get_AUC Demonstration and Test Codes

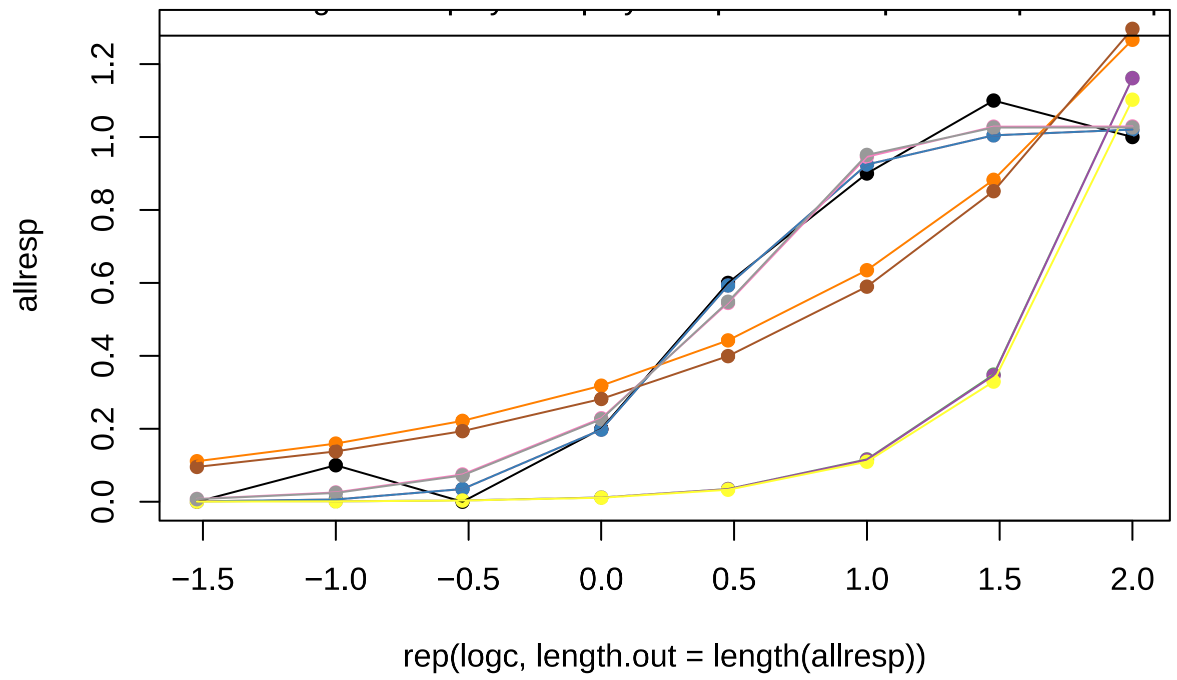
2023-08-18

Demonstration

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

# use verbose to show the winning model and plot all models
output <- tcplfit2_core(conc, resp, .8, verbose = TRUE, do.plot = TRUE)
```

