Calculating the Area Under the Concentration-Response Curve with tcplfit2

Center for Computational Toxicology and Exposure

Introduction

This vignette walks through how to estimate the area under the curve (AUC) for concentration-response curves with `tcplfit2`, with various applications. The AUC can be interpreted as a measure of overall efficacy/potency, which users may want to include as part of their analyses.

Main Feature

The `concRespCore` function has a logical argument `AUC` controlling whether the area under the curve (AUC) is calculated for the winning model and returned alongside the other modeling results (e.g. model parameters and hit-call details). This argument defaults to `TRUE`, such that the AUC is always included in the output unless otherwise specified (i.e. `AUC=FALSE`).

```
# some example data
conc <- list(.03, .1, .3, 1, 3, 10, 30, 100)
resp <- list(0, .2, .1, .4, .7, .9, .6, 1.2)
row <- list(conc = conc,
            resp = resp,
            bmed = 0,
            cutoff = 1,
            onesd = .5)

# AUC is included in the output
concRespCore(row, conthits = TRUE)
#>      n_gt_cutoff cutoff fit_method top_over_cutoff      rmse  a  b       tp
```

```
#> cnst           1      1       hill        1.225599 0.1750312 NA NA 1.225599
#>              p  q       ga la        er     bmr     bmdl     bmdu       caikwt
#> cnst 0.7752844 NA 2.554272 NA -2.467853 0.6745 2.341811 4.822553 6.064845e-05
#>           mll   hitcall     ac50 ac50_loss      top        ac5      ac10
#> cnst 4.492301 0.9650614 2.554272        NA 1.225599 0.05726215 0.1501198
#>           ac20      acc    ac1sd      bmd                        conc
#> cnst 0.4272681 17.43267 1.580024 3.314775 0.03|0.1|0.3|1|3|10|30|100
#>                            resp errfun      AUC
#> cnst 0|0.2|0.1|0.4|0.7|0.9|0.6|1.2    dt4 1.969363
```

The following sections demonstrate how to estimate the AUC when curve fitting is performed with `concRespCore` as well as separate calls to `tcplfit2_core` and `tcplhit2_core`. Additionally, we provide several types of potential curve fits with the resulting AUC and how to interpret it.

Applications

Positive Curves

This section provides an example of how to use `get_AUC` function in `tcplfit2` to calculate area under the curves (AUC) for a given concentration-response curve. First, we need some curves obtained from curve-fitting on with a few data examples.

```
# This is taken from the example under tcplfit2_core
conc_ex2 <- c(.03, .1, .3, 1, 3, 10, 30, 100)
resp_ex2 <- c(0, .1, 0, .2, .6, .9, 1.1, 1)

# fit all available models in the package
# use do.plot = TRUE to show all fitted curves
oldpar <- par(no.readonly = TRUE)
on.exit(par(oldpar))
par(xpd = TRUE)
output_ex2 <- tcplfit2_core(conc_ex2, resp_ex2, .8, do.plot = TRUE)
```

