NeuroData Deliverable 10/9

tagarwa2

October 2016

1 Dataset-wide distance and MNR plot

Algorithm 1 Load timeseries and produce a dataset-wide distance estimation and MNR plot.
Input: Folder path containing timeseries .rds files from a dataset. Each .rds file must contain the timeseries data for one subject/session.
Output: Two .png files containing dataset-wide distance and MNR plots.

1: Create 3D matrix M and store all .rds file data (1 dimension signifying the subject and 2 dimensions containing the corresponding timeseries data)
2: Produce 3D matrix CORR by applying 'cor' function on M. Again, CORR's first dimension will signify the subject and the other 2 dimensions will contain the corresponding correlations.
3: Initialize empty array A for number of thresholding points N.
4: Create array B with thresholding values, going from minimum value to max value in CORR with N steps.
5: **for** *tloc* in 1:N **do**
6: Temporarily copy CORR into CORR2
7: Make CORR2 values 0 if less than threshold
8: Make CORR2 values 1 if greater than threshold
9: Compute distance matrix from thresholded CORR2 using stats dist function.
10: Compute rdf matrix from distance matrix.
11: Compute mnr matrix from rdf matrix (mean of rdf matrix).
12: Store mnr matrix for current *tloc* in master matrix MNRM[tloc].
13: Compute KDEM matrix by performing hellinger distance algorithm with distance matrix calculated above.
14: Create KDE plot using ggplot on KDEM matrix.
15: Create MNR plot using ggplot on MNRM matrix.

To test the above algorithm, we can use different inputs with known solutions/outputs, so that we can compare them to our algorithm's outputs. The three types of inputs would be as follows:

1) Dataset with 50 subject scans, with each scan containing 70 voxels, with each voxel having 240 timestamps of intensity. All intensity values will be different for a particular scan, but will be the same across scans. For example, every scan will have same data, which will be a vector going from 1 to 240*70. In this case, we know that the MNR score should be 0.5 and all distances will be 0.
2) Dataset with 50 subject scans, with each scan containing 70 voxels, with each voxel having 240 timestamps of intensity. All intensity values will be NaN. In this case, we know that the script will fail and no MNR or distances will be produced.
3) Dataset with 50 subject scans, with each scan containing 70 voxels, with each voxel having 240 timestamps of intensity. Intensity values will be the same within a sample but different across scans. For example, the

1st scan will be all 1s, the 2nd scan will be all 2s, etc, and the 50th scan will be all 50s. In this case, we know that the MNR score should be 1.

Since we know exactly how the inputs for our "simulations" should look, we simply have to write code to generate the outlined inputs. The code to generate all three inputs is below:

```
#input 1
x <- 240*70
vector1 <- c(1:x)
input1 <- matrix(data = vector1, nrow = 70, ncol = 240)
for(i in 1:50) {
  filename <- paste(c("C:/Users/Tanay Agarwal/Desktop/College Classes/NeuroData 1/Delivera
  saveRDS(input1, filename)
}

#input 2
input2 <- matrix(NaN, nrow = 70, ncol = 240)
for(i in 1:50) {
  filename <- paste(c("C:/Users/Tanay Agarwal/Desktop/College Classes/NeuroData 1/Delivera
  saveRDS(input2, filename)
}

#input 3
for(i in 1:50) {
  input3 <- matrix(data = i, nrow = 70, ncol = 240)
  filename <- paste(c("C:/Users/Tanay Agarwal/Desktop/College Classes/NeuroData 1/Delivera
  saveRDS(input3, filename)
}
```

