Standardized effect size in 3 levels

Han Bossier

23-1-2018

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
- No between-study heterogeneity

2 Theory

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:

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

and

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

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:

$$\begin{aligned}
\Leftrightarrow \quad \sum_{m=1}^{M} \frac{\hat{\beta}_m}{M} &\approx \mathrm{E}(\hat{\beta}) && (4)\\
&= \beta = 3 \\
\Leftrightarrow \quad \sum_{m=1}^{M} \frac{\widehat{\mathrm{Var}(\beta_m)}}{M} &\approx \mathrm{E}(\widehat{\mathrm{Var}(\beta)}) && (5)\\
&= \sigma^2 (X'X)^{-1} = 100^2 (X'X)^{-1} \\
\Leftrightarrow \quad \sum_{m=1}^{M} \frac{\mathrm{t}}{M} &\approx \mathrm{E}(\mathrm{t}_m) && (6)\\
&= \frac{\hat{\beta}}{\sqrt{\mathrm{Var}(\hat{\beta})}}
\end{aligned}$$

2.2 Second level

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

$$Y_G = \beta_0^* + \beta_1^* X_G + \varepsilon^*. \tag{7}$$

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:

$$\begin{aligned}
\mathrm{Var}(\varepsilon^*) &= \sigma_G^2 + \mathrm{Var}_\beta(\widehat{\beta_1}) && (8)\\
&= 0 + \sigma^2 (X'X)^{-1}
\end{aligned}$$

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:

$$\Leftrightarrow \quad \sum_{m=1}^{M} \frac{\hat{\beta}_m^*}{M} \approx \mathrm{E}(\hat{\beta}^*) \tag{9}$$
$$= \beta^* = \beta = 3$$
$$\Leftrightarrow \quad \sum_{m=1}^{M} \frac{\widehat{\mathrm{Var}(\beta_m^*)}}{M} \approx \mathrm{E}(\widehat{\mathrm{Var}(\beta^*)}) \tag{10}$$
$$= \sigma^{*2}(X_G'X_G)^{-1}$$
$$= \sigma^2(X'X)^{-1}(X_G'X_G)^{-1}$$
$$\Leftrightarrow \quad \sum_{m=1}^{M} \frac{\mathrm{t}_m^*}{M} \approx \mathrm{E}(\mathrm{t}^*) \tag{11}$$
$$= \frac{\hat{\beta}^*}{\sqrt{\mathrm{Var}(\hat{\beta}^*)}}$$

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:

$$\begin{aligned} \mathrm{d} &= \frac{\mathrm{t}}{\sqrt{n}} \\ &= \mathrm{t}\left[\sqrt{(X_G'X_G)}\right]^{-1} \\ &= \frac{\hat{\beta}^*}{\left[\sqrt{\sigma^2(X'X)^{-1}(X_G'X_G)^{-1}}\right]}\sqrt{(X_G'X_G)^{-1}}. \end{aligned} \tag{12}$$

Or simplified:

$$\mathrm{d} = \frac{\hat{\beta}^*}{\sigma\sqrt{(X'X)^{-1}}}. \tag{13}$$

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:

$$\frac{\sum_{m=1}^{M} g_m}{M} = \frac{3}{100\sqrt{(X'X)^{-1}}} \times J. \tag{14}$$

2.3 Third level

The estimated value at the third level (meta-analysis) corresponds with 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. However, we do not add between-study variability in the Monte-Carlo simulation setting. Furthermore, as we use the same amount of subjects for each study, the expected value of the within-study variability will be equal for each study. Hence 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:

$$\begin{aligned} \mu &= E(g) \\ &= g, \end{aligned} \tag{15}$$

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

3 Monte-Carlo simulation set-up

In the following code section, we define:

