Ancestral Graphs example with Stop-signal data

Sara Jahfari April 2017

Before running throught the example bellow get an intution for the method and stats reading Lourens Waldorp's 2011 NI paper here.

1. Getting started

Ancestral Graphs relies on a number of packages such as graphs, RBGL and ggm. These packages are updated in a fast rate, and for now you will have to run an older version of R (version 3.0.2) in order to use the validated versions of these packages for AG. You can find the checked package versions in the folder AG_codes/Rpackage_checked.

For me it works really well to use RSwich to swich between the newest version of R and this older version for AG. After installing the older version of R, and using RSwich make sure that you are running R version 3.0.2 (2013-09-25).

```
# are you running the correct version of R?
sessionInfo()
```

Now set the Base directory path to the AG_stopexample folder on your computer

```
# set datadir Base
dir_base='/Users/sarajahfari/Github/AG_stopexample'

#dir AG_codes, example_codes,
dir_agcodes=paste(dir_base,'/AG_codes',sep='')
dir_excodes=paste(dir_base,'/Example_codes',sep='')
dir_data=paste(dir_base,'/Example_data',sep='')
```

and get/source the other required codes and packages

```
# source checked packages
source("http://bioconductor.org/biocLite.R")
biocLite("graph")
source("http://bioconductor.org/biocLite.R")
biocLite("RBGL")
install.packages(paste(dir_agcodes, "/Rpackage_checked/ggm-1.995-3.tar.gz",
    sep = ""), repos = NULL, type = "source")

# source AG_codes
file.sources = list.files(paste(dir_agcodes, "/rFunctions2",
    sep = ""), pattern = "*.R", full.names = TRUE, recursive = T)
sapply(file.sources, source, .GlobalEnv)
```

2. Get data, and define trials/conditions/regions

Read the single subjects data list with single trial estimates for a number of ROI's that will be used to infer connectivity. Next, I always create an index file that is easy to use with the AG codes.

```
# this reads the single subject data termed yDat, and creates
# the condition indexes in a seperate list
source(paste(dir_excodes, "/index_yDatstop.R", sep = ""), chdir = F)
head(yDat[[1]])
```

```
##          CaudateR40exc maxGPeR30exc maxGPiR30exc maxSTNR25exc      IFGR
## 01GoL_1      -0.595375     0.442298     0.066915    -0.031055 -0.329973
## 01GoL_10      0.105857    -0.026887     0.057072    -0.083707 -0.144638
## 01GoL_11      0.161698     0.380750     0.500733    -0.017357  0.291495
## 01GoL_12     -0.032499     0.463627    -0.033962    -0.052651  0.278092
## 01GoL_13      0.106706     0.105174     0.095832     0.159087  0.207860
## 01GoL_14     -0.500952    -0.032343    -0.253548     0.308119  0.607734
##          MotorCBA4aL PreSMARsmall ThalamusR40exc     type
## 01GoL_1    -0.398792    -0.369048      -0.183989  01GoL_1
## 01GoL_10   -0.194387     0.450988       0.012323 01GoL_10
## 01GoL_11    0.816016     0.066621       0.534921 01GoL_11
## 01GoL_12    0.268594    -0.064747       0.272632 01GoL_12
## 01GoL_13    0.298172     0.209098       0.401299 01GoL_13
## 01GoL_14   -0.012316    -0.080152      -0.199831 01GoL_14
```

Then, define the trial types of this task which you would like to model, and the nodes which should be in your model space

```
# these are the conditions that are evalauted seperatly in
# FitmodelsStop.R (see below)
cond = list(ST, SR, Go, GoL, GoR)
names(cond) = c("ST", "SR", "Go", "GoL", "GoR")

# this is just a list with the row_indexes per subject for
# the succesfull stop trials
head(cond$ST)
```

```
## $`sub-001`
##  [1]  50  51  52  53  54  55  56  57  58  59  60  61 124 125 126 127 128
## [18] 129 130 131 132 133 134 135
##
## $`sub-002`
##  [1]  50  51  52  53  54  55  56  57  58  59  60 125 126 127 128 129 130
## [18] 131 132 133 134 135 136
##
## $`sub-003`
##  [1]  51  52  53  54  55  56  57  58  59  60  61 126 127 128 129 130 131
## [18] 132 133 134 135 136 137 138 139 140
##
## $`sub-004`
##  [1]  51  52  53  54  55  56  57  58  59  60  61 126 127 128 129 130 131
## [18] 132 133 134 135 136 137 138 139 140 141
##
## $`sub-005`
##  [1]  49  50  51  52  53  54  55  56  57  58 122 123 124 125 126 127 128
## [18] 129 130 131 132 133 134
##
## $`sub-006`
##  [1]  50  51  52  53  54  55  56  57  58  59  60  61  62  63 124 125 126
## [18] 127 128 129 130 131 132 133
```

```
# if responses are made with the left and right it can be
# difficult to fit the model with collapsed trials but also
# note that this wil reduce your number of trials by half
# cond = list(STL, STR, SRL, SRR,GoL,GoR)
# names(cond)=c('STL', 'STR', 'SRL', 'SRR','GoL','GoR')
```

```
# Define the nodes which should be in your model space
MOroi = c("CaudateR40exc", "PreSMARsmall", "IFGR", "maxSTNR25exc",
    "maxGPeR30exc", "maxGPiR30exc", "ThalamusR40exc")

C.Lab = MOroi  # these are the regions that AG will use to evaluate connectivity between
```

