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.

```
# 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 (flow rate for HMA)

First, set a random seed for your work.

```
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 (Environmental factor, length=500) 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 $CANO_i = 1.5 + 0.1 \times X_i + \epsilon_i$ where the standard deviation of the random normal error ϵ_i is a normal distributed: $\epsilon \sim \mathcal{N}(0, 0.45)$. 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.

$$FeNO = C_{aw} + (C_A - C_{aw}) \times e^{-\frac{Daw}{flow}}$$

```
# 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=200,           # 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
                       # function to generate participant-level covariate X
                       # as function of number of study participants
                       Xfn=function(n){rnorm(n,0,1)},
                       # vector of regression coefficients relating X to NO parameters,
                       # in this order: CA, logCaw, logDaw
                       beta=truebeta,
                       # vector of flow rates in multiple flow FeNO protocol
                       Flow=c(rep(30,2),rep(50,2),rep(100,2),rep(300,2)),

                       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 logCaw, CA
                       coralphalogCawlogDaw = -0.35,# correlation of logCaw, logDaw
                       coralphalogDawCa     = -0.38 # correlation of 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

out2<-DataGeneratorCS_UbN(Ndat=10,            # number of study participants
                          alpha=c(1.5,3.5,2.5),# vector of NO parameter population means, when X=0, in
                          Xfn=function(n){rnorm(n,0,1)},# function to generate participant-level covari
                          beta=c(0.1,0.1,0.1),
                          flowTarget=c(rep(30,2),rep(50,2),rep(100,2),rep(300,2)),# vector of regressio
                          Flow=FlowFun(10,0.9), # Must be a Ndat*max length of flow matrix, missing val
                          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
)
dat <- out2$dat
datX<- out2$datX
```

- The resultant dataset is in the usual 'long' data format:

id	eno	logeno	flow	flowTarget
1	18.550100	2.920475	31.01645	30
1	18.016264	2.891275	30.58051	30
1	14.224177	2.654943	48.81613	50
1	12.760105	2.546324	50.36992	50
1	7.676364	2.038146	98.97814	100
1	8.119653	2.094287	99.61364	100
1	3.702673	1.309055	298.62592	300
1	3.665347	1.298923	301.72294	300
2	3.507760	1.254978	30.90222	30
2	3.284286	1.189149	29.16724	30

- The intercepts and slopes of the linear regression moedls: NOparam ~ X of the simulated data, used in the plotting section:

	Ca	logCaw	logDaw
(Intercept)	1.6343842	3.9089567	2.3183797
datX[, 2]	0.1965633	0.2297409	-0.0420262

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 0.92 (95% CI: -0.39,2).

For 1 unit increase in covariate X, CANO increases 0.18, (95% CI: -0.88, 1.25).

Interpretation for CawNO and DawNO from their log version:

The population mean CawNO is 3.69 (95% CI: 3.17,4).

For 1 unit increase in covariate X, CawNO increases -13.9%, (95% CI: -43.28%, 30.69%).

The population mean DawNO is 2.84 (95% CI: 1.93,4).

For 1 unit increase in covariate X, DawNO increases 22.04%, (95% CI: -41.41%, 154.19%).

- These interpretations are similar for following results.

2.2 TS_HMA: Two-stage Högman & Merilänen Algorithm

*add target flow: low, medium, high comments

```
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)
```

2.3 TS_NLME: Two-stage nonlinear mixed effects model

```
if (!file.exists("TSNLME_cc.Rdata")) {
    set.seed(2020)
    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 version
    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)
```