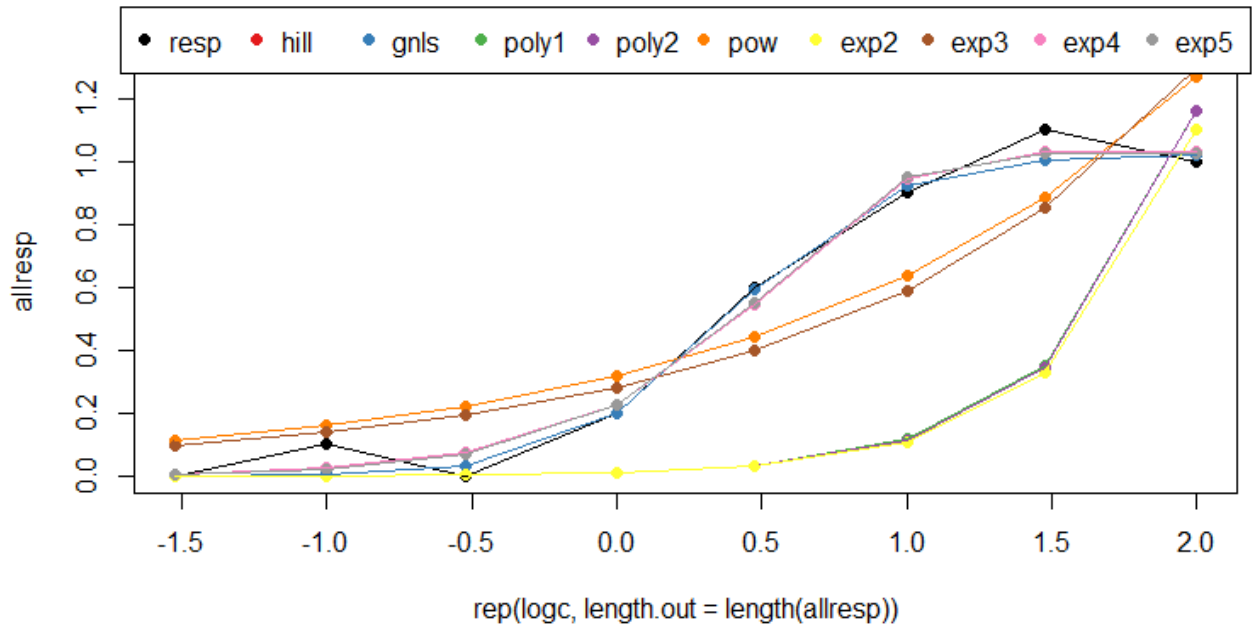

Figure 1: This figure depicts all fit concentration-response curves. The models are polynomial 1 and 2, power, hill, gain-loss, and exponential 2 to exponential 5.

We can use the `get_AUC` function to calculate the AUC for a single model. Inputs to this function are: the name of the model, lower and upper concentration bounds (usually the lowest and the highest concentrations in the data, respectively), and the estimated model parameters. The code chunk below demonstrate how to calculate AUC for the hill model, starting with extracting information from `tcplfit2_core` output then inputting this information into the `get_AUC` function. After estimating the AUC, we plot the hill curve and shade the corresponding area under the curve.

```
fit_method <- "hill"
# extract the parameters
modpars <- output_ex2[[fit_method]][output_ex2[[fit_method]]$pars]

# plug into get_AUC function
# for hill and gnls, no need to convert concentration used for bounds to log-scale
# they will be converted inside the function
estimated_auc1 <- get_AUC(fit_method, min(conc_ex2), max(conc_ex2), modpars)
estimated_auc1
#> [1] 1.64823

# extract the predicted responses from the model
pred_resp <- output_ex2[[fit_method]][["modl"]]

# plot to see if the result make sense
# the shaded area is what the function tries to find
plot(log10(conc_ex2), pred_resp)
```

```
lines(log10(conc_ex2), pred_resp)
polygon(c(log10(conc_ex2), max(log10(conc_ex2))), c(pred_resp, min(pred_resp)), col=rgb(1,
        0, 0,0.5))
```

