SUNVS - A Surface-based Brain Network Viewer Toolbox

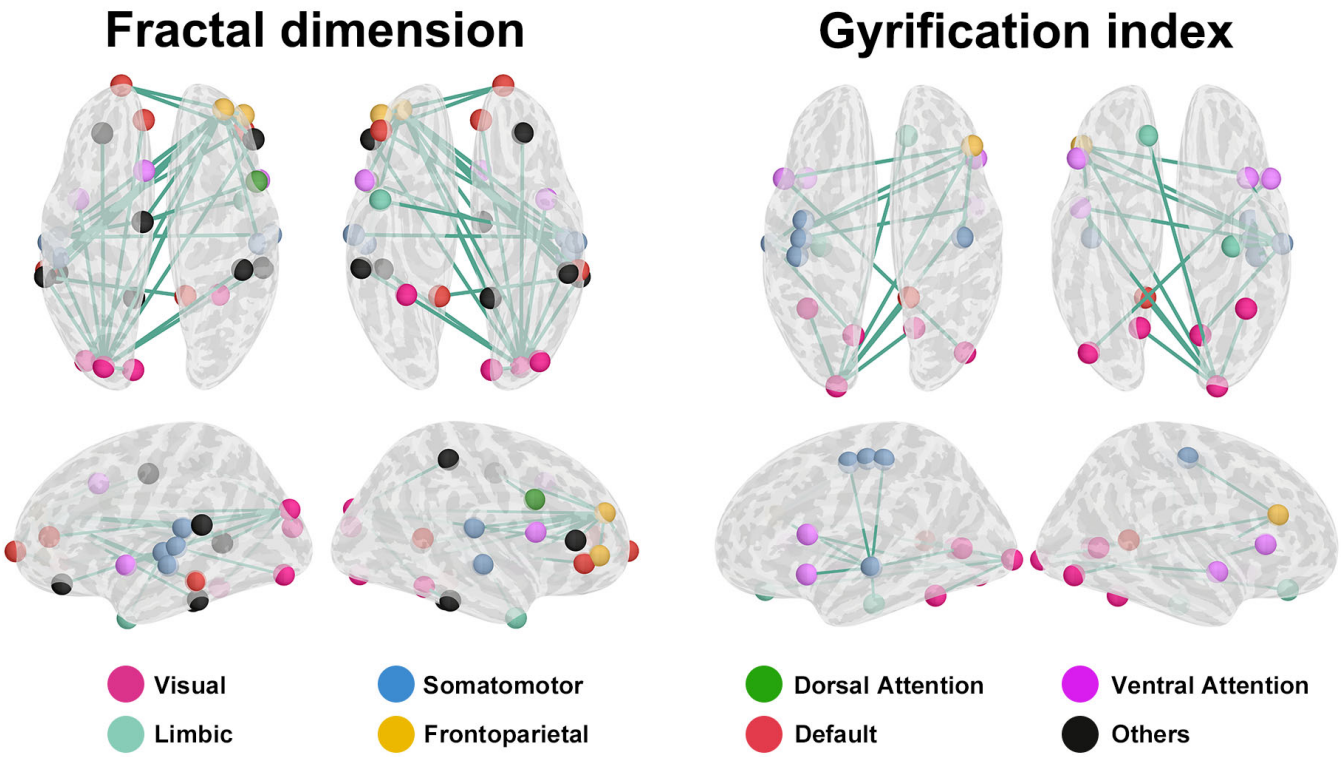
Grab your towel and don't panic

*Please **feel free** to use this toolbox.*

Website: *https://github.com/c14h19no2/SUNVS*

DOI: *10.5281/zenodo.4044779*

I am pleasure if you'd like to cite me as follows: *Wang, Ningkai. (2020). c14h19no2/SUNVS - A Surface-based Brain Network Viewer Toolbox. Zenodo. doi:10.5281/zenodo.4044779*



