Checking some fMRI analysis procedures.

Han Bossier

19 februari 2016

Introduction

In this small report, I will generate an fMRI time series for 100 subjects. The experiment consists of a simple blocked design (one condition, ON/OFF for 20 sec each). The TR = 2 sec, there are in total 200 scans. The data consists of a null effect in a small grid of 2 x 2 x 2 voxels (total of 8 voxels). I will then relate the design of the experiment within subjects to the simulated time series through either the *lm* function in R, or by manually calculating the estimates ($\hat{\beta}$).
After all subjects are processed, we will go the second level GLM in which we estimate the group estimates manually or by averaging the individual $\hat{\beta}$ estimates. Then we proceed to calculating the test statistic by applying a simple OLS procedure. This is similar as a one sample t-test (only one group of subjects). Hence we both use the function *t.test* as well manually calculating the T-map.

We start with some global options for simulating the data:

```
####**************####
#### Global options
####**************####
TR <- 2
nscan <- 200
total <- TR*nscan
on1 <- seq(1,total,40)
onsets <- list(on1)
duration <- list(20)
effect.null <- list(0)          ## No effect
effect <- list(1)               ## Effect of 1 for designmatrix
DIM <- c(2,2,2)
nsub <- 100

TrueLocations <- c(4,4,4)
TrueWhiteNoise <- c(1,0,0,0,0,0)
TrueRadius <- 1
COPE <- VARCOPE <- TMAP <- array(NA,dim=c(prod(DIM),nsub))

####**************####
#### Design matrices
####**************####
# Design Matrices via neuRosim:
#     * We need two design vectors:
#     * The first one have an intercept (needed for analysis).
#        * This will be the column of the design matrix in the analysis.
#     * The second one is used to generate data with a NULL effect.
design.Cond1 <- simprepTemporal(onsets = list(on1), durations = list(duration[[1]]),
                       hrf = "double-gamma", TR = TR, totaltime = total,
                       effectsize = list(effect[[1]]))

design.null <- simprepTemporal(regions = 1, onsets = onsets, durations = duration,
                       hrf = "double-gamma", TR = TR, totaltime = total,
                        effectsize = effect.null)

# X-matrix in order to fit the model later on.
x <- matrix(c(simTSfmri(design.Cond1, nscan=nscan, TR=TR, noise="none")),ncol=1)
```

Now consider the case for one subject ($s = 1$):

```
s <- 1
# Define two regions (which does nothing as there is no effect, )
regions <- simprepSpatial(regions = 1, coord = TrueLocations, radius = list(TrueRadius), form ="cube", 

# Weighting structure.
#   * Order = white, temporal, low-frequency, physyiological, task related and spatial.
w <- TrueWhiteNoise

# Base value
base <- 5

# Actual simulated data
sim.data <- simVOLfmri(design=design.null, image=regions, base=base, dim=DIM, SNR=0.5,
             type ="gaussian", noise= "mixture", spat="gaussRF", FWHM=2, weights=w, verbose = TRUE)
  # Transform it to correct dimension (Y = t x V)
  Y.data <- t(matrix(sim.data,ncol=nscan))

####*************####
#### ANALYZE DATA: 1e level
####*************####

# Fitting GLM model.
model.lm <- lm(Y.data ~ x)
b1 <- coef(model.lm)['x',]
COPE[,s] <- b1
# Manual calculate COPE values:
  # Design matrix: extended with 1's for the intercept
  xIn <- cbind(1,x)
  # Contrast: 0,1 as we are not interested in b0
  CONTRAST <- c(0,1)
manb1 <- CONTRAST %*% (solve(t(xIn) %*% xIn )) %*% t(xIn) %*% Y.data
```

We can now compare the object b1 and the manually calculated b1's:

```
VisualCheckSub <- cbind(c(b1),c(manb1)) ;VisualCheckSub
```

```
##            [,1]       [,2]
## [1,] -1.1244526 -1.1244526
## [2,]  1.4500775  1.4500775
## [3,] -0.9837369 -0.9837369
## [4,] -0.2611795 -0.2611795
## [5,] -0.3355936 -0.3355936
## [6,]  1.2202463  1.2202463
## [7,] -0.3564218 -0.3564218
## [8,]  1.4213000  1.4213000
```

