Supplemental Data 3

R scripts to perform the selection and validation of the calibration model

A video “user guide” containing instructions on how to setup and execute the R scripts is available at `https://youtu.be/azpD2GG0qNA`. Please watch it before using the R scripts.

The “.R” files ready for use are available at `https://bit.ly/2VlLVPx`.

```
####################
## File to be run ##
####################

## Parameters to be set
directory <- "~/Desktop/Bridge/Calibration" # set work directory : folder where the calibration data is
                                           # and the two R sheets Calibration2Ks and Calibration2CVM
filename <- "Model.txt" # name of the file that contains the raw data (calibration data)
result_filname <- "My_Results.txt" # name of the file that contains the results
dec <- "." # , if the decimals are ex. 0,5 . if the decimals are ex 0.5
nbgrille = 50 # global paramter for integral approximations
b <- 1000 #global paramter for bootstraping
stat <- 2 # 1: KS, 2: CVM
alpha <- 0.05 ## The alpha to use in hypothesis testing
## end of parameter setting


##########
## Code ##
##########
source("CodeV5.R") ## Contains the main coded functions
A <- read.table(filename,header=F,dec=dec) ## reading the data
A <- as.matrix(A)
stats = c("Kolmogorov Smirnov","Cramer von Mises")


Decision <- rep(0,3)
DCV <- rep(0,3)

# weight : 0
weight <- 0;
P0 = fitPlus(A,weight,b,nbgrille,stat)
coP0 <- matrix(c(rev(P0$Linear$param),0,rev(P0$Quadratic$param)),ncol=3,byrow=T) ## Contains the fitted coefficients
colnames(coP0) <- c("b0","b1","b2")
```

```r
rownames(coP0) <- c("Linear","Quadratic")


List0 <- list(Fitted_Param = coP0, Pval_Lin = P0$Linear$pvalN,Stat_Lin = P0$Linear$statN ,
      Pval_Quad = P0$Quadratic$pvalN,Stat_Quad = P0$Quadratic$statN, Pval_PartialF = P0$pvalFtest)

Decision[1] = 1 + (List0$Pval_PartialF <0.05)

# weight : 1
weight <- 1;
P1 = fitPlus(A,weight,b,nbgrille,stat)
coP1 <- matrix(c(rev(P1$Linear$param),0,rev(P1$Quadratic$param)),ncol=3,byrow=T) ## Contains the fitted coefficients
colnames(coP1) <- c("b0","b1","b2")
rownames(coP1) <- c("Linear","Quadratic")


List1 <- list(Fitted_Param = coP1, Pval_Lin = P1$Linear$pvalN,Stat_Lin = P1$Linear$statN ,
  Pval_Quad = P1$Quadratic$pvalN,Stat_Quad = P1$Quadratic$statN, Pval_PartialF = P1$pvalFtest)

Decision[2] = 1 + (List1$Pval_PartialF <0.05)

# weight : 2
weight <- 2;
P2 = fitPlus(A,weight,b,nbgrille,stat)
coP2 <- matrix(c(rev(P2$Linear$param),0,rev(P2$Quadratic$param)),ncol=3,byrow=T) ## Contains the fitted coefficients
colnames(coP2) <- c("b0","b1","b2")
rownames(coP2) <- c("Linear","Quadratic")


List2 <- list(Fitted_Param = coP2, Pval_Lin = P2$Linear$pvalN,Stat_Lin = P2$Linear$statN ,
  Pval_Quad = P2$Quadratic$pvalN,Stat_Quad = P2$Quadratic$statN, Pval_PartialF = P2$pvalFtest)

