brainKCCA: a simple tutorial

Xubo Yue

March 2018

1 Introduction

In this article, we introduce an R package, brainKCCA, for a region-level functional connectivity network analysis using the voxel-level resting state functional magnetic resonance imaging (fMRI) data. This package adopts a multi-attribute canonical graph approach to measuring the strength of functional connectivity for the whole brain network and assessing the statistical significance via permutation tests. The core algorithm is implemented with the parallel computing option, which is scalable to the analysis of massive neuroimaging data. This package also provides a useful brain network visualization tool which can flexibly show all the network connections among predefined regions in a three dimensional brain from any viewpoint. We provide hands-on tutorials on how to analyze the resting state fMRI data using this package and provide an example to the analysis of functional connectivity networks in the Autism Brain Imaging Data Exchange (ABIDE) study.

2 Software Demonstration

2.1 Package installation

Currently, this package is available in Neuroconductor (`https://neuroconductor.org/package/details/brainKCCA`) and GitHub (`https://github.com/xuboyue`). Users can install it by using the following syntax:

```
R> source("https://neuroconductor.org/neurocLite.R")

> # From the Latest Release on NeuroC
> neuro_install('brainKCCA', release = "stable", release_repo =
+     latest_neuroc_release(release = "stable"))
> neuro_install('brainKCCA', release = "current", release_repo =
+     latest_neuroc_release(release = "current"))

> # from GitHub
> neuro_install('brainKCCA', release = "stable", release_repo = "github")
> neuro_install('brainKCCA', release = "current", release_repo = "github")
```

The dependencies are mainly utils, elasticnet, rgl, CCA, kernlab, brainR, oro.nifti, misc3d, knitr and parallel.

2.2 Data format

The resting-state functional magnetic resonance imaging (Rs-fMRI) data of patients/controls are preprocessed based on the Configurable Pipeline for the analysis of Connectomes (C-PAC). The default images were registered in 2mm MNI space with dimension $91 \times 109 \times 91$. The flexibility of core functions in brainKCCA allows other possible setting of processed Rs-fMRI data such as 3mm MNI resolution and different dimensionality.

Figure 1 and Figure 2 shows the scan of the region file registered in 2mm and 3mm MNI space (in software **MRIcron**).

Figure 3 demonstrates the layout of region file registered in 2mm MNI space. As aforementioned, the dimensionality of this file is $91 \times 109 \times 91$.

Figure 4 shows one example of the format of Region List required by this package. The first three columns are the XYZ coordinates of brain region and the next following two columns are the region index. The last column is the name of brain region. Users are allowed to create their own region list file (only need to specify region index and region name, the data for coordinates are supposed to be stored in the Region code files) or use the default region list (116 regions, 2mm resolution) provided in the R package.

The default region index is defined in AAL116 system, which can be accessed from, for example, brainGraph package.

```
R> require(brainGraph)
> aal116
         name     x.mni   y.mni   z.mni       lobe hemi index
  1:  PreCG.L -38.65000  -5.680  50.940    Frontal    L     1
  2:  PreCG.R  41.37000  -8.210  52.090    Frontal    R     2
  3: SFGdor.L -18.45000  34.810  42.200    Frontal    L     3
  4: SFGdor.R  21.90000  31.120  43.820    Frontal    R     4
  5: ORBsup.L -16.56000  47.320 -13.310    Frontal    L     5
 ---
112:  VERMIS6   1.14140 -67.059 -15.123 Cerebellum    R   112
113:  VERMIS7   1.14580 -71.930 -25.141 Cerebellum    L   113
114:  VERMIS8   1.15210 -64.429 -34.080 Cerebellum    R   114
115:  VERMIS9   0.86467 -54.875 -34.896 Cerebellum    L   115
116: VERMIS10   0.35584 -45.800 -31.683 Cerebellum    R   116
```

2.3 Data processing

The **nii2RData** function is designed to process nii/nii.gz data (located in **imgPath**), in argument **niiFile1**, which contains patients' fMRI information, and generate R data files. User can call this function and store generated R data for further analysis. Users can name their saved files in **saveName** argument. The saved R data files are located in **datPath**.

```
R> testcase1 <- nii2RData(niiFile1 = "preproc_con21_rest_MNI_2mm",
    resolution = "2mm", saveName = "patient1.RData", regionCode="", niiFile2="",
    imgPath=getwd(), datPath=getwd())
```

2.4 Calculation of multiple regions connectivity

This function is designed to calculate the strength of connectivity among multiple brain regions. It can either accept user-defined region file and region code to partition brain region or employ its default region data to perform partition. To call this function without user-defined region data, data file and region of interest can be directly provided to this function:

```
R> result0<-permkCCA_multipleRegion(imageDat=c("preproc_con21_rest_MNI_2mm",
+   "preproc_con23_rest_MNI_2mm"), region=c(1,60,70), resolution="2mm",
+   saveName="None", kernel= "rbfdot", regionCode="", niiFile2="",
+   imgPath=getwd(), datPath=getwd(),
+   parallel=FALSE, loc="local", perm=50, saveData="None")
 Checking data format...
Reading nii files...this progress may take a long time...
  |======================================================================| 100%
 reading data  1
```

```
  |======================================================================| 100%
 reading data  2
  |======================================================================| 100%
 Completed...

Calculating KCC...
  |======================================================================| 100%
  |======================================================================| 100%
  |======================================================================| 100%
  |======================================================================| 100%
  |======================================================================| 100%
  |======================================================================| 100%
```

When the function is running, it will generate progress bars with detailed explanations to notify users the current progress.

