A User's Guide to Graphical-Model-based Multivariate Analysis (GAMMA) Suite

Version 1.2

Rong Chen
December 2011

1. Introduction

Graphical-Model-based Multivariate Analysis (GAMMA) suite is a freely available cross-platform Bayesian data-mining software package for the analysis of high-dimensional data [1-3]. The problem domain of interest is neuroimaging data analysis, primarily lesion-based studies, functional MR-based studies, and morphometric studies.

The input to the GAMMA suite is a data set **D** and a group-membership variable C. In an example lesion-deficit study, we aim to detect regions which can differentiate the patients with or without reading impairment. In this example, the group-membership variable C represents whether or not a subject has reading impairment (C=0: normal, C=1: reading impairment). The data set **D** includes a set of lesion maps generated based on MR perfusion-weighted images.

In the mathematical language, V_i^k represents the i^{th} predictor variable for subject k; V_i^k is a binary variable, i.e., its state is in {0, 1}. Let $\mathbf{V}^k$ denote the collection of V_i for subject k. The input to algorithms in the GAMMA suite is $\mathbf{D}=\{(\mathbf{V}^1, C^1), \ldots, (\mathbf{V}^k, C^k)\}$. **$\mathbf{V}^k$ MUST be as a binary image in Nifti or Analyze format**. **$\{\mathbf{V}^k\}$ must be defined in the same stereotaxic space across subjects; that is, all image volumes must be normalized to the same template**. We refer to $\mathbf{V}^k$ as an *effect map.* All images should have the SAME dimension and SAME spacing. For example, if the first image is 256*256*128 with 1*1*1.5 mm^3. Then all other images should have the same header information. Please make sure all input images have the correct header information. GAMMA does not check for this.

C is a *binary group-membership variable,* which may be a demographic variable, such as gender, or a clinical variable, such as the results of cognitive assessment. Being binary, C assumes a state in {0, 1}; **in general, we code the group of interest (e.g., the experimental group) as 1, and the control group as 0**.

Features of the GAMMA suite include:

- Identify predictor variables characterizing group differences using a Bayesian nonparametric algorithm (biomarker detection).
- Generate a probabilistic model to describe associations among predictor variables and the group-membership variable (descriptive modeling).
- Ensemble learning to generate stabilized models.
- Modeling of contextual data via Markov random fields.
- Machine-learning algorithms for Bayesian-network generation, including on-line updating of the BDe score.
- Loopy belief propagation for inference and induction.

Source code for the GAMMA suite is freely available under the GNU General Public License (GPL: http://www.gnu.org/copyleft/gpl.html). The core and GUI compile under Mac OS X ®, Linux, and Microsoft Windows ®. The GAMMA suite does not depend on other software packages. Please send your comments or bug reports to gamma.developer@gmail.com or rong.chen.mail@gmail.com.

2. Installation

GAMMA is written in C++, and the GUI is developed based on Qt4. Since the source code is available, users can re-compile it on their own platforms, whether Unix/Linux, Windows, or Mac OS X.

The GAMMA suite provides two end-user programs: GAMMA core and GAMMA GUI. The GAMMA core includes several command-line programs. The GAMMA GUI is a graphical user interface implemented using Qt4.

The released version of the GAMMA suite includes three compressed files: gamma_suite_src.zip, gamma_suite_program.zip, and test_data.zip.

- **gamma_suite_src.zip** includes source code.
- **gamma_suite_program.zip** includes documentation and end user programs.
- **test_data.zip** includes test data.

Here is the GAMMA development and testing environment.

	Linux	Windows	Mac OS X
Development	gcc, Ubuntu 8.1	XP, MSYS, gcc	gcc, Mac OS X 10.5.8
Testing	Ubuntu 8.1, Cent OS 5.3	XP, Vista, Windows7	Mac OS X 10.5.8

GAMMA core development and testing environment

	Linux	Windows	Mac OS X
Development	Qt creator 1.2.1, Qt 4.5.3, Ubuntu 8.1	Qt creator 1.2.1, Qt 4.5.3, XP	Qt creator 1.2.1, qt 4.5.3, Mac OS X 10.5.8
Testing	Ubuntu 8.1, Cent OS	XP, Vista, Windows7	Mac OS X 10.5.8

	5.3		

GAMMA GUI development and testing environment

The installation process consists of the following steps:

1. Download gamma_suite_program.zip.
2. Unzip it to a directory (assume in the rest of this section that the directory is '~/soft/gamma_suite'). Precompiled binaries will be found in '~/soft/gamma_suite/end_user_program'.
    a. GAMMA core in '~/soft/gamma_suite/end_user_program/gamma_core'. Users can choose the binaries for their own platform.
    b. GAMMA GUI in '~/soft/gamma_suite/end_user_program/gamma_gui'. Users can choose the binaries for their own platform.
3. Set up program paths. Users should first copy these binaries to a directory where users want to start the program. For example, under Linux, this directory could be ~/bin/. Let $GAMMA denote this directory. Users can permanently add $GAMMA to the operating system (OS) search path as follows:
    - *bash users under Linux*: edit the file .bash_profile in user's home directory. Change the PATH variable and add $GAMMA to PATH.
    - *Mac OS X users*: edit the file .profile in user's home directory. Change the PATH variable and add $GAMMA to PATH.
    - *Windows users*: Right click [My Computer], open [properties]. Then under [advanced→environment variables], change the variable PATH (add the gamma program directory to PATH).