绘图指引

0. 前期准备

1. 本绘图工具包的功能依赖于 *CAT12*，因此在使用前请首先下载 *CAT12* 及 *SPM12*，并正确设置路径 (*setpath*)
2. 本工具包适用于 164k *.gii* 格式的 *surface* 文件（共包含 163842 个顶点），双侧半球的 .gii 文件需分开储存
3. 为方便软件识别，左脑的 surface *.gii* 文件建议更改前缀 *lh.*，右脑的 surface *.gii* 文件更改前缀

为 *rh.*，如将 *myLeftBrain.gii* 更改为 *lh.myLeftBrain.gii*，*myRightBrain.gii* 更改为 *rh.myRightBrain.gii*

1. 工具包内容

1.1. 主函数

本绘图工具包包括 5 个函数，绘图时我们主要会用到以下 4 个：

```
1. sunvs_display % 用颜色标注脑区
2. sunvs_net_viewer % 展示脑网络中的节点和连边
3. sunvs_gen_centroid_coor % 根据图谱所定义的ROI和选定的皮层形状，快速生成ROI的中心坐标
4. sunvs_ROI_annot2gii % 根据特定图谱中脑区编号，快速生成相应gii文件
```

1.2. 依赖函数与文件

此外，本工具包内还存在如下函数和文件，为 *sunvs_net_viewer* 和 *sunvs_display* 函数提供依赖。

```
1. sunvs_plot_3dsphere
% sunvs_net_viewer的依赖函数，用于绘制3D节点

2. nodalBoundaryList文件夹中
2.1. [lr]h.inflated.Uniform.gii
% 为 sunvs_net_viewer 函数提供默认底版

2.2. [lr]h.nodalBoundaryList_[atlas].gii
% 包含图谱的脑区边界信息，可用作 underlay，对脑图的可视化效果进行优化。
```

2. 在 *Surface* 底版上绘制脑图

2.1. 在 *Surface* 底版上展示特定脑区 // *sunvs_display*

2.1.1 绘制单侧半球

要显示 *surface* 上的特定脑区，我们会用到 *sunvs_display* 函数。在使用该函数之前，我们首先需要得到记录了脑区信息（可以是脑区的显著性，分属模块的编号，以及皮层厚度等等）的 *.gii (gifti)* 文件。
假设我们已经得到了需要输出的 *.gii* 文件，分为左右半球，我们只需要绘制出其中的左半球，其名为 *'lh.mySurface.gii'*，文件名的前缀 *'lh.'* 代表其为左脑在 *matlab* 命令行窗口输入：

```
sunvs_display('lh.mySurface.gii');
```

软件即会自动绘制左半球。

2.1.2. 绘制双侧半球，参数：'multisurf'

当然，很多时候我们会有绘制双侧半球的需求，此时我们只需添加参数 *'multisurf'*，并将该参数的值设置为1：

```
sunvs_display('lh.mySurface.gii', 'multisurf', '1');
```

软件包即会自动找到 *'lh'* 的对侧半球，即 *'rh'*，将左右半球同时绘制出来。同理，如果你选择 *'rh'* 时将 *'multisurf'* 参数设置为 1，代码也会自动在 *'rh'* 的同路径下寻到 *'lh'*。

2.1.3. 选择大脑形状，参数：*'usefsaverage'*

Surface 是把皮层抽象化得到的二维平面，理论上在拓扑结构不变的前提下可以扭曲成任何形状，例如膨胀起来便于观察；可以呈现为褶皱的 *central surface* 或是平滑的 *pial surface*；甚至可以将其信息映射到一个球面上。这种高度自由的特性为我们的数据可视化提供了许多便利，我们可以用不同的载体来呈现 *Surface* 上的信息。

比如：

```
sunvs_display('lh.mySurface.gii', 'multisurf', '1', 'usefsaverage', 'inflated');
% 使用膨胀的 surface 作为数据可视化的大脑底版，优势在于可以将原本隐藏的脑沟暴露在外，便于观察。


sunvs_display('lh.mySurface.gii', 'multisurf', '1', 'usefsaverage', 'IXI555');
% 使用 CAT12 DARTEL 模版的 surface 作为数据可视化的大脑底版，优势在于其与真实大脑形状相近。
% 可以看到，利用多个参数的组合，我们可以在指定函数呈现左右半球的同时使用某个特定底版。
```