The **imageDat** argument can accept two types of argument: (1) the brain imaging data file (file name with extension .nii or .nii.gz. However, you do not need to write the extension in this argument), then the function **nii2RData** will be invoked to transform nii/nii.gz file(s) into R Data; (2) the output generated by function **nii2RData**. One advantage of this setting allows user to transform nii data files and store the output R data files so that they do not need to re-read data files in the future. The orginal nii or nii.gz files are supposed to located in path of **imgPath**.

The **region** argument can accept vectors of input of brain regions. The preference of resolution ("2mm" or "3mm") can be specified in the **resolution** arguments.

Users can write name (for example, "pro1.RData") in argument **saveName** to save processed data generated by **nii2RData** function. The **saveData** argument allows users to save generated R data files (for example, "result1.RData"). All results are saved under path **datPath**.

The **kernel** option can instruct function to calculate correlation by different kernel. "rbfdot" represents the the radial basis kernel function "Gaussian". **perm** is the number of permutation in the permutation test.

If users do not want to use the default region data, they are supposed to specify their own region information in the **regionCode** and **niiFile2** arguments. To call this function with user-defined region file and region code:

```
R> result1<-permkCCA_multipleRegion(imageDat=c("preproc_con21_rest_MNI_2mm",
+   "preproc_con23_rest_MNI_2mm"), region=c(1,60,70), regionCode="RegionList.txt",
+   niiFile2="AAL_MNI_2mm.nii")
# the outputs are omitted for brevity
```

The customized region file can have more (or less) than the default 116 partitions and different region name. Additionally, users are supposed to provide corresponding region partition data in the **niiFile2** argument. Three key components are stored in the variables **result0** and **result1**: the region index, the p value and the name of corresponding regions.

The parallel computing issue shall be discussed in the fourth section.

2.5 Summary of results

The results generated by **permkCCA_multipleRegion** can be summarized by the function **summary_kcca**. The first argument of summary function is the result generated by **permkCCA_multipleRegion**. The second argument is the significance level. Typically it can be 0.05 or 0.01. The fourth argument indicates which format used to save table. For example, "markdown" indicates outputting summary information as R markdown table.

```
R> summary_kcca(kcca_object=result0, significance=0.05, patientID=1
+   , saveFormat="markdown", threshold=0.2)

 summary of kcca object  1  generated by 'permkCCA_multipleRegion' function:
```

```
|index1 |index2 |region1 |region2 |pvalue |
|:------|:------|:-------|:-------|:------|
|1      |60     |PreCG.L |SPG.R   |0      |
|1      |70     |PreCG.L |PCL.R   |0      |
|60     |70     |SPG.R   |PCL.R   |0      |
>
```

Note that if users have more than one patient data, they need to specify which patient they would like to see in **patientID** argument. If users would like to summarize connectivity network in group level, they can set `saveFormat="group"`, then this function will generate an object (with regions that are significant) which can be used to plot (see next section). Note that it requires more than 1 patient's data. The **threshold** argument is used to determine threshold of significance.

```
R> group_data <- summary_kcca(kcca_object=result0, significance=0.05,
+   saveFormat="group", threshold=0.2)
```

2.6 Visualization of multiple regions connectivity

This section is devoted to demonstrate approaches to build functional connectivity network. The results generated by **permkCCA_multipleRegion** can be treated as one argument in function **multipleRegion_plot**:

```
R> multipleRegion_plot(input=result0, regionCodeProvided = FALSE,
+   view = "coronal", color = "blue", screenShot = "None")
 You have  2  patients' data
Which data you would like to see? 1
p value of kccc between  PreCG.L  and  SPG.R  is:  0
p value of kccc between  PreCG.L  and  PCL.R  is:  0
p value of kccc between  SPG.R  and  PCL.R  is:  0
> rgl.snapshot("result1.png")
```

The **input** argument accepts result generated by **permkCCA_multipleRegion**. The FALSE in **regionCodeProvided** indicates that the user defined region code has not been provided. The **view** can be specified as "coronal", "axial", "SL" (left sagittal) and "SR" (right sagittal). The **color** argument defines the color of connection lines. The resulting connectivity network will be generated and the **rgl.snapshot** function from package rgl can save the screenshot as png file. The resulting connectivity network is reported in **Figure 5**.

The function also can plot multiple views of brains (**Figure 6**).

```
> multipleRegion_plot(result0, view = c("coronal", "axial", "SL", "SR"))

 You have  2  patients' data
Which data you would like to see? 1
p value of kccc between  PreCG.L  and  SPG.R  is:  0
p value of kccc between  PreCG.L  and  PCL.R  is:  0
p value of kccc between  SPG.R  and  PCL.R  is:  0
> rgl.snapshot("result2.png")
```

When you obtained group-level object, you can construct network by:

```
> multipleRegion_plot(group_data, view = c("coronal", "axial", "SL", "SR"))
> rgl.snapshot("result3.png")
```

3 Performance and Analysis

We demonstrate the performance of connectivity calculation in this section. We will use all 90 brain regions with resolution 3mm, and perform multiple region calculation by 50 times permutation. The running time is measured under cluster environment by parallel computing with 16 processors.

```
R> result2<-permkCCA_multipleRegion(imageDat="UM_1_0050272_func_preproc",
+   region=c(1:90), resolution="3mm", regionCode="RegionList90.txt",
+   niiFile2="AAL_90_3mm.nii", parallel=TRUE, loc="cluster")
```

The argument **parallel=TRUE** will initiate parallel computing and **loc="cluster"** indicates that the codes will be executed in cluster. If users would like to run parallel computing in local computer, **loc="local"** can be specified. The average running time is about 5 hours.