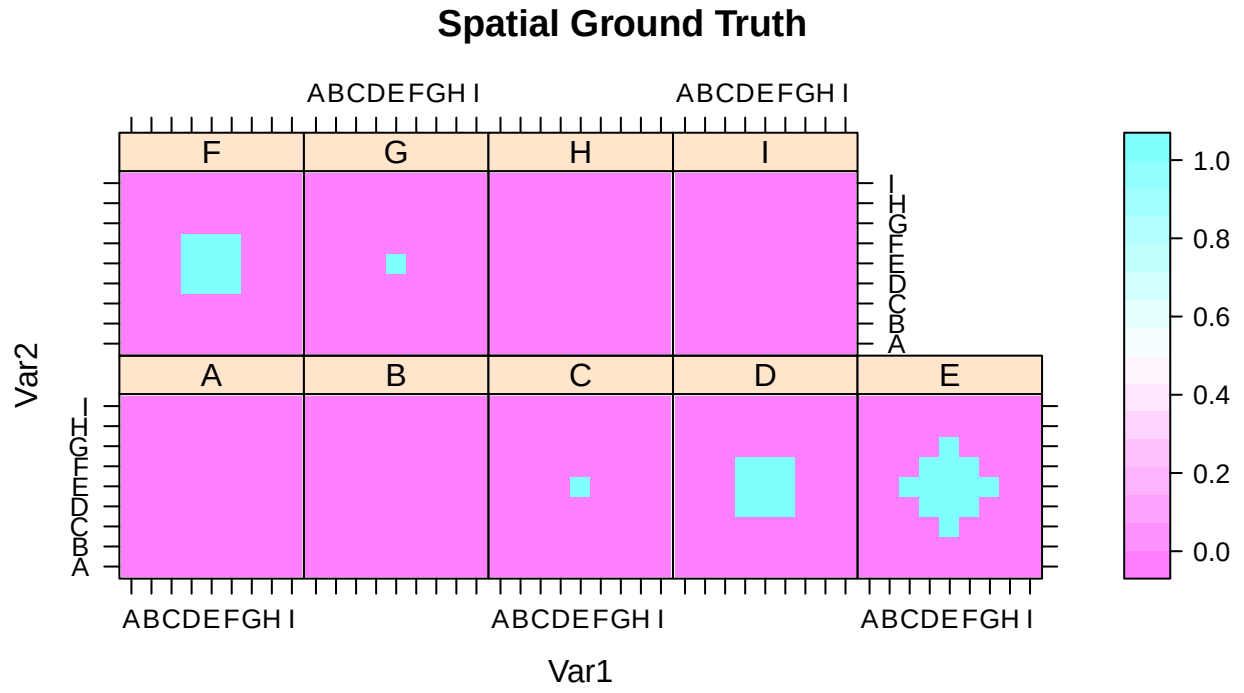
- number of subjects
- number of studies
- characteristics of signal in each subject
- characteristics of image

```
# 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)
levelplot(GroundTruth, main = 'Spatial Ground Truth')
```

