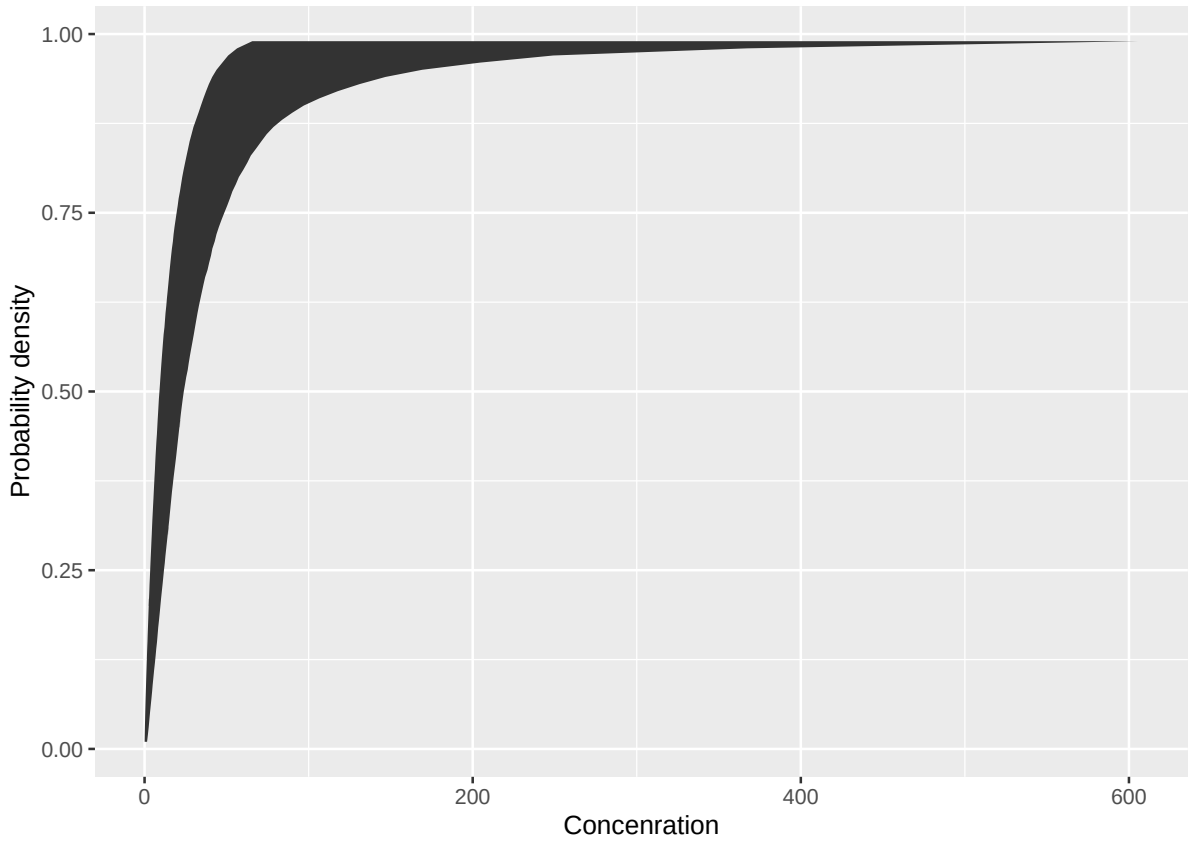

The second is `geom_ssdsegments()` which plots the range of censored species sensitivity data

```
ggplot(ccme_boron) +
  geom_ssdsegment(aes(x = Conc, xend = Conc * 2)) +
  ylab("Probability density") +
  xlab("Concenration")
```



