Standardized effect sizes in 3 levels with between-study heterogeneity

Han Bossier

1-2-2018

Contents

1 Introduction

In this report, I calculate the true values for parameter estimates at each level of a 3-level statistical analysis of fMRI data. We then compare these with observed values using a Monte-Carlo simulation study. We start at subject level, generating a time series. Then we combine subjects using *OLS*. Finally we combine the results using a random effects meta-analysis model. We generate data on a small grid of voxels (no smoothing, pure white noise). But only save one voxel in the center with the activation. We use 1000 Monte-Carlo simulations to check our expectations.

Some general parameters for this report:

- 29 subjects per study
- 50 studies per MA
- No between-subject heterogeneity
- WITH between-study heterogeneity

2 Theory

In this report, we assume independence between all three levels (eg. no subjects are resampled).

2.1 First level

We generate for each subject a time series using the General Linear Model. We have for each voxel the following model:

$$Y = \beta_0 + \beta_1 X + \varepsilon \tag{1}$$

In this report, we set $\beta_0 = 100$, $\beta_1 = 3$, X is a design matrix obtained by convoluting an ON/OFF blocked design with a canonical HRF and $\varepsilon \sim N(0, \sigma^2)$, where $\sigma = 100$. Note that σ corresponds to within-subject variability. The observed values in Y correspond to the time series measured as the BOLD signal. We fit the model using OLS.

From theory, we know that:

$$\mathrm{Var}(\hat{\beta}) = \sigma^2 (X'X)^{-1} \tag{2}$$

and

$$\mathrm{t} = \frac{\hat{\beta}}{\sqrt{\mathrm{Var}(\hat{\beta})}}. \tag{3}$$

2.1.1 Design matrix

To obtain X, we use *neuRosim*. In the section/code below, the design matrix is denoted as *pred - base*. Furthermore β_0 is denoted as *base*. We fit a simple linear regression model (only one predictor) to estimate the average effect of the only condition in our design. First we set the following parameters:

- TR

- number of scans
- onsets of blocks ON
- duration of blocks (in sec)

Then we use the functions `neuRosim::simprepTemporal` to generate the parameters and `neuRosim::simTSfmri` to generate the time series of the design matrix. This results in a vector X of length T, where T stands for time. In the section *Monte-Carlo simulation set-up*, we also plot $\beta_1 X$ denoted as the *scaled signal*.

Note that we will calculate the value for $(X'X)^{-1}$ in the section *Monte-Carlo simulation set-up*, after setting up the design matrix. It is roughly equal to 0.03.

2.1.2 True values

We will save the estimated parameters for a number of subjects in the Monte-Carlo simulation setting below. Over all **M** simulations, we should get:

$$\Leftrightarrow \quad \sum_{m=1}^{M} \frac{\hat{\beta}_m}{M} \approx \mathrm{E}(\hat{\beta}) \tag{4}$$
$$= \beta = 3$$

$$\Leftrightarrow \quad \sum_{m=1}^{M} \frac{\widehat{\mathrm{Var}(\beta_m)}}{M} \approx \mathrm{E}(\widehat{\mathrm{Var}(\beta)}) \tag{5}$$
$$= \sigma^2 (X'X)^{-1} = 100^2 (X'X)^{-1}$$

2.2 Second level

At the second level, we have for each voxel the following model:

$$Y_G = \beta_0^* + \beta_1^* X_G + \varepsilon^*. \tag{6}$$

Now $Y_G = \hat{\beta}_1$, the vector of estimated first level parameters. In our case, $\beta_0^* = 0$ and X_G is the second level design matrix which is equal to a column of 1's with length equal to the number of subjects. Furthermore, $\varepsilon^* \sim N(0, \sigma^{*2})$. We fit the model using OLS. Note that as the OLS estimate for β_1 is equal to the average of Y, the true value for $\beta_1^* = 3$, which is the average over all first level estimates.

