Multiple flow FeNO in population research: R code methods demo

The following demo file provides example R code to implement six methods for research involving multiple flow exhaled nitric oxide (FeNO) measurements in a study population, where the goal is to relate estimated NO parameters to factors of interest (i.e., covariate(s) X).

0. Preliminaries: set up your R session

IMPORTANT! Install JAGS, if not already installed, follow instructions at: https://mcmc-jags.sourceforge.io/

Load the following required packages, installing first if not already available. To use the R2jags package, JAGS must be pre-installed.

```
# set working directory
setwd("K:/paper/newpackage/PopFeNO")
# load required packages
require(MASS)
require(lme4)
require(nlme)
require(reshape2)
require(R2jags)
```

1. Data simulation

First, set a random seed for your work.

```
set.seed(2022)
```

Simulate data from a cross-sectional study with multiple flow FeNO and a single standard normal covariate X using the function DataGeneratorCS() in DataG.R, which outputs a simulated dataset for Ndat participants, with data for each participant on mean flow and FeNO for all maneuvers as well as that participant's value of X.

```
# Simulate a multiple flow FeNO dataset in a study population
source("DataG.R")       # code to generate multiple flow FeNO datasets
flow=c(rep(30,2),rep(50,2),rep(100,2),rep(300,2))
truebeta=c(0.1,0.1,0.1)
out <- DataGeneratorCS(Ndat=500,           # number of study participants
                       alpha=c(1.5,3.5,2.5),# vector of NO parameter population means,
                                            # when X=0, in this order: CA, logCaw, logDaw
                       Xfn=function(n){rnorm(n,0,1)}, # function to generate participant-level covariate
                                                      # as function of number of study participants
                       beta=truebeta, # vector of regression coefficients relating X to NO parameters,
                                      # in this order: CA, logCaw, logDaw
                       Flow=c(rep(30,2),rep(50,2),rep(100,2),rep(300,2)), # vector of flow rates in
```

```
                                                                    # multiple flow FeNO protocol
                      SD=0.1,               #SD of the error
                      sdalphaCa             = 0.45, # population SD of CA
                      sdalphalogCaw         = 0.65, # population SD of logCaw
                      sdalphalogDaw         = 0.55, # population SD of logDaw
                      coralphalogCawCa      = 0.66, # correlation of NO parameters: logCaw, CA
                      coralphalogCawlogDaw = -0.35,# correlation of NO parameters: logCaw, logDaw
                      coralphalogDawCa      = -0.38 # correlation of NO parameters: logDaw, CA
                     )
# output is two versions of the same dataset, plus dataset only including X and id
# one dataset for use in Bayesian methods via JAGS (datJAGS)
# one dataset for all other methods (dat)
dat     <- out$dat
datJAGS <- out$datJAGS
datX    <- out$datX
```

The dat dataset is in the usual 'long' data format:

```
dat[1:10,]
```

```
##    id       eno   logeno flow flowTarget
## 1   1 19.416852 2.966141   30         30
## 2   1 23.020485 3.136384   30         30
## 3   1 12.260047 2.506346   50         50
## 4   1 13.544361 2.605970   50         50
## 5   1  7.956771 2.074023  100        100
## 6   1  9.577102 2.259375  100        100
## 7   1  3.964285 1.377326  300        300
## 8   1  3.379312 1.217672  300        300
## 9   2 16.057166 2.776155   30         30
## 10  2 11.648306 2.455161   30         30
```

2. Estimate NO parameter associations with X

2.1 TS_NLS: Two-stage nonlinear least squares 1: NLS_StageI: Estimated NO paramteres ordered by id, exist NAs 2: NLS_StageIX: Combine NO paramters and covariate X by id 3: Fit individual grouped linear regression models for each NO parameters

```
source("TS_NLS.R")

# Stage I
NLS_StageI<-TS_NLS_StageI(dat)

# create dataset including both Stage I estimates and X
NLS_StageIX <- merge(NLS_StageI,datX,by="id")

# Stage II -  edit to include any additional Stage II covariates (e.g., confounder adjustments)
TS_NLS_Ca     <-lme(Ca ~ X,     random=~1|id, data = NLS_StageIX, na.action = na.omit)
TS_NLS_logCaw <-lme(logCaw ~ X, random=~1|id, data = NLS_StageIX, na.action = na.omit)
TS_NLS_logDaw <-lme(logDaw ~ X, random=~1|id, data = NLS_StageIX, na.action = na.omit)
```