Decision[3] = 1+ (List2$Pval_PartialF <0.05)

## Variance Test for weight selection
Ano  = A[,-1]/sqrt(A[,1])
spoids = sum(1/sqrt(A[,1]))
var1 = apply(Ano/spoids,1,var)
Ano2 = A[,-1]/A[,1]
spoids2 = sum(1/(A[,1]))
var2 = apply(Ano2/spoids2,1,var)
var0 = apply(A[,-1]/length(A[,1]),1,var)
s1 = signif(sum((var0-mean(var0))^2),3)
s2 = signif(sum((var1-mean(var1))^2),3)
s3 = signif(sum((var2-mean(var2))^2),3)
sss = c(s1,s2,s3)
z1 = paste("Variance test for weight selection" )
z2 = paste("Scores:" , "No weight:",s1,"x^(-1):",s2, " x^(-2):",s3 )
ccc = c("no weight","x^(-1)","x^(-2)")
```

```r
ccb = paste("Selected weight: ",ccc[which(sss==min(sss))],sep="")
zaa = paste(z1,z2,ccb,sep="\n")

best <- which(sss==min(sss))

awiner = eval(parse(text = paste("List",best-1,sep="")))

## Ftest Hétéro
pop1 = A[1,-1]
pop2 = A[nrow(A),-1]
ttt = var.test(pop1,pop2,alternative = "less")
bb = paste("F-test for heteroscedasticity")
bb1 = paste("p-value: ",signif(ttt$p.value,4))
ddd = "No"
if(ttt$p.value<alpha ){ddd="Yes"}
bb2 = paste("Weighting needed:",ddd,sep=" ")
popl = paste(bb,bb1,bb2,sep="\n")


v = paste("Partial F-test for model order selection")
v1 = paste("p-value",signif(awiner$Pval_PartialF,4),sep=": ")
v2 = paste("Model selected: ",c("linear","quadratic")[Decision[best]],sep="")
vv = paste(v,v1,v2,sep="\n")


w = paste("Normality of the standerdized residuals")
w1 = paste("Test used: ",stats[stat],sep="")
aw= eval(parse(text = paste("awiner$Pval_",c("Lin","Quad")[Decision[best]],sep="")))
w2 = paste("p-value: ",signif(aw,4))
ww = "No"
if( alpha<aw)
{
  ww = "Yes"
}
ww1 = paste("Validation test passed:",ww,sep=" ")
vw =paste(w,w1,w2,ww1,sep="\n")

md = c("Linear","Quadratic")[Decision[best]]
c("1","1/x","1/x^2")[best]
if(ww=="Yes")
{
zw=paste("Model selected: ",md,", ",c("1","1/x","1/x^2")[best],sep="" )
zw2 = paste("Calibration equation:")
if(md=="Linear")
{
  co1 = awiner$Fitted_Param[1,1]
  co2 = awiner$Fitted_Param[1,2]
  zw1 = paste(signif(co2,4)," x + ",signif(co1,4),sep="")
}
if(md=="Quadratic")
{
  co1 = awiner$Fitted_Param[2,1]
  co2 = awiner$Fitted_Param[2,2]
  co3 = awiner$Fitted_Param[2,3]
```

```
  zw1 = paste(signif(co3,4)," x^2 + ",signif(co2,4)," x + ",signif(co1,4),sep="")
}
vw2 = paste(zw,zw2,zw1,sep="\n")
}
if(ww == "No"){
  vw2 = paste("No model selected, validation test failed")
}

## Plots
namePlot = strsplit(result_filname,".",fixed=TRUE)[[1]][1]
# Plot of variance
var_level = apply(A[,-1],1,var)
pdf(file=paste(namePlot,"Variance.pdf",sep="_"))
plot(A[,1],var_level,xlab="Concentration",ylab="Variance",main="Variance plot",mgp = c(2,
    0.8, 0), axes = T)
dev.off()
## Calibration curve
f <- function(x)
{
  predic(rev(awiner$Fitted_Param[Decision[best],]),x)
}