```
## hill >>> 367 0
## gnls >>> 373 0
## poly1 >>> 103 0
## poly2 >>> 0 0
## power >>> 606 0
## Exp2 >>> 0 0
## Exp3 >>> 463 0
## Exp4 >>> 184 0
## Exp5 >>> 521 0
## [1] "aic values:"
##       cnst       hill       gnls      poly1      poly2        pow       exp2
##  18.672517 -17.429907 -13.429907  14.008619  16.032201   3.749644  16.053970
##       exp3       exp4       exp5
##   6.805700 -16.646138 -14.655030
## Winner:  hill
```



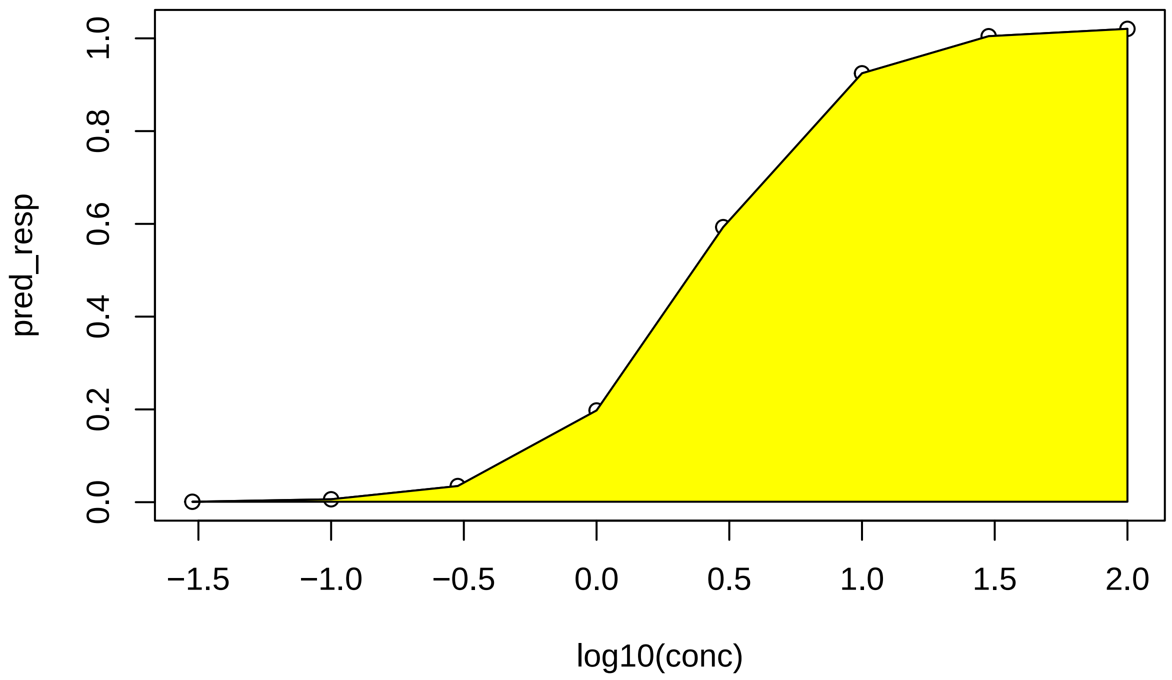
Let say we want to get AUC for the winning model, in this case, hill. We can do that with get_AUC() with inputs: the name of the model, lower and upper bounds (of the area), and estimated model parameters. In the following codes I will use the function and then plot to check if the result make sense.

```r
fit_method <- "hill"
# extract the parameters
modpars <- output[[fit_method]][output[[fit_method]]$pars]
# extract the predicted responses from the model
pred_resp <- output[[fit_method]][["modl"]]

# plug in the function
# for hill and gnls, no need to convert conc to log-scale it will do so inside the function
get_AUC(fit_method, min(conc), max(conc), modpars)
```

```
## [1] 1.64823
```

```r
# plot to see if the result make sense
# the shaped area is what the function tries to find
plot(log10(conc), pred_resp)
lines(log10(conc), pred_resp)
polygon(c(log10(conc), max(log10(conc))), c(pred_resp, min(pred_resp)), col="yellow")
```



We can do the same for other models too (excluding constant model).

```r
fitmodels = c("poly1", "poly2", "pow", "exp2", "exp3", "exp4", "exp5")

for (model in fitmodels){

  fit_method <- model
  # extract corresponding model parameters
  # these results for all model fitted are available in the result from tcplfit2_core
  modpars <- output[[fit_method]][output[[fit_method]]$pars]

  # predicted response can also be extract from results
  pred_resp <- output[[fit_method]][["modl"]]

  # get AUC
  print(get_AUC(fit_method, min(conc), max(conc), modpars))

  #plot(conc, pred_resp)
  #lines(conc, pred_resp)
  #polygon(c(conc, max(conc)),c(pred_resp, min(pred_resp)), col="yellow")

