Supplementary Data 3

Evaluation of the Endogenous Concentration Using a Quadratic and a Linear Regression

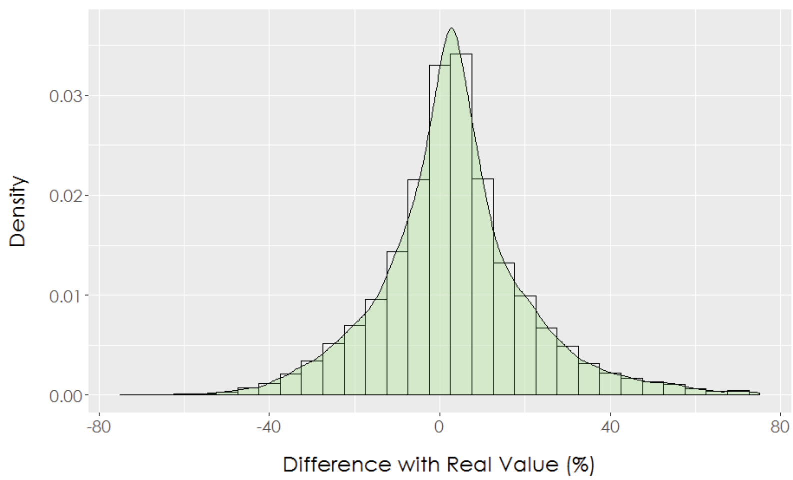

Figure 1: Percent difference observed between the real endogenous value and the measured endogenous value when a linear calibration model is used on quadratic data.

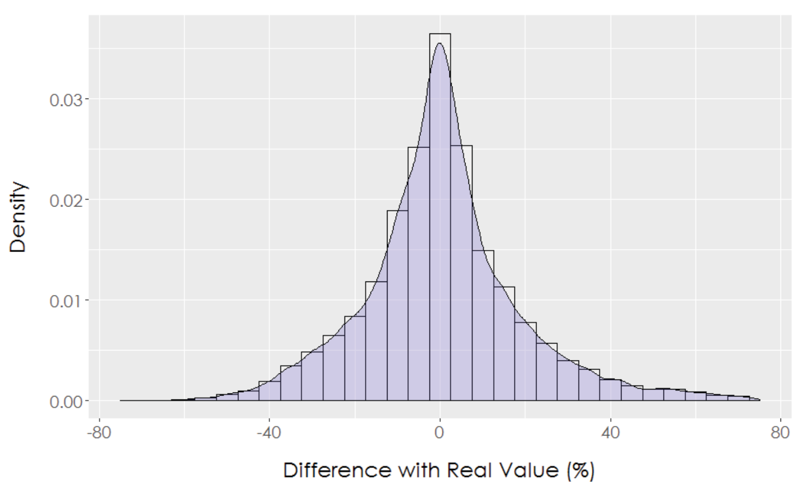

Figure 2: Percent difference observed between the real endogenous value and the measured endogenous value when a quadratic calibration model is used on quadratic data.

Distribution density is more symmetrical, with a median closer to zero when the quadratic regression is used on quadratic data rather than the linear one.

The R script used to produce those histogram is reproduced here:

```r
# XE_Lin_vs_Quad.R
# A script to compare estimation of the endogenous concentration via a cropped linear calibration curve
# and a quadratic calibration curve if the underlying data is quadratic (mimicking the LC-MS/MS case).
# By Brigitte Desharnais, original version on 2017-04-28,
# substantial re-work on 2018-09-21.

# Create vectors to stock the observed differences to the known endogenous concentration.
Q_Diff <- vector(mode="numeric", length=10000)
L_Diff <- vector(mode="numeric", length=10000)

# Vector of concentration levels to be used.
X <- c(0, 3, 6, 30, 60, 150, 255, 300)

# Number of calibration levels.
n <- length(X)

# Repeat z iterations of generating false data, finding the experimental endogenous concentration and
# comparing it to the real endogenous concentration.
for (z in 1:10000){
  # The preliminary study of 14 experimental BHB curves allowed to estimate the
  # normal range of b0, b1 and b2. This was used to set the boundaries of the uniform distributions here.
  # Random selection of b0 (intercept).
  b0 <- runif(1, min=0.08, max=0.4)

  # Random selection of b1 (linear parameter).
  b1 <- runif(1, min=0.07, max=0.1)

  # Random selection of b2 (quadratic parameter).
  b2 <- runif(1, min=-0.00009, max=-0.00000008)

  # Generation of predicted responses (measurements).
  y <- (b2*X*X) + (b1*X) + b0

  # Generate a level dependent measurement error for heteroscedastic data, type 1/(x^2).
  percent <- runif(1, min=1, max=20)
  sd <- (percent/100)*y
  RSD <- sd/y*100

  # Simulation of experimental data, i.e. noise obscured measurements.
  # Set number of replicates.
  rep = 1

  # Create empty matrix to store the results.
  Y <- matrix(nrow=n, ncol=rep)