Cal.dots = predic(rev(awiner$Fitted_Param[Decision[best],]),A[,1])
pdf(file=paste(namePlot,"Calibration curve.pdf",sep="_"))
plot(rep(A[,1],each=(ncol(A[,-1]))),c(t(A[,-1])),xlab="Concentration",ylab="Signal",main="
    Calibration curve",mgp = c(2, 0.8, 0), axes = T)
curve(f,add=T)
dev.off()
mt = paste(popl,zaa,vv,vw,vw2,sep="\n \n")
write(mt,file=result_filname)
cat(mt)
```

```
  # A : sample of data
  # nbgrille : preision for the approximation of the grid
  # variance to be used in the normal distribution function

  ## Computes sqrt(n) * sup_x | F_n(x) - Phi(x) | where Phi is the normal distribution
    function mean 0
  ## variance sigma, F_n is the empirical distribution function computed from A

library("mvtnorm")

pval <- function(dis,cv){
  mean(cv>dis)
}

OptParam <- function(X,Y,poids,indice){
  ## trouve les param optimaux pour indice = 1: modèle linéaire, 2 : modèle quadratique
  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)
}
```

```
predic <- function(param,x)
{
  xf <- matrix(rep(x,each = length(param)),ncol=length(param),byrow=T)
  ex <- matrix(rep(0:(length(param)-1),length(x)),ncol=length(param),byrow=T)
  par <- matrix(rep(rev(param),length(x)),ncol=length(param),byrow=T)
  Temp <- par * xf^ex
  return(apply(Temp,1,sum))
}

CV <- function(don,indice,poids)
{
  X = rep(don[,1],ncol(don[,-1]))
  Y = c(don[,-1])
  CVc = 0
  corr <- (1/X^poids)
  for(i in 1:length(Y))
  {
    p <- OptParam(X[-i],Y[-i],poids,indice)
    pr <- (predic(p,X[i]) - Y[i])
    CVc = CVc + (pr)^2 * corr[i]
  }
  return(CVc/sum(corr))
}

fit <- function(A,poids,b,nbgrille,stat,indice){
  ## A : matrix of data, column 1 contains the covariates
  ## poids : 1/X^poids i.e poids = 0 : no additional weight, poids =2 : inverted quadratique weight
  ## b : number of bootstrap replicates to use, suggestion is 1000
  ## nbgrille : number of points used to approximate the statistic, suggestion is 50
  ## stat : 1 is Kolmogorov smirnoff statistic, 2 is cramer von mises
  ## indice : 1 is linear, 2 is quadratic 3 is cubic

  don <- as.matrix(A)
  nbr <- ncol(A)-1

  W <- diag(rep(1/abs(don[,1])^poids,nbr))  # Weights

# Optimal parameters
  X = rep(don[,1],ncol(don[,-1]))
  Y = c(don[,-1])
  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
  meanXp <- apply(Xp,2,mean)
  matInv <- solve(t(Xp)%*%W%*%Xp)
  matt <- matInv%*%t(Xp)%*%W
  param.optimaux <- rev(matt%*%Y)

  #Residuals
  poidsob <- 1/abs(X)^poids
  poidsob <- poidsob/sum(poidsob)
  epsilonp <- (predic(param.optimaux,X) - Y) * sqrt(poidsob)
```

```
  var.param <- matInv * var(epsilonp) ## variance of estimated residuals

  ## Bootstrap
  n <- length(epsilonp)
  sigma <- sqrt(var(epsilonp))

  ## Computationi of the data process
  grille = (1:(nbgrille))/(nbgrille+1) ## Approximation over the grid