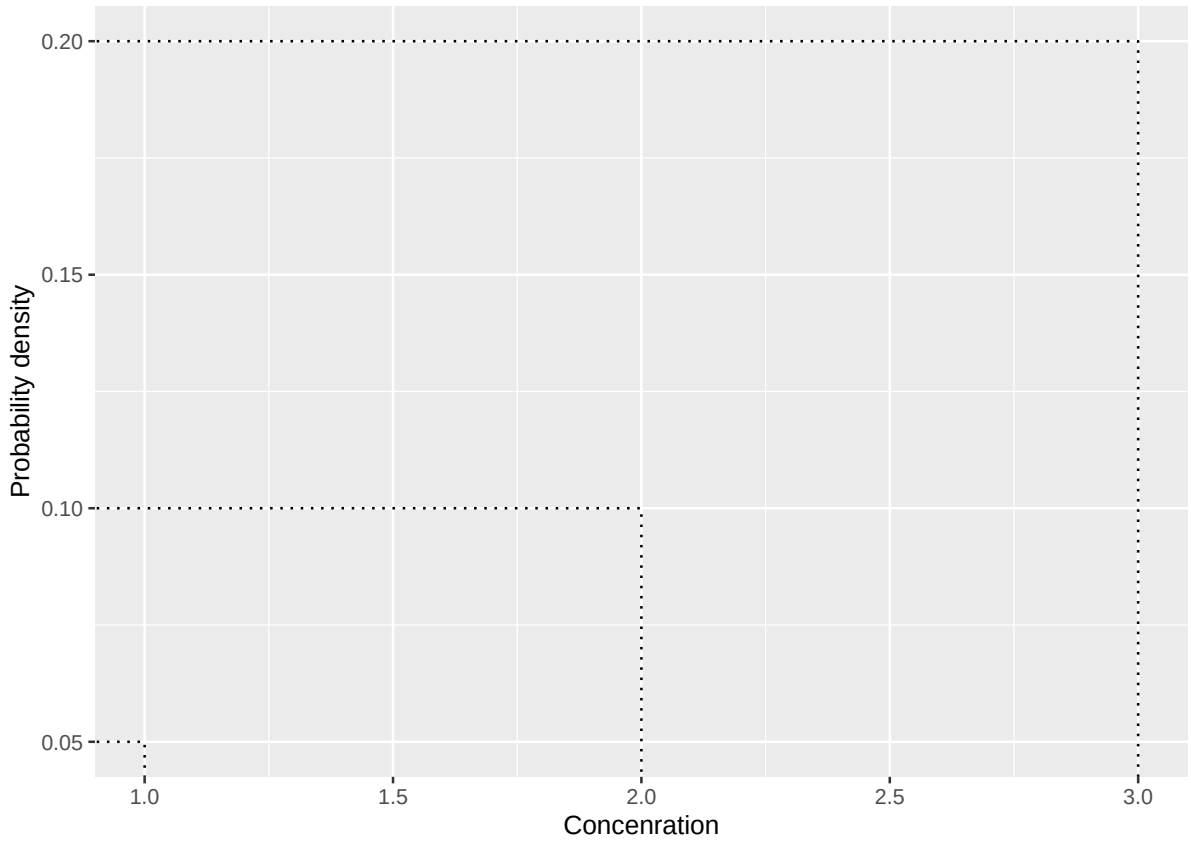

The third is `geom_xribbon()` which plots species sensitivity confidence intervals

```
ggplot(boron_pred) +
  geom_xribbon(aes(xmin = lcl, xmax = ucl, y = proportion)) +
  ylab("Probability density") +
  xlab("Concenration")
```



