## Evaluating Statistical Bounding Methods for Semi-Quantitative Non-Targeted Analysis Using ENTACT Data (R Markdown)

Louis C. Groff II

2/25/2021

This markdown document shows the statistical methods performed for the ENTACT Mixtures LC Semi-Quant manuscript.

Load necessary libraries and shift directories as needed:

library(ggplot2)  
library(readxl)  
library(xlsx)  
library(investr)  
library(data.table)  
library(gridExtra)  
datadir <- 'C:/Users/lgroff/OneDrive - Environmental Protection Agency (EPA)/Profile/Documents/Data/ENTACT Semi Quant/Standard Mixtures/JRS Tracer Analysis'

Load In ESI+ data:

posdata\_1 <- read\_xlsx(paste0(datadir,'/Supp\_Table1\_Pos.xlsx'))  
posdata\_2 <- read\_xlsx(paste0(datadir,'/Supp\_Table2\_Pos.xlsx'))

## Method 1. Inverse Prediction Using Calibration Curves:

See Section 3.2.1 in manuscript

1. Preallocate empty arrays and lists:

chemunique\_pos <- unique(posdata\_1$New\_ID) #unique chemical ID list  
posdata\_2$y0 <- NA  
posdata\_2$Known\_Conc <- NA  
posdata\_2$Conc\_CC <- NA  
posdata\_2$Conc\_CC\_Upper <- NA  
lmlist\_pos <- vector('list',length(chemunique\_pos))  
plotlist\_pos <- lmlist\_pos  
predlist\_pos <- lmlist\_pos

1. set RNG seed and determine row indices to select y0 for each individual chemical (calculated as a random number between 0-1 \* length of the individual chemical data subset):

set.seed(16384)  
tmpidx\_pos<-sample(1:100,nrow(posdata\_2),size=nrow(posdata\_2))/100

1. Run cal. curve for loop for each unique chemical (including isomers)
2. Store RNG-selected observed intensity (y0) and concentration at y0, store in posdata\_2.
3. calculate linear model on log(Normalized\_Intensity) vs. log(Conc.), store in lmlist\_pos.
4. calculate Conc\_CC (calibration curve estimate) at y0 from regression coefficients, stored in posdata\_2.
5. calculate 99% prediction interval (99% PI) for the regression (need n > 3 data points), store in predlist\_pos.
6. Use calibrate() function within investr package to calculate the upper concentration bound from the 99% PI (need n > 3 data points), encapsulated in a try() statement since it fails for some compounds, but don’t want to break the loop on a failure, store in posdata\_2.
7. store ggplots of each calibration curve in plotlist\_pos:

for (i in 1:length(chemunique\_pos)){  
 tmpdf <- posdata\_1[posdata\_1$New\_ID==chemunique\_pos[[i]],]  
 tmpdf <- tmpdf[complete.cases(tmpdf$Normalized\_Intensity),] #remove NAs  
   
 posdata\_2$y0[[i]] <- tmpdf[[ceiling(tmpidx\_pos[[i]]\*nrow(tmpdf)),'Normalized\_Intensity']]  
 posdata\_2$Known\_Conc[[i]] <- tmpdf[[ceiling(tmpidx\_pos[[i]]\*nrow(tmpdf)),'Concentration']]  
   
 #store regression model objects:  
 lmlist\_pos[[i]] <- lm(tmpdf,formula = log10(Normalized\_Intensity)~log10(Concentration))  
   
 #store concentration predictions for individual chemical cal curves:  
 posdata\_2$Conc\_CC[[i]] <- 10^((log10(posdata\_2$y0[[i]])-lmlist\_pos[[i]]$coefficients[1])/lmlist\_pos[[i]]$coefficients[2])  
   
 #store 99% prediction interval data for each chemical cal curve:  
 predlist\_pos[[i]] <- try(predict(lmlist\_pos[[i]],  
 level=0.99,  
 interval='prediction'), silent=T)  
 #run calibrate on n > 3 intensities to get 99% upper bound:  
 tmpgg <- ggplot(tmpdf,aes(x=log10(Concentration),y=log10(Normalized\_Intensity)))+  
 geom\_point()+  
 geom\_smooth(method='lm',formula=y~x,se=FALSE)  
 labs(x='Log10 Concentration',  
 y= 'Log10 Normalized Intensity',  
 title = unique(tmpdf$Preferred\_Name))  
 if (length(tmpdf$Intensity) > 3){  
 tmpcal<-try(calibrate(lmlist\_pos[[i]],  
 y0=log10(posdata\_2$y0[[i]]),  
 level=0.99,  
 interval='inversion'),silent=T)  
 #store individual regression plots in a list:  
 tmpgg <- tmpgg+  
 geom\_ribbon(aes(ymin=predlist\_pos[[i]][,'lwr'],  
 ymax=predlist\_pos[[i]][,'upr']),  
 alpha=0.25)  
 if (class(tmpcal) != 'try-error'){  
 posdata\_2$Conc\_CC\_Upper[[i]] <- 10^tmpcal$upper  
 }  
 }  
 plotlist\_pos[[i]] <- tmpgg  
}