}
```

```
## [1] 58.09263
## [1] 57.89675
## [1] 97.43487
## [1] 55.01408
## [1] 96.18956
## [1] 98.80625
## [1] 98.65734
```

Examine how this function behave with negative curves

Taking example 3 from the vignette which gives data for negative curves and curves that have area both above and below 0.

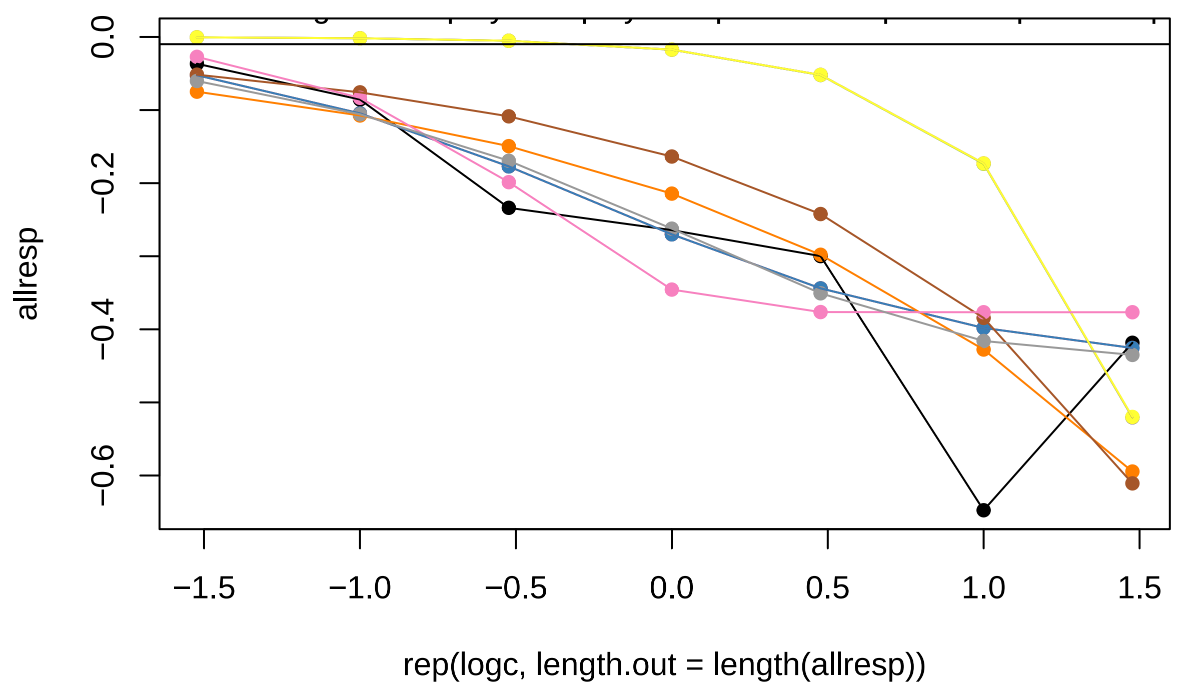
```
# Taking the code from example 3 in the vignette
library(stringr)  # string management package
```

```
## 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
output_negative <- tcplfit2_core(conc, resp, cutoff, do.plot = TRUE)
```



Take one of the model to check - exp3

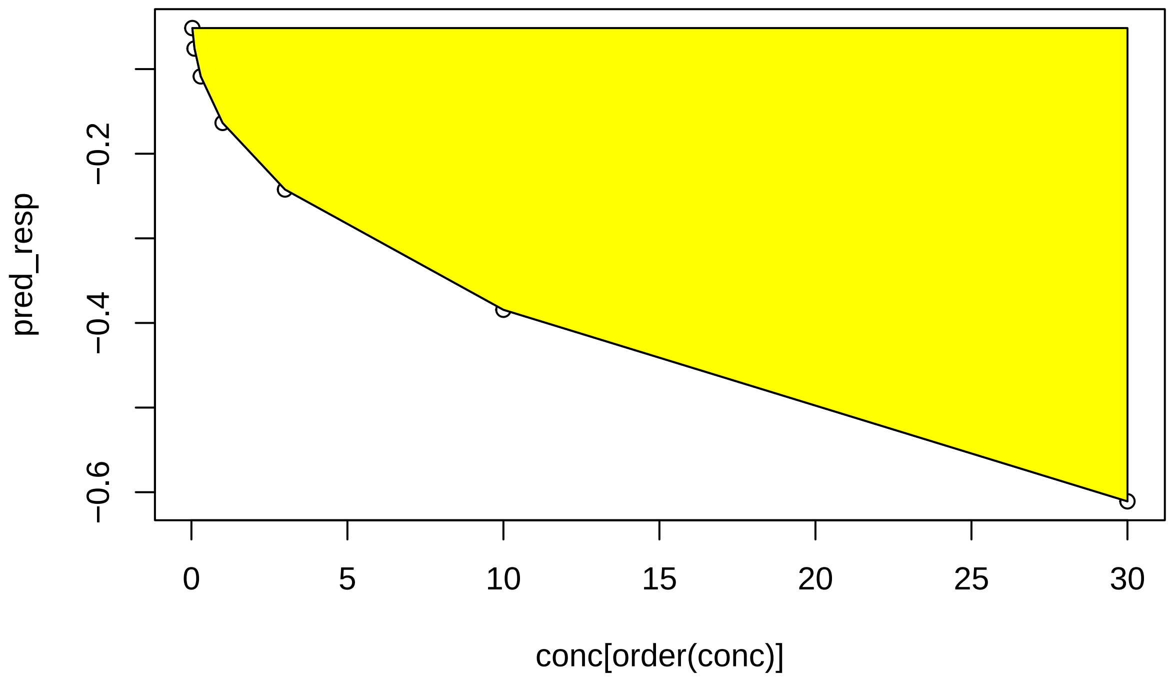
```r
fit_method <- "exp3"