Now let's continue and process 100 subjects:

```
# For loop over nsub
for(s in 1:nsub){
  # Define two regions (which does nothing as there is no effect, )
  regions <- simprepSpatial(regions = 1, coord = TrueLocations, radius = list(TrueRadius), form ="cube"

  # Weighting structure.
  #   * Order = white, temporal, low-frequency, physyiological, task related and spatial.
  w <- TrueWhiteNoise

  # Base value
  base <- 5

  # Actual simulated data
  sim.data <- simVOLfmri(design=design.null, image=regions, base=base, dim=DIM, SNR=0.5,
               type ="gaussian", noise= "mixture", spat="gaussRF", FWHM=2, weights=w, verbose = TRUE)
    # Transform it to correct dimension (Y = t x V)
    Y.data <- t(matrix(sim.data,ncol=nscan))

  ####****************####
  #### ANALYZE DATA: 1e level
  ####****************####

  # Fitting GLM model.
  model.lm <- lm(Y.data ~ x)
  b1 <- coef(model.lm)['x',]
  COPE[,s] <- b1
}
```

We now check the second level analysis:

```
####****************####
#### GROUP ANALYSIS: 2e level
####****************####

# Group COPE (average)
GCOPE <- apply(COPE,1,mean,na.rm=TRUE)
  # Manual calculate group COPE values:
  Xg <- matrix(1,nrow=nsub)
    # We need the pseudo inverse of Xg, use function ginv from package MASS
      # pseudo inverse can also be calculated as:
      PseudInver <- solve(t(Xg) %*% Xg) %*% t(Xg)
  manGCOPE <- ginv(Xg) %*% t(COPE)

# Now we will do the OLS estimation of the variance
GVARCOPE <- apply(COPE,1,var,na.rm=TRUE)

# TMAP
GTMAP <- GCOPE/sqrt(GVARCOPE/(nsub))

# Now check this T-map
manT.test <- apply(COPE,1,t.test)
  manT.val.tmp <- unlist(manT.test)
manT.val <- as.numeric(manT.val.tmp[names(manT.val.tmp)=='statistic.t'])
```

We compare the function *ginv()* for calculating a pseudo-inverse with the object *PseudInver*, the objects *GCOPE* and manually calculated second level COPES (*manGCOPE*) and finally the t-map calculated by applying the function *t.test* and our manually calculated *GTMAP*.

```
# Pseudo-inverse: only look at first value
PSINVER <- cbind(c(PseudInver[1]),c(ginv(Xg)[1]));PSINVER
```

```
##      [,1] [,2]
## [1,] 0.01 0.01
```

```
  # Is the rest the same?
  all.equal(PseudInver,ginv(Xg))
```

```
## [1] TRUE
```

```
# GCOPE
VisualCheckGCOPE <- cbind(c(GCOPE),c(manGCOPE));VisualCheckGCOPE
```

```
##             [,1]        [,2]
## [1,] -0.02550313 -0.02550313
## [2,]  0.05574133  0.05574133
## [3,]  0.31603731  0.31603731
## [4,]  0.29888078  0.29888078
## [5,]  0.05960421  0.05960421
## [6,]  0.06813962  0.06813962
## [7,] -0.01694830 -0.01694830
## [8,] -0.12099100 -0.12099100
```

```
# T-map
VisualCheckTval <- data.frame(GTMAP,manT.val);VisualCheckTval
```

```
##        GTMAP   manT.val
## 1 -0.1605031 -0.1605031
## 2  0.3231439  0.3231439
## 3  1.8763137  1.8763137
## 4  1.9260572  1.9260572
## 5  0.3344425  0.3344425
## 6  0.4581959  0.4581959
## 7 -0.1033202 -0.1033202
## 8 -0.6797874 -0.6797874
```