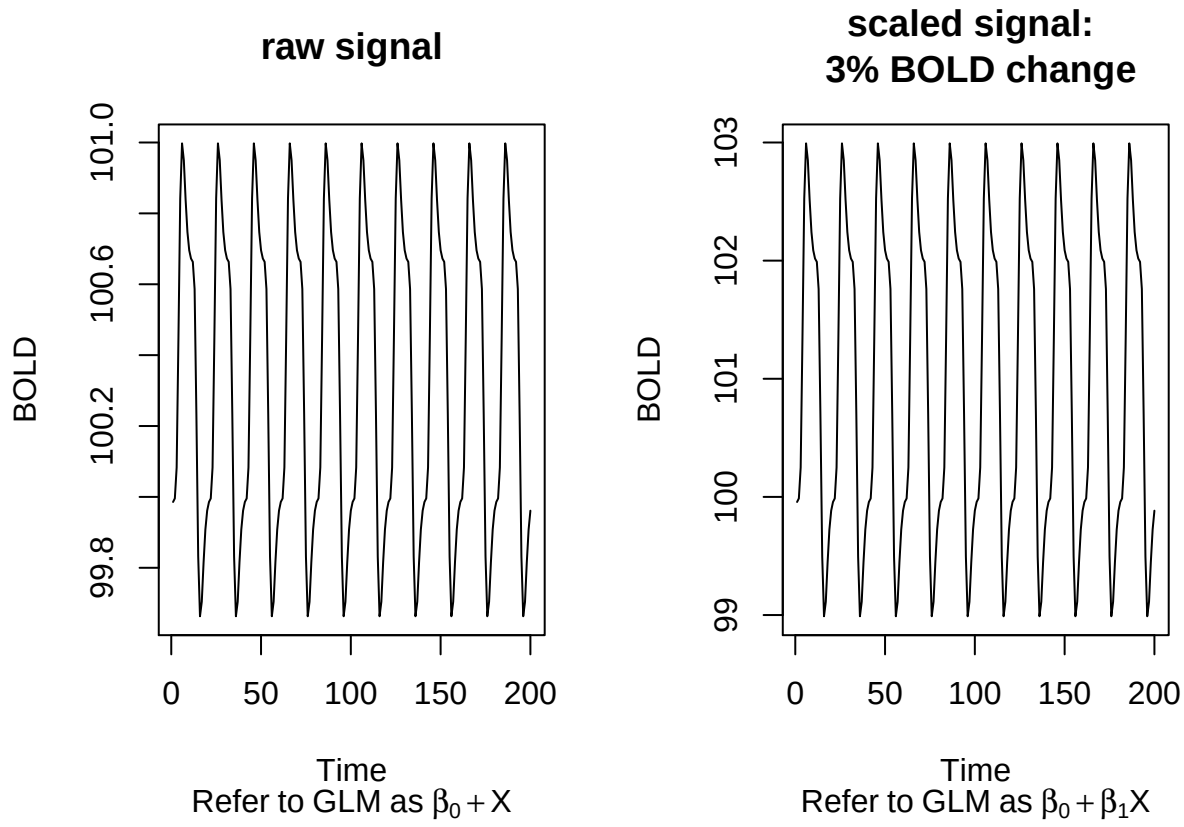


Next we create a true signal (no noise). We convert the signal to the appropriate scale (we choose a base signal of 100) and make sure the amplitude of the signal corresponds to % BOLD signal. This is done through:

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

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)

# 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')
```




```
# Now get the smoothed (raw) signal in each voxel
Rawsignal <- GroundTruth %o% signal_BOLDC
```

To generate data, we start with setting variance parameters:

```
# Sigma of white noise
whiteSigma <- 100
# Between study variability
tau <- 0
```

Now we generate data for each subject and study. We start with creating a value for the study-specific mean. Note that there is no between-study variability. Hence it does not really make sense to do this, but later we might add between-study heterogeneity.

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

The following **R** code is given as example (not run).

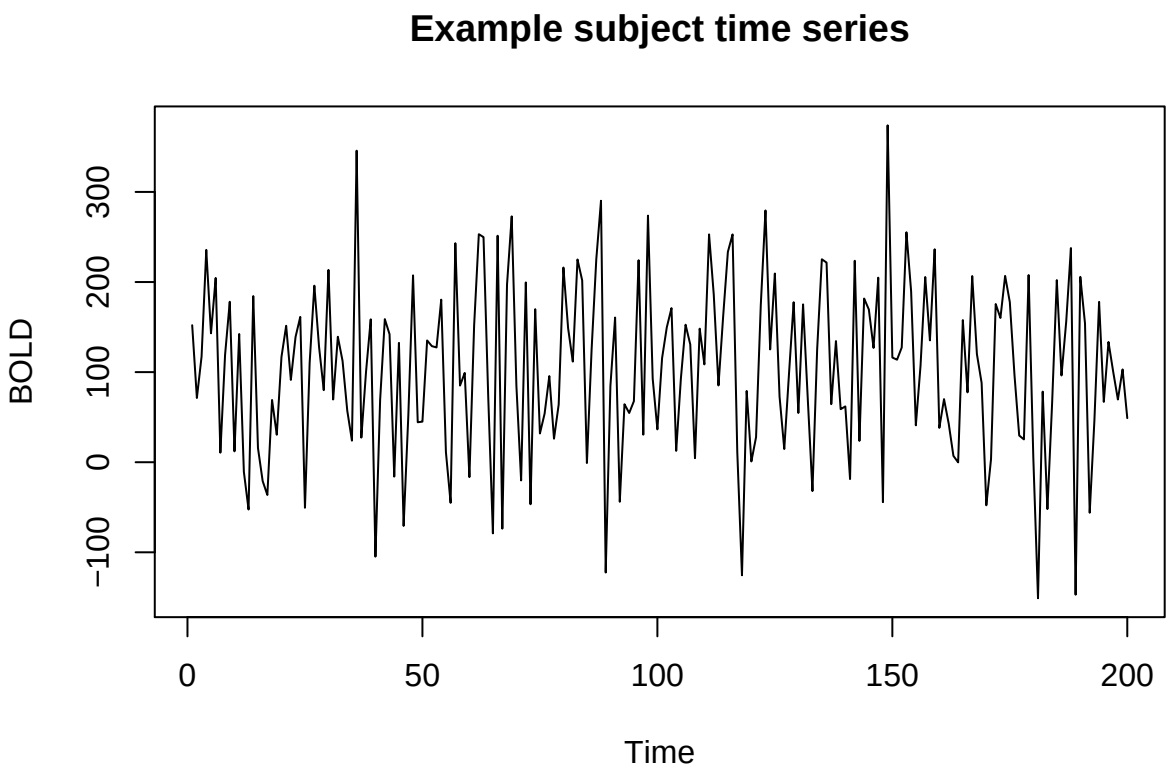
```
# Seed
set.seed(pi)
# Data frame with results:
MAvec <- tibble(sim = integer(),
                      Wavg = numeric(),
                      sigma = numeric(),
                      nstud = numeric())
# Empty vectors
COPE <- VARCOPE <- array(NA,dim=c(prod(DIM),nsub))
STCOPE <- STVARCOPE <- STVALUE <- 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.
  # Distributed with mean true signal and variance tau
  StudData <- Rawsignal + array(
    array(rnorm(n = prod(DIM), mean = 0, sd = tau),
          dim = DIM), dim = c(DIM, nscan))
  # For loop over nsub
  for(s in 1:nsub){
    # Make white noise
    whiteNoise <- array(rnorm(n = (prod(DIM) * nscan), mean = 0,
                              sd = whiteSigma), dim = c(DIM, nscan))

    # Create image for this subject
    SubjData <- StudData + whiteNoise

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

    ###################
    #### 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)

# 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:

```
par(mfrow = c(1,1))
plot(SubjData[TrueLocations[1],
              TrueLocations[2],
              TrueLocations[3],], 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.

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

# Empty data frames
VectorMA <- tibble(sim = integer(),
                   Wavg = numeric(),
                   sigma = 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')) %>%
    rename(studID = subID) %>%
    bind_rows(studSimDat,. )

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

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)) %>%
  summarise(AvgCope = mean(COPEsub),
            AvgVarCope = mean(VARCOPEsub),
            AvgTVal = mean(Tval))
firstRes
```

```
# A tibble: 1 x 3
  AvgCope AvgVarCope AvgTVal
    <dbl>      <dbl>   <dbl>
1    3.35        254   0.210
```

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
3.347581	253.8918	0.2097940	Observed	First

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)) %>%
  summarise(AvgCope = mean(COPEstud),
            AvgVarCope = mean(VARCOPEstud),
            AvgTVal = mean(Tval),
            AvgG = mean(HedgesG))
secondRes
```

```
# A tibble: 1 x 4
  AvgCope AvgVarCope AvgTVal  AvgG
    <dbl>      <dbl>   <dbl> <dbl>
1    3.05       8.81    1.06 0.191
```

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
3.046237	8.806766	1.055664	0.1907338	Observed	Second

4.3 Third level

Finally, we have `1000` number of Monte-Carlo simulations where we save the estimate for the weighted average.

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

```
# A tibble: 1 x 1
  AvgMu
  <dbl>
1 0.181
```

Comparing with $E(g)$, we have:

```
EhedgeGMA <- EhedgeG

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

Table 3: Third level analysis

AvgMu	Value	Level
0.1830116	Expected	Third
0.1810194	Observed	Third

We do observe what we expect on all three levels.

5 Conclusion

All seem fine. We do observe what we expect to find.