Now that we have the test inputs and pseudocode, we can write the actual algorithm code, found below:

```
# an example of a driver that loads timeseries data and produces
# a dataset wide distance estimation (along with mnr) and kde plot
# in a figure
#
# written by Eric Bridgeford

dirn <- dirname(parent.frame(2)$ofile)
dirn <- dirname(sys.frame(1)$ofile)
setwd(dirn)

## Sources ---------------------------------------------------------------------

source('C:/Users/Tanay Agarwal/Desktop/College Classes/NeuroData 1/fngs/data_processing/Ru
source('C:/Users/Tanay Agarwal/Desktop/College Classes/NeuroData 1/fngs/data_processing/Ru
source('C:/Users/Tanay Agarwal/Desktop/College Classes/NeuroData 1/fngs/data_processing/Ru
require('ggplot2')
require('reshape2')
require('Rmisc')
require('dlm')
source('C:/Users/Tanay Agarwal/Desktop/College Classes/NeuroData 1/Reliability/Code/FlashR
source('C:/Users/Tanay Agarwal/Desktop/College Classes/NeuroData 1/Reliability/Code/FlashR
```

```
source('C:/Users/Tanay Agarwal/Desktop/College Classes/NeuroData 1/Reliability/Code/FlashR
source('C:/Users/Tanay Agarwal/Desktop/College Classes/NeuroData 1/Reliability/Code/R/proc
source('C:/Users/Tanay Agarwal/Desktop/College Classes/NeuroData 1/Reliability/Code/R/proc
source('C:/Users/Tanay Agarwal/Desktop/College Classes/NeuroData 1/fngs/data_processing/Ru

#datasets <- c('NKI', 'BNU1', 'BNU2', 'HNU1', 'DC1')
#datasets <- c('NKI', 'BNU1', 'BNU2')
datasets <- c('SWU')
outpath <- 'C:/Users/Tanay Agarwal/Desktop/College Classes/NeuroData 1/Deliverables 10-10/
#kfopts <- c('kf', 'no kf')
kfopts <- c('no kf')
#scan_pos <- c(2, 3, 3, 3, 3)
scan_pos <- c(1, 1)
inpath <- 'C:/Users/Tanay Agarwal/Desktop/College Classes/NeuroData 1/Deliverables 10-10/in

## Loading Timeseries ------------------------------------------------------------------

atlases <- c('desikan_2mm')
for (at in atlases[1]) {
  # dir.create(paste(outpath, at, "/", sep=""))
  for (dataset in datasets) {
    opath <- outpath #paste(outpath, dataset, "/", at, "/", sep="")
    # dir.create(opath)
    tpath <- inpath #paste(inpath, dataset, "/", at, "/", sep="")
    tsnames <- list.files(tpath, pattern="\\.rds", full.names=TRUE)
    #print(tsnames)
    tsobj <- open_timeseries(tsnames, sub_pos=3)
    #print(tsobj)
    ts <- tsobj[[1]]
    sub <- tsobj[[3]]
    #print(sub)
    maxmnr <- 0

    for (opt in kfopts) {
      if (isTRUE(all.equal(opt, 'kf'))) {
        kf <- obs2kf(ts)
        #zsc <- signal2zscore(ts)
        corr <- obs2corr(kf)

      } else {
        #zsc <- signal2zscore(ts)
        corr <- obs2corr(ts)
        #print(ts)
        #corr <- cor(do.call(rbind, ts))
      }

      #print(corr)

      ## Change Convention from preferred vara[[sub]][array] to vara[sub,array] for use wi
      nroi <- dim(corr[["1"]])[1]
      nscans <- length(corr)
```

```
  #print(nroi)
  wgraphs <- array(rep(NaN, nroi*nroi*nscans), c(nroi, nroi, nscans))

  #corr[is.nan(corr)] == 0

  counter <- 1
  for (subject in names(corr)) {
    #print(corr[[subject]])
    wgraphs[,,counter] <- corr[[subject]]
    #wgraphs[counter,,] <- corr[[subject]]
    counter <- counter + 1
  }
  #print(wgraphs[1,,])
  #print(max(wgraphs))

  ## Compute MNR ------------------------------------------------------------

  thresh_obj <- thresh_mnr(wgraphs, sub, N = 25)

  mnrthresh <- thresh_obj[[1]]
  Dthresh <- thresh_obj[[2]]
  mnrthresh <- 0

  ranked_graphs <- rank_matrices(wgraphs)
  Drank <- distance(ranked_graphs)
  mnrrank <- mnr(rdf(Drank, sub))

  Draw <- distance(wgraphs)
  mnrraw <- mnr(rdf(Draw, sub))
  #print(c(mnrraw, mnrrank, mnrthresh))
  optmax_mnr <- max(c(mnrraw, mnrrank, mnrthresh))
  if (isTRUE(optmax_mnr > maxmnr)) {
    maxmnr <- optmax_mnr
    kfo <- opt
    if (isTRUE(all.equal(maxmnr, mnrraw))) {
      Dmax <- Draw
      winner <- 'raw'
    } else if (isTRUE(all.equal(maxmnr, mnrrank))) {
      Dmax <- Drank
      winner <- 'rank'
    } else {
      Dmax <- Dthresh
      winner <- 'thresh'
    }
  }
}

## Produce Plots for MNR ------------------------------------------------------

kdeobj <- hell_dist(Dmax, sub)
kde_dist <- data.frame(kdeobj[[1]]$y, kdeobj[[2]]$y, kdeobj[[1]]$x)
```