```

```r
# Generation of random normally distributed measurements and storage in Y matrix.
for(i in 1:n)
{Temp <- rnorm(rep, y[i], sd[i])
 Y[i,] <- Temp}

# Load the function OptParam
# (Taken from CodeV5.R of the calibration project, written by FCL)
OptParam <- function(X,Y,poids,indice){
  # Finds the calibration parameters for
  # Weight (poids) 1 (=0) or 1/x (=1) or 1/(x^2) (=2).
  # Model order (indice) linear (1) or quadratic (2).
  W <- diag(1/abs(X)^poids)
  xf <- matrix(rep(X,each = indice +1),ncol=indice+1,byrow=T)
  ex <- matrix(rep(0:(indice),length(X)),ncol=indice +1 ,byrow=T)
  Xp = xf^ex
  matInv <- solve(t(Xp)%*%W%*%Xp)
  param.optimaux <- rev(matInv%*%t(Xp)%*%W%*%Y)
  return(param.optimaux)
}

# Sterilize data set to prevent NaN in weighted functions,
# i.e. replace concentrations of 0 by 1e-8.
for (k in 1:n){
  if (X[k] == 0){
    X[k] <- 1E-8}
  }

# Perform linear regression without the two top levels (standards 6 and 7).
L_Params <- OptParam(X[1:6],Y[1:6],2,1)

# Perform quadratic regression with all standards.
Q_Params <- OptParam(X,Y,2,2)

# Solve for XE in all cases.
# The real XE : based on parameters used to generate data.
XE_R_1 <- abs((-b1 + sqrt((b1*b1)-(4*b2*b0)))/(2*b2))
XE_R_2 <- abs((-b1 - sqrt((b1*b1)-(4*b2*b0)))/(2*b2))

# Select the smallest solution of the two (i.e. the x intercept).
if(XE_R_1 < XE_R_2){
   XE_R <- XE_R_1
}else{
   XE_R <- XE_R_2
}

# XE calculated from the quadratic regression on simulated experimental data.
XE_Q_1 <- abs((-Q_Params[2] + sqrt((Q_Params[2]*Q_Params[2])-(4*Q_Params[1]*Q_Params[3])))/(2*Q_Params[1]))
XE_Q_2 <- abs((-Q_Params[2] - sqrt((Q_Params[2]*Q_Params[2])-(4*Q_Params[1]*Q_Params[3])))/(2*Q_Params[1]))

# Select the smallest solution of the two (i.e. the x intercept).
if(XE_Q_1 < XE_Q_2){
   XE_Q <- XE_Q_1
```

```
  }else{
    XE_Q <- XE_Q_2
  }

  # Store the difference between the simulated experimental value and the real value in Q_Diff.
  Q_Diff[z] <- ((XE_Q - XE_R)/XE_R)*100

  # XE calculated from the linear regression on the simulated experimental data.
  XE_L <- L_Params[2]/L_Params[1]

  # Store the difference between the simulated experimental value and the real value in L_Diff.
  L_Diff[z] <- ((XE_L-XE_R)/XE_R)*100

}

# Load necessary packages.
library(extrafont)
library(ggplot2)
library(dplyr)

# Format results for graph production.
Data <- cbind(L_Diff, Q_Diff)
colnames(Data) <- c("Linear", "Quadratic")
Data <- tbl_df(Data)

# Generate histograms for the percent differences observed.
# For linear regression.
ggplot(Data, aes(x=Linear)) +
  geom_histogram(aes(y=..density..), binwidth = 5, colour="black", fill="white", alpha=0.2) +
  geom_density(alpha=0.2, fill="#5ed134")+
  scale_x_continuous(name="Difference with Real Value (%)", limits=c(-75,75)) +
  scale_y_continuous(name="Density")+
  ggtitle("Difference Observed with Linear Regression") +
  theme(plot.title = element_text(size = 20, face = "bold", family = "Century Gothic"),
        axis.title = element_text(size = 18, family = "Century Gothic"),
        axis.title.y = element_text(margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.title.x = element_text(margin = margin(t = 20, r = 0, b = 0, l = 0)),
        axis.text = element_text(size = 14, family = "Century Gothic"),
        plot.margin=unit(c(1,1,0.5,0.5),"cm"))

# For quadratic regression.
ggplot(Data, aes(x=Quadratic)) +
  geom_histogram(aes(y=..density..), binwidth = 5, colour="black", fill="white", alpha=0.2) +
  geom_density(alpha=0.2, fill="#4b3cc1")+
  scale_x_continuous(name="Difference with Real Value (%)", limits=c(-75,75)) +
  scale_y_continuous(name="Density")+
  ggtitle("Difference Observed with Quadratic Regression") +
  theme(plot.title = element_text(size = 20, face = "bold", family = "Century Gothic"),
        axis.title = element_text(size = 18, family = "Century Gothic"),
        axis.title.y = element_text(margin = margin(t = 0, r = 20, b = 0, l = 0)),
```

```
        axis.title.x = element_text(margin = margin(t = 20, r = 0, b = 0, l = 0)),
        axis.text = element_text(size = 14, family = "Century Gothic"),
        plot.margin=unit(c(1,1,0.5,0.5),"cm"))
```