If users want to install the GAMMA suite by recompiling source code, please refer to Sections 2.1 and 2.2. Otherwise, users can skip those two sections, and go directly to Section 3.

2.1 Compiling GAMMA core

If users want to compile GAMMA core from source code, we recommend gcc for Unix/Linux, Windows, or Mac OS X. Users can download gcc from http://gcc.gnu.org/install/binaries.html.

The compiling steps for Unix/Linux and Mac OS X are as follows:

1. Download gamma_suite_src.zip.
2. Unzip it to a directory. If the directory were named '~/soft/gamma_suite', then the source code for GAMMA suite would be found in '~/soft/gamma_suite/src', that for GAMMA core would be found in '~/soft/gamma_suite/src/gamma_core', and that for GAMMA GUI would be found in '~/soft/gamma_suite/src/gamma_gui'.
3. Compile.
    a. For Linux/Unix, run gamma_install.sh under directory '~/soft/gamma_suite/src/gamma_core'.
    b. For Mac OS X, run gamma_install_mac.sh.
    c. For Windows, run gamma_install_win.sh
4. Executable binaries are in directory '~/soft/gamma_suite/src/gamma_core/bin
5. Copy binaries to a directory, and add the directory containing these executable binaries to the OS search path.

For Windows users, please refer to section Appendix A.2 for compiling the source code.

2.2 Compiling GAMMA GUI

If users want to compile GAMMA GUI, they must install Qt creator (http://qt.nokia.com/downloads). We recommend that users go to http://qt.nokia.com/downloads, then choose 'Qt SDK: Complete Development Environment'. This Qt SDK includes Qt libraries, Qt Creator IDE, and Qt development tools, for different platforms.

Download gamma_suite_src.zip. Unzip it to a directory. If the directory were named '~/soft/gamma_suite', then the source code for GAMMA suite would be found in '~/soft/gamma_suite/src', the project file of GAMMA GUI is under the directory gamma_suite/src/gamma_gui.

3. GAMMA Suite

The GAMMA suite provides two end-user programs: GAMMA core and GAMMA GUI. All end-user programs use two GAMMA-format project files: gamma.prj.filelist is a description of data sources and image information, and gamma.configuration stores parameter settings.

gamma.prj.filelist is a simple text file; its format is as follows:

```
number_subj 28                ← the total number of cases in the study
number_var 12289              ← the total number of variables in the study
dim 64 64 3                   ← image dimensions
spacing 1 1 1                 ← image spacing
id file_name fv               ← header line (never edited)
1 subj_1.img 0                ← subject 1, its image file, its fv
2 subj_2.img 0
3 subj_3.img 1
4 subj_4.img 1
…
```

The second row, number_var, is the total number of variables in the study. This number is equal to the total number of predictor variables plus 1 (this extra variable is the group-membership variable). For example, in a study with image size 64 * 64 * 3, we have 12288 voxels (predictor variables). Then number_var = (64 * 64 * 3) + 1 = 12289.

GAMMA is primary designed for neuroimaging data analysis. The image formats supported by GAMMA suite are Analyze and NIfTI. GAMMA assumes that all input images are binary (0 represents 'no effect' and 1 represent 'effect'. GAMMA also assumes that all input images are in the same stereotaxic space.

gamma.configuration is a simple text file; its format is as follows.

```
mask_type 1                   ← mask type
mask_th 0.3                   ← threshold associated with mask
beta 1                        ← Markov Random Field smoothing parameter
T_gem 5                       ← Generalized Expectation Maximization iteration number
```

```
T_lbp 5                    ← Loopy Belief Propagation iteration number
max_roi 2                  ← maximum number of ROIs allowed
model_order -1             ← how to search the number of clusters
surf_flag 0                ← external neighborhood file
ver 1                      ← verbose
```

The parameter mask_type is set when users want to filter non-relevant variables. For example, in a lesion-deficit study, users often start by excluding voxels that are lesioned for only a small proportion of subjects. The GAMMA suite provides three masking methods:

- mask_type = 0: remove voxels whose values are zero (e.g., background voxels).
- mask_type = 1: remove voxels V for which norm(V) < mask_th; norm(V) is the total number of subjects whose value is 1 at that voxel location.
- mask_type = -1: don't use masking.

How to choose a masking method depends on the application. Removing background voxels is a reasonable default.
Usually, users use the default values for beta, T_gem, and T_lbp.
Typically, beta is in [0.1, 10], T_gem is in [5, 10], and T_lbp is in [5, 20].
The parameter max_roi is the maximum number of ROIs allowed; typically, users choose an integer in [2, 5]. If a study is undersampled (the study with small sample size), GAMMA cannot support a more complicated model, so max_roi should be set to 1 or 2.
The parameter surf_flag determines the spatial model used for image voxels. If surf_flag = 0, GAMMA uses the default spatial model, a 3D lattice. If surf_flag = 1, GAMMA uses an external file to specify neighborhood structures.

Because GAMMA routines load all study images into memory, users may encounter 'out of memory' errors. To reduce memory requirements, users can either process effect maps by removing background voxels (cropping), or by downsampling images.

3.1 How to use GAMMA core

There are three programs in GAMMA core. They all require the files gamma.prj.filelist and gamma.configuration. Users can use any text editor to create these two files.
Under Linux and Mac OS X, if users have 'permission problems', users can use 'chmod +x' to fix it. For example, if OS shows an error message as 'program_name: Permission denied', then users can open a terminal, go to the program directory, type
$chmod +x program_name

Here is a list of available programs.