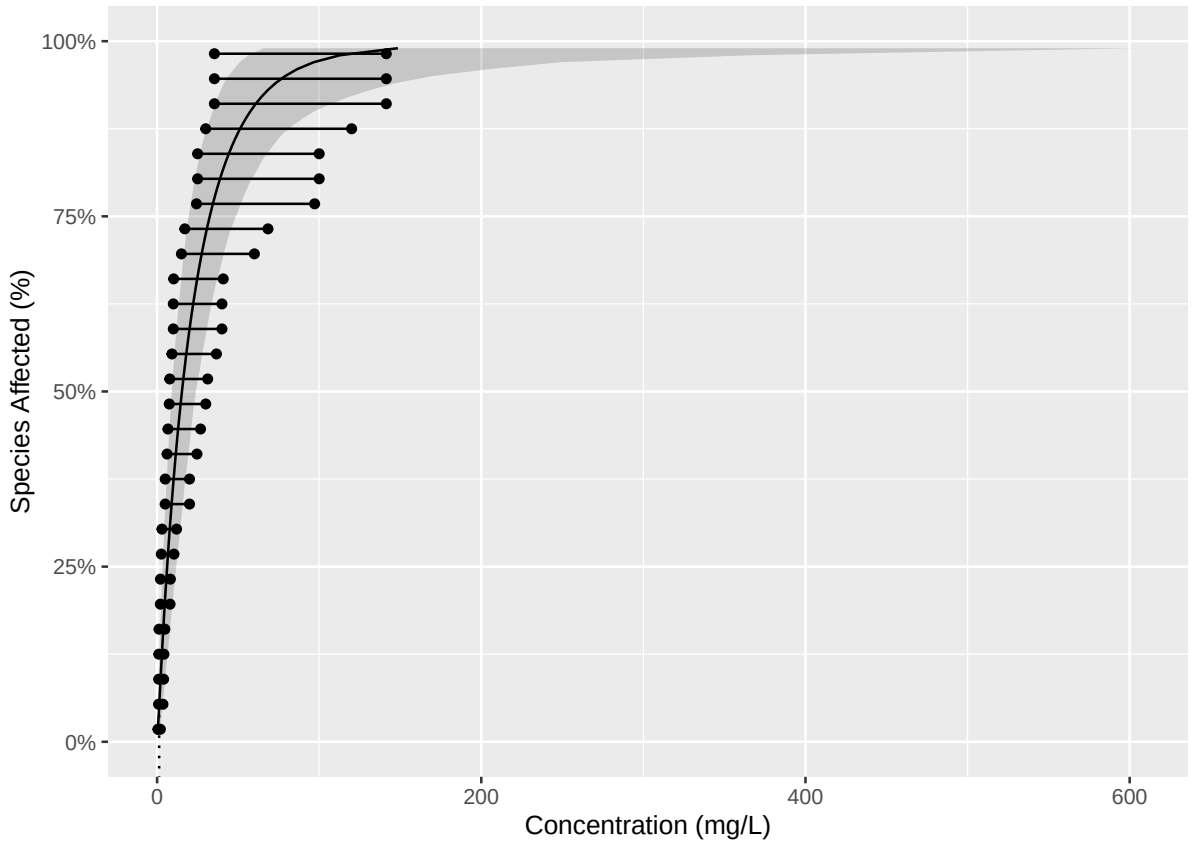

And the fourth is `geom_hcintersect()` which plots hazard concentrations

```
ggplot() +
  geom_hcintersect(xintercept = c(1, 2, 3), yintercept = c(0.05, 0.1, 0.2)) +
  ylab("Probability density") +
  xlab("Concenration")
```