We performed a case study of 60 patients data (with 34 autism and 26 controls) by using cluster parallel computing in High Performance Computing (HPC) cluster and jobs were submitted via Portable Batch System (PBS) files, where the commands and cluster resources used for jobs are defined. The script of PBS files are as follows:

```
#!/bin/sh
####  PBS preamble
#PBS -N brainKCCA
#PBS -M maxyxb@umich.edu
#PBS -m abe
#PBS -t 1-60
# Change the number of cores (ppn=1), amount of memory, and walltime:
#PBS -l nodes=1:ppn=16,mem=40Gb,walltime=8:00:00
#PBS -j oe
#PBS -V
# Change "example_flux" to the name of your Flux allocation:
#PBS -A maxyxb
#PBS -q fluxod
#PBS -l qos=flux
####  End PBS preamble
#  Show list of CPUs you ran on, if you're running under PBS
if [ -n "$PBS_NODEFILE" ]; then cat $PBS_NODEFILE; fi
#  Change to the directory you submitted from
if [ -n "$PBS_O_WORKDIR" ]; then cd $PBS_O_WORKDIR; fi
#  Put your job commands here:
module load R
R CMD BATCH --no-restore --no-save script.R script.out
```

For each PBS job, the following R codes were recursively used:

```
R> array_ID <- as.numeric(Sys.getenv('PBS_ARRAYID'))
> #set working directory to your imaging files
> setwd("./UM_1")
> fileName <- gsub(".nii.gz","",list.files())[array_ID]
> setwd("..")
> testcase <- nii2RData(niiFile1 = fileName, resolution="3mm",
+   regionCode="RegionList90.txt", niiFile2="AAL_90_3mm.nii",
+   imgPath="./UM_1", datPath="./UM_1_result")
> result <- permkCCA_multipleRegion(imageDat=testcase, region=c(1:90),
+   resolution="3mm", regionCode="RegionList90.txt",
+   niiFile2="AAL_90_3mm.nii", parallel=TRUE, loc="cluster")
> tableTemp<-summary_kcca(result,0.05,1,"excel")
> write.csv(tableTemp,paste(fileName, ".csv", sep=""), col.names=FALSE)
> group_data_2 <- meanConnection(path=getwd(), threshold=0.2)
```

60 csv files were generated and the **meanConnection** function was used to process all these files. The users are supposed to separate cases and controls csv files. Users can specify locations of their csv files in **path** argument. The percentage of connection are calculated based on significance in each region pairs. The threshold of percentage is calculated based on binomial test to confirm the connections were not false positive (in this example, the threshold is 0.25). The **meanConnection** function will return a data frame with significant region index and region name (without p value) and the **multipleRegion_plot** function can accept this dataframe as an argument and generate region-level connectivity network (**Figure 7-8**). Users need to set `significance=NA` in this case.

```
> multipleRegion_plot(group_data_2, NA, view = c("coronal", "axial", "SL", "SR"))
> rgl.snapshot("result3.png")
```