- gammacoregamma: GAMMA algorithm [1]
- gammacoreel: GAMMA with ensemble learning [3]
- gammacorelatent: Regional-state inference [2]

Usage of these modules is as follows:
3.1.1 GAMMA
Command line: gammacoregamma gamma.prj.filelist
Parameters:

gamma.prj.filelist - gamma file list
Output:
label_field_gamma.{hdr, nii}: label field in analyze or nifti format
belief_map_gamma.{hdr, nii}: belief map in analyze or nifti format
bn.gamma: generated BN in plain-text format
bn.gamma.xdsl: BN in xml format
roi.gamma: label field in ROI format.
log.gamma: run-time information
Example:
gammacoregamma gamma.prj.filelist
This command tells GAMMA to analyze data provided in gamma.prj.filelist

3.1.2 GAMMA with ensemble learning
Command line: gammacoreel gamma.prj.filelist method resampling_size
Parameters:
gamma.prj.filelist: gamma file list
method: an integer variable. It takes values from {1 - jackknife; 2 - bootstrap}
resampling_size: An integer representing the number of resampling iterations.
Example:
gammacoreel gamma.prj.filelist 2 200
This command tells GAMMA with ensemble learning to perform bootstrap resampling with resampling size = 200
Output:
Label_field_gammael.{hdr, nii}: label field in analyze or nifti format
prob_map_gammael.{hdr, nii}: associated probability map in analyze or nifti format
bn.el.gamma: generated BN in plain text format
bn.el.xdsl: BN in xml format.
roi.el.gamma: label field in ROI format.
log.el.gamma: run-time information

3.1.3 Regional-state inference (RSI)
Command line: gammacorelatent gamma.prj.filelist roiFile method
Parameters:
gamma.prj.filelist: gamma file list
roiFile: roi definition
method: {lm, voting}. lm : Latent model; voting : Voting
Output:
roi_state_gamma.csv: regional state in csv format
roi_state_gamma.arff: regional state in arff format
Example:
gammacorelatent gamma.prj.filelist roi.gamma lm
This command tells RSI to perform latent model inference for ROI defined in file roi.gamma

This program requires a ROI definition file. In this file, voxels are arranged as a column vector. The format of this file is as follows:

```
_______
number of ROIs, number of voxels
```

```
label1, label2, ... labelN

______
Label is 1 for ROI 1, 2 for ROI 2, ...
```

An example ROI file is as follows

```
2 256
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 2 2 2 0 0 0 0 0
0 0 0 0 0 0 0 0 2 2 2 0 0 0 0 0
0 0 0 0 0 0 0 0 2 2 2 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Users can use gammacoregamma or gammacoreel to generate ROI files.

The comma-separated values (csv) format is a widely used text format. Users can view and edit csv files using Excel. The arff format is supported by Weka, a widely used data-mining software package (http://www.cs.waikato.ac.nz/ml/weka/).

3.2 How to use GAMMA GUI

Here is how to start the GAMMA GUI.

- Linux users. Open a xterm, and type gamma_gui_start.sh if users set up program paths (details of setting up program paths are in page 2, section 2). If users don't set up program paths, open a xterm, go to the folder including the GAMMA GUI, and type ./gamma_gui_start.sh. For example, if GAMMA GUI is in '~/soft/gamma_suite/end_user_program, you can open a xterm,
  $ cd '~/soft/gamma_suite/end_user_program
  $ ./ gamma_gui_start.sh
- Mac OS X user. GAMMA GUI is provided as a dmg file. Double-click the dmg file to open it. Locate GAMMA GUI application's icon within this new Finder window. Drag and drop it into your "Applications" directory. Details of using dmg file are in http://www.ofzenandcomputing.com/zanswers/779. After this, just double click the GAMMA GUI icon.
- Windows user. Double click the GAMMA GUI program.

Under Linux and Mac OS X, if users have 'permission problems', users can use 'chmod +x' to fix it. For example, if OS shows an error message as 'program_name: Permission denied', then users can open a terminal, go to the program directory, type
$chmod +x program_name

3.2.1 Prepare study data
Put all relevant image data into a directory. GAMMA GUI requires two additional files.
Image list file: a plain text file. The format is (subject id, subject image file). Here is an example

```
1 subj_1.img                          ← subject id = 1, image file = subj_1.img
2 subj_2.img
3 subj_3.img
…
```

Clinical variable file – clinical variable is provided as a CSV file. The group membership variable *C* MUST be the last variable. The first variable must be the subject id. Here is an example