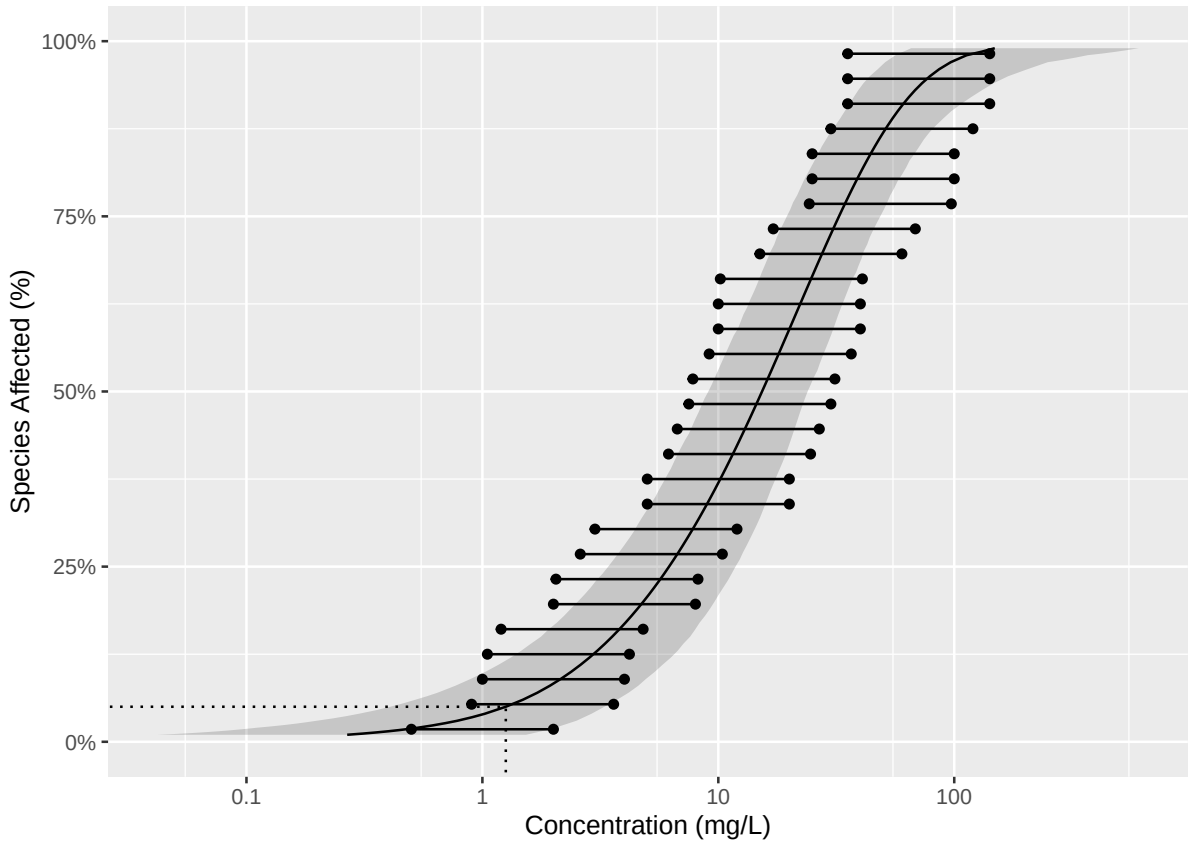

They can be combined together as follows

```
gp <- ggplot(boron_pred, aes(x = est)) +
  geom_xribbon(aes(xmin = lcl, xmax = ucl, y = proportion), alpha = 0.2) +
  geom_line(aes(y = proportion)) +
  geom_ssdsegment(data = ccme_boron, aes(x = Conc / 2, xend = Conc * 2)) +
  geom_ssdpoint(data = ccme_boron, aes(x = Conc / 2)) +
  geom_ssdpoint(data = ccme_boron, aes(x = Conc * 2)) +
  scale_y_continuous("Species Affected (%)", labels = scales::percent) +
  expand_limits(y = c(0, 1)) +
  xlab("Concentration (mg/L)")
print(gp + geom_hcintersect(xintercept = boron_hc5$est, yintercept = 5 / 100))
```




To log the x-axis add the following code.

```
gp <- gp + coord_trans(x = "log10") +
  scale_x_continuous(
    breaks = scales::trans_breaks("log10", function(x) 10^x),
    labels = comma_signif
  )
print(gp + geom_hcintersect(xintercept = boron_hc5$est, yintercept = 0.05))
```




The most recent plot can be saved as a file using `ggsave()`, which also allows the user to set the resolution.

```
ggsave("file_name.png", dpi = 600)
```