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


get_AUC(fit_method, min(conc), max(conc), modpars) # returns -12.92
```

```
## [1] -12.92738
```

```r
# 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="yellow")
```

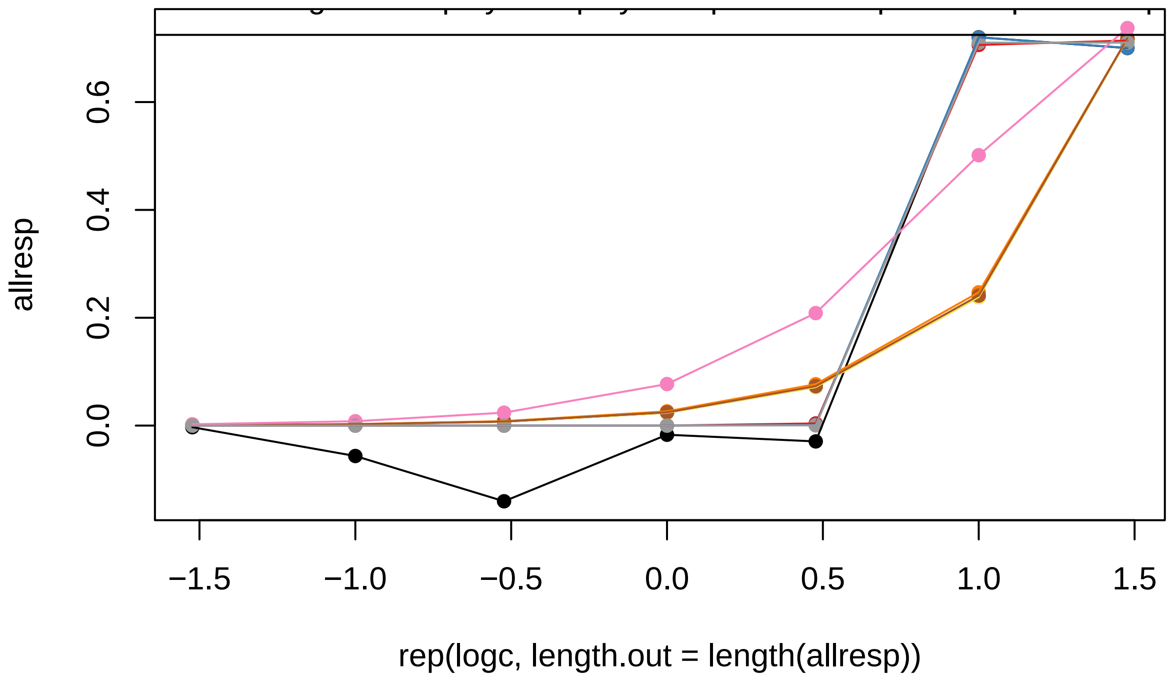


Things to consider: The x-axis in plot above is in regular unit so it looks different from the figure above which uses log-conc for x-axis. The absolute value 12.92 seems to be a legit estimate for the area above the curve (technically can't call it AUC anymore), but it's returned as a negative number. Need more consideration of how to work with sign.

Then use another example to check what happen if a curve have area below and above x-axis.

```r
conc=as.numeric(str_split(signatures[2,"conc"],"\\|")[[1]])
resp=as.numeric(str_split(signatures[2,"resp"],"\\|")[[1]])
cutoff=signatures[2,"cutoff"]

# plot all models, this is an example of negative curves
output_biphasic <- tcplfit2_core(conc, resp, cutoff, do.plot = TRUE)
```



Compare results

Compare the result of my function to the result from Katie's code

```r
conc <- c(.03, .1, .3, 1, 3, 10, 30, 100)

# Katie's code
gnls_curve <- function(top, ga, gw, la, lw, lconc){
    gain <- 1/(1+10^((ga - lconc)*gw))
    loss <- 1/(1+10^((lconc - la)*lw))
    return(top*gain*loss)
}

mapply(function(lower,
upper,
top,
ga,
gw,
la,
lw) integrate(gnls_curve,
lower,
upper,
top=top,
ga=ga,
gw=gw,
la=la,
lw=lw)$value,
lower = min(log10(conc)),
upper = max(log10(conc)),
top = 1.023238,
ga = log10(2.453007),
gw = 1.592714,
la = log10(4288.993065),
lw = 5.770323)
```

```
## [1] 1.64823
```

```
# my function
ps <- list(tp = 1.023238, ga = 2.453007, p = 1.592714, la = 4288.993065,
           q = 5.770323, er = -3.295309 )
get_AUC("gnls", min(conc), max(conc), ps)
```

```
## [1] 1.64823
```