```
1,0                                    ←subject id = 1, C = 0
2,0
3,1
…
```

3.2.2 Run GAMMA GUI
Start the program.

1. The first step is to create a GAMMA project [menu Project→New project]. GAMMA GUI will display a dialog to obtain the location of the project's home directory (the directory where data are stored). After the user specifies this directory, GAMMA GUI asks for the image list file, and imports clinical variable file. After this step, GAMMA GUI will set the project parameters, as shown in Figure 1. This step generates gamma.prj.filelist and gamma.configure.
2. Change parameters. First open a project, then use the [Project→Set parameters] command to change the project's parameters.
3. Run GAMMA. Select [Project→Open project], and then [Algorithm→GAMMA]. GAMMA's results can be viewed using the [Report→GAMMA result] menu item.
4. Run GAMMAEL. Select [Project→Open project], then [Algorithm→GAMMA EL]. Results can be viewed with [Report->GAMMA EL result].
5. To run regional state inference for the ROI(s) generated by GAMMA, use [Project→Open project], then [Algorithm→GAMMA ROI]. Results will be in the files roi_state_gamma.csv and roi_state_gamma.arff.
6. To run regional state inference for the ROI(s) generated by GAMMAEL, use the [Project→Open project], then [Algorithm→GAMMA EL ROI]. Results will be in the files roi_state_gammael.csv and roi_state_gammael.arff.
7. To run regional state inference for the ROIs defined by an external file, use [Project→Open project], then [Algorithm→GAMMA ROI]. Set the file name in the dialog to the name of the external file. Results will be in the files roi_state_gamma.csv and roi_state_gamma.arff.

A video file gamma_gui.mov demonstrates how to use GAMMA GUI.
For troubleshooting, please read appendix A.3.

4. Example Analyses

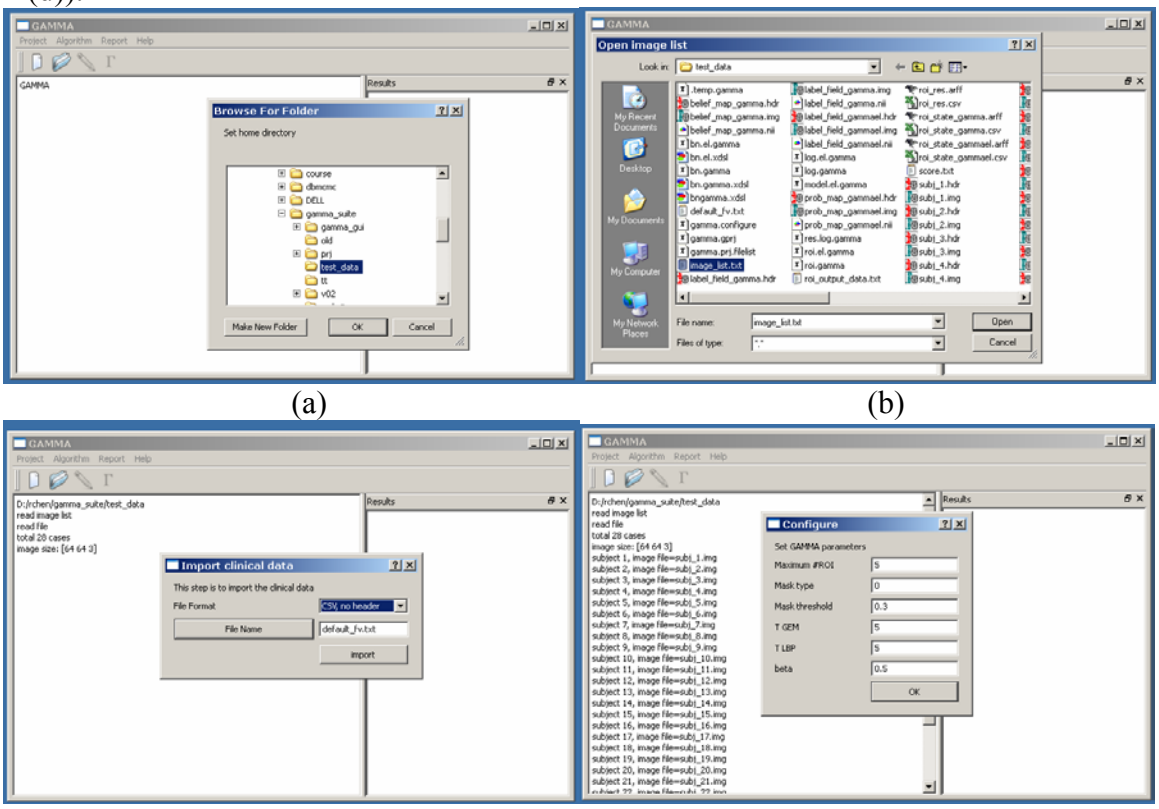
test_data.zip includes test data.

4.1 Simulated test data

We have included a sample data set in the installation directory gamma_suite/test_data/test_small (download test_data.zip and unzip it). In this data set, a voxel in the effect map represents whether or not a region is atrophic. We indicate by [$A = 1$; $B = 1$; $F = 1$] that region A was atrophic, region B was atrophic, and there was a functional deficit. These data consist of 28 samples: 10 for which [$A = 0$; $B = 0$; $F = 0$], 10 for which [$A = 1$; $B = 0$; $F = 1$], and 8 for which [$A = 0$; $B = 1$; $F = 1$]. We added simulated "salt and pepper" noise to these images. In these simulated images, both regions A and B are rectangles. In the directory test_small, you will find 28 effect maps (subj_1.img … subj_28.img), one image list file (image_list.txt), and one clinical variable file (default_fv.txt).

To perform the analysis, complete the following steps:

Step1. Create a project: [Project→New project], set the project home directory (Figure 1 (a)), set the image file list (Figure 1 (b)), set the clinical variable file (Figure 1 (c)), and set the parameters (Figure 1 (d)).



(a) (b)

(c) (d)

Figure 1. Steps in creating a new project.

Step 2. Run GAMMA [Algorithm→GAMMA], and check the results (Figure 2), which will have been saved in the directory test_small/res/. Users can compare the output from their analysis to those results stored in that directory, and check whether GAMMA GUI correctly runs under their system.

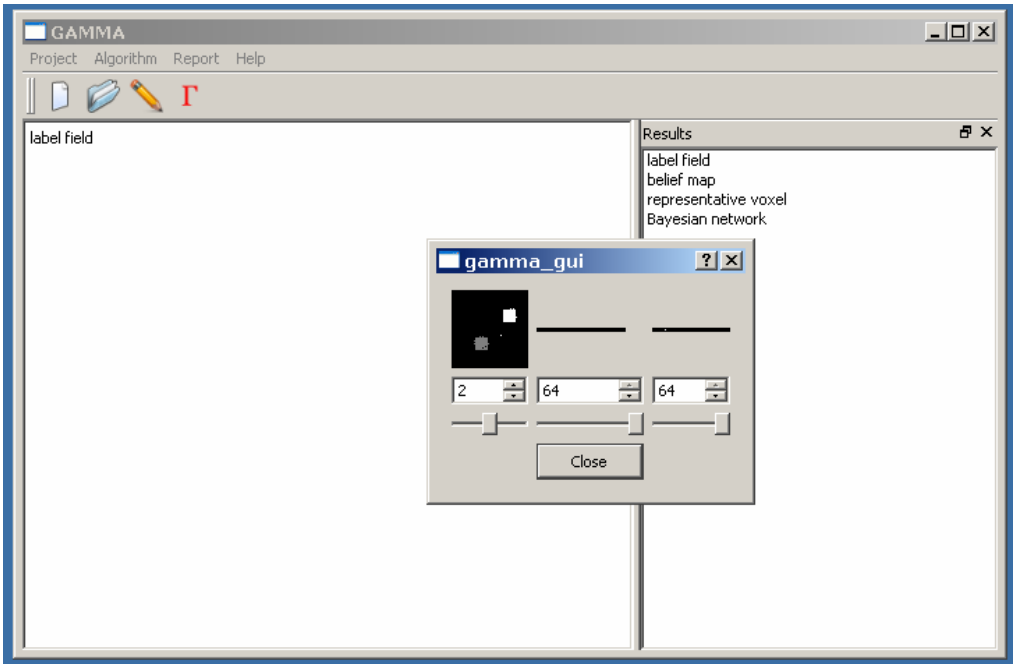


Figure 2. View GAMMA results.