此外，软件包还可以使用 *CAT12* 提供的 *central surface* 作为底版，亦可以自己选定 *.gii* 文件作为底版(*'custom'*, *'文件路径'*)，具体使用方法可以查阅函数注释，此处不再赘述。

2.1.4. 选择大脑底版纹理，参数：*'useUnderlay'*

我们在可视化时常用的 surface 底版是 *'inflated'*，即膨胀的脑。该底版将原本隐藏的脑沟暴露在外，便于观察，但其带来的缺点也很明显，即抹去了原本的沟回信息，我们在一个光滑的脑上难以辨认一个区域原本所属的脑区。*'useUnderlay'* 参数即用于给上述大脑底版添加纹理，以作为解决方案。

目前的纹理分为两种：

1. 沟回信息，参数值为 **'mc'**，选择该纹理时，大脑底版上会出现黑白间隔的图案，指示该位置是脑沟还是脑回
2. 脑区信息，参数值为 *'a2009s'*, *'DK40'* 等 surface 图谱名缩写，选择该纹理时，大脑底版上会出现所选图谱的所有脑区的边界线。

使用方法很简单，挑选好你想要的纹理，例如你在先前的大脑分区、统计阶段都使用了 a2009s 图谱，可使用 a2009s 的脑区边界，在命令行输入：

```
sunvs_display('lh.mySurface.gii', 'useUnderlay', 'a2009s');
```

软件即会在 *a2009s* 图谱定义的所有脑区周围绘制上白边。

2.1.5. 选择透明度，参数：*'TransParency'*

有时我们需要看到对侧的脑，亦或是单纯为了美观，我们可以对脑设置透明度，此时使用 *'TransParency'* 参数即可，数值范围为 *0-1*，值越高越透明。

```
sunvs_display('lh.mySurface.gii', 'TransParency', 0.4);
```

2.1.6. 选择观看视角，参数：*'view'*

基于 *'view'* 参数，我们可以指定图片在生成时的呈现方位。其中：

> 'l' = left
> 'r' = right
> 'a' = anterior
> 'p' = posterior
> 's' = superior (Default)
> 'i' = inferior

若我们希望呈现脑的左半侧面，只需添加 'view' 参数：

```
sunvs_display('lh.mySurface.gii', 'view', 'l');
```

2.1.7. 选择颜色表，参数：*'Colormap'*

若要给 ROI 指定颜色，我们可以使用 *'Colormap'* 参数，主要有以下几种用法：

```
sunvs_display('lh.mySurface.gii', 'Colormap', jet(64));
% .gii 文件中的所有值将 map 到 jet 这一配色方案的 colormap 中，其中这一 corlormap 分为 64 个
```

我们可以尝试看一下 *jet()* 函数的输出，以较简短的 *jet(8)* 为例：

```
>> jet(8)

ans =

    0    0    1
    0    0.5  1
    0    1    1
    0.5  1    0.5
    1    1    0
    1    0.5  0
    1    0    0
    0.5  0    0
```

可以看到，它给出了一个 8*3 的矩阵，其中 8 行代表 8 个颜色梯度，3 列分别代表颜色的 RGB 值，该 *RGB* 值可从常用的 256 值的 *RGB* 值转换而来，例如蓝色是 [0, 0, 256]，对每个值除以 256 来进行标准化，即得到了 [0,0,1]，即 *jet(8)* 中的第一个值。而我们 *.gii* 文件中的值则会 *map* 到这张表的颜色中，若我们的 *.gii* 文件中有 6 个值，我们拥有最小值的脑区会得到 *colormap* 中第一个颜色，也就是蓝色。