Interpretation for CANA:

The population mean CANO is 1.3972161 (95% CI: 1.3364499,1.4579824). For 1 unit increase in covariate X, CANO increases 0.0690305, (95% CI: `r intervals(TS_NLS_Ca,which="fixed")[[1]][2,1]`, 0.129065).

```
source("TS_HMA.R")

# Stage I, specify target flow rates for  HMA (low, medium, high)
HMA_StageI  <- TS_HMA_StageI(dat, flowLMH=c(30,100,300))

# create dataset including both Stage I estimates and X
HMA_StageIX <- merge(HMA_StageI,datX,by="id")

# Stage II -  edit to include any additional Stage II covariates (e.g., confounder adjustments)
TS_HMA_Ca      <-lme(Ca ~ X,     random=~1|id,data = HMA_StageIX, na.action = na.omit)
TS_HMA_logCaw  <-lme(logCaw ~ X, random=~1|id,data = HMA_StageIX, na.action = na.omit)
TS_HMA_logDaw  <-lme(logDaw ~ X, random=~1|id,data = HMA_StageIX, na.action = na.omit)
```

2.2 TS_HMA: Two-stage Högman & Merilänen Algorithm

```
source("TS_NLME.R")
#Stage I
TSNLME_StageIout <- TS_NLME_StageI(dat,tol1=0.1,tol2=0.01,outputFit=TRUE)# include X for later unified
TSNLME_StageI    <- TSNLME_StageIout$ests
TSNLME_StageIfit <- TSNLME_StageIout$fit  # save fit to speed up U_NLME

# create dataset including both Stage I estimates and X
TSNLME_StageIX <- merge(TSNLME_StageI,datX,by="id")

# Stage II -  edit to include any additional Stage II covariates (e.g., confounder adjustments)
TS_NLME_Ca     <-lme(Ca ~ X,     random=~1|id, data = TSNLME_StageIX, na.action = na.omit)
TS_NLME_logCaw <-lme(logCaw ~ X, random=~1|id, data = TSNLME_StageIX, na.action = na.omit)
TS_NLME_logDaw <-lme(logDaw ~ X, random=~1|id, data = TSNLME_StageIX, na.action = na.omit)
```

2.3 TS_NLME: Two-stage nonlinear mixed effects model

```
# The function is for single X. If you want to fit with multiple X, just modify the "fixed" and start s
source("U_NLME.R")
# direct approach
U_NLMEout<-U_NLME_direct(dat,datX,tol=0.1)
# update approach
U_NLMEout_u<-U_NLME_update(TSNLME_StageIout,dat,datX,tol=0.1)
# anova(U_NLMEout,U_NLMEout_u) # compare two approaches
```

2.4 UNLME: Unified nonlinear mixed effects model

2.5 TS_HB: Two-stage Hierarchical Bayesian method

- Load existing results if exists.

```
source("TS_HB.R")
set.seed(2022)
# Stage I
if(!file.exists("TSHB_cc.Rdata")){
    TSHB_S1<-TSHB_iter(beta0_prior=c(2,4,3),
                   rhat=1.1,addon.iter=4000,Max_update=10,
                   n.final=3000,N.iterT=3000,N.burnin=2500,N.thinM=1,N.chain=3,
                   flow=flow,dat=datJAGS,
                   tracing=c("beta0_Ca","beta0_logCaw","beta0_logDaw"))
    save(TSHB_S1,file="TSHB_cc.Rdata")
}else{
    load("TSHB_cc.Rdata")
}

TSHB_S1_dat<-data.frame("Ca"=TSHB_S1$summary[grepl("^Ca",rownames(TSHB_S1$summary)),1],
                    "logCaw"=TSHB_S1$summary[grepl("^logCaw",rownames(TSHB_S1$summary)),1],
                    "logDaw"=TSHB_S1$summary[grepl("^logDaw",rownames(TSHB_S1$summary)),1]
                    )
TSHB_S1_dat$id <- as.numeric(unlist(lapply(rownames(TSHB_S1_dat),function(x) strsplit(strsplit(x,"\\[")
TSHB_S1_dat    <- TSHB_S1_dat[order(TSHB_S1_dat$id),]
TSHB_StageIX   <- cbind(TSHB_S1_dat,datX)

# Stage II -  edit to include any additional Stage II covariates (e.g., confounder adjustments)
TS_HB_Ca     <-lme(Ca ~ X,     random=~1|id, data = TSHB_StageIX, na.action = na.omit)
TS_HB_logCaw <-lme(logCaw ~ X, random=~1|id, data = TSHB_StageIX, na.action = na.omit)
TS_HB_logDaw <-lme(logDaw ~ X, random=~1|id, data = TSHB_StageIX, na.action = na.omit)
```