Step 3. Run GAMMAEL with Jackknife resampling [Algorithm→GAMMA EL] (Figure 3). Results will be saved in directory test_small/res/

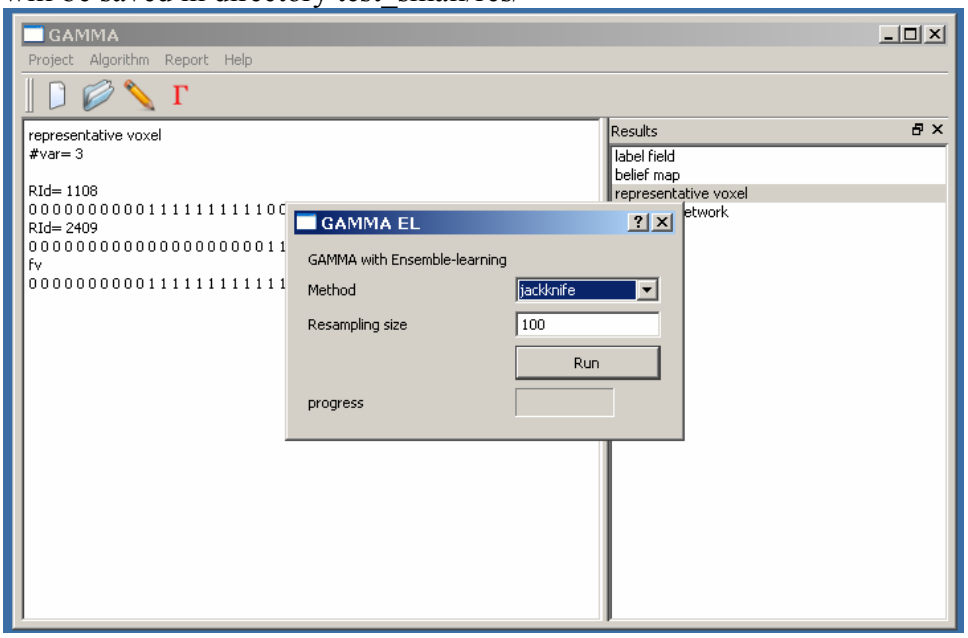


Figure 3. GAMMEL with Jackknife resampling.

Step 4. Run regional state inference for ROI generated by GAMMA or GAMMAEL. Use [algorithm→GAMMA ROI] and [algorithm→GAMMA EL ROI]. Results will be saved in the directory test_small/res/. Because regional state inference is a stochastic algorithm, the final result depends on the initial state. If two results are different only with respect to state coding, they are

effectively identical. For example, if for a ROI, one state variable is [1 1 0 0], and the other state variable is [0 0 1 1], these two variables encode the same information.

4.2 A lesion-deficit study

Due to ethic and regulatory issues, we cannot provide real clinical data as a test data set for the GAMMA suite.

We constructed simulated lesion maps based on the CH2 or AAL template [4]. This data set is in gamma_suite/test_data/test_lesion (download test_data.zip and unzip it). This study included 30 subjects (15 in group A and 15 in group B).

First, we chose a region of interest. In this experiment, we chose Brodmann area (BA) 39 in the CH2 atlas as the region of interest. Second, we introduced lesions; that is, voxels in BA 39 would be abnormal with probability λ, and voxels outside of BA 39 would be abnormal with probability μ. λ and μ represent the signal and noise levels, respectively. Third, to make the simulation more realistic, we applied an in-plane median filter to each subject's lesion map to increase spatial smoothness.

In our experiment, for subjects in group B, each brain voxel had a probability of 0.1 of assuming value '1' (being lesioned). For subjects in group A, a voxel inside the brain but not in BA 39 had a probability of 0.1 of being lesioned, and a voxel in BA 39 had a probability of 0.6 of being lesioned.

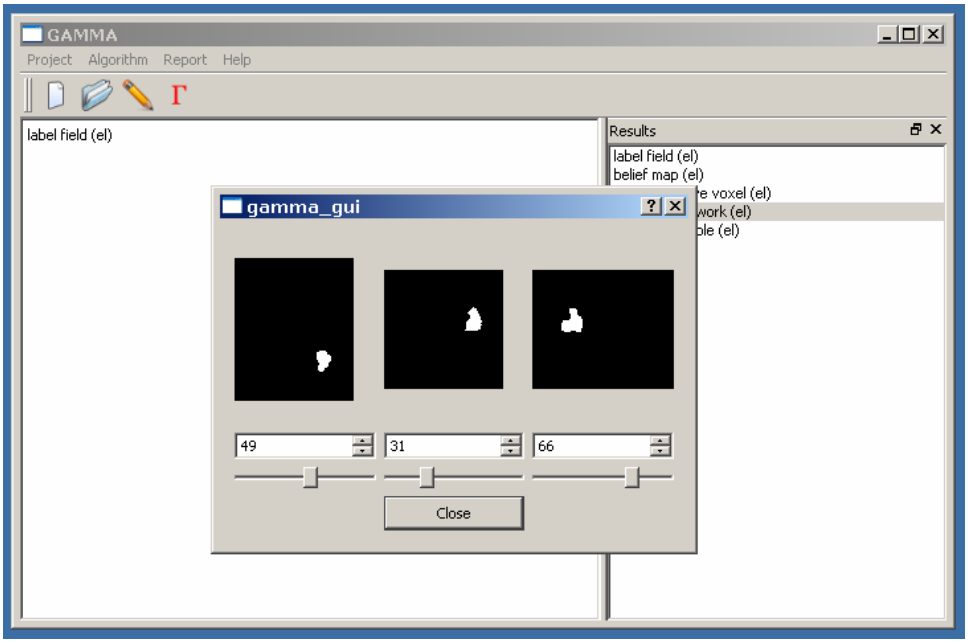


Figure 4. Using GAMMA GUI to analyze lesion-deficit data

We used GAMMAEL with Jackknife resampling to detect regions characterizing the group difference. GAMMAEL required approximately 10 minutes to generate a model consisting of a single brain region whose status was predictive of *C,* using a Linux desktop with an Intel Xeon 3.4G CPU and 4Gb memory. For a given subject, if voxels in this region demonstrated volume reduction, the BN generated by GAMMA would estimate a 0.94 probability of this subject's being in the group A. Figure 4 shows a screenshot of using GAMMA GUI to view the generated label field.

We also used regional state inference to infer the regional state for the generated ROI, using a latent-variable model. RSI required approximately 10 minutes. Subsequently, we used the generated regional state variable to predict *C* using naïve Bayes and support vector machines (SVM). For both naïve Bayes and SVM, the classification accuracy using ten-fold cross-validation was 1.0. Users can use Weka to analyze the regional state file (roi_state_gamma.arff) to verify this result. All results are saved in test_lesion/res/.

5. Discussion

Algorithms in the GAMMA suite are designed to handle binary effect maps. The problem domain of interest is neuroimaging data analysis, primarily lesion-based studies, functional MR-based studies, and morphometric studies. The majority of neuroimaging studies belong to these three kinds of study types. Therefore, algorithms in the GAMMA suite have wide applicability.