类似的 colormap 还有很多，例如 *parula*, *hsv*, *hot*, *pink*, *flag*, *gray*, *cool*, *copper*, *white* 等等，可以在它们的 *help* 文件中找到更多相似的 *colormap*， 此处不赘述。

既然我们知道了 *colormap* 的构成和数据结构，我们就可以自己定义 colormap 来更精细地操纵图中的色彩。例如，我们的 *.gii* 文件中有 6 个脑区，它们各自分属于 [1 2 3 2 3 1] 模块，那么我们直接给每个脑区的 *vertices* 赋值 [1 2 3 2 3 1]，并创造一个 *colormap* 矩阵，给每个脑区直接定义其颜色（共 3 种，各自对应一个模块）：

```
>> myColormap = [1    0    0; ...
                 0    1    0; ...
                 0    0    1; ...
                 0    1    0; ...
                 0    0    1; ...
                 1    0    0];
```

最后将 *.gii* 文件路径和 *colormap* 导入函数

```
sunvs_display('lh.mySurface.gii', 'Colormap', myColormap);
```

得到的脑图上就能出现分属 3 种模块的 6 个脑区了。

2.1.8. 输出图片到指定目录，参数：*'imgprint'*，*'imgprintDir'* 与 *'dpi'*

呈现图的同时输出图片到指定目录，要用到两个参数：*'imgprint'* 与 *'imgprintDir'*：

```
sunvs_display('lh.mySurface.gii', 'imgprint', 1, 'imgprintDir', '/Users/Username');
% 若不指定 'imgprintDir'，只设置 'imgprint' 为 1，程序会自动输出图片到 matlab 目前的的工作路径
% 若不设置 'imgprint' 为 1，只指定 'imgprintDir' 的文件目录，程序也可正常运行。
```

输出的图片为 *.tif* 格式，文件名跟随 *.node* 文件，分辨率默认为 600 *dpi*，也可以通过 *'dpi'* 参数来更改。

2.2. 在 Surface 底版上绘制脑网络 // *sunvs_net_viewer*

要在 Surface 底版上绘制脑网络，我们首先要制作节点文件(*.node file*)与连边文件(*.edge file*)。

2.2.1. 节点文件

.node 是一个文本文件，以 *.node* 为后缀名（这里沿用了 *Brainnet viewer* 的格式与命名方式）。文件中包括 3 类信息：

1. 每个节点的 3 维坐标信息
2. 每个节点分属的模块（即颜色）
3. 每个节点的重要性（即大小）

文件内容示例如下：

```
-24    48   -2     1    1    Node1
-34   -77   -7     2    2    Node2
12    -32    62    3    1    Node3
52    -11    17    2    2    Node4
17     52    3     3    2    Node5
-7     35    10    1    0    Node6
```

其中，

1. 每一行代表一个节点
2. 每一行的前 3 列代表这个节点的三维坐标，分别为 *X*，*Y*，*Z* 轴坐标。坐标信息可以通过函数 *sunvs_gen_centroid_coor* 获取，使用方法见本文档 ***第 3.1. 部分***。
3. 每一行的第 4 列代表这个节点的模块信息，每个模块都会被指定一种颜色
4. 每一行的第 5 列代表这个节点的重要性信息，该值越大，图中该节点就越大。若不希望显示某个节点，那么把它的大小设置为 0 即可，该节点及其连边都将被隐藏。

2.2.2. 连边文件

连边文件是以 *.edge* 为后缀的文本文件，文件中是一个 N*N 的对称矩阵，其中 N 为节点数量。

内容示例如下：

```
0    1    0    0    0    0
1    0    0    1    0    0
0    0    0    0    1    0
0    1    0    0    1    0
0    0    1    1    0    0
0    0    0    0    0    0
```

注：虽然示例内容是二值矩阵，但实际也可以绘制加权矩阵，亦可以绘制负连接。

2.2.3. 网络绘制

网络绘制需要用到 *sunvs_net_viewer* 函数，我们可以选择只呈现节点：

```
sunvs_net_viewer('myNodes.node');
```