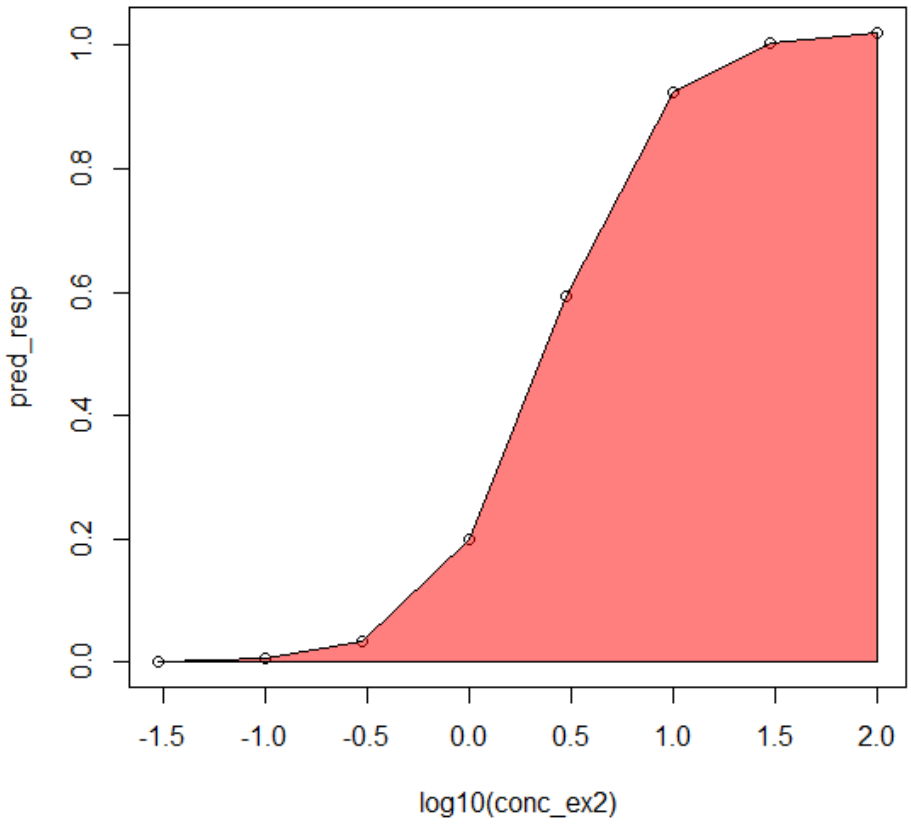

Figure 2: The red region is the area under the fitted hill curve. The AUC estimated with `get_AUC` is 1.64823. This estimate seems to align with the area of the shaded region.

We can also calculate the AUC for other models too (with exception of the constant model).

```
# hill and gnls will return a much smaller number because they are in log10-scale
fitmodels = c("gnls", "poly1", "poly2", "pow", "exp2", "exp3", "exp4", "exp5")
mylist <- list()
for (model in fitmodels){

  fit_method <- model
  # extract corresponding model parameters
  modpars <- output_ex2[[fit_method]][output_ex2[[fit_method]]$pars]

  # get AUC
  mylist[[fit_method]] <- get_AUC(fit_method, min(conc_ex2), max(conc_ex2), modpars)

}
```

```
# print AUC's for other models
data.frame(mylist,row.names = "AUC")
#>         gnls    poly1    poly2      pow     exp2     exp3     exp4     exp5
#> AUC 1.64823 58.09263 57.89675 97.43487 55.01408 96.18956 98.80625 98.65734
```

Negative curves

This section demonstrates the behavior of the `get_AUC` function with negative curves. Here, we use some example data from example 3 in the tcplfit2 vignette.

```
# Taking the code from example 3 in the vignette
library(stringr)  # string management package
#> Warning: package 'stringr' was built under R version 4.2.3
data("signatures")

# use row 5 in the data
conc=as.numeric(str_split(signatures[5,"conc"],"\\|")[[1]])
resp=as.numeric(str_split(signatures[5,"resp"],"\\|")[[1]])
cutoff=signatures[5,"cutoff"]

# plot all models, this is an example of negative curves
oldpar <- par(no.readonly = TRUE)
on.exit(par(oldpar))
par(xpd = TRUE)
output_negative <- tcplfit2_core(conc, resp, cutoff, do.plot = TRUE)
```