List of Figures

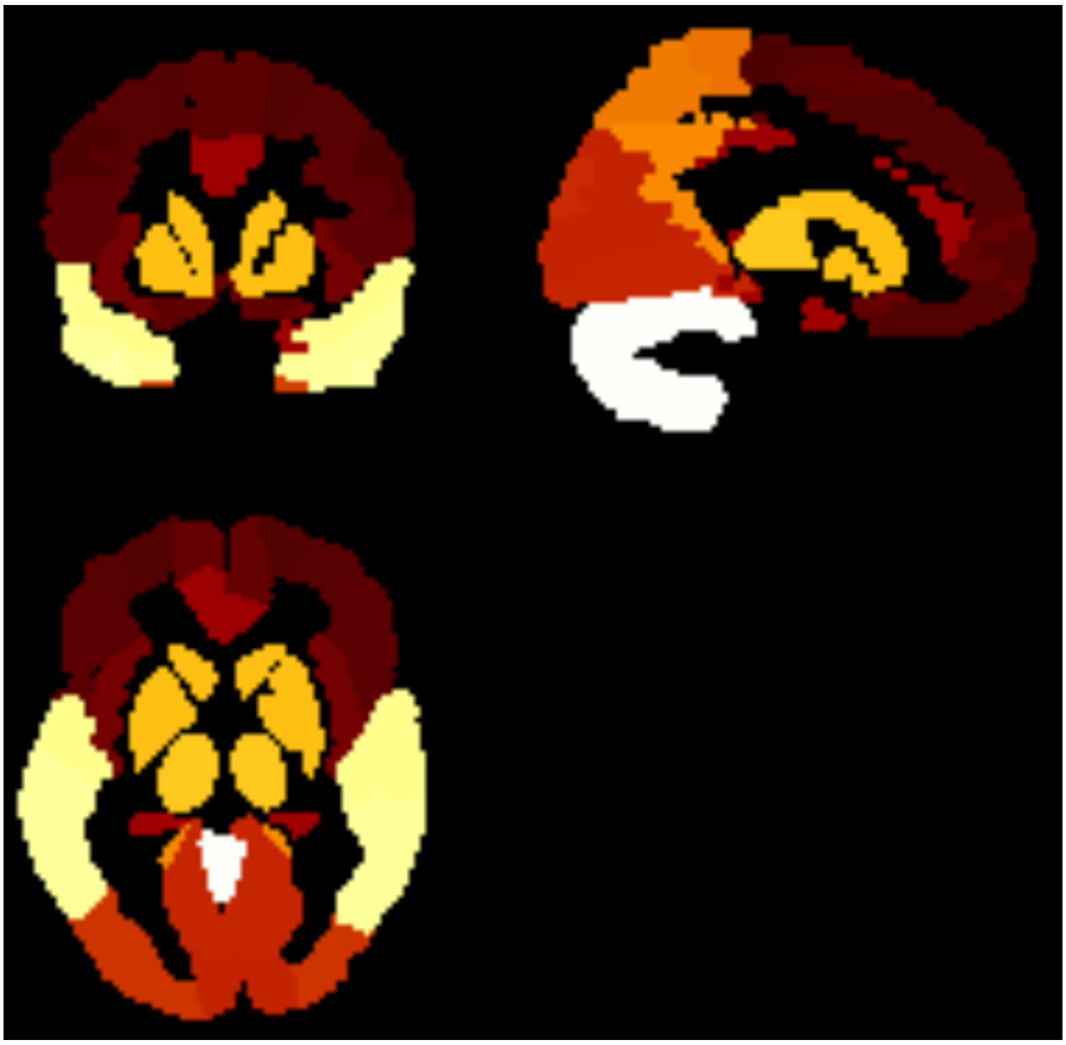


Figure 1: 2mm MNI region file