也可以选择同时呈现节点与连边：

```
sunvs_net_viewer('myNodes.node', 'myEdges.edge');
```

此外，由于该函数底层调用了 *sunvs_display* 函数，因此 *sunvs_display* 的参数理论上 *sunvs_net_viewer* 基本都可调用，用法相同，此处不赘述。

2.2.4. 节点与连边的尺寸，参数： *'NodeWeight'* 与 *'EdgeWeight'*

除了 *sunvs_display* 所能使用的参数外，本软件包还针对网络节点与连边设置了一些参数，可以自由选择：

在呈现节点时，在 a) 使用相同尺寸 b) 使用 *.node* 文件中的权重 之间切换

```
sunvs_net_viewer('myNodes.node', 'myEdges.edge', 'NodeWeight', 1);
% 使用 .node 文件中的权重
```

```
sunvs_net_viewer('myNodes.node', 'myEdges.edge', 'NodeWeight', 0);
% 使用相同尺寸
```

与此相似，在呈现连边时，可在 a) 使用相同粗细 b) 使用 *.edge* 文件中的权重 之间切换

```
sunvs_net_viewer('myNodes.node', 'myEdges.edge', 'EdgeWeight', 1);
% 使用 .edge 文件中的权重
```

```
sunvs_net_viewer('myNodes.node', 'myEdges.edge', 'EdgeWeight', 0);
% 使用 .edge 文件中的权重
```

2.2.5. 模块的颜色与纹理，参数： *'ModuleColor'* 与 *'ModuleTexture'*

我们可以使用 *'ModuleColor'* 与 *'ModuleTexture'* 为脑图中不同模块的节点添加颜色与纹理，方法如下：

2.2.5.1. 模块颜色，参数： *'ModuleColor'*

同前文 *sunvs_display* 中的 *colormap* 参数相似，我们可以自己定义 *ModuleColor* ，以精细地操纵图中的色彩。例如，我们的 *.node* 文件中有 6 个脑区，它们各自分属于 [1 2 3 2 3 1] 3 大模块，那么我们直接给 *.node* 文件中每个节点的第 4 列分别赋值 [1; 2; 3; 2; 3; 1]，并创造一个 *ModuleColor* 矩阵，给每个模块定义其颜色（共 3 种，各自对应一个模块）：

```
>> myModuleColor = [1    0    0; ...
                    0    1    0; ...
                    0    0    1];
```

最后将 *.node* 文件路径和 *ModuleColor* 导入函数

```
sunvs_net_viewer('myNodes.node', 'myEdges.edge', 'ModuleColor', myModuleColor);
```

得到的脑图上就能出现分属 3 种模块的 6 个脑区了。

2.2.5.2. 模块纹理，参数： *'ModuleTexture'*

'ModuleTexture' 参数的功能与 *'ModuleColor'* 类似，不同的是 *'ModuleColor'* 为不同模块的节点添加颜色，而 *'ModuleTexture'* 则是为不同模块的节点添加纹理。要使用这一参数，我们首先需要新建一个文本文档，后缀名可改为 *.module*，便于日后识别。在这个文档中，我们可以为每个模块的节点添加不同图片作为模块的纹理特征，以 2.2.5.1. 中出现的模块分区为例，其 *.module* 文件内容可以是：

```
/Users/Username/myModuleTextures/Texture01.png
/Users/Username/myModuleTextures/Texture02.png
/Users/Username/myModuleTextures/Texture03.png
```

最后将 *.node* 文件路径、 *ModuleTexture* 及 *.module* 文件路径导入函数

```
sunvs_net_viewer('myNodes.node', 'myEdges.edge', 'ModuleTexture', ‘myModuleColor.module’
```

即完成了模块的纹理绘制。

2.2.6. 正负连接的颜色，参数： *'LineColorPos'* 与 *'LineColorNeg'*

对于连接矩阵中的正连接与负连接，软件包默认使用了不同的配色（红色与蓝色）。也可以通过使用 *'LineColorPos'* 与 *'LineColorNeg'* 参数自行设置正负连接的 *RGB* 值。