2.4 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 start
    # statement of the function
    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.

```
# Stage I if(!file.exists('TSHB_cc.Rdata')){
set.seed(2020)
source("TS_HB.R")


TSHB_S1 <- TSHB_iter_UB(beta0_prior = c(2, 4, 3), rhat = 1.1,
    addon.iter = 60, Max_update = 10, n.final = 60, N.iterT = 550,
    N.burnin = 500, N.thinM = 1, N.chain = 3, flow = flow, dat = dat,
    tracing = c("beta0_Ca", "beta0_logCaw", "beta0_logDaw", "sdCa",
        "sdlogCaw", "sdlogDaw", "corlogCawCa", "corlogCawlogDaw",
        "corlogDawCa", "sigma_c"))
save(TSHB_S1, file = "TSHB_cc_ub.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$id <- as.numeric(unlist(lapply(rownames(TSHB_S1_dat),
    function(x) strsplit(strsplit(x, "\\[")[[1]][2], "\\]")[[1]])))
TSHB_S1_dat <- TSHB_S1_dat[order(TSHB_S1_dat$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)

	2.5%	mean	97.5%	Rhat
beta0_Ca	-993.9257764	-196.2995768	1.8616967	2.913813
beta0_logCaw	2.7353890	3.8047858	4.4149104	2.105715
beta0_logDaw	1.4905670	2.3803340	2.9132965	7.130810
sdCa	0.2885420	27.5031470	169.7547641	8.627952
sdlogCaw	0.2956694	0.7092545	1.4566963	3.829610
sdlogDaw	0.1425450	0.4524099	1.0625146	2.793215
corlogCawCa	-0.0871290	0.6430785	0.9482338	1.458423
corlogCawlogDaw	-0.3722056	0.6159421	0.9770478	1.897797
corlogDawCa	-0.2480491	0.4774214	0.8763660	1.382292
sigma_c	0.0864459	0.1025492	0.1201962	1.068549

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_UB(beta0_prior = c(2, 4, 3), betaC_prior = c(0.1,
    0.1, 0.1), rhat = 1.1, addon.iter = 100, Max_update = 1,
    n.final = 100, N.iterT = 1200, N.burnin = 1000, N.thinM = 1,
    N.chain = 3, flow = flow, dat = dat, X = datX[, 2], tracing = c("beta0_Ca",
        "beta0_logCaw", "beta0_logDaw", "beta1_Ca", "beta1_logCaw",
        "beta1_logDaw", "sdCa", "sdlogCaw", "sdlogDaw", "corlogCawCa",
        "corlogCawlogDaw", "corlogDawCa", "sigma_c"))
save(UHB_sim, file = "UHB_cc_ub.Rdata")
# }else{ load('UHB_cc.Rdata') }
```

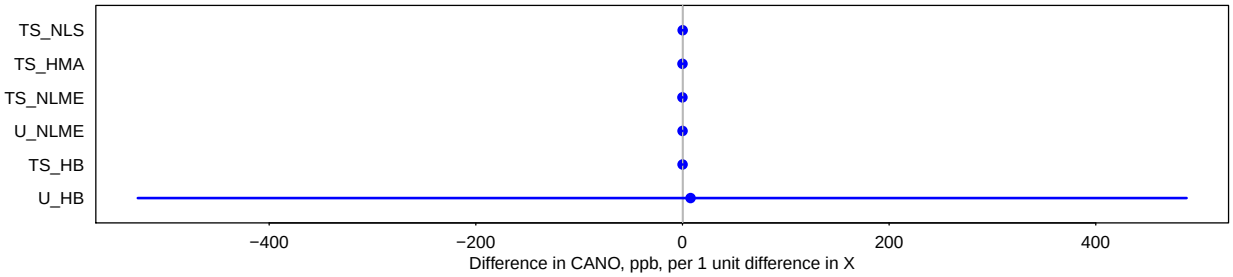
2.6.ex Convergence diagnostic for U_HB via Rhat