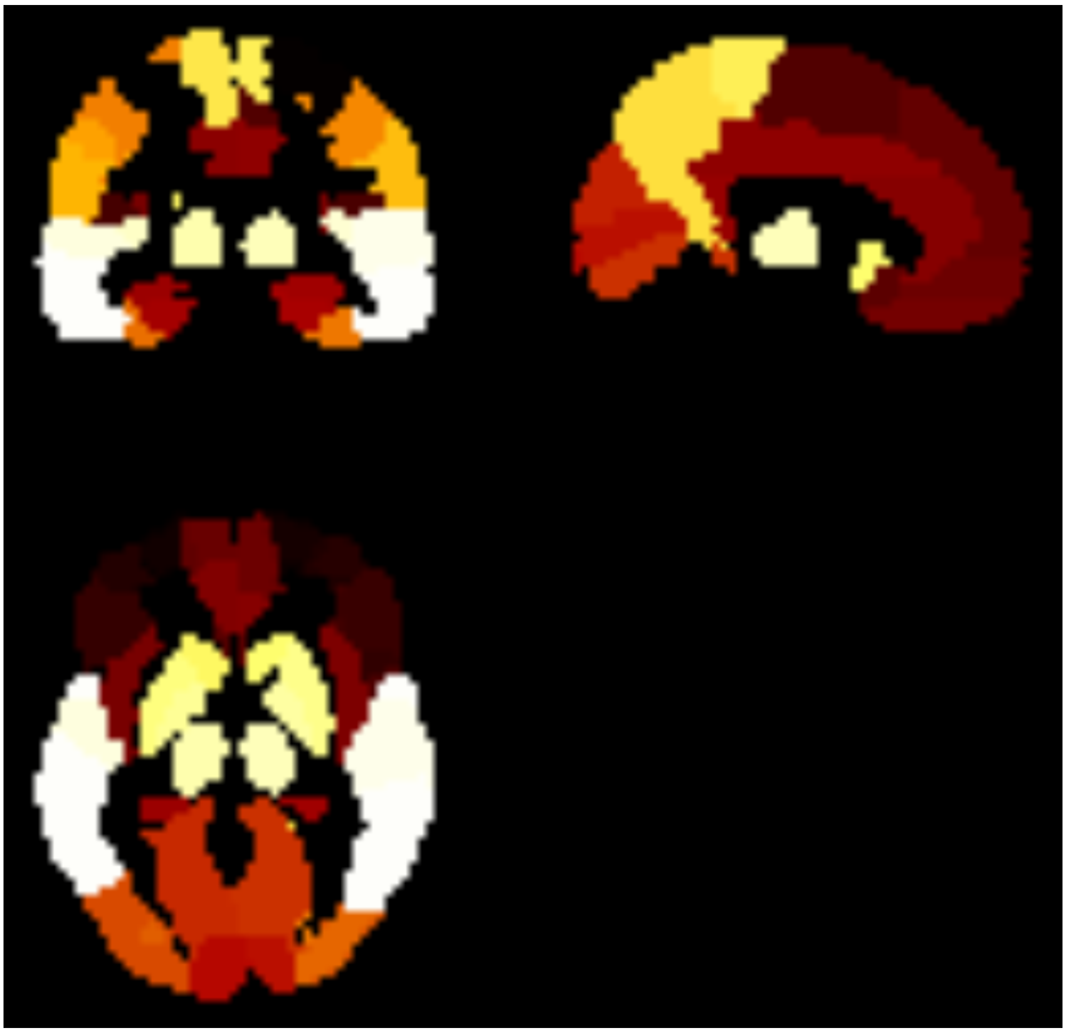
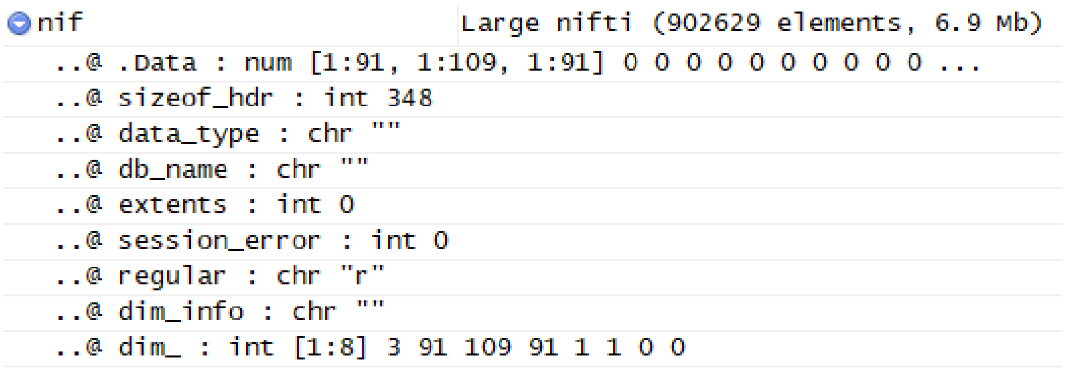
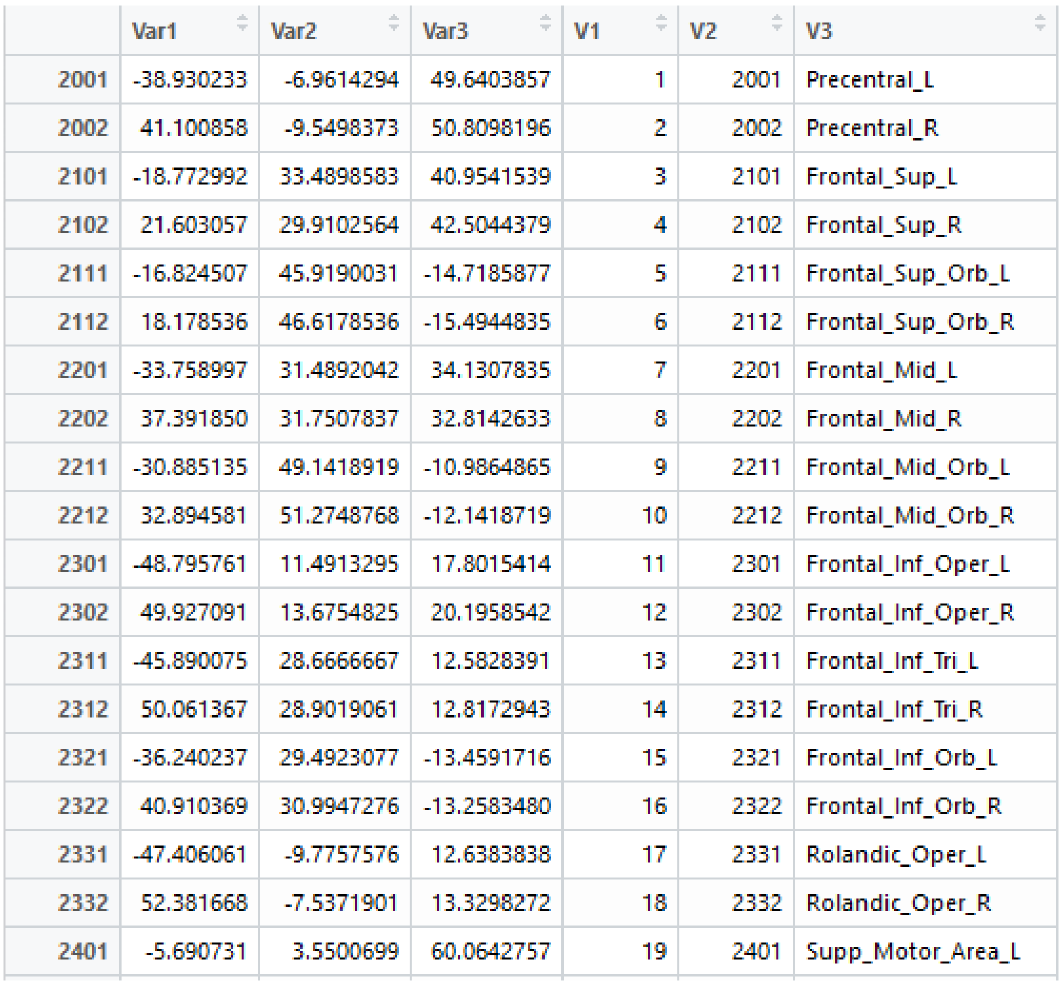


