Multiple flow FeNO in population research: R code methods demo

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

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.

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

1. Data simulation

First, set a random seed for your work.

```r
set.seed(2020)
```

Next, we simulate data from a cross-sectional study with multiple flow FeNO on each partcipant and a single covariate of interest, X, which varies at the participant-level (i.e., it varies across participants, but is constant within participant across FeNO maneuvers). We have written a function DataGeneratorCS() in DataG.R, which outputs a simulated dataset containing data for each participant on mean flow and FeNO for all FeNO maneuvers as well as that participant's value of X.

In this example, we generate data for 500 participants with X following a standard normal distribution (though in practice X can have any distribution) and multiple flow FeNO (8 maneuvers per participant, two each at the target flow rates of 30, 50, 100, and 300 ml/s). Participant-level NO parameters (CANO, CawNO and DawNO) are each generated as a linear function of X, except that rather than CawNO or DawNO, we generate logCawNO and logDawNO to better replicate the approximately log normal distribution observed in practice and to avoid negative estimates in later modeling. For example, participant-level CANO_i = 1.5 + 0.1*X_i + epsilon_i where the standard deviation of the random normal error is 0.45. While we could generate each of the three NO parameters seperately for each participant, instead we generate the three NO parameters from a multivariate normal distribution to introduce correlation between NO parameters. Given the NO parameters, FeNO at each maenuvers can then be generated according to the standard two compartment model XXJingying please add formulationXX with random normal error of 0.1.

```
# 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=1000,             # 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 resultant dataset is in the usual 'long' data format:

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

```
##    id      eno    logeno flow flowTarget
## 1   1 6.928492 1.9356421   30         30
## 2   1 6.450914 1.8642218   30         30
## 3   1 5.166481 1.6421918   50         50
## 4   1 5.887342 1.7728046   50         50
## 5   1 3.782733 1.3304469  100        100
## 6   1 3.262054 1.1823571  100        100
## 7   1 1.749397 0.5592711  300        300
## 8   1 1.946140 0.6658478  300        300
## 9   2 6.715659 1.9044420   30         30
## 10  2 7.187736 1.9723762   30         30
```

- The intercepts and slopes of the linear regression moedls: NOparam ~ X of the simulated data:

```
estParam<-lm(datJAGS$NOparam~datJAGS$X)$coef
estParam
```

```
##                     Ca     logCaw     logDaw
## (Intercept) 1.53471895 3.54146356 2.46989797
## datJAGS$X   0.09307022 0.09869727 0.08740731
```

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 CANO:

The population mean CANO is 1.38 (95% CI: 1.34,1).

For 1 unit increase in covariate X, CANO increases 0.08, (95% CI: `r round(intervals(TS_NLS_Ca,which="fixed")[[1]][2,`
0.13).

```


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

```


if(!file.exists("TSNLME_cc.Rdata")){
  source("TS_NLME.R")
  #Stage I
  TSNLME_StageIout <- TS_NLME_StageI(dat,tol1=0.1,tol2=0.1,outputFit=TRUE)# include X for later unified
  save(TSNLME_StageIout,file="TSNLME_cc.Rdata")
}else{
  load("TSNLME_cc.Rdata")
}
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

2.4 UNLME: Unified nonlinear mixed effects model

- Notation: There are two ways to fit the unified NLME model. One through direct simulation, the other updates the two-stage version.

```
if(!file.exists("UNLME_cc.Rdata")){
  set.seed(2020)
  # The function is for single X. If you want to fit with multiple X, just modify the "fixed" and    st
  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
  save(U_NLMEout,file="UNLME_cc.Rdata")
}else{
  load("UNLME_cc.Rdata")
}
```

2.5 TS_HB: Two-stage Hierarchical Bayesian method

- Load existing results if exists.

```
source("TS_HB.R")

# Stage I
if(!file.exists("TSHB_cc.Rdata")){
    set.seed(2020)
    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         1.5090358  1.5499208  1.59100448 1.006103
## beta0_logCaw     3.4134646  3.4946240  3.57935519 1.042173
## beta0_logDaw     2.4181594  2.5164054  2.60836531 1.035312
## sdCa             0.4225800  0.4554148  0.48888886 1.009385
## sdlogCaw         0.5753498  0.6428229  0.71811159 1.094953
## sdlogDaw         0.3826897  0.4936405  0.60249867 1.092135
## corlogCawCa      0.5399083  0.6394098  0.73048487 1.023285
## corlogCawlogDaw -0.4425728 -0.2489738 -0.02690783 1.120940
## corlogDawCa     -0.4174139 -0.2354739 -0.01930473 1.032991
## sigma_c          0.0989317  0.1007837  0.10267102 1.008575
```

2.6 U_HB: Unified Hierarchical Bayesian method

- Load existing results if exists.