  grrep <- t(matrix(rep(grille,each=n),byrow=T,ncol=n))
  epsilonrep <- matrix(rep(epsilonp,each=nbgrille),ncol=nbgrille,byrow=T)/sigma
  Fn <- apply(pnorm(epsilonrep)<=grrep,2,mean)
  monproc <- sqrt(n)*(Fn - grille)

  proc=matrix(rep(0,b*nbgrille),ncol=b)
  Xb = predic(param.optimaux,X);
  for(i in 1:b)
  {
    ech_boot = sample(epsilonp,n,replace=T)
    Yb = Xb + ech_boot / sqrt(poidsob)
    param.optimauxb <- rev(matt%*%Yb)
    epsilonpb <- (predic(param.optimauxb,X) - Yb) * sqrt(poidsob)
    epsilonrepb <- matrix(rep(epsilonpb,each=nbgrille),ncol=nbgrille,byrow=T)/sqrt(var(epsilonpb))
    Fnb <- apply(pnorm(epsilonrepb)<=grrep,2,mean)
    monprocb <- sqrt(n)*(Fnb - Fn)
    proc[,i] = monprocb
  }

  ## Computation of the pvalue
  distr <- rep(0,b)
  mastat <- 0
  if(stat==1){  distr <- apply(abs(proc),2,max)
    mastat <- max(abs(monproc)) }
  if(stat==2){ distr <- apply(proc^2,2,mean)
    mastat<- mean(monproc^2) }
  quantileb = quantile(distr,0.95)
  pvalN = mean(distr>mastat)

liste <- list(param=param.optimaux,varparam=var.param,pvalN=pvalN,statN=mastat,qBoot=quantileb,esp=epsilonp)
return(liste)
}

fitPlus <- function(A,poids,b,nbgrille,stat)
{
  ## Performs the fit plus the partial F text
  fitLin <- fit(A,poids,b,nbgrille,stat,1)
  fitQ <- fit(A,poids,b,nbgrille,stat,2)

  SSREGL <- sum(fitLin$esp^2)
  SSREGQ <- sum(fitQ$esp^2)
  degf <- length(fitQ$esp)-3
  stat <- (SSREGL-SSREGQ)/(SSREGQ/(degf))
  pvalF <- 1-pf(stat,1,degf)
```

```r

  L2 <- list(Linear= fitLin,Quadratic = fitQ,Ftest=stat,pvalFtest=pvalF)
  return(L2)
}
```

```r
## Code KS

KS <- function(A,nbgrille,sigma){
  # A : sample of data
  # nbgrille : preision for the approximation of the grid
  # variance to be used in the normal distribution function

  ## Computes sqrt(n) * sup_x | F_n(x) - Phi(x) | where Phi is the normal distribution function mean 0
  ## variance sigma, F_n is the empirical distribution function computed from A

  n<- length(A)
  fn <- function(t){sqrt(n) * abs(mean(A <= qnorm(t,0,sd=sqrt(sigma))) - t ) }
  max(sapply((1:nbgrille)/nbgrille,fn))
}
pval <- function(dis,cv){
  mean(cv>dis)
}


# Linear fit using KS statistic
fit1KS <- function(A,poids,b,nbgrille){
don <- as.matrix(A)
nbr <- ncol(A)-1

W <- diag(rep(1/abs(don[,1])^poids,nbr))  # matrix of weights

# Finding the optimal parameters for the regression
x1 <- don[,1]
Xp <- matrix(c(rep(1,length(don[,1])*nbr),rep(x1,nbr)),ncol=2)
matInv <- solve(t(Xp)%*%W%*%Xp)
param.optimaux <- rev(matInv%*%t(Xp)%*%W%*%c(don[,-1]))

# Computing the residuals
temp <- param.optimaux[1]*don[,1]+  param.optimaux[2]
temp2 <- matrix(rep(temp,nbr),nrow=length(don[,1]))
temp3 <- (don[,-1] - temp2)

poidsob <- 1/abs(don[,1])^poids
poidsob <- poidsob/sum(poidsob)