An important question is validity of using binary effect maps. For lesion-based studies, most studies focus on how the lesioned voxels (or brain regions) are associated with deficit. Usually investigators manually delineate lesioned voxels. Therefore, labeling a voxel as lesioned/non-lesioned is a natural selection. For functional MR-based studies, we focus on whether or not voxels are activated during a task, and how this activation pattern is associated with other variables. Therefore, using binary activation map is valid. For morphometric studies, we can consider tissue loss in a voxel location is associated with a latent variable which represents whether or not this voxel is affected by the disease. This justifies the practice of using binary effect maps. However, using binary effect maps is a limitation of our algorithms. This is because binarizing continuous variables may cause loss of information.

Appendix

A.1 Supported image formats

The image formats supported by GAMMA suite are Analyze and NIfTI. Clinical MRI systems often provide images in DICOM format. We can use MRIConvert (http://lcni.uoregon.edu/~jolinda/MRIConvert/) or dcm2nii (http://www.cabiatl.com/mricro/) to convert DICOM images to NIfTI format.

A.2 Compiling GAMMA core under Windows

1) Install MinGW (http://www.mingw.org/).
2) Install MSYS (http://www.mingw.org/). When users install MSYS, please correctly setup the path for MinGW.
3) Download gamma_suite_src.zip.
4) Unzip it to a directory. Assume the directory is '~/soft/gamma_suite'.
5) Open a MSYS terminal.
6) Run gamma_install_win.sh in directory '~/soft/gamm_suite/src/gamma_core'
7) The executable binaries gammacoregamma.exe, gammacoreel.exe, and gammacorelatent.exe are found in the directory '~/soft/gamma_suite/src/gamma_core/bin'.
8) Add the GAMMA program directory to the OS search path. Right click 'My Computer', select [Properties]. Then under [Advanced→Environment variables], change the variable PATH by adding the GAMMA program directory.

A.3 GAMMA GUI troubleshooting

- Under Windows, if users get an error message such as 'qtcored4.dll not found', add your Qt bin directory (such as C:\Qt\2009.04\qt\bin) to the PATH environment variable as follows: My computer→properties→advanced→Environment Variables→PATH. Also, you should have all Qt dll files (QtGuid4.dll, QtCored4.dll, etc) in this directory
- GAMMA GUI was developed based on Qt. However, it is not as stable as GAMMA core. I checked an empty Qt GUI application under Linux using Valgrind (Valgrind is a tool to detect memory management problems. It's available for Linux or Mac OS X). Valgrind reports that Qt itself (or third party programs used by Qt4) may have the memory leaks, which could cause the GAMMA GUI to crash when a project uses a lot of memory. I checked GAMMA core using Valgrind, and found no memory-management errors. Therefore, users with a memory-intensive project can first use GAMMA GUI to create a project, and then use GAMMA core to run GAMMA, GAMMAEL, or regional state inference, without using the GUI.
- Under Mac OS X, when the user clicks [Report→GAMMA result] or [Report→GAMMA EL result], the list menu may not appear. This is a known Qt4 problem under Mac OS X. Just click the empty space inside the right dock window of GAMMA GUI, and the list menu should appear.

A.4 Examples of image-processing pipelines