Note that the variance of ε^* contains two components. First we have between subject variability σ_G^2. Second, since we use $Y_G = \hat{\beta}_1$ as input, we have variability of the estimated parameters at the first level. This is denoted as $\mathrm{Var}_\beta(\hat{\beta}_1)$. From theory, we know what the variance of the estimated first level β parameters is. Since we do not induce between subject variability in our simulations, we get:

$$\mathrm{Var}(\varepsilon^*) = \sigma_G^2 + \mathrm{Var}_\beta(\widehat{\beta_1}) \tag{7}$$
$$= 0 + \sigma^2 (X'X)^{-1}$$

2.2.1 True values

We will save the estimates for β_G and its variance of each study in one meta-analysis for some Monte-Carlo runs. Over all **M** simulations, we should get:

$$\Leftrightarrow \quad \sum_{m=1}^{M} \frac{\hat{\beta}^*_m}{M} \approx \mathrm{E}(\hat{\beta}^*) \tag{8}$$
$$= \beta^* = \beta = 3$$
$$\Leftrightarrow \quad \sum_{m=1}^{M} \frac{\widehat{\mathrm{Var}(\beta^*_m)}}{M} \approx \mathrm{E}(\widehat{\mathrm{Var}(\beta^*)}) \tag{9}$$
$$= \sigma^{*2}(X'_G X_G)^{-1}$$
$$= \sigma^2 (X'X)^{-1}(X'_G X_G)^{-1}$$

Note that $(X'_G X_G)^{-1}$ will be fixed as we always have the same amount of subjects. It is equal to `0.0344828`

2.2.2 Standardized effect sizes

Standardized effect sizes (such as Cohen's d) at the second level (study level) are generally calculated by dividing the mean effect with the standard deviation. Or by multiplying the t-value with $\frac{1}{\sqrt{n}}$. In our case, this corresponds to:

$$\begin{aligned} \mathrm{d} &= \frac{\mathrm{t}}{\sqrt{n}} \\ &= \mathrm{t}\left[\sqrt{(X'_G X_G)}\right]^{-1} \\ &= \frac{\hat{\beta}^*}{\left[\sqrt{\sigma^2 (X'X)^{-1}(X'_G X_G)^{-1}}\right]} \sqrt{(X'_G X_G)^{-1}}. \end{aligned} \tag{10}$$

Or simplified:

$$\mathrm{d} = \frac{\hat{\beta}^*}{\sigma\sqrt{(X'X)^{-1}}}. \tag{11}$$

We are mainly interested in Hedges' g. This is obtained by multiplying d with a correction factor J:

```
NeuRRoStat::corrJ
```

```
function(N){
  1-(3/((4*(N-1))-1))
}
<environment: namespace:NeuRRoStat>
```

The expected value of Hedges' g over all simulations is equal to:

$$\frac{\sum_{m=1}^{M} \mathrm{g}_m}{M} = \frac{3}{100\sqrt{(X'X)^{-1}}} \times J. \tag{12}$$

2.3 Third level

2.3.1 General linear model

Before estimating the population effect size, we can also run a third level GLM using OLS. Again, we have for each voxel the following model:

$$Y_M = \beta_0^{**} + \beta_1^{**} X_M + \varepsilon^{**}. \tag{13}$$

Now $Y_M = \hat{\beta}_1^*$, the vector of estimated second level parameters. In our case, $\beta_0^{**} = 0$ and X_M is the third level design matrix which is equal to a column of 1's with length equal to the number of studies. Furthermore, $\varepsilon^{**} \sim N(0, \sigma^{**2})$. We fit the model using OLS. Note that as the OLS estimate for β_1^{**} is again equal to the average over all first level estimates.

The variance of ε^{**} now contains three components. First we have between study variability σ_M^2. Then we have variability of the estimated second level parameter estimates, denoted as $\text{Var}_\beta(\hat{\beta}_1^*)$. Finally we have variability of the estimated parameters at the first level, denoted as $\text{Var}_\beta(\hat{\beta}_1)$. Since we do not induce between subject variability in our simulations ($\sigma_G^2 = 0$), we get:

$$\begin{aligned} \text{Var}(\varepsilon^{**}) &= \sigma_M^2 + \text{Var}_\beta(\widehat{\beta_1^*}) + \text{Var}_\beta(\widehat{\beta_1}) \\ &= \sigma_M^2 + \sigma_G^2 (X_G' X_G)^{-1} + \sigma^2 (X'X)^{-1} \\ &= \sigma_M^2 + 0 + \sigma^2 (X'X)^{-1} \end{aligned} \tag{14}$$

2.3.1.1 True values

We will set the between-study variability (σ_M^2) to 100.