2.2.7. 为节点添加 *Label*，参数： *'Label'*

若在 *.node* 文件的第六列中设置了 *Label*，那么只需要：

```
sunvs_net_viewer('myNodes.node', 'myEdges.edge', 'Label', 1);
```

即可在脑图上显示每个节点的 *Label*。

2.2.8. 综合示例

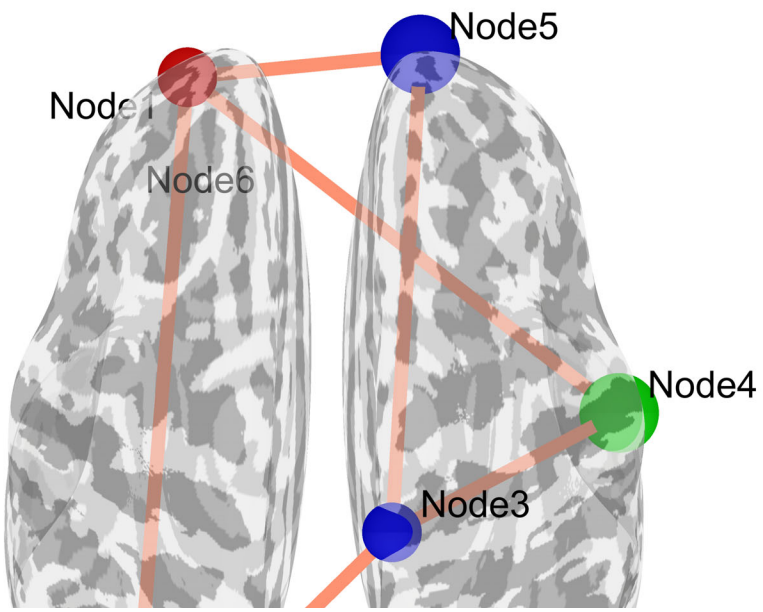
我们想要根据上文示例中的 *.node* 与 *.edge* 文件输出脑图，并希望：

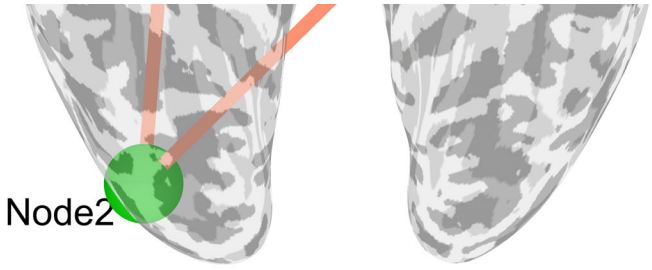
1. 使用膨胀的 *surface mesh*
2. 使用大脑沟回的 *underlay*
3. 使用自定义的模块颜色
4. 使用 *.node* 中配置的节点大小权重
5. 使用 *.edge* 中配置的连边粗细权重
6. 使用 0.5 的透明度
7. 输出 *.node* 中配置的节点 *Label*
8. 采用上方视角
9. 输出图片到当前文件夹

那么我们需要中命令行窗口输入：

```
sunvs_net_viewer('Content_EPAS.node', 'Content_EPAS.edge', 'usefsaverage', 'inflated',..
'useUnderlay', 'mc', 'ModuleColor', myModuleColor, 'NodeWeight', 1, 'EdgeWeight', 1,...
'TransParency', 0.5, 'Label', 1, 'view', 's','imgprint', 1);
```