Figure 2: 3mm MNI region file



```
nif                          Large nifti (902629 elements, 6.9 Mb)
   ..@ .Data : num [1:91, 1:109, 1:91] 0 0 0 0 0 0 0 0 0 0 ...
   ..@ sizeof_hdr : int 348
   ..@ data_type : chr ""
   ..@ db_name : chr ""
   ..@ extents : int 0
   ..@ session_error : int 0
   ..@ regular : chr "r"
   ..@ dim_info : chr ""
   ..@ dim_ : int [1:8] 3 91 109 91 1 1 0 0
```

Figure 3: Region Code



	Var1	Var2	Var3	V1	V2	V3
2001	-38.930233	-6.9614294	49.6403857	1	2001	Precentral_L
2002	41.100858	-9.5498373	50.8098196	2	2002	Precentral_R
2101	-18.772992	33.4898583	40.9541539	3	2101	Frontal_Sup_L
2102	21.603057	29.9102564	42.5044379	4	2102	Frontal_Sup_R
2111	-16.824507	45.9190031	-14.7185877	5	2111	Frontal_Sup_Orb_L
2112	18.178536	46.6178536	-15.4944835	6	2112	Frontal_Sup_Orb_R
2201	-33.758997	31.4892042	34.1307835	7	2201	Frontal_Mid_L
2202	37.391850	31.7507837	32.8142633	8	2202	Frontal_Mid_R
2211	-30.885135	49.1418919	-10.9864865	9	2211	Frontal_Mid_Orb_L
2212	32.894581	51.2748768	-12.1418719	10	2212	Frontal_Mid_Orb_R
2301	-48.795761	11.4913295	17.8015414	11	2301	Frontal_Inf_Oper_L
2302	49.927091	13.6754825	20.1958542	12	2302	Frontal_Inf_Oper_R
2311	-45.890075	28.6666667	12.5828391	13	2311	Frontal_Inf_Tri_L
2312	50.061367	28.9019061	12.8172943	14	2312	Frontal_Inf_Tri_R
2321	-36.240237	29.4923077	-13.4591716	15	2321	Frontal_Inf_Orb_L
2322	40.910369	30.9947276	-13.2583480	16	2322	Frontal_Inf_Orb_R
2331	-47.406061	-9.7757576	12.6383838	17	2331	Rolandic_Oper_L
2332	52.381668	-7.5371901	13.3298272	18	2332	Rolandic_Oper_R
2401	-5.690731	3.5500699	60.0642757	19	2401	Supp_Motor_Area_L

Figure 4: Region List

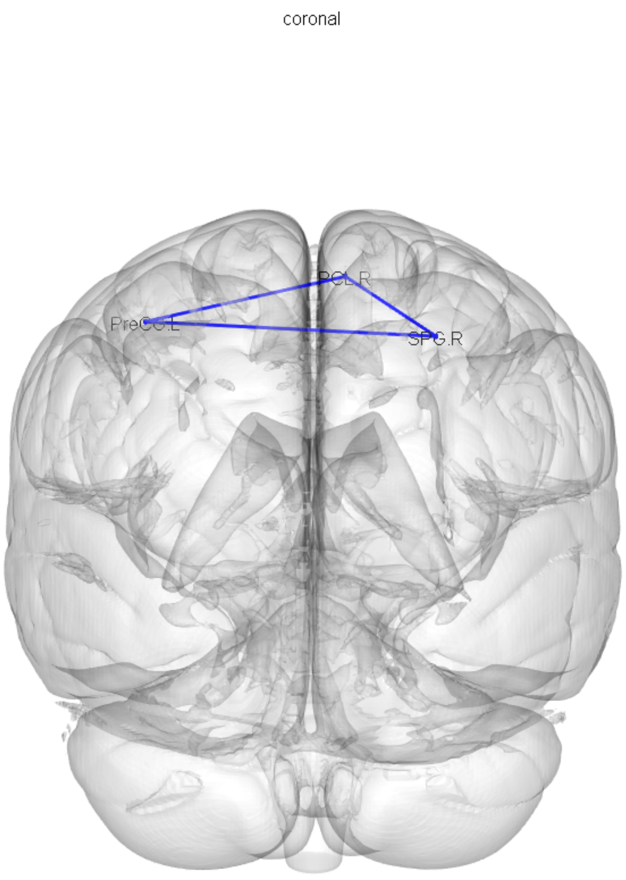


Figure 5: Regions-level connectivity network (coronal and axial view)