## Warning in predict.lm(lmlist\_pos[[i]], level = 0.99, interval = "prediction"): predictions on current data refer to \_future\_ responses

(551 Occurrences)

From the above calibration curve regressions, if regression slope is zero, set intercept to NA since that leads to underestimation of concentration estimate from RF=10^intercept, store RF = y0/concentration instead:

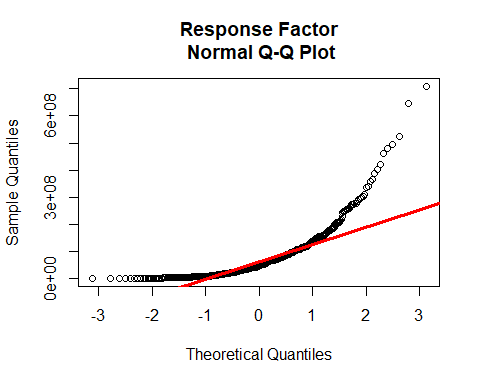
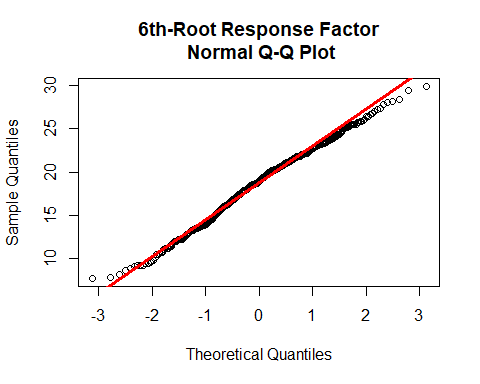
for(i in 1:length(posdata\_2$Slope\_Full)){  
 if(posdata\_2$Slope\_Full[[i]]==0){  
 posdata\_2$Intercept\_Full[[i]] <- NA  
 posdata\_2$RF[[i]] <- posdata\_2$y0[[i]]/posdata\_2$Known\_Conc[[i]]  
 }   
}

Store error quotients for calibration curve analysis in posdata\_2:

#Conc\_0.99\_CC:  
posdata\_2$Err\_CCUppervsCCEst <- posdata\_2$Conc\_CC\_Upper/posdata\_2$Conc\_CC

## Method 2: Inverse Prediction Using a Bounded Response Factor

See section 3.2.2. in manuscript. Show quantile-quantile plots for RF distribution and RF(1/6)

Store tRF = RF(1/6) in posdata\_2, and calculate tRF\_0.01 and tRF\_0.99 using Eq. 1 in manuscript.

alpha = 0.01 #0.01 for 99% pred. interval  
level = 1-alpha/2  
posdata\_2$RF\_6thRoot <- posdata\_2$RF^(1/6) #6th-root transformation  
  
#Upper prediction interval bound for a Normal 1D Distribution:  
pos\_RF6th\_upper<- mean(posdata\_2$RF\_6thRoot,na.rm=T)+qt(level,length(posdata\_2$RF\_6thRoot)-1)\*sd(posdata\_2$RF\_6thRoot,na.rm=T)\*sqrt(1+1/length(posdata\_2$RF\_6thRoot))  
  
#Lower prediction interval bound for a Normal 1D Distribution:  
pos\_RF6th\_lower<- mean(posdata\_2$RF\_6thRoot,na.rm=T)-qt(level,length(posdata\_2$RF\_6thRoot)-1)\*sd(posdata\_2$RF\_6thRoot,na.rm=T)\*sqrt(1+1/length(posdata\_2$RF\_6thRoot))

Since concentration is inversely proportional to response factor (C=I/RF), use lower prediction interval bound (tRF\_0.01) to calculate upper bound concentration predictions (Conc\_0.99\_RF). Store predictions and error quotients in posdata\_2:

#Conc\_0.99\_RF:  
posdata\_2$Conc\_DefaultRF\_Upper <- posdata\_2$y0/pos\_RF6th\_lower^6  
  
#Error Quotients:  
posdata\_2$Err\_RFUppervsCCEst<- posdata\_2$Conc\_DefaultRF\_Upper/posdata\_2$Conc\_CC

## Method 3: Inverse Prediction Using Ionization Efficiency Estimation

See section 3.2.3 in manuscript. Load in data from Kruve, et al.:

datadir2<-'C:/Users/lgroff/OneDrive - Environmental Protection Agency (EPA)/Profile/Documents/Data/ENTACT Semi Quant/Standard Mixtures/Anneli Updates'  
posdata\_anneli <- read\_xlsx(paste0(datadir2,'/ENTACT\_pos\_pred.xlsx'))