工具包即会绘制出下图，并将其以 600 *dpi* 的分辨率保存到当前工作路径。





我们可以看到，图中的节点 *Node6* 并不清晰，这种情况下，我们可以考虑将透明度调高，或者将 *useUnderlay* 设置为 'none'。

3. 便捷工具

3.1. 生成脑区中心坐标 // *sunvs_gen_centroid_coor*

如前所述，我们若要绘制脑网络，首先需要生成 *.Node* 文件，*.Node* 文件中存储来每个脑网络节点的三维坐标。考虑到一个脑区是立体的（*Volume*）或平面的（*Surface*），内部包含大量的坐标，我们缺少一个代表性的坐标点（如先前文献给出的坐标点）来表征我们的脑区时，我们可以使用脑图谱中每个脑区的中心坐标来替代，该坐标由对脑区内所有体素（*Volume*）/顶点(*Vertices*)的坐标信息取算术平均数得到。对于我们的 *Surface* 作图，工具包内提供了 *sunvs_gen_centroid_coor* 函数来方便快捷地计算中心坐标，使用方法如下。

3.1.1. 图谱为 *.annot* 格式

当图谱为 *.annot* 格式时，我们需要准备两个文件：*.gii* 文件和 *.annot* 文件，其中

1. *.gii* 文件：*.gii* 文件的选择至关重要，该文件在这里用于提供 *Vertices* 的**坐标信息**。如前文（*sunvs_display* 的 *usefsaverage* 参数部分）所述，我们的 *.gii* 文件决定了脑的形状，也就决定了所有 *Vertices* 的坐标。由于我们计算坐标的目的是为了在 *Surface* 上绘制节点，因此**我们在这里选择的 *.gii* 文件必须和 *sunvs_net_viewer* 中使用的 *usefsaverage* 保持一致**
2. *.annot* 文件决定了分区方式，只要直接把所用图谱的 *.annot* 文件路径放进来即可。

```
[CenCoor] = sunvs_gen_centroid_coor(Path_filename, Path_annot);
% Path_filename 为 .gii 的文件路径
% Path_annot 为 .annot 的文件路径
```

3.1.2. 图谱为 *.gii* 格式

当图谱为 *.gii* 格式时，分区方式与坐标信息都由 *.gii* 文件提供，只需要输入

```
[CenCoor] = sunvs_gen_centroid_coor(Path_filename);
% Path_filename 为 .gii 的文件路径
```

3.2. 便捷生成 .gii 文件以在 *Surface* 底版上展示特定脑区 // *sunvs_ROI_annot2gii*

我们在使用 *sunvs_display* 在 *Surface* 底版上展示特定脑区时，需要使用到已经给特定脑区赋值的 *.gii* 文件，但是手动生成该文件还是较为麻烦。这里，本工具包提供了一个函数来便捷生成所需的 *.gii* 文件，先看函数：

```
sunvs_ROI_annot2gii(pathAnnot, indROIs, valueROIs)
```

其中，*pathAnnot* 是 *.annot* 图谱的文件路径，*indROIs* 是 *ROI* 的脑区编号，*valueROIs* 是要给对应脑区赋的值。举例来说，我们使用 *a2009s* 图谱（共包含 150 个脑区，左右脑各 75 个）对一套数据进行分区，数据分析发现，第 15、16 号脑区和第 96、150 号（即右半球的第 21、75 个脑区）脑区在组间存在显著效应，其 *p* 分别等于 0.001，0.004，0.017 和 0.023。那么，我们分别对左右半球进行操作：

```
sunvs_ROI_annot2gii('lh.a2009s_150_FreeSurfer_164k.annot', [15, 16]], [0.001, 0.004]]);
% 左脑


sunvs_ROI_annot2gii('rh.a2009s_150_FreeSurfer_164k.annot', [21, 75]], [0.017, 0.023]]);
% 右脑
```

该函数亦可使用 *'usefsaverage'* 参数，使用方法与 *sunvs_display* 相同。

4. tips

1. 在使用本工具包作图时，可以先使用灰度或 *copper* 等双色渐变的 *colormap*，在生成 *.tif* 文件后再进入 *Photoshop*，从背景中抠出脑图，使用脑图作为蒙版(Mask),再对蒙版添加渐变映射，这样可以较直观地操纵配色。
2. 待补充