The estimates for β_M and its variance are saved. Over all $\mathbf{M}$ simulations, we should get:

$$\Leftrightarrow \quad \sum_{m=1}^{M} \frac{\hat{\beta}_m^{**}}{M} \approx \text{E}(\hat{\beta}^{**}) \tag{15}$$

$$= \beta^{**} = \beta^* = \beta = 3$$

$$\Leftrightarrow \quad \sum_{m=1}^{M} \frac{\widehat{\text{Var}(\beta_m^{**})}}{M} \approx \text{E}(\widehat{\text{Var}(\beta^{**})}) \tag{16}$$

$$= \sigma^{**2} (X_M' X_M)^{-1}$$

$$= \sigma_M^2 (X_M' X_M)^{-1} \sigma^2 (X'X)^{-1}$$

Note that $(X_M' X_M)^{-1}$ will be fixed as we always have the same amount of studies It is equal to `0.02`

2.3.2 Population effect size

The estimated population effect size (μ) at the third level (meta-analysis) corresponds to the weighted average of all standardized effect sizes at the second level. We use a random-effects model with the method of moments estimator for between-study heterogeneity. The weights correspond to the inverse of the sum of within- and between study variability.

Note that we use the same amount of subjects for each study. Hence the expected value of the within-study variability will be equal for each study. Asymptotically, all weights are the same and the weighted average is equal to an unweighted average. Hence the true value for μ, the population effect is equal to:

$$\begin{aligned} \mu &= E(g) \\ &= g, \end{aligned} \tag{17}$$

as $E(g)$ is a constant in our set-up.

The question now is, what will the estimated between-study heterogeneity of the population effect size be? I don't think it is that $\tau^2 = \sigma_M^2$. Could it be that it is equal to the variance of the estimated third level β^{**} parameter? In other words:

$$\begin{aligned} \tau^2 &= \text{Var}(\beta^{**}) \\ &= \sigma_M^2 (X_M' X_M)^{-1}. \end{aligned} \tag{18}$$

Let us find out.

3 Monte-Carlo simulation set-up

In the following code section, we define:

- number of subjects
- number of studies
- characteristics of signal in each subject
- characteristics of image

```
# number of simulations
nsim <- 1000

# Number of subject: median sample size at 2018 = 28.5 (Poldrack et al., 2017)
nsub <- 29
# Number of studies
nstud <- 50

# Signal characteristics
TR <- 2
nscan <- 200
total <- TR*nscan
on1 <- seq(1,total,40)
onsets <- list(on1)
duration <- list(20)
# Base of signal: also intercept in GLM
base <- 100
# %BOLD change
BOLDC <- 3

# Image characteristics
DIM <- c(9,9,9)
voxdim <- c(3.5, 3.5, 3.51) # Voxelsize
```

```
ext <- 1                    #  Extend
nregio <- 1
TrueLocations <- c(5,5,5)   # Center of image and activation
# In 1D-vector location
TrueLocVec_tmp <- array(NA, dim = DIM)
TrueLocVec_tmp[TrueLocations[1],
               TrueLocations[2],
               TrueLocations[3]] <- 1
TrueLocVec <- as.numeric(which(array(TrueLocVec_tmp, dim = prod(DIM)) == 1,
                               arr.ind = TRUE))
```

In the following section, we create the ground truth area. To do so, we first need to generate a temporary design with a true signal (neuRosim needs a design to generate an area).

```
# We generate a temporary design for getting a true signal
truthdesign <- neuRosim::simprepTemporal(1, 1, onsets = 1, effectsize = 1,
                               durations = 1, TR = 1, acc = 0.1)

# Now use this to get a sphere shaped area
area <- neuRosim::simprepSpatial(regions = 1, coord = list(TrueLocations),
                        radius = ext, form = "sphere", fading = 0)
truth <- neuRosim::simVOLfmri(design = truthdesign, image = area,
                     dim = DIM, SNR = 1, noise = "none")[,,,1]
GroundTruth <- ifelse(truth > 0, 1, 0)
```