## Fitting and plotting distributions to multiple groups such taxa and/or chemicals

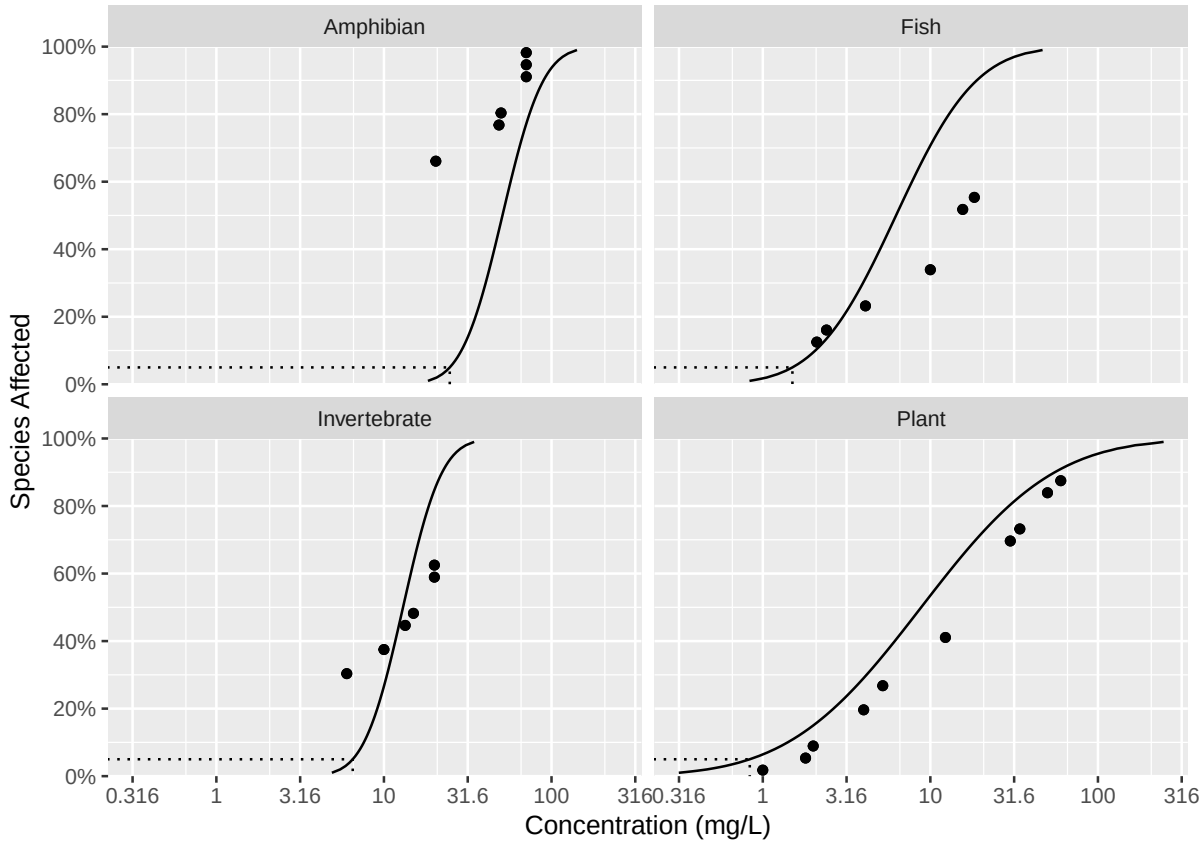
An elegant approach using some tidyverse packages is demonstrated below.

```
library(ssddata)
library(ssdtools)
library(ggplot2)
library(dplyr)
library(tidyr)
library(purrr)

boron_preds <- nest(ccme_boron, data = c(Chemical, Species, Conc, Units)) %>%
  mutate(
    Fit = map(data, ssd_fit_dists, dists = "lnorm"),
    Prediction = map(Fit, predict)
  ) %>%
  unnest(Prediction)
```

The resultant data and predictions can then be plotted as follows.

```
library(ssdtools)
library(ssddata)
```

```
ssd_plot(ccme_boron, boron_preds, xlab = "Concentration (mg/L)", ci = FALSE) +
  facet_wrap(~Group)
```




## Embellishing Plots with an Exposure Distribution

For example, suppose we want to superimpose an environmental concentration cumulative distribution and compute the exposure risk as outlined in Verdonck et al. (2003).

Finding a suitable probability distribution to describe the exposure concentration is beyond the scope of this document – we will assume that this has been done elsewhere. In particular, suppose that the exposure concentration follows a log-normal distribution with a mean of -2.3 and a standard deviation of 1 on the logarithmic scale. From the exposure distribution, we construct a data frame with the concentration values and the cumulative probability of seeing this exposure or less in the environment.