3. Define the connectivity between nodes in different models

```
# source the models to use, these are the defined PFC-BG
# models to evaluate for connectivity/fits
source(paste(dir_excodes, "/StopModel.R", sep = ""), chdir = F)

# try [this is an interactive plot, you can change the layout
# of nodes and get out with esc]
drawGraph(AgO)  # see StopModel.R for definitions, but this should be the hyperdirect-Indirect model

# Now do the connectivity evaluation for each subject, each
# model/condition, to compute aic/bic and n-fits
source(paste(dir_excodes, "/FitmodelsStop.R", sep = ""), chdir = F)
```

Now evaluate fit and random effects AIC/BIC at group level

```
# this is the grouplevel AIC (see FitmodelsStop for code) for
# each model(cols)/condition(rows)
round(AMT, digits = 0)
```

```
##     hindi   hyp indir   dir nfit-hindi nfit-hyp nfit-indir nfit-dir
## ST   7161  8298  7509  7748         43       43         43       43
## SR   7456  8772  7955  8238         43       43         43       43
## Go  21763 25600 22884 23953          0        0          1        0
## GoL 11762 13772 12387 12902         11        5         16       13
## GoR 11397 13353 11998 12516         13       13         19       18
```

```
# this is the grouplevel BIC (see FitmodelsStop for code) for
# each model(cols)/condition(rows)
round(BMT, digits = 0)
```

```
##     hindi   hyp indir   dir nfit-hindi nfit-hyp nfit-indir nfit-dir
## ST   7241  8357  7574  7807         43       43         43       43
## SR   7536  8832  8020  8298         43       43         43       43
## Go  21864 25675 22965 24028          0        0          1        0
## GoL 11852 13840 12460 12969         11        5         16       13
## GoR 11486 13419 12070 12582         13       13         19       18
```

4. Select the winning model (hindi) to derive single subject connectivity estimates

```
# agO was the hindi model in StopModel.R
hindi = agO

# conditions to get beta for [these are the conditions where
# the model fitted all ppn]
```

```
cond2 = list(ST, SR)
names(cond2) = c("ST", "SR")

# Make ST and SR lists using yDat
ST = list()
SR = list()

for (i in 1:length(yDat)) {
    ST[[i]] = yDat[[i]][cond2[[1]][[i]], C.Lab]
    SR[[i]] = yDat[[i]][cond2[[2]][[i]], C.Lab]
}


# compute covarience matrix
covList.ST = lapply(ST, cov)
covList.SR = lapply(SR, cov)

fitST = list()
fitSR = list()

for (i in 1:length(yDat)) {
    fitST[[i]] <- fitAncestralGraph(hindi, covList.ST[[i]], dim(ST[[i]])[1])
    fitSR[[i]] <- fitAncestralGraph(hindi, covList.SR[[i]], dim(SR[[i]])[1])
}

# compute the variance of beta per subject
var.g.ST <- ag.var.group(fitST, covList.ST, ST)
var.g.SR <- ag.var.group(fitSR, covList.SR, SR)

# save matrix for var beta and beta itself (zijn nog leeg)
varbeta.ST = matrix(, dim(var.g.ST)[3], dim(var.g.ST)[1])
varbeta.SR = matrix(, dim(var.g.SR)[3], dim(var.g.SR)[1])
beta.ST = matrix(, length(yDat), length(ag.theta(fitST[[1]])))
beta.SR = matrix(, length(yDat), length(ag.theta(fitSR[[1]])))

for (i in 1:length(yDat)) {
    varbeta.ST[i, ] = diag(var.g.ST[, , i])
    varbeta.SR[i, ] = diag(var.g.SR[, , i])
    beta.ST[i, ] = ag.theta(fitST[[i]])
    beta.SR[i, ] = ag.theta(fitSR[[i]])
}

# these are the standarized connections (to use for group
# comparisons)
sbeta.ST = round(beta.ST/varbeta.ST, dig = 8)
sbeta.SR = round(beta.SR/varbeta.SR, dig = 8)
```