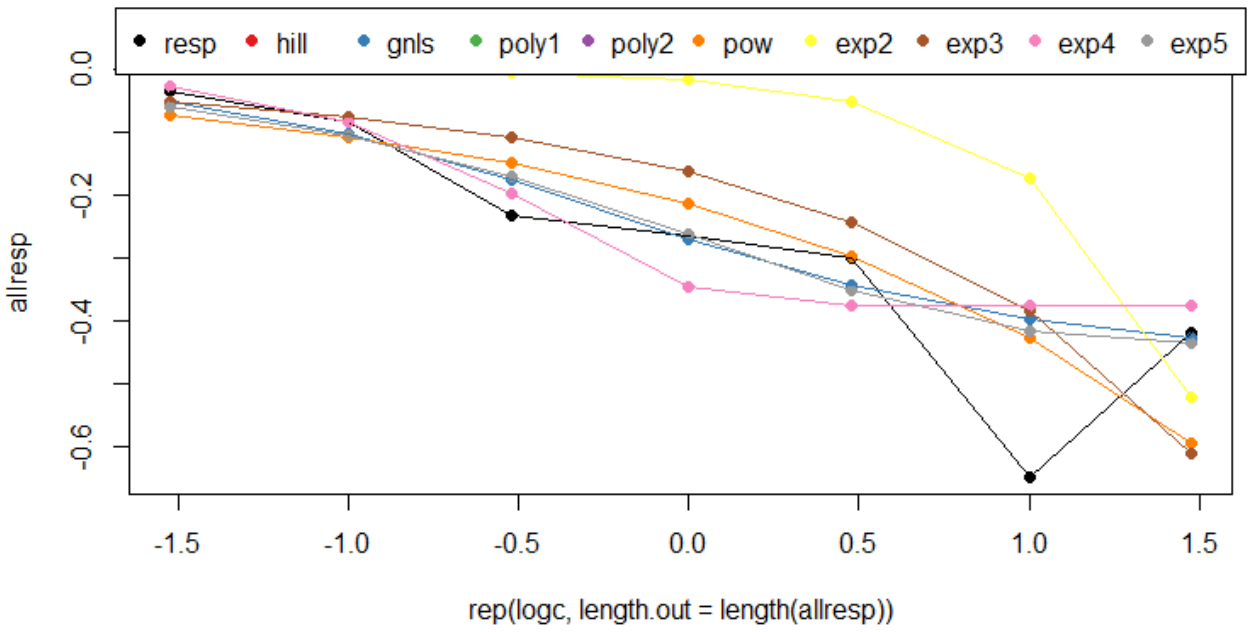

Figure 3: This plot depicts all fitted concentration-response curves. All curves are decreasing starting from 0 and are below x-axis.

The code chunk below calculates the AUC for exponential 3 model with the `get_AUC` function.

```
fit_method <- "exp3"

# extract corresponding model parameters and predicted response
modpars <- output_negative[[fit_method]][output_negative[[fit_method]]$pars]
pred_resp <- output_negative[[fit_method]][["modl"]]

estimated_auc2 <- get_AUC(fit_method, min(conc), max(conc), modpars)
estimated_auc2
#> [1] -12.92738

# plot this curve
pred_resp <- pred_resp[order(conc)]
plot(conc[order(conc)], pred_resp)
lines(conc[order(conc)], pred_resp)
polygon(c(conc[order(conc)], max(conc)), c(pred_resp, max(pred_resp)), col=rgb(1, 0,
        0,0.5))
```