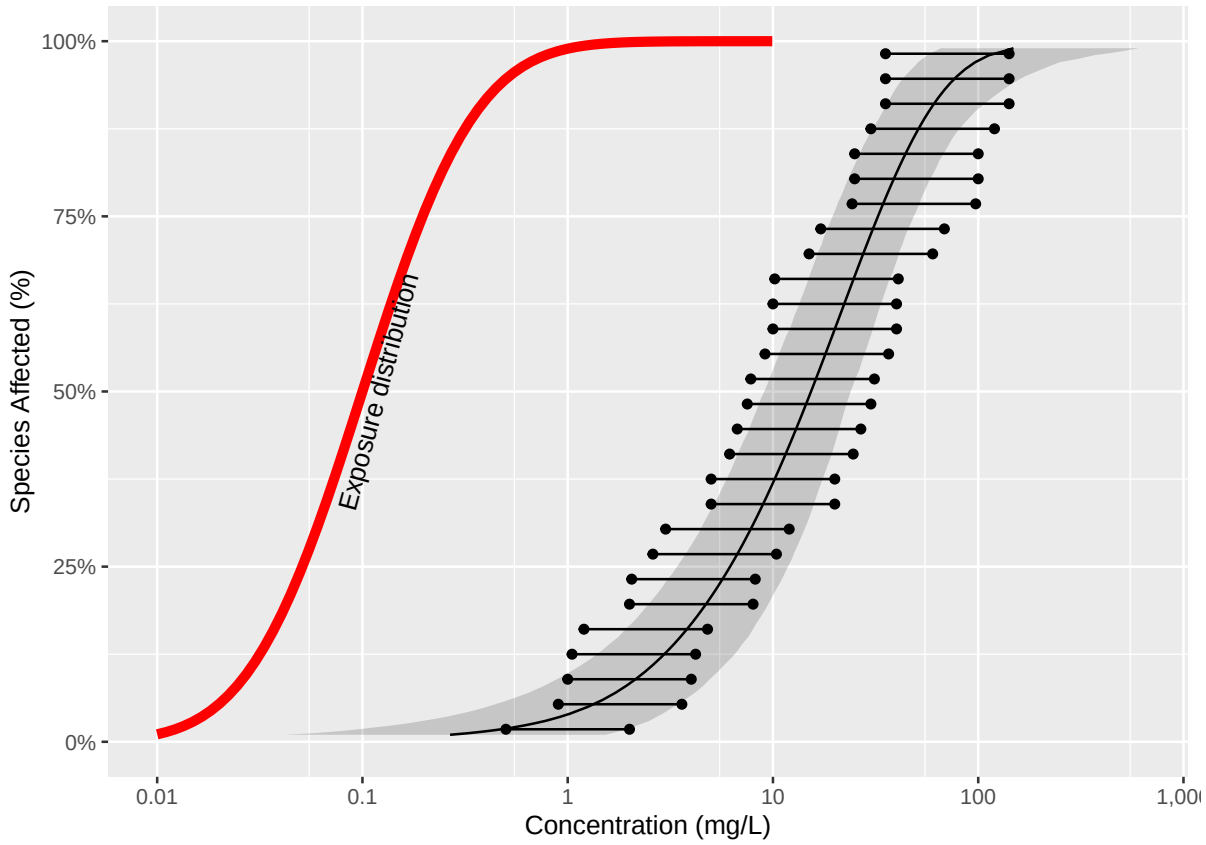
Notice that some care is needed because the ssdtools plot is on the logarithmic base 10 scale and not the natural logarithm base $e$ scale.

```
ex.cdf <- data.frame(Conc = exp(seq(log(.01), log(10), .1))) # generate a grid of concentrations
ex.cdf$ex.cdf <- plnorm(ex.cdf$Conc,
  meanlog = ex.mean.log,
  sdlog = ex.sd.log
) # generate the cdf
```

We now add this to the plot

```
gp +
  geom_line(data = ex.cdf, aes(x = Conc, y = ex.cdf), color = "red", linewidth = 2) +
  annotate("text",
    label = paste("Exposure distribution"),
```

```
    x = 1.08 * ex.cdf$Conc[which.max(ex.cdf$ex.cdf > 0.5)], y = 0.5, angle = 75
  )
```