Generally, applying GAMMA to clinical studies includes two stages: image preprocessing, and GAMMA analysis. Here are three examples of the image-preprocessing step.

- ***A cross-sectional study of brain morphometry.*** The task is to find the brain regions that can characterize group differences. In this case, C is a subject's group membership. For each subject, we acquire structural MR images and measurements of C. After image-preprocessing steps, including segmentation, registration, and smoothing, we obtain a map in which each voxel's intensity is proportional to the volume of that region. After thresholding (we can choose the sample mean or median as threshold), we obtained a binary *effect map,* in which voxels with value 1 represent volume reduction, and voxels with value 0 represent normal.
- ***A longitudinal study of brain morphometry.*** The task is to find brain regions that characterize group differences. As in the previous example, C represents a subject's group membership. For each subject, we obtain structural MR images for two different times, $t1$ and $t2$, along with measurements of C. For each subject's pair of MR images, after preprocessing steps, including as segmentation, registration, and smoothing, we obtain a map in which each voxel's intensity is proportional to the volume of that region. After thresholding (subtracting $t1$ volume map from the $t2$ volume map and thresholding by zero), we obtain an effect map. The input to GAMMA is a set of effect maps and associated function-variable states. Figure 5 shows this process.
- ***Detecting group differences in brain activation recorded by fMRI***. In this scenario, the task is to identify brain regions characterizing group differences. Data are preprocessed using SPM (http://www.fil.ion.ucl.ac.uk/spm/) or Voxbo (http://www.voxbo.org), resulting in a statistical parametric map (T-map or F-map) for each contrast. We then choose a significance level α to threshold these statistical parametric maps, and generate binary maps representing voxel activation: each voxel assumes a value in {0, 1}, corresponding to off (no activation) and on (activation), respectively. We submit these binary effect maps and C as input to GAMMA.

- ***Lesion-deficit analysis***. For lesion studies, we delineate abnormal brain voxels based on MR or CT images. We can either manually label abnormal regions or use automatic segmentation tools to delineate these lesions. We refer to the delineated abnormal brain regions as the subject's lesion map. In a subject's lesion map, if a voxel is damaged, it is labeled as `1'; otherwise, it is labeled as `0'. We provide binary lesion maps (effect maps) and *C* as input to GAMMA.

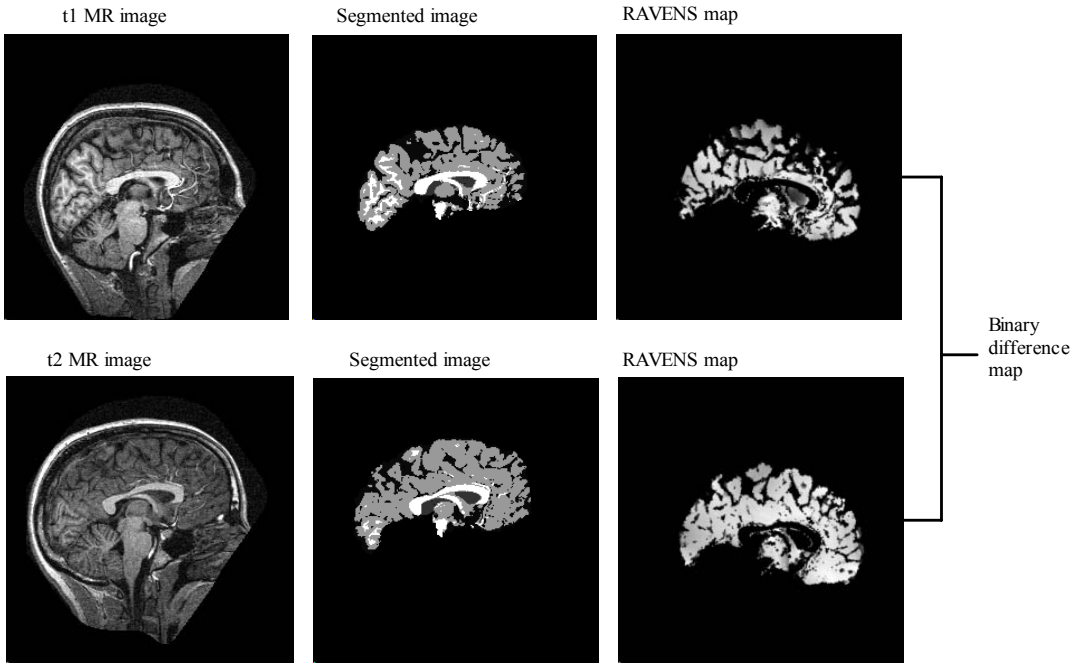


Figure 5. Longitudinal study for brain morphometry.

A.5. Markov random fields (MRFs)

MRF is a probabilistic graphical model. In the GAMMA suite, MRF is not implemented as a class. MRF is implemented based on the class dbnSamples which represents a 3D image volume. The definition of spatial structure and potential function of MRF is in dbnGAMMA; the inference algorithm (loopy belief propagation) is in dbnGAMMA and dbnMessage.

In this package, we use the two layer MRF model. An example is in Figure 6. In this MRF model, **Y**={Y1, Y2, Y3} are observed node (the evidence node), and **X**={X1, X2, X3} are unobserved. X1, X2, X3 are connected. The neighborhood of X2 is {X1, X3}. We want to infer the value of **X** based on **Y**. An example case is image segmentation. We want to infer the label (grass, cloud, or human) based on the observed image intensity.