Pre-allocate empty columns, filter updated log(IE) predictions based on DTXSID matches:

posdata\_2$logIE\_Pred <- NA  
posdata\_anneli <- posdata\_anneli[posdata\_anneli$DTXSID %in% posdata\_2$DTXSID,]  
for (i in 1:length(posdata\_2$New\_ID)){  
 #If length of matched filtered subset is > 0, store the unique log(IE)  
 #value in posdata\_2.   
 if (sum(posdata\_anneli$DTXSID == posdata\_2$New\_ID[[i]])>0){  
 posdata\_2$logIE\_Pred[[i]] <- as.numeric(unique(posdata\_anneli[posdata\_anneli$DTXSID == posdata\_2$New\_ID[[i]],'logIE\_pred\_new'])[[1]][[1]])  
 }  
 #include log(IE) for isomers (store one value for all isomers):  
 if (sum(posdata\_anneli$DTXSID == posdata\_2$DTXSID[[i]])>0 &   
 sum(posdata\_anneli$DTXSID == posdata\_2$New\_ID[[i]])==0){  
 posdata\_2$logIE\_Pred[[i]] <- as.numeric(unique(posdata\_anneli[posdata\_anneli$DTXSID == posdata\_2$DTXSID[[i]],'logIE\_pred\_new'])[[1]][[1]])  
 }  
}

## Warning: NAs introduced by coercion

Compute linear regression on log(IE) vs. tRF:

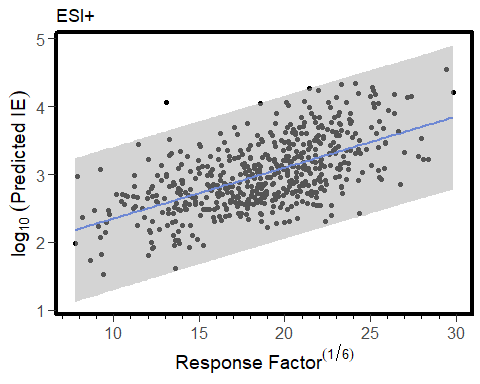
anneli\_lm\_pos<-lm(posdata\_2,  
 formula=posdata\_2$logIE\_Pred~posdata\_2$RF\_6thRoot)

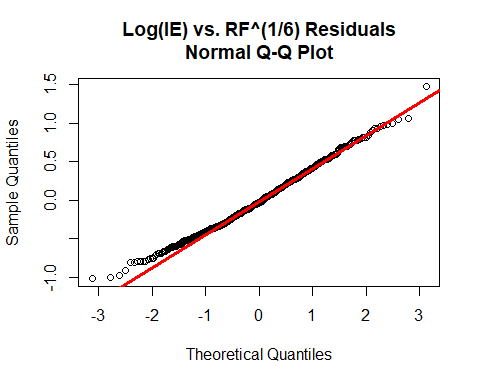
The slope is 0.0750841 and intercept is 1.5997875

Calculate 99% Prediction Interval:

anneli\_predint\_pos<-predict(anneli\_lm\_pos,  
 newdata=posdata\_2,  
 interval='prediction',  
 level=0.99)

Plot results:



Examine fit residuals: 

Allocate empty columns for calibrate() analysis to produce IE-predicted RFs (particularly, tRF\_0.01\_IE):

posdata\_2$logIE\_RF\_0.99 <- NA  
posdata\_2$logIE\_RF\_0.01 <- NA  
posdata\_2$logIE\_RF\_fit <- NA

Run inverse predictions using the log IE for each chemical:

for (i in 1:length(posdata\_2$RF\_6thRoot)){  
 if (!is.na(posdata\_2$logIE\_Pred[[i]])){  
 tmpcal<-calibrate(formula=posdata\_2$logIE\_Pred~posdata\_2$RF\_6thRoot,  
 y0=posdata\_2$logIE\_Pred[[i]],  
 interval='inversion',  
 level=0.99)  
 posdata\_2$logIE\_RF\_0.01[[i]]<-tmpcal$lower  
 posdata\_2$logIE\_RF\_0.99[[i]]<-tmpcal$upper  
 posdata\_2$logIE\_RF\_fit[[i]]<-tmpcal$estimate  
 }  
}

To ensure our error quotient is never larger than the largest error produced by the default RF method, we impute tRF\_0.01 in for any tRF\_0.01\_IE < tRF\_0.01 from the 1D tRF distribution:

for (i in 1:length(posdata\_2$logIE\_RF\_0.01)){  
 if (!is.na(posdata\_2$logIE\_RF\_0.01[[i]]) &  
 posdata\_2$logIE\_RF\_0.01[[i]] < pos\_RF6th\_lower){  
 posdata\_2$logIE\_RF\_0.01[[i]]<-pos\_RF6th\_lower  
 }  
}

Following imputation, again, we use C=I/RF\_0.01\_IE^6 to determine Conc\_0.99\_IE, and calculate the error quotients:

#Conc\_0.99\_IE:  
posdata\_2$logIE\_Conc\_Upper <- posdata\_2$y0/posdata\_2$logIE\_RF\_0.01^6  
  
#Error Quotients:  
posdata\_2$Err\_IEUppervsCCEst <- posdata\_2$logIE\_Conc\_Upper/posdata\_2$Conc\_CC