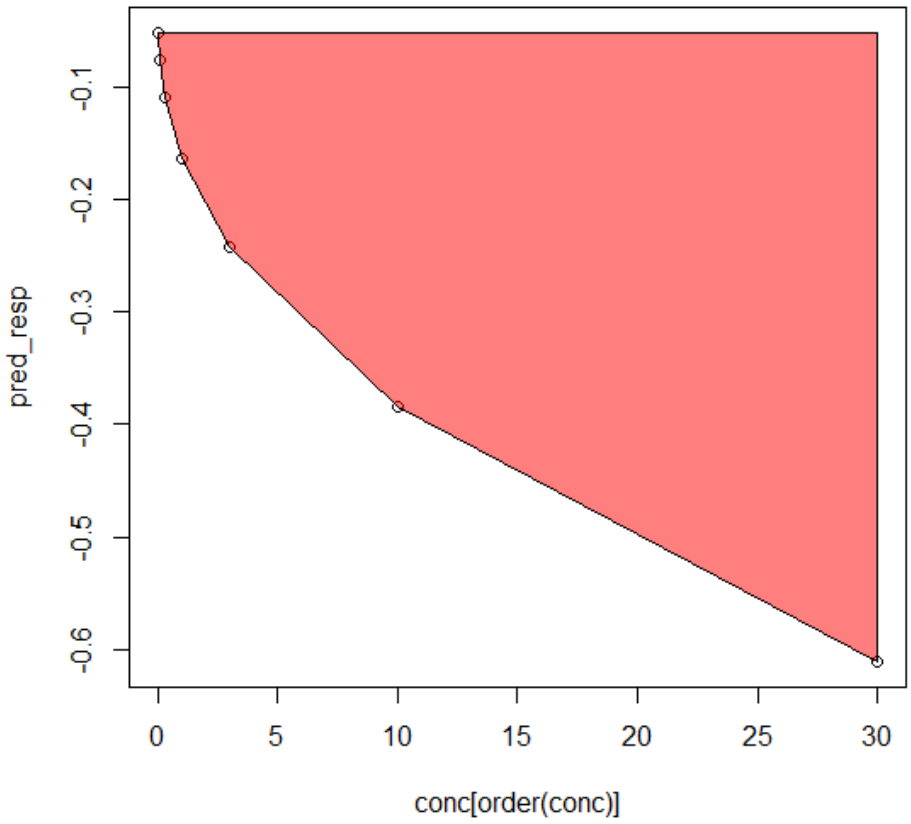

Figure 4: Notice the function returns a negative AUC value, -12.92738. The absolute value, 12.92738, seems to align with the area between the curve and x-axis. Note: The x-axis in this plot is in the original units, so it looks different from the curve in figure 3, which has the x-axis in log_{10} units.

As demonstrated, when integrating over a curve in the negative direction, the function will return a negative AUC value. However, some users may want to consider all "areas" as positive values. For this reason, we added the `return.abs = TRUE` argument in `get_AUC` to convert negative AUC values to positive values when returned. This argument is by default `FALSE`.

```
get_AUC(fit_method, min(conc), max(conc), modpars, return.abs = TRUE)
#> [1] 12.92738
```

Bi-phasic Curves

Currently, none of the models in `tcplfit2` are capable of fitting bi-phasic curves. However, this section demonstrates what happens if a user did want to estimate the AUC for a bi-phasic curve.

The following chunk simulates a polynomial 2 curve that has area below and above the x-axis. Please note, the polynomial 2 model implemented in this package is re-parameterized such that the baseline response is always assumed to be 0.

The Polynomial 2 model in `tcplfit2` is implemented as $a * (\frac{x}{b} + \frac{x^2}{b^2})$. Here, we use $a = 1$ and $b = (-2)$ to simulate a bi-phasic curve, which can be represented in the typical form as $\frac{1}{4}x^2 - \frac{1}{2}x$.

```
# simulate a poly2 curve
ps <- unlist(list(a = 1, b = -2))
conc_sim <- c(0, 0.03, 0.10, 0.30, 1.00, 2.00, 2.50, 3.00, 3.20)
resp_sim <- poly2(ps, conc_sim)

# plot the curve
plot(conc_sim, resp_sim)
lines(conc_sim, resp_sim)
abline(h = 0)
polygon(conc_sim[1:6], resp_sim[1:6], col=rgb(1, 0, 0,0.5))
polygon(c(conc_sim[6:9], 3.20), c(resp_sim[6:9], 0), col=rgb(0, 0, 1,0.5))
```