epsilonp <- c(temp3)*(sqrt(rep(poidsob,nbr))) # residuals

var.param <- matInv * var(epsilonp) # the variance of the estimated paramters

## Bootstrap
  n <- length(epsilonp)
  sigma <- sqrt(var(epsilonp))
  distrCv <- rep(0,b)

for(j in 1:b){
```

```
  B <- rnorm(n,0,sqrt(sigma))
  C <- rmvnorm(1,mean=c(0,0),var.param)
   drift <-  (C[1] + C[2]*mean(don[,1])) # Drift function to used when esimated residuals are involved
  n<- length(A)
  fn <- function(t){
    sqrt(n) * abs(mean(A <= qnorm(t,0,sd=sqrt(sigma))) -
        t +  drift*dnorm(qnorm(t,0,sqrt(sigma)),0,sd=sqrt(sigma))/sqrt(n)) }
  distrCv[j] <- max(sapply((1:nbgrille)/nbgrille,fn))
  }
cv0 <- KS(epsilonp,nbgrille,sigma)
  pvalN <- pval(cv0,distrCv)

liste <- list(param=param.optimaux,res1=cbind(rep(poidsob,nbr),c(temp3)),epsp=epsilonp,varparam=var.param,pvalN=pvalN)
return(liste)
}

#quadratic fit using KS statistic
fit2KS <- function(A,poids,b,nbgrille){
don <- as.matrix(A)
nbr <- ncol(A)-1

W <- diag(rep(1/abs(don[,1])^poids,nbr))  # Weights

# Optimal parameters
x1 <- don[,1]
Xp <- matrix(c(rep(1,length(don[,1])*nbr),rep(x1,nbr),rep(x1^2,nbr)),ncol=3)
matInv <- solve(t(Xp)%*%W%*%Xp)
param.optimaux <- rev(matInv%*%t(Xp)%*%W%*%c(don[,-1]))

#Residuals
temp <- param.optimaux[1]*don[,1]^2 +  param.optimaux[2]*don[,1] +  param.optimaux[3]
temp2 <- matrix(rep(temp,nbr),nrow=nrow(don))
temp3 <- (don[,-1] - temp2)

poidsob <- 1/abs(don[,1])^poids
poidsob <- poidsob/sum(poidsob)
epsilonp <- c(temp3)*(sqrt(rep(poidsob,nbr)))
var.param <- matInv * var(epsilonp)

  ## Bootstrap
  n <- length(epsilonp)
  sigma <- sqrt(var(epsilonp))
  distrCv <- rep(0,b)

for(j in 1:b){
  B <- rnorm(n,0,sqrt(sigma))
  C <- rmvnorm(1,mean=c(0,0,0),var.param)
   drift <-  (C[1] + C[2]*mean(don[,1]) + C[3]*mean(don[,1]^2)) # Drift function to used when esimated residuals are involved
  n<- length(A)
  fn <- function(t){
```

```
    sqrt(n) * abs(mean(A <= qnorm(t,0,sd=sqrt(sigma))) -
        t +  drift*dnorm(qnorm(t,0,sqrt(sigma)),0,sd=sqrt(sigma))/sqrt(n)) }
  distrCv[j] <- max(sapply((1:nbgrille)/nbgrille,fn))
  }
cv0 <- KS(epsilonp,nbgrille,sigma)
  pvalN <- pval(cv0,distrCv)
  pvalN <- pval(cv0,distrCv)

liste <- list(param=param.optimaux,varparam=var.param,pvalN=pvalN)
return(liste)
}
```

```
## Code CVM
CVM <- function(A,nbgrille,sigma)
{
  # A : sample of data
  # nbgrille : preision for the approximation of the grid
  # variance to be used in the normal distribution function

  ## Computes sqrt(n) * \integral( F_n(x) - Phi(x) )^2 dx where Phi is the normal
    distribution function mean 0
  ## variance sigma, F_n is the empirical distribution function computed from A
  cv =0
  n <- length(A)
    for(i in 1:nbgrille){
    cv <- cv +  n/nbgrille * (mean(A <= qnorm(i/nbgrille,mean=0,sd=sqrt(sigma)) )- i/
    nbgrille )^2