2.5.ex Convergence diagnostic for TS_HB via Rhat (See Gelman and Rubin (1992), Brooks and Gelman (1998)])

- Print out the estimation (95% CL) and Rhat (converge if <1.1)

```
TSHB_S1$summary[c("beta0_Ca","beta0_logCaw","beta0_logDaw","sdCa","sdlogCaw","sdlogDaw","corlogCawCa","
                      "sigma_c"),c("2.5%","mean","97.5%","Rhat")]
```

```
##                       2.5%       mean     97.5%     Rhat
## beta0_Ca         1.49753345  1.5500093 1.6037103 1.077980
## beta0_logCaw     3.34077083  3.4193315 3.5234249 1.069067
## beta0_logDaw     2.50648100  2.6206368 2.7006227 1.147185
## sdCa             0.40415293  0.4448635 0.4875842 1.034334
## sdlogCaw         0.54649166  0.6178530 0.6861821 1.207797
## sdlogDaw         0.09509015  0.1695061 0.2905103 3.280325
## corlogCawCa      0.46963490  0.5885458 0.6918710 1.060723
## corlogCawlogDaw -0.16443516  0.4106666 0.6810458 1.795172
## corlogDawCa     -0.47858958 -0.1729028 0.4829335 1.298679
## sigma_c          0.09760268  0.1000675 0.1026716 1.008098
```

2.6 U_HB: Unified Hierarchical Bayesian method

- Load existing results if exists.

```
source("U_HB.R")
set.seed(2022)
if(!file.exists("UHB_cc.Rdata")){
    UHB_sim<-UHB_iter(beta0_prior=c(2,4,3),
                  betaC_prior =c(0.1,0.1,0.1),
                  rhat=1.1,addon.iter=1000,Max_update=3,n.final=1000,
                  N.iterT=1500,N.burnin=1000,N.thinM=1,N.chain=3,
                  flow=flow,dat=datJAGS,X=datJAGS$X,tracing=c("beta1_Ca","beta1_logCaw","beta1_logDaw"))
    save(UHB_sim,file="UHB_cc.Rdata")
}else{
    load("UHB_cc.Rdata")
}
```

2.6.ex Convergence diagnostic for U_HB via Rhat

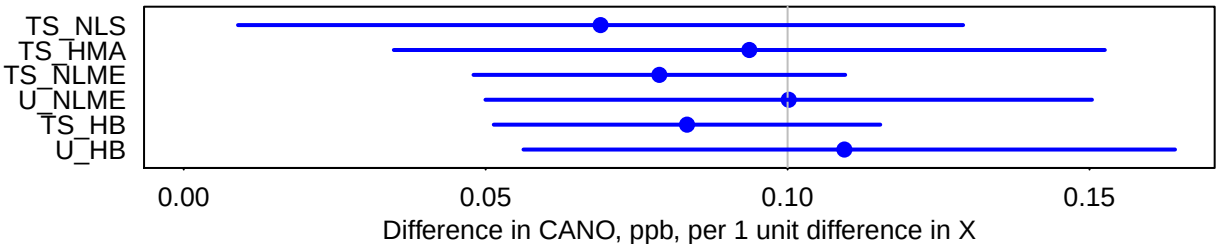
- Print out the estimation (95% CL) and Rhat (converge if <1.1)