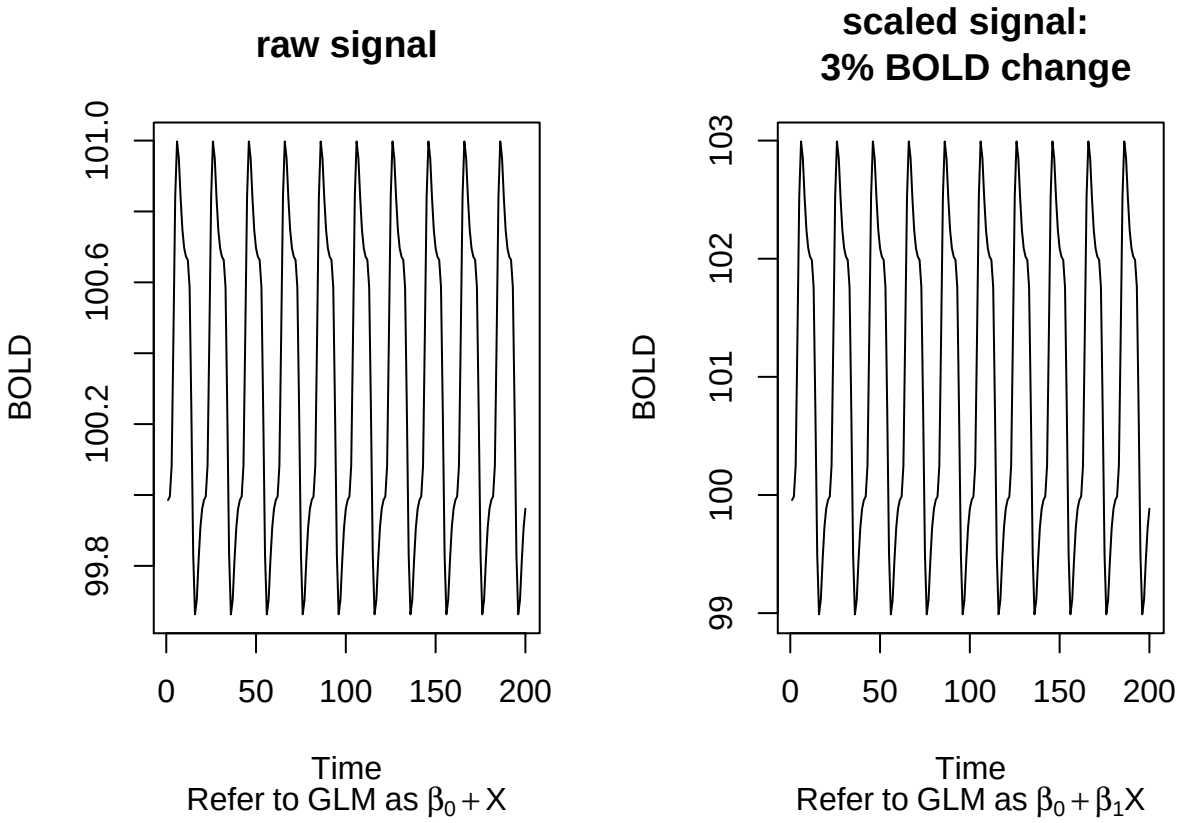
Next we create a true signal (no noise). We convert the signal to the appropriate scale (we choose a base signal of 100) and make sure the amplitude of the signal corresponds to % BOLD signal. This is done through:

$$\text{true signal} = \%BOLD \times (\text{raw signal} - 100) + 100$$

This scales the amplitude of *raw signal* to % BOLD change.

```
# Generating a design matrix
X <- neuRosim::simprepTemporal(total,1,onsets = onsets,
                      effectsize = 1, durations = duration,
                      TR = TR, acc = 0.1, hrf = "double-gamma")

# Generate time series for ONE active voxel: predicted signal, this is the design
pred <- neuRosim::simTSfmri(design=X, base=100, SNR=1, noise="none", verbose=FALSE)

# Now we create the BOLD signal by converting to % BOLD signal changes
# Need to be in appropriate scale
# Note that we can rewrite the following as: beta_0 + beta_1 * X, with:
  # beta_0 = base
  # beta_1 = BOLDC
  # X = (pred - base)
signal_BOLDC <- BOLDC * (pred-base) + base

## Design parameters: will need to use this for calculating var(beta)
# Extend the design matrix with an intercept
xIN <- cbind(1,pred)

# Contrast: not interested in intercept
CONTRAST <- matrix(c(0,1),nrow=1)
```

```r
# Calculate (X'X)^(-1) with contrast
design_factor <- CONTRAST %*% (solve(t(xIN) %*% xIN )) %*% t(CONTRAST)


# Plot
par(mfrow = c(1,2))
plot(pred, type = 'l', main = 'raw signal',
     sub = expression(Refer ~ to ~ GLM ~ as ~ beta[0] + X),
     ylab = 'BOLD',
     xlab = 'Time')
plot(signal_BOLDC, type = 'l', main = paste0('scaled signal: \n ', BOLDC,'% BOLD change'),
     sub = expression(Refer ~ to ~ GLM ~ as ~ beta[0] + beta[1] * X),
    ylab = 'BOLD', xlab = 'Time')
```



To generate data, we start with setting variance parameters:

```r
# Sigma of white noise
whiteSigma <- 100
# Between study variability
tau <- 10
```

Now we generate data for each subject and study. We start with creating a value for the study-specific BOLD effect. This will induce between-study variability. Note, we do not add between-subject heterogeneity.