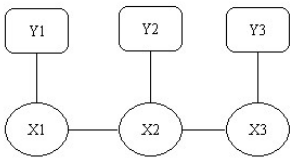


Figure 6. A MRF model.

In this package, X is a categorical variable (the label). MRF for **X** is a Potts model [1]. The smoothing parameter is β. The potential function between Xi and Xj is defined in function create_transition_matrix_kclsutering (in dbnGAMMA.cpp). Users can also use their own potential function (transition matrix) for MRF inference.
The connection between Y and X is defined in function create_evidence_kclustering (in dbnGAMMA.cpp). In this package, we use the Gaussian model. Users can change it to other models appropriate for their own applications.
Here is the list of related functions.

- Represent a set of (or a single) image volume – use class dbnSamples
- Define the neighborhood system (spatial structure) - create_nbhd_system_kclustering (if users use second order MRF) or create_nbhd_system_fromfile (if users specify the neighborhood) in dbnGAMMA.cpp
- LBP for MRF inference - lbp_inference in dbnGAMMA.cpp
- Infer **X** based on **Y** using LBP- lbp_seg in dbnGAMMA.cpp

Users can find details of these functions in the html file.

Frequently asked questions (FAQ)

Q1. 'Out of memory' errors.
A. Because GAMMA routines load all study images into memory, users may encounter 'out of memory' errors. To reduce memory requirements, users can either process effect maps by removing background voxels (cropping), or by downsampling images.

Q2. How to recover the Results subwindow after closing it?
A: [Help -> Results]

Q3. OS shows an error message as 'program_name: Permission denied'. How to fix it?

A: Under Linux and Mac OS X, if users have 'permission problems', users can use 'chmod +x' to fix it. For example, if OS shows an error message as 'program_name: Permission denied', then users can open a terminal, go to the program directory, type
$chmod +x program_name

Q4. Under Mac OS X, when the user clicks [Report→GAMMA result] or [Report→GAMMA EL result], the list menu may not appear.
A: This is a known Qt4 problem under Mac OS X. Just click the empty space inside the right dock window of GAMMA GUI, and the list menu should appear.

References

1. Chen R, Herskovits EH. Graphical-model based morphometric analysis. *IEEE Transaction on Medical Imaging.* 2005;24(10):1237-1248.

2. Chen R, Herskovits EH. A Bayesian Network Classifier with Inverse Tree Structure for Voxel-wise MR Image Analysis. Paper presented at: the eleventh conference of SIGKDD, 2005.

3. Chen R, Herskovits EH. Graphical-model-based multivariate analysis of functional magnetic resonance data. *NeuroImage.* 2007;35:635-647.
4. Holmes CJ, Hoge R, Collins L, Woods R, Toga AW, Evans AC. Enhancement of MR images using registration for signal averaging. *J Comput Assist Tomogr.* Mar-Apr 1998;22(2):324-333.
5. Smith SM, Matthews PM, Gurd JM, Slater A. Multi-subject statistic map combination. Paper presented at: HBM98, 1998.
6. Larsen S, Kikinis R, Talos IF, Weinstein D, Wells W, Golby A. Quantitative comparison of functional MRI and direct electrocortical stimulation for functional mapping. *Int J Med Robot.* Sep 2007;3(3):262-270.
7. Lashkari D, Sridharan R, Vul E, Hsieh P-J, Kanwisher N, Golland P. Nonparametric Hierarchical Bayesian Model for Functional Brain Parcellation. Paper presented at: MMBIA: IEEE Computer Society Workshop on Mathematical Methods in Biomedical Image Analysis, 2010.