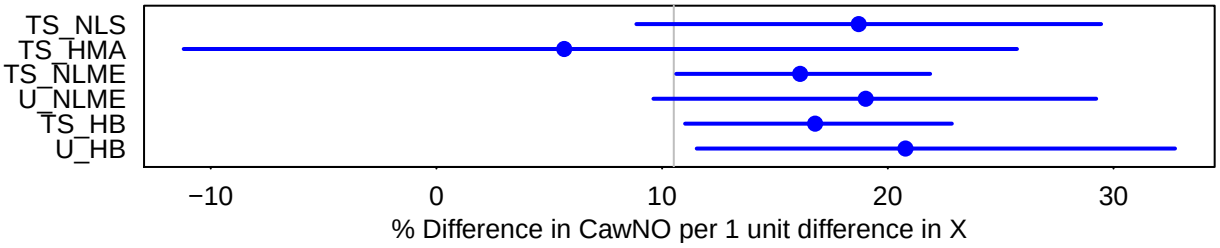
```
UHB_sim$summary[c("beta0_Ca","beta0_logCaw","beta0_logDaw",
                  "beta1_Ca","beta1_logCaw","beta1_logDaw",
                  "sdCa","sdlogCaw","sdlogDaw",
                  "corlogCawCa","corlogCawlogDaw","corlogDawCa",
                  "sigma_c"),c("2.5%","mean","97.5%","Rhat")]
```

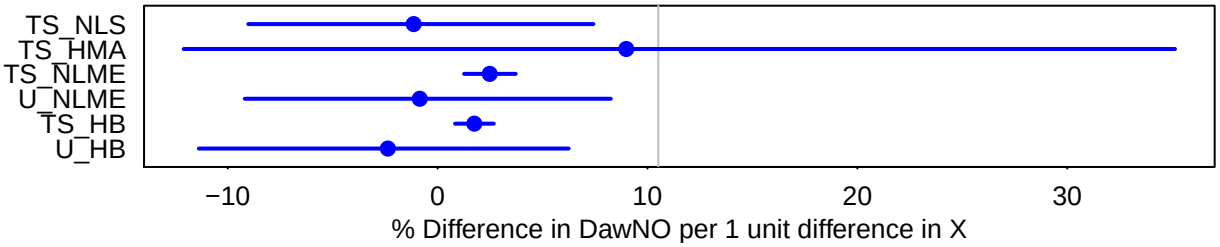
```
##                       2.5%        mean      97.5%     Rhat
## beta0_Ca         1.50734338  1.56716769 1.62650389 1.391810
## beta0_logCaw     3.39601309  3.56316972 3.81813904 3.362638
## beta0_logDaw     2.17308229  2.45901273 2.65526517 3.566671
## beta1_Ca         0.05626447  0.10940456 0.16415525 1.050764
## beta1_logCaw     0.10916633  0.18882113 0.28310534 1.068274
## beta1_logDaw    -0.12078440 -0.02400036 0.06057182 1.122616
## sdCa             0.39790539  0.44804412 0.49821050 1.543450
## sdlogCaw         0.54126778  0.62818253 0.74339953 2.397158
## sdlogDaw         0.27629953  0.43399990 0.69885785 4.365563
## corlogCawCa      0.47066993  0.63056685 0.77348775 2.434320
## corlogCawlogDaw -0.51631785 -0.13254616 0.18855645 3.765336
## corlogDawCa     -0.46534325 -0.24830977 0.08582834 2.663337
## sigma_c          0.09686811  0.09945313 0.10206027 1.015140
```

3. Create plot comparing estimated NO parameter associations across 6 methods

- Y axis: 6 methods
- X axis: coefficient effect size: The values used in simulation were all equaled to 0.1 for CANO, logCawNO and logDawNO. Which means for 1 unit increase in the covariate X, the corresponding NO paramters CANO, logCawNO and logDawNO increase 0.1 unit. The geometric interpretation for CawNO was that it was (exp(0.1)-1) times higher for 1 unit increase in the covariate, so was for DawNO


TS_NLS
TS_HMA
TS_NLME
U_NLME
TS_HB
U_HB
0.00
0.05
0.10
0.15
Difference in CANO, ppb, per 1 unit difference in X


TS_NLS
TS_HMA
TS_NLME
U_NLME
TS_HB
U_HB
−10
0
10
20
30
% Difference in CawNO per 1 unit difference in X


TS_NLS
TS_HMA
TS_NLME
U_NLME
TS_HB
U_HB
−10
0
10
20
30
% Difference in DawNO per 1 unit difference in X