Then we take the study true signal and add white noise to the time series.

The following **R** code is given as example.

```r
# Seed
set.seed(pi)
```

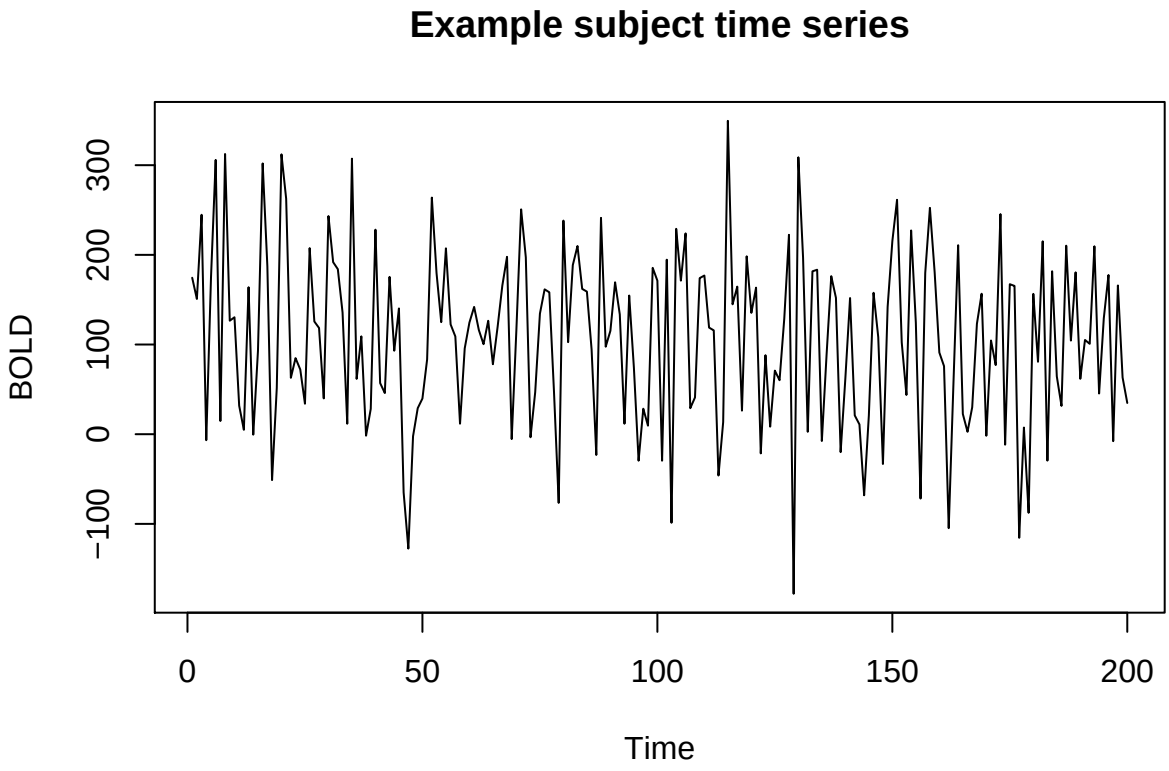
```r
# Data frame with results:
MAvec <- tibble(sim = integer(),
                      Wavg = numeric(),
                      sigma = numeric(),
                      nstud = numeric())
# Empty vectors
COPE <- VARCOPE <- array(NA,dim=c(prod(DIM),nsub))
STCOPE <- STVARCOPE <- STVALUE <- array(NA,dim=c(prod(DIM),nstud))

# For loop over studies
for(t in 1:nstud){
    # Create the delta: subject specific true effect, using tau as between-study
    #   heterogeneity.
    # This is done by generating a study specific BOLD signal at center of activation.
    BOLDCS <- BOLDC + rnorm(n = 1, sd = tau)

    # Need to be in correct scale
    signal_BOLDCS <- BOLDCS * (pred-base) + base

    # Now get the unsmoothed (raw) signal for this study
    StudData <- GroundTruth %o% signal_BOLDCS

    # Transform to voxel * nscan matrix (instead of 4D image)
    StudDataT <- array(StudData, dim = c(prod(DIM), nscan))

  # For loop over nsub
  for(s in 1:nsub){
    # Multilevel data generation:
    # White noise around signal in each voxel.
    # We take study signal and add white noise (using apply)
    # No smoothing of noise as we are unable to calculate the true value of
    #   the effect size if we do so!!
    SubjData <- t(apply(StudDataT, MARGIN = 1,
          FUN = function(voxel){voxel + rnorm(n = nscan, mean = 0,
                                              sd = whiteSigma)}))

    # Transform it to correct dimension (Y = t x V)
    Y.data <- t(SubjData)

    ###################
    #### ANALYZE DATA: 1e level GLM
    ###################
    # COPE (beta 1) --> fit GLM