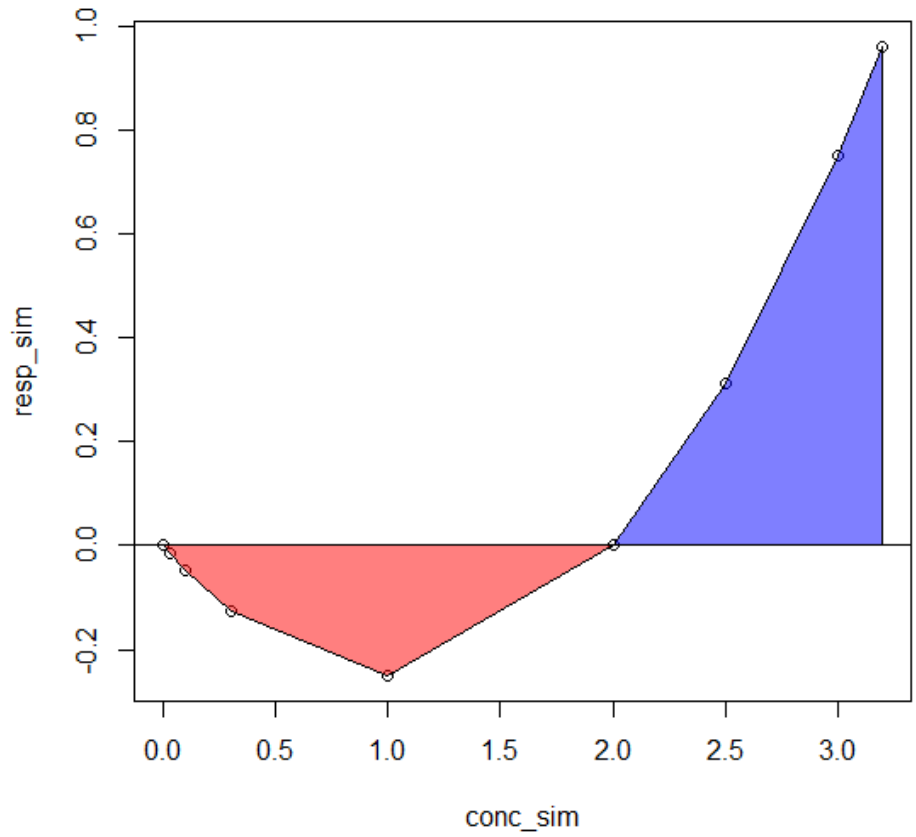

Figure 5: This plot illustrates the simulated bi-phasic polynomial 2 curve. The curve initially decreases, then increases and crosses the x-axis at `x = 2` (x-intercept).

```
# get AUC for the simulated Polynomial 2 curve
get_AUC("poly2", min(conc_sim), max(conc_sim), ps)
#> [1] 0.1706667
```

Integrating over a curve with area above and below the x-axis, the current function returns the difference between the total area above the x-axis and the total area below the x-axis (i.e. blue region minus red region). In this example, area above the x-axis is slightly larger than the area below the x-axis so the function returned a positive difference.

Since the AUC is used as an indicator of overall effect size in a toxicological context, then it makes more sense to report the total area of the above and below x-axis (i.e., blue region plus red region) for bi-phasic curves. To do that, we need to calculate each area separately and take the sum of those areas. The code below demonstrate how this is done for the previously simulated polynomial 2 curve. It should be noted, one needs to identify where the x-intercept(s) occur in order to calculate each area. In this example, we know the x-intercept is at `x = 2`.

```
# calculate area of each section separately and then sum them together
x_intercept <- 2
total_auc <- abs(get_AUC("poly2", min(conc_sim), x_intercept, ps)) + get_AUC("poly2",
         x_intercept, max(conc_sim), ps)
total_auc
#> [1] 0.8373333
```

The total area of red region plus blue region is 0.8373333.

AUC after hit-calling

In some cases, users may want to run the `tcplfit2_core` and `tcplhit2_core` functions separately, and only obtain the AUC for the winning model from `tcplhit2_core`. Thus, `tcplfit2` also includes a wrapper function for `get_AUC`, called `post_hit_AUC`, which allows users to estimate the AUC for the winning model only.

`tcplhit2_core` provides output in a data frame format with a single row containing the concentration-response data, the winning model name along with the fitted parameter values, and hit-calling results. The code chunk below provides an example demonstrating how to use the wrapper function `post_hit_AUC`. Internally, the wrapper function extracts information from the one-row data frame output and pass it to `get_AUC`, which calculates the AUC. Thus, manual entry of the model name, parameters values, etc. into `get_AUC` is not necessary with `post_hit_AUC`.

The winning model from the positive curve example is the Hill model. Comparing the AUC from the previous example and the AUC returned from the `post_hit_AUC` here should be identical, i.e. 1.64823.

```
out <- tcplhit2_core(output_ex2, conc_ex2, resp_ex2, 0.8, onesd = 0.4)
out
#>      n_gt_cutoff cutoff fit_method top_over_cutoff       rmse  a  b       tp
#> cnst           3    0.8       hill        1.279049 0.05022924 NA NA 1.023239
#>             p  q       ga la        er     bmr     bmdl     bmdu       caikwt
#> cnst 1.592714 NA 2.453014 NA -3.295307 0.5396 2.33877 2.979982 1.446965e-08
#>           mll   hitcall     ac50 ac50_loss      top       ac5      ac10
#> cnst 12.71495 0.9999999 2.453014        NA 1.023239 0.3862094 0.6174049
#>          ac20      acc    ac1sd      bmd                         conc
#> cnst 1.027285 5.466819 1.856853 2.627574 0.03|0.1|0.3|1|3|10|30|100
#>                          resp errfun
#> cnst 0|0.1|0|0.2|0.6|0.9|1.1|1    dt4
post_hit_AUC(out)
#> [1] 1.64823
```