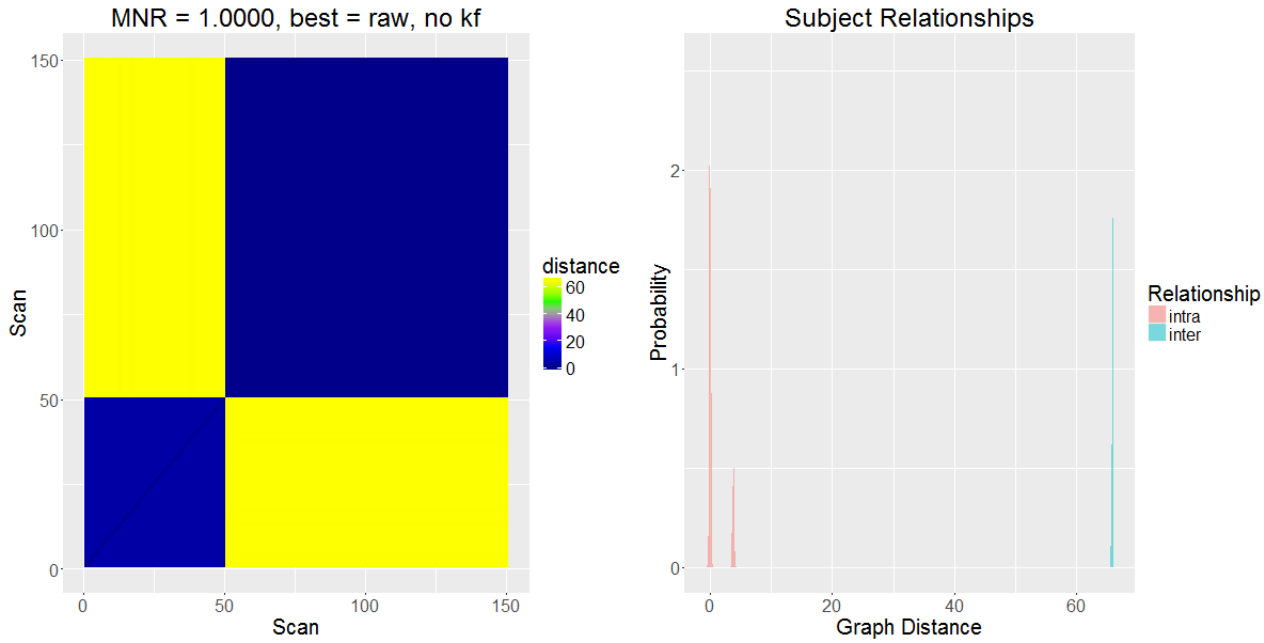
```
    colnames(kde_dist) <- c("intra", "inter", "Graph_Distance")
    meltkde <- melt(kde_dist, id="Graph_Distance")
    colnames(meltkde) <- c("Graph_Distance", "Relationship", "Probability")


    distance_title <- sprintf('MNR_=_%.4f,_best_=_%s,_%s', maxmnr, winner, kfo)
    distance_plot <- ggplot(melt(Dmax), aes(x=Var1, y=Var2, fill=value)) +
      geom_tile() +
      scale_fill_gradientn(colours=c("darkblue","blue","purple","green","yellow"),name="dis
      xlab("Scan") + ylab("Scan") + ggtitle(distance_title) +
      theme(text=element_text(size=20))

    kde_plot <- ggplot()+geom_ribbon(data=meltkde, aes(x=`Graph Distance`, ymax=Probability
      ggtitle("Subject_Relationships") + theme(text=element_text(size=20))

    png(paste(outpath, dataset, "_", kfopts, "_", at, ".png", sep=""), height=600, width =
    multiplot(distance_plot, kde_plot, layout=matrix(c(1,2), nrow=1, byrow=TRUE))
    dev.off()
  }
}
```

The results of running the algorithm on input 1 are:



Algorithm 2 Fungus Pipeline

Inputs:

1) fmri file
2) struct file
3) atlas file
4) atlas brain file
5) mask file
6) labels file
7) output directory path

Outputs:

1) extracted ROI and voxel intensity timeseries
2) quality control summary
3) temporary outputs after every intermediate step while producing above 2 outputs

1: Load in data from command line inputs.
2: Perform preprocessing to remove subject scanner motion and align the fmri in the subject-specific brainspace.
3: Align image to generalized brainspace.
4: Perform nuisance correction on aligned image.
5: Extract ROI timeseries from image and store in 2D matrix, where one dimension represents a voxel and the other dimension contains the intensities for that voxel at specific timestamps.
6: Produce quality control summary for current pipeline run.