```
source("U_HB.R")

if(!file.exists("UHB_cc.Rdata")){
    set.seed(2020)
    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

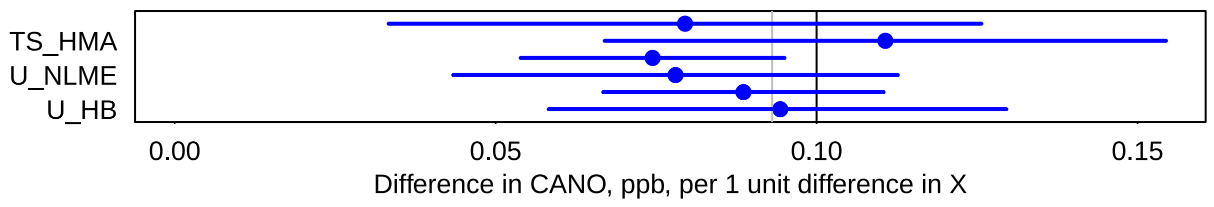
- Print out the estimation (95% CL) and Rhat (converge if <1.1)

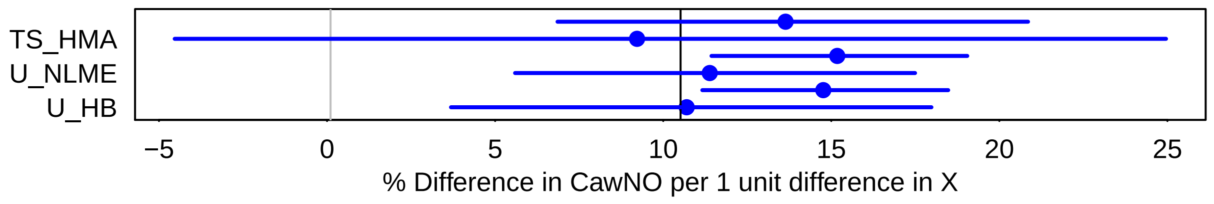
```
UHB_sim$summary[c("beta0_Ca","beta0_logCaw","beta0_logDaw",
                  "beta1_Ca","beta1_logCaw","beta1_logDaw",
                  "sdCa","sdlogCaw","sdlogDaw",
                  "corlogCawCa","corlogCawlogDaw","corlogDawCa",
                  "sigma_c"),c("2.5%","mean","97.5%","Rhat")]
```

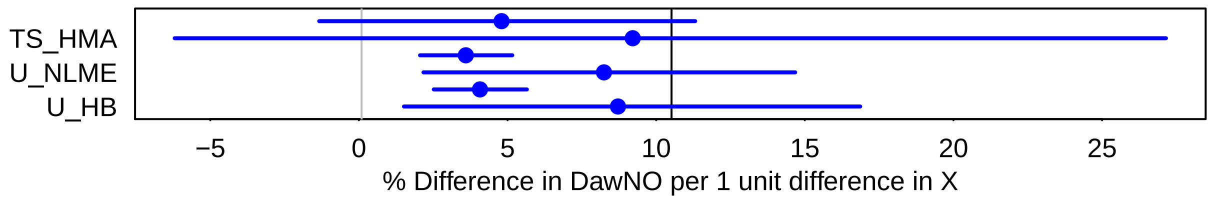
```
##                        2.5%        mean      97.5%     Rhat
## beta0_Ca         1.51119850  1.55091324  1.5915931 1.005527
## beta0_logCaw     3.40857005  3.49420617  3.5963433 1.033231
## beta0_logDaw     2.40312590  2.52247501  2.6206695 1.032590
## beta1_Ca         0.05826295  0.09433501  0.1295614 1.001967
## beta1_logCaw     0.03621160  0.10163802  0.1653405 1.002494
## beta1_logDaw     0.01503875  0.08355172  0.1558483 1.004417
## sdCa             0.41540423  0.44688566  0.4805696 1.005201
## sdlogCaw         0.57906136  0.64148660  0.7056561 1.023145
## sdlogDaw         0.38331218  0.48990496  0.6050677 1.065012
## corlogCawCa      0.53824119  0.63252838  0.7250663 1.010522
## corlogCawlogDaw -0.46469195 -0.30680658 -0.1044257 1.049161
## corlogDawCa     -0.47915510 -0.30687743 -0.1284974 1.014665
## sigma_c          0.09889641  0.10073696  0.1026550 1.007205
```

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







4: Comparison by table:

- Betas:

```
all2<-rbind(allResults[1:3,3],allResults[4:6,3],allResults[7:9,3],allResults[10:12,3],allResults[13:15,
colnames(all2)<-c("CaNO","logCaw","logDaw")
rownames(all2)<-sixMethods
kable(round(all2,2))
```

	CaNO	logCaw	logDaw
TS_NLS	0.08	0.13	0.05
TS_HMA	0.11	0.09	0.09
TS_NLME	0.07	0.14	0.04
U_NLME	0.08	0.11	0.08
TS_HB	0.09	0.14	0.04
U_HB	0.09	0.10	0.08

```
getwd()
```

```
## [1] "K:/paper/newpackage/PopFeNO"
```