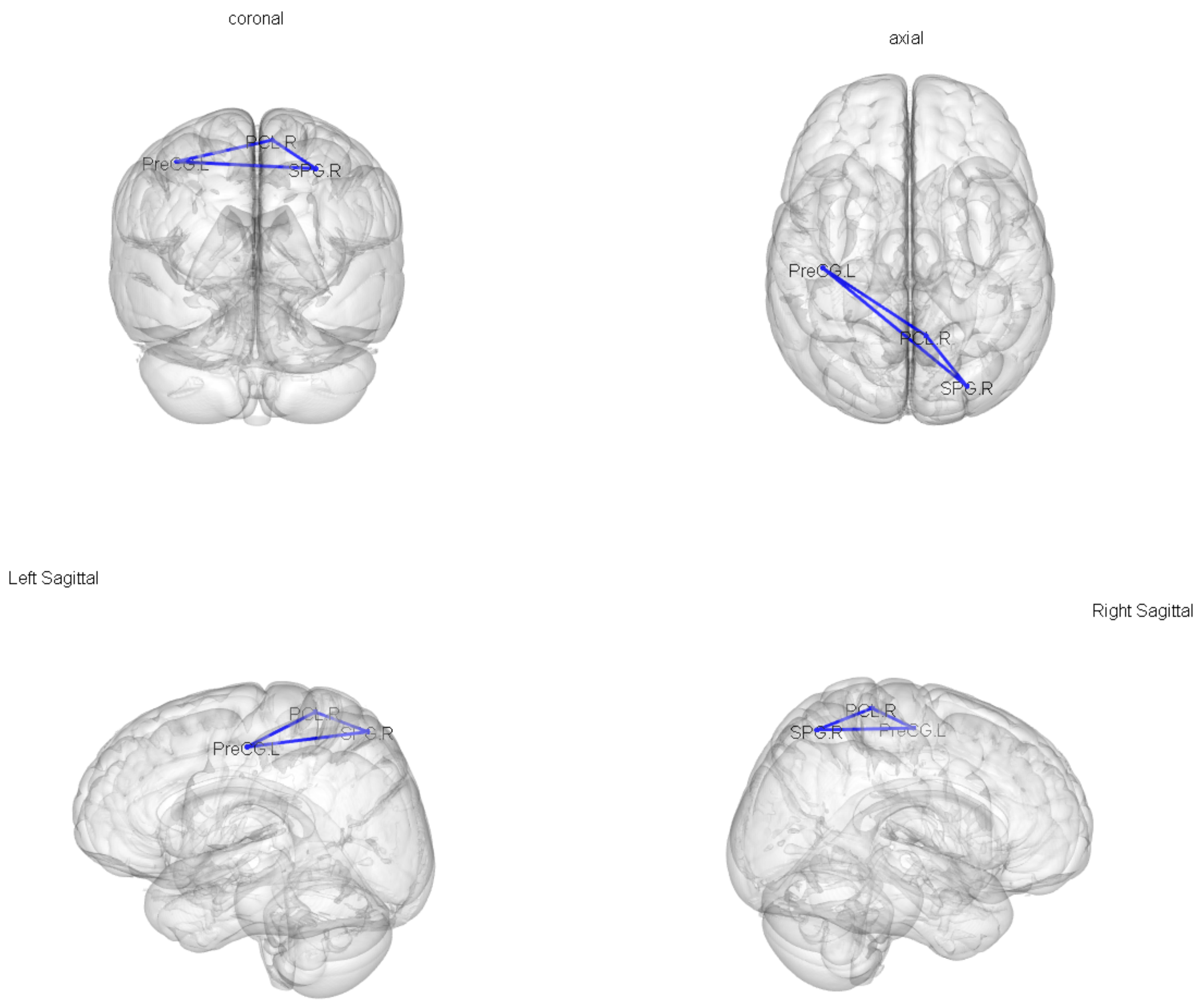


Figure 6: multiple regions connectivity network (multiple views)