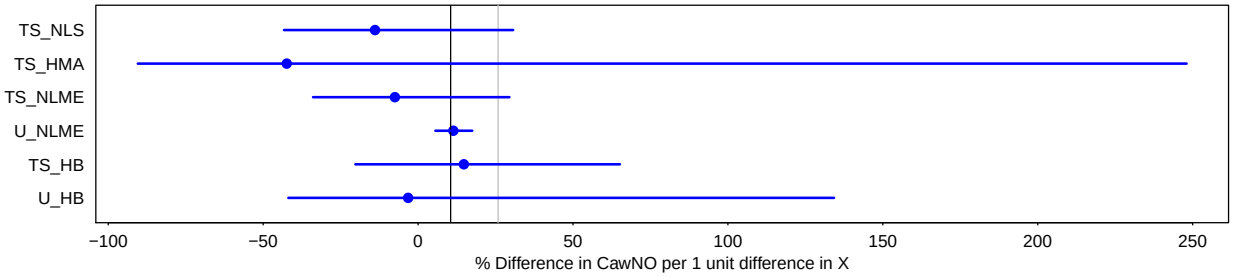
- Print out the estimation (95% CL) and Rhat (converge if <1.1)

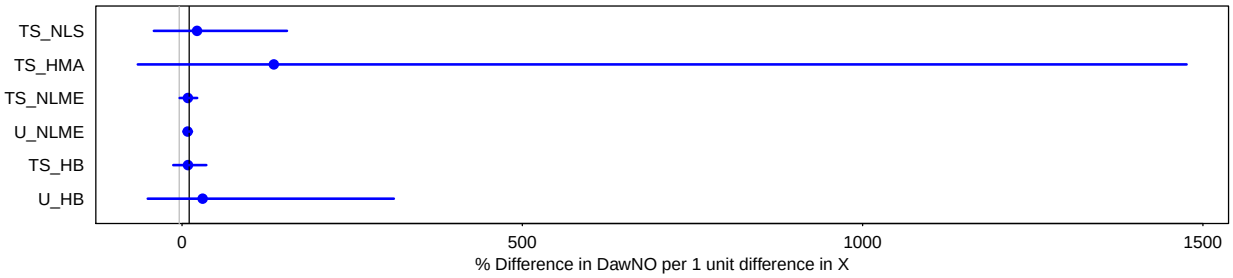
	2.5%	mean	97.5%	Rhat
beta0_Ca	-2484.8934838	-1114.5250868	-204.6047328	1.053201
beta0_logCaw	3.5367370	4.4002989	5.5108719	5.086159
beta0_logDaw	0.4978336	1.8039490	2.6926961	4.485909
beta1_Ca	-527.1484290	7.8222306	487.8997552	1.382090
beta1_logCaw	-0.5422396	-0.0325873	0.8512145	5.371599
beta1_logDaw	-0.7005638	0.2644322	1.4139378	3.744597
sdCa	17.9203646	65.1039120	151.9736111	1.062158
sdlogCaw	0.1019269	0.4961355	1.2141681	5.502402
sdlogDaw	0.2021531	0.8597509	1.8339613	2.600239
corlogCawCa	-0.7908604	-0.0153289	0.9222895	1.162105
corlogCawlogDaw	-0.8490496	-0.1363497	0.5805836	1.727325
corlogDawCa	-0.7196850	0.1077570	0.9033382	1.089476
sigma_c	0.0883555	0.1056248	0.1268269	1.062361

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
- Refernce lines: Black lines indicate the values used in data simulation. Gray lines indicate the values obtained by regression on the simulated NO parameters and Xs.


TS_NLS
TS_HMA
TS_NLME
U_NLME
TS_HB
U_HB
−400
−200
0
200
400
Difference in CANO, ppb, per 1 unit difference in X


TS_NLS
TS_HMA
TS_NLME
U_NLME
TS_HB
U_HB
−100
−50
0
50
100
150
200
250
% Difference in CawNO per 1 unit difference in X


TS_NLS
TS_HMA
TS_NLME
U_NLME
TS_HB
U_HB
0
500
1000
1500
% Difference in DawNO per 1 unit difference in X