    model.lm <- lm(Y.data ~ pred)
    b1 <- coef(model.lm)['pred',]
    COPE[,s] <- b1

    # VARCOPE --> estimate residual (we need to extend the design matrix with an intercept)
    xIN <- cbind(1,pred)
    BETA <- coef(model.lm)
    res <- (t(Y.data - xIN %*% BETA) %*% (Y.data - xIN %*% BETA))/(nscan - 2)
    res <- diag(res)
    # Contrast: not interested in intercept, remove it again
```

```
    CONTRAST <- matrix(c(0,1),nrow=1)
    # Calculate varcope
    VARCOPE[,s] <- CONTRAST %*% (solve(t(xIN) %*% xIN )) %*% t(CONTRAST) %*% res

    # Clean objects
    rm(model.lm, b1, xIN, BETA, res, CONTRAST)
  }

  ##################
  #### GROUP ANALYSIS: 2e level using OLS
  ##################

  # Group COPE
  STCOPE[,t] <- apply(COPE, 1, mean)

  # Group Y variable
  Gr.Y <- t(COPE)

  # Group VARCOPE:
  # First constant: (X'X)^-1
  # X is the design matrix, column of 1's
  GrX <- matrix(1, nrow = nsub)
  GrCt <- solve(t(GrX) %*% GrX)
  # Residuals
  GrRes <- (t(Gr.Y - GrX %*% matrix(STCOPE[,t], nrow = 1))) %*%
            (Gr.Y - GrX %*% matrix(STCOPE[,t], nrow = 1))/
            (nsub - 1)
  GrRes <- diag(GrRes)
  # Denominator: checked using t.test
  STVARCOPE[,t] <- GrRes %*% GrCt

  # T value
  STVALUE[,t] <- STCOPE[,t] / sqrt(STVARCOPE[,t])

  # Clean objects
  rm(Gr.Y, GrRes)
}

# MA on voxel 365
voxCOPE <- STCOPE[TrueLocVec,]
voxVARCOPE <- STVARCOPE[TrueLocVec]
voxTval <- STVALUE[TrueLocVec,]

# Hedges g
voxHedgeG <- hedgeG(t = voxTval, N = nsub)

# Variance of g
voxVarG <- varHedge(voxHedgeG, N = nsub)

# Weighted average
WA <- as.numeric(rma(yi = voxHedgeG, vi = voxVarG,
    method = 'DL')$beta)
```

```r
# Bind results in data frame
MAvec <- data.frame(sim = 1,
           Wavg = WA,
           sigma = whiteSigma,
           nstud = nstud) %>%
  bind_rows(MAvec,. )
```