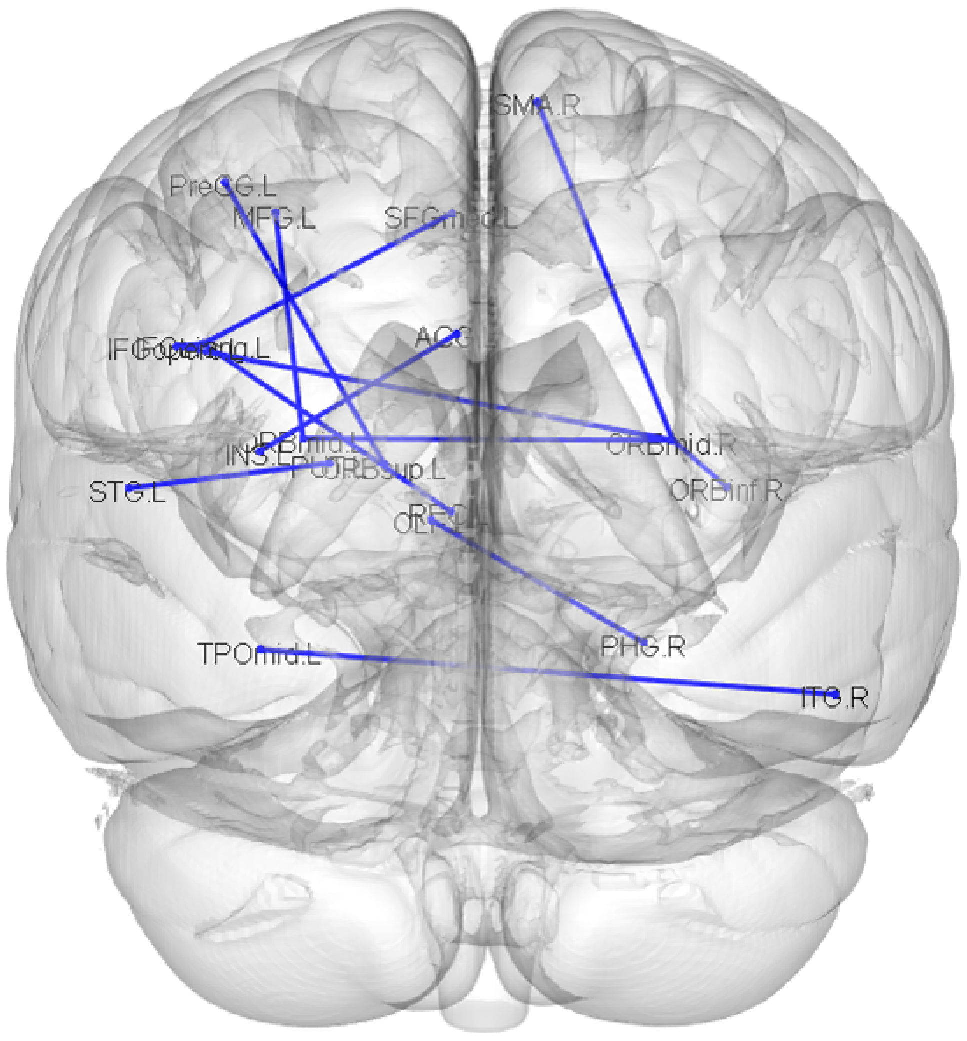


Figure 7: multiple regions connectivity network for case

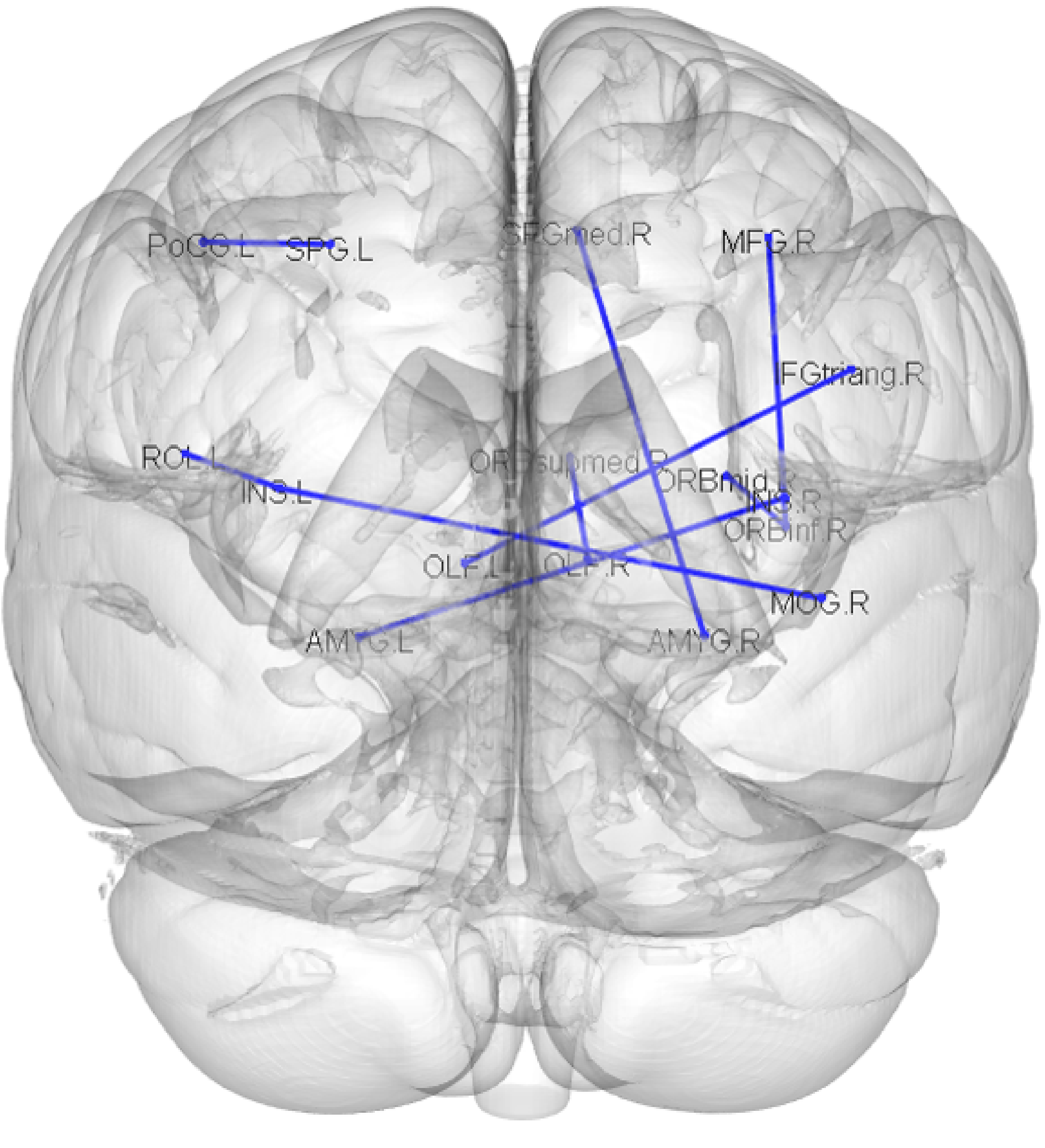


Figure 8: multiple regions connectivity network for control