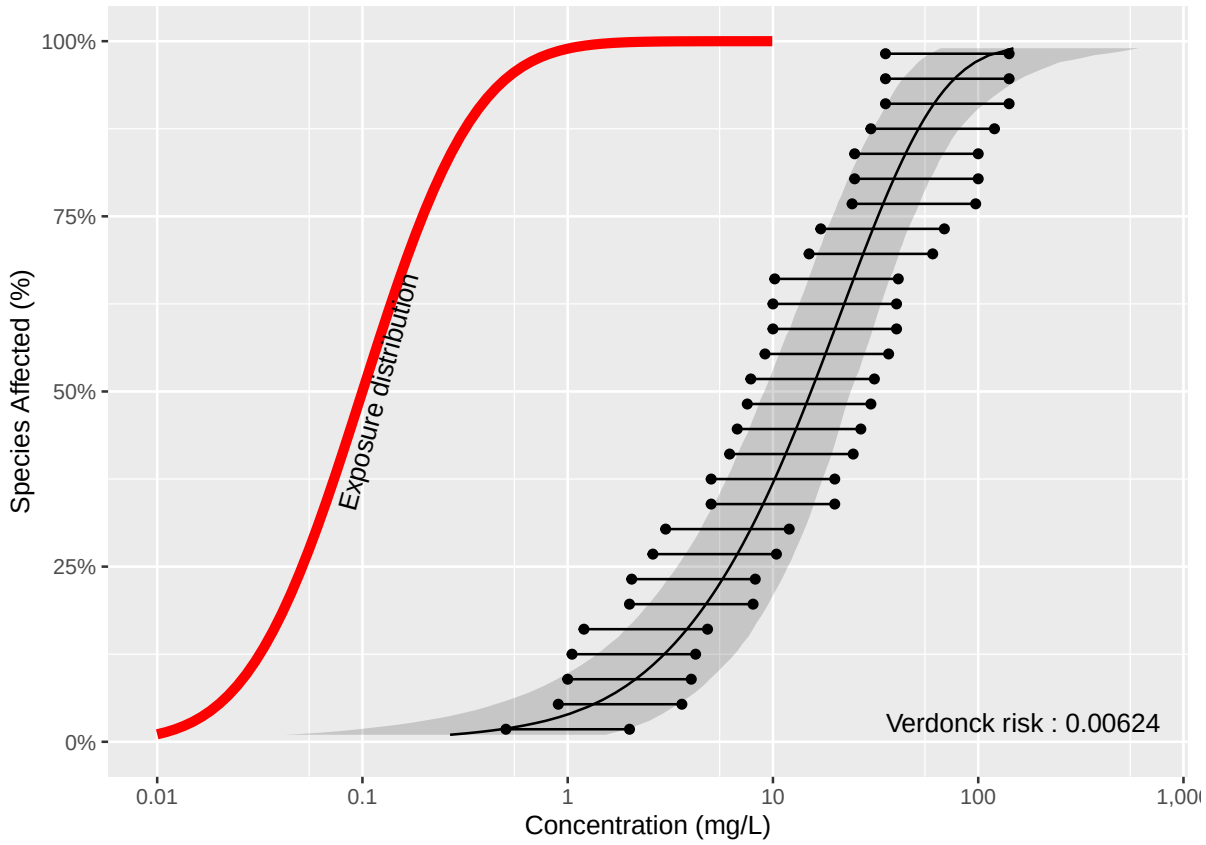

The `ssdtools` package contains a function `ssd_exposure()` that computes the risk as defined by Verdonck et al (2003) representing the average proportion of species at risk.

```
set.seed(99)
ex.risk <- ssd_exposure(fits, meanlog = ex.mean.log, sdlog = ex.sd.log)
ex.risk
```

```
## [1] 0.0062416
```

The risk of 0.00624 can also be added to the plot in the usual way:

```
gp +
  geom_line(dat = ex.cdf, aes(x = Conc, y = ex.cdf), color = "red", linewidth = 2) +
  annotate("text",
    label = paste("Exposure distribution"),
    x = 1.08 * ex.cdf$Conc[which.max(ex.cdf$ex.cdf > 0.5)], y = 0.5, angle = 75
  ) +
  annotate("text",
    label = paste("Verdonck risk :", round(ex.risk, 5)),
    x = Inf, y = 0, hjust = 1.1, vjust = -.5
  )
```




## References

Verdonck FAM, Aldenberg T, Jaworska J, Vanrolleghem PA (2003) Limitations of Current Risk Characterization Methods in Probabilistic Environmental Risk Assessment. Environmental Toxicology and Chemistry 22:2209. https://doi.org/10.1897/02-435

## Licensing

# Appendix M – Additional technical details Vignette

# Additional technical details

sstools Team

2024-05-17

## Small sample bias

The ssdtools package uses the method of Maximum Likelihood (ML) to estimate parameters for each distribution that is fit to the data. Statistical theory says that maximum likelihood estimators are asymptotically unbiased, but does not guarantee performance in small samples. A detailed account of the issue of small sample bias in estimates can be found in the following pdf.

## The inverse Pareto and inverse Weibull as limiting distributions of the Burr Type-III distribution

### Burr III distribution

The probability density function, $f_X(x; b, c, k)$ and cumulative distribution function, $F_X(x; b, c, k)$ for the Burr III distribution (also known as the *Dagum* distribution) as used in `ssdtools` are:

### Inverse Pareto distribution

Let $X \sim Burr(b, c, k)$ have the *pdf* given in the box above. It is well known that the distribution of $Y = \frac{1}{X}$ is the *inverse Burr* distribution (also known as the *SinghMaddala* distribution) for which:

$$f_Y(y; b, c, k) = \frac{c\, k \left(\frac{y}{b}\right)^c}{y\left[1+\left(\frac{y}{b}\right)^c\right]^{k+1}} \quad b, c, k, y > 0$$

$$F_Y(y; b, c, k) = 1 - \frac{1}{\left[1+\left(\frac{y}{b}\right)^c\right]^k} \quad b, c, k, y > 0$$

We now consider the limiting distribution when $c \to \infty$ and $k \to 0$ in such a way that the product $ck$ remains constant, i.e. $ck = \lambda$.