As an example, this is the time series of the latest subject at the voxel with activation:

```r
par(mfrow = c(1,1))
plot(SubjData[365,], type = 'l', main = 'Example subject time series',
     ylab = 'BOLD', xlab = 'Time')
```



4 Simulation Results

To check our results, we have run the code from section *Monte-Carlo simulation set-up* 1000 times. We saved the following objects in each simulation:

- the estimated β and its variance parameter of the latest 29 subjects from the latest simulated study. This is to check the estimated parameters at the first level.
- the estimated β parameter and its variance from the latest 50 studies in the latest generated MA. To check parameter estimates at second level.
- the estimated weighted average, this is the parameter of interest at the third level.

First we read in the **R** objects.

```r
# Location of data
LocDat <- '/Volumes/2_TB_WD_Elements_10B8_Han/PhD/Simulation/Results/VectorMA/Results_act'

# Empty data frames
```

```
VectorMA <- tibble(sim = integer(),
                   Wavg = numeric(),
                   EstTau2 = numeric(),
                   sigma = numeric(),
                   tau = numeric(),
                   nstud = numeric())
VectorGLM <- tibble(sim = integer(),
                   GLMcope = numeric(),
                   EstRes = numeric(),
                   sigma = numeric(),
                   tau = numeric(),
                   nstud = numeric())
studSimDat <- tibble(Value = numeric(),
                     param = factor(levels= c('COPEstud', 'VARCOPEstud')),
                     studID = integer(),
                     voxID = integer())
subSimDat <- tibble(Value = numeric(),
                     param = factor(levels= c('COPEsub', 'VARCOPEsub')),
                     subID = integer(),
                     voxID = integer())

# For loop over the split-up simulations (every .rda file contains 10 simulations)
for(i in 1:c(nsim/10)){
  # Read in subject data: latest subject of latest study in MA
  subSimDat <- readRDS(paste0(LocDat, '/subSimDat_', i, '.rda')) %>%
    bind_rows(subSimDat,. )

  # Read in study data: latest study in MA
  studSimDat <- readRDS(paste0(LocDat, '/studSimDat_', i, '.rda')) %>%
    bind_rows(studSimDat,. )

  # Read in MA data
  VectorMA <- readRDS(paste0(LocDat, '/MAvec_', i, '.rda')) %>%
    bind_rows(VectorMA,. )

  # Read in GLM data
  VectorGLM <- readRDS(paste0(LocDat, '/GLMvec_', i, '.rda')) %>%
    bind_rows(VectorGLM,. )
}
```

Now let us check the results.

4.1 First level

We have 29 subjects from which we record the COPE, VARCOPE and calculate the T-statistic. We only saved these values for every 10th simulation (small coding mistake). Hence we get `nsim/10` times $29 = 2900$ values. We average over all these values to get the empirical estimate.

```
firstRes <- subSimDat %>%
  mutate(simID = rep(1:c(nsim/10), each = nsub*2)) %>%
  spread(key = param, value = Value) %>%
  mutate(Tval = COPEsub / sqrt(VARCOPEsub)) %>%
  # First summarise within each simulation => over all subjects within a study
```

```
  group_by(simID) %>%
  summarise(AvgCopeS = mean(COPEsub),
            AvgVarCopeS = mean(VARCOPEsub),
            AvgTValS = mean(Tval)) %>%
  ungroup() %>%
  # Now over all simulations
  summarise(AvgCope = mean(AvgCopeS),
          AvgVarCope = mean(AvgVarCopeS),
          AvgTVal = mean(AvgTValS))
firstRes
```

```
# A tibble: 1 x 3
  AvgCope AvgVarCope AvgTVal
    <dbl>      <dbl>   <dbl>
1    3.36        254   0.211
```

Now we compare with the expected values:

```
Ebeta1 <- BOLDC
EvarBeta1 <- whiteSigma**2 * design_factor
Et1 <- Ebeta1/sqrt(EvarBeta1)
# Print in data frame
knitr::kable(data.frame(AvgCope = Ebeta1, AvgVarCope = EvarBeta1, AvgTVal = Et1,
           Value = 'Expected', Level = 'First', stringsAsFactors = FALSE) %>%
  bind_rows(., mutate(firstRes, Value = 'Observed', Level = 'First')),
  caption = "First level analysis")
```

Table 1: First level analysis

AvgCope	AvgVarCope	AvgTVal	Value	Level
3.000000	254.3824	0.1880952	Expected	First
3.358907	253.6302	0.2112117	Observed	First

Note that this may seem off. However, we have added between-study variability. So each study has its own true value. Here, we look at subjects at level 1, while the true value is calculated at the third level...

4.2 Second level

We have `50` studies from which we record the COPE, VARCOPE and calculate the T-statistic. We only saved these values for every 10th simulation (small coding mistake). Hence we get `nsim/10` times `50` = `5000` values. We average over all these values to get the empirical estimate.

Let us also re-calulculate Hedges' g, using the estimate for the t-value.

```
secondRes <- studSimDat %>%
  mutate(simID = rep(1:c(nsim/10), each = nstud*2)) %>%
  spread(key = param, value = Value) %>%
  mutate(Tval = COPEstud / sqrt(VARCOPEstud)) %>%
  mutate(HedgesG = hedgeG(t = Tval, N = nsub)) %>%
  # Average within all MA's => in each simulation
  group_by(simID) %>%
  summarise(AvgCopeS = mean(COPEstud),
            AvgVarCopeS = mean(VARCOPEstud),
            AvgTValS = mean(Tval),
```

```
            AvgGS = mean(HedgesG)) %>%
  ungroup() %>%
  # Average across simulations
  summarise(AvgCope = mean(AvgCopeS),
            AvgVarCope = mean(AvgVarCopeS),
            AvgTVal = mean(AvgTValS),
            AvgG = mean(AvgGS))
secondRes
```

```
# A tibble: 1 x 4
  AvgCope AvgVarCope AvgTVal  AvgG
    <dbl>      <dbl>   <dbl> <dbl>
1    3.23       8.76    1.14 0.206
```