Finally, select the defined directed and undirected connections and put this in a usable list which can be saved as tsv

```
# select the defined directed and undirected connections
beta.ST=beta.ST[,c(1:8,10)]
beta.SR=beta.SR[,c(1:8,10)]
sbeta.ST=sbeta.ST[,c(1:8,10)]
```

```r
sbeta.SR=sbeta.SR[,c(1:8,10)]


colnames(beta.ST) =paste('ST-',c('caud->gpe','presma->caud','presma->stn','ifg->caud',
                                 'ifg->stn','stn->gpi','gpe->gpi','gpi->thalamus',
                                 'presma-ifg'),sep='')

colnames(beta.SR) =paste('SR-',c('caud->gpe','presma->caud','presma->stn','ifg->caud',
                                 'ifg->stn','stn->gpi','gpe->gpi','gpi->thalamus',
                                 'presma-ifg'),sep='')

colnames(sbeta.ST) =colnames(beta.ST)
colnames(sbeta.SR) =colnames(beta.SR)

Beta_output=list(Connectivity_ST=beta.ST, Connectivity_SR=beta.SR,standarized_con_ST=sbeta.ST,standariz

for (rn in 1:length(Beta_output)) {rownames(Beta_output[[rn]])=names(yDat)}
}
```

```r
# arrow is direct in col.names, and - (presma-ifg) is undirected
head(Beta_output$Connectivity_ST)
```

```
##         ST-caud->gpe ST-presma->caud ST-presma->stn ST-ifg->caud
## sub-001  -0.32534493     -0.55749704   -0.007316717  0.266109294
## sub-002  -0.48860562     -0.33756271   -0.276528991  0.002147515
## sub-003   0.08619859     -0.40180361   -0.041358759  0.035535550
## sub-004  -0.24045212     -0.20431636    0.628892409 -0.358360363
## sub-005  -0.12537864     -0.38403947    0.126770183 -0.227053967
## sub-006  -0.44078481     -0.02563003    0.128984510 -0.443016775
##          ST-ifg->stn  ST-stn->gpi ST-gpe->gpi ST-gpi->thalamus
## sub-001 -0.141143328  0.003738614  -0.4264254      -0.14515365
## sub-002  0.114201882  0.055687908  -0.7558637      -1.57800436
## sub-003  0.003940724  0.208368482  -0.8793710      -0.29854938
## sub-004 -0.816364543 -0.207181870  -0.2668323      -0.43127709
## sub-005 -0.396544915  0.061684750  -0.6557039      -0.05550985
## sub-006 -0.395269876  0.105049133  -0.5514878      -0.22893107
##         ST-presma-ifg
## sub-001     0.4615812
## sub-002     0.3413610
## sub-003     0.1925970
## sub-004     0.2600340
## sub-005     0.0893940
## sub-006     0.3113932
```

```r
# arrow is direct in col.names, and - (presma-ifg) is undirected
head(Beta_output$standarized_con_ST)
```

```
##         ST-caud->gpe ST-presma->caud ST-presma->stn ST-ifg->caud
## sub-001  -0.00009765     -0.00005072    -0.00000064   0.00002781
## sub-002  -0.00000096     -0.00074737    -0.00002196   0.00001327
## sub-003   0.00402757     -0.70181803    -0.02425740   0.02596681
## sub-004  -0.00017681     -0.00009260     0.00002705  -0.00017788
## sub-005  -0.00122267     -0.00751744     0.00037188  -0.00204955
## sub-006  -0.00001853     -0.00018219     0.00050155  -0.00007106
##         ST-ifg->stn ST-stn->gpi ST-gpe->gpi ST-gpi->thalamus ST-presma-ifg
```

```
## sub-001 -0.00001255  0.00000026 -0.00000768      -0.00000097    0.00034208
## sub-002  0.00012644  0.00000100 -0.00000632      -0.00000051    0.00003346
## sub-003  0.01039742  0.02229524 -0.00324629      -0.00356488    0.01971200
## sub-004 -0.00007555 -0.00000733 -0.00001859      -0.00001560    0.00133518
## sub-005 -0.00057094  0.00007966 -0.00022248      -0.00005725    0.00468278
## sub-006 -0.00000868  0.00000170 -0.00000289      -0.00000834    0.00011922
```

All of these steps are also in the *EvaluateStop_example.R*. If you have questions let me know. Have fun!