  }
  cv
}
pval <- function(dis,cv){
  mean(cv>dis)
}


#Linear fit with CVM statistic
fit1CVM <- function(A,poids,b,nbgrille){
don <- as.matrix(A)
nbr <- ncol(A)-1
nbniv <- nrow(don)

# Weights
W <- diag(rep(1/abs(don[,1])^poids,nbr))
# Optimal parameters
x1 <- don[,1]
Xp <- matrix(c(rep(1,length(don[,1])*nbr),rep(x1,nbr)),ncol=2)
matInv <- solve(t(Xp)%*%W%*%Xp)
param.optimaux <- rev(matInv%*%t(Xp)%*%W%*%c(don[,-1]))

#Residuals
temp <- param.optimaux[1]*don[,1]+  param.optimaux[2]
temp2 <- matrix(rep(temp,nbr),nrow=length(don[,1]))
temp3 <- (don[,-1] - temp2)

poidsob <- 1/abs(don[,1])^poids
```

```r
poidsob <- poidsob/sum(poidsob)
epsilonp <- c(temp3)*(sqrt(rep(poidsob,nbr)))

var.param <- matInv * var(epsilonp)

## Bootstrap
  n <- length(epsilonp)
  sigma <- sqrt(var(epsilonp))
  distrCv <- rep(0,b)

for(j in 1:b){

  B <- rnorm(n,0,sqrt(sigma))
  C <- rmvnorm(1,mean=c(0,0),var.param)
  drift <-  (C[1] + C[2]*mean(don[,1])) # Drift function to used when esimated residuals are involved
  cv <- 0
  for(i in 1:nbgrille){
    cv <- cv +  n/nbgrille * (mean(B <= qnorm(i/nbgrille,mean=0,sd=sqrt(sigma)) )- i/nbgrille + drift*dnorm(qnorm(i/nbgrille,0,sqrt(sigma)),0,sd=sqrt(sigma))/sqrt(n) )^2
  }
  distrCv[j] <- cv
  }
cv0 <- CVM(epsilonp,nbgrille,sigma)
  pvalN <- pval(cv0,distrCv)

liste <- list(param=param.optimaux,varparam=var.param,pvalN=pvalN)
return(liste)
}

# Quadratic fit statistic CVM
fit2CVM <- function(A,poids,b,nbgrille){
don <- as.matrix(A)
nbr <- ncol(A)-1
nbniv <- nrow(A)


#Weights
W <- diag(rep(1/abs(don[,1])^poids,nbr))
#Optimal paramters
x1 <- don[,1]
Xp <- matrix(c(rep(1,length(don[,1])*nbr),rep(x1,nbr),rep(x1^2,nbr)),ncol=3)
matInv <- solve(t(Xp)%*%W%*%Xp)
param.optimaux <- rev(matInv%*%t(Xp)%*%W%*%c(don[,-1]))


#Residuals
temp <- param.optimaux[1]*don[,1]^2 +  param.optimaux[2]*don[,1] +  param.optimaux[3]
temp2 <- matrix(rep(temp,nbr),nrow=nbniv)
temp3 <- (don[,-1] - temp2)

poidsob <- 1/abs(don[,1])^poids
poidsob <- poidsob/sum(poidsob)
epsilonp <- c(temp3)*(sqrt(rep(poidsob,nbr)))
```

```
var.param <- matInv * var(epsilonp)

## Bootstrap
  n <- length(epsilonp)
  sigma <- sqrt(var(epsilonp))
  distrCv <- rep(0,b)

for(j in 1:b){

  B <- rnorm(n,0,sqrt(sigma))
  C <- rmvnorm(1,mean=c(0,0,0),var.param)
  drift <-  (C[1] + C[2]*mean(don[,1]) + C[3]*mean(don[,1]^2)) # Drift function to used when esimated residuals are involved
  cv <- 0
  for(i in 1:nbgrille){
    cv <- cv +  n/nbgrille * (mean(B <= qnorm(i/nbgrille,mean=0,sd=sqrt(sigma)) )- i/nbgrille + drift*dnorm(qnorm(i/nbgrille,0,sqrt(sigma)),0,sd=sqrt(sigma))/sqrt(n) )^2
  }
  distrCv[j] <- cv
  }
cv0 <- CVM(epsilonp,nbgrille,sigma)
  pvalN <- pval(cv0,distrCv)

liste <- list(param=param.optimaux,varparam=var.param,pvalN=pvalN)
return(liste)
}
```