We compare again with the expected values:

```
# First constant: (X_G'X_G)^-1
# X is the group design matrix, column of 1's
GrX <- matrix(1, nrow = nsub)
GrCt <- solve(t(GrX) %*% GrX)
Ebeta1S <- Ebeta1
EvarBeta1S <- EvarBeta1 * GrCt
Et1S <- Ebeta1S/sqrt(EvarBeta1S)
# Also Hedges' g
EhedgeG <- Ebeta1S/(whiteSigma * sqrt(design_factor)) * corrJ(N = nsub)
EhedgeG
```

```
          [,1]
[1,] 0.1830116
```

```
# Check with formula
CheckHedgeG <- hedgeG(t = Et1S, N = nsub)
CheckHedgeG
```

```
          [,1]
[1,] 0.1830116
```

```
# Print in data frame
knitr::kable(data.frame(AvgCope = Ebeta1S, AvgVarCope = EvarBeta1S,
                        AvgTVal = Et1S, AvgG = EhedgeG,
           Value = 'Expected', Level = 'Second', stringsAsFactors = FALSE) %>%
  bind_rows(., mutate(secondRes, Value = 'Observed', Level = 'Second')),
  caption = "Second level analysis")
```

Table 2: Second level analysis

AvgCope	AvgVarCope	AvgTVal	AvgG	Value	Level
3.000000	8.771807	1.012924	0.1830116	Expected	Second
3.226255	8.762535	1.137993	0.2056087	Observed	Second

Again, this may seem off due to studies having their own true value.

4.3 Third level

Finally, we have 1000 number of Monte-Carlo simulations where we save the estimate for the weighted average and the estimated between-study heterogeneity (τ^2).

```
thirdRes <-
  VectorMA %>%
  summarise(AvgMu = mean(Wavg),
            AvgTau2 = mean(EstTau2))
thirdRes
```

```
# A tibble: 1 x 2
  AvgMu AvgTau2
  <dbl>   <dbl>
1 0.176   0.285
```

Comparing $\hat{\mu}$ with μ and $\widehat{\tau}^2$ with τ^2, we have:

```
MAX <- matrix(1, nrow = nstud)
MACt <- solve(t(MAX) %*% MAX)
ETau2 <- tau^2 * MACt
  #tau^2 - (tau^2 * MACt * whiteSigma^2 * design_factor)
#EvarBeta1*GrCt
EhedgeGMA <- EhedgeG

# Print in data frame
knitr::kable(data.frame(AvgMu = EhedgeGMA, AvgTau2 = ETau2,
           Value = 'Expected', Level = 'Third', stringsAsFactors = FALSE) %>%
  bind_rows(., mutate(thirdRes, Value = 'Observed', Level = 'Third')),
  caption = "Third level analysis - meta-analysis")
```

Table 3: Third level analysis - meta-analysis

AvgMu	AvgTau2	Value	Level
0.1830116	2.0000000	Expected	Third
0.1757591	0.2849729	Observed	Third

We now do the same for the GLM.

```
thirdResGLM <-
  VectorGLM %>%
  summarise(AvgCOPE = mean(GLMcope),
            AvgSigma2 = mean(EstRes))
thirdResGLM
```

```
# A tibble: 1 x 2
  AvgCOPE AvgSigma2
    <dbl>     <dbl>
1    2.94       108
```

Comparing $E(\hat{\beta})$ with β and the unexplained variance of the GLM, we have:

```
# Print in data frame
knitr::kable(data.frame(AvgCOPE = BOLDC, AvgSigma2 = tau**2,
           Value = 'Expected', Level = 'Third', stringsAsFactors = FALSE) %>%
  bind_rows(., mutate(thirdResGLM, Value = 'Observed', Level = 'Third')),
  caption = "Third level analysis - GLM")
```

Table 4: Third level analysis - GLM

AvgCOPE	AvgSigma2	Value	Level
3.000000	100.0000	Expected	Third
2.944897	107.9663	Observed	Third

Hmm, it does seem to be off a little bit.

5 Conclusion

Not there yet.