Now,

$$\lim_{\substack{(c,k)\to(\infty,0)\\ ck=\lambda}} \{F_Y(y; b, c, k)\} = 1 - \lim_{\substack{(c,k)\to(\infty,0)\\ ck=\lambda}} \frac{1}{\left[1+\left(\frac{y}{b}\right)^c\right]^k}$$

*and*

$$\lim_{\substack{(c,k)\to(\infty,0)\\ ck=\lambda}} \left[1 + \left(\frac{y}{b}\right)^c\right]^k = \lim_{\substack{(c,k)\to(\infty,0)\\ ck=\lambda}} \left\{ \left(\frac{y}{b}\right)^{ck} \left[1 + \left(\frac{b}{y}\right)^c\right]^k \right\}$$

*and*

$$\begin{aligned}
&\lim_{\substack{(c,k)\to(\infty,0)\\ ck=\lambda}} \left\{ \left(\frac{y}{b}\right)^{ck} \left[1 + \left(\frac{b}{y}\right)^c\right]^k \right\} = \lim_{\substack{(c,k)\to(\infty,0)\\ ck=\lambda}} \left\{ \left(\frac{y}{b}\right)^{ck} \right\} \lim_{\substack{(c,k)\to(\infty,0)\\ ck=\lambda}} \left\{ \left[1 + \left(\frac{b}{y}\right)^c\right]^k \right\} \\
&= \lim_{\substack{(c,k)\to(\infty,0)\\ ck=\lambda}} \left\{ \left(\frac{y}{b}\right)^{ck} \right\} \cdot 1 \\
&= \left(\frac{y}{b}\right)^{\lambda}
\end{aligned}$$

Therefore,

$$\lim_{\substack{(c,k)\to(\infty,0)\\ ck=\lambda}} \{F_Y(y;b,c,k)\} = 1 - \left(\frac{b}{y}\right)^{\lambda} \quad y \geq b$$

which we recognise as the (American) Pareto distribution. So, if the limiting distribution of $Y = \frac{1}{X}$ is a Pareto distribution, then the limiting distribution of $X = \frac{1}{Y}$ is the (American) *inverse Pareto* distribution:

$$f_X(x;\alpha,\beta) = \lambda b^{\lambda} x^{\lambda-1}; 0 \leq x \leq \tfrac{1}{b}; \lambda, b > 0$$
$$F_X(x;\alpha,\beta) = (xb)^{\lambda}; 0 \leq x \leq \tfrac{1}{b}; \lambda, b > 0$$

For completeness, the MLEs of this distribution have closed-form expressions and are given by:

$$\hat{\lambda} = \left[\ln\left(\frac{g_X}{\hat{b}}\right)\right]^{-1}$$
$$\hat{b} = \frac{1}{\max\{X_i\}}$$

and $g_X$is the *geometric mean* of the data.

**Inverse Weibull distribution**

Let $X \sim Burr(b,c,k)$ have the *pdf* given in the box above. We make the transformation

$$Y = \frac{b k^{\frac{1}{c}} \theta}{X}$$

where $\theta$ is a parameter (constant). The distribution of $Y$ is also a Burr distribution and has *cdf*

$$G_Y(y) = 1 - \frac{1}{\left[1 + \left(\frac{y}{k^{\frac{1}{c}}\theta}\right)^{c}\right]^{k}}$$

.We are interested in the limiting behaviour of this Burr distribution as $k \to \infty$. Now,

$$\lim_{k\to\infty} G_Y(y) = 1 - \lim_{k\to\infty}\left[1 + \left(\frac{y}{k^{\frac{1}{c}}\theta}\right)^{c}\right]^{-k}$$

$$= 1 - \lim_{k\to\infty}\left[1 + \frac{\left(\frac{y}{\theta}\right)^{c}}{k}\right]^{-k}$$

$$= 1 - \exp\left[-\left(\tfrac{y}{\theta}\right)^{c}\right]$$
$$\left\{\text{using the fact that } \lim_{n\to\infty}\left(1 + {}^{z}\!/_{n}\right)^{-n} = e^{-z}\right\}$$

We recognise the last expression as the *cdf* of a Weibull distribution with parameters $c$ and $\theta$.

## Licensing