

BrainNet Viewer User Manual

MATLAB Python Desktop and Docker Web Editions

Version 2.0.0-beta.1

Mingrui Xia

National Key Laboratory of Cognitive Neuroscience and Learning
Beijing Key Laboratory of Brain Imaging and Connectomics
IDG/McGovern Institute for Brain Research
Beijing Normal University, Beijing, China

Contents

This manual explains how to load brain visualization data, configure surfaces, nodes, edges, volumes and ROIs, inspect the result, save Scenes and export figures. It also covers mesh tools, camera animation, command-line rendering and troubleshooting. Shared scientific workflows are described once, with separate MATLAB, Python Desktop and Docker/Web deployment instructions. Check coordinates and statistical meaning before interpreting a figure. Screenshots are MATLAB illustrations unless labelled otherwise. Numeric control tables describe MATLAB defaults unless an edition note states otherwise; use the shared tutorials for Python and Web differences.

• 1 Installation and launch.....	3
• 2 Workspace and first visualization.....	5
• 3 Data formats and coordinates.....	6
• 4 Scenes and presets.....	10
• 5 Layout and camera controls.....	12
• 6 Global appearance and rendering.....	15
• 7 Surface appearance and maps.....	18
• 8 Nodes.....	22
• 9 Edges.....	24
• 10 Volume mapping and statistics.....	27
• 11 ROI drawing.....	31
• 12 Colormaps and presets.....	33
• 13 Figure export.....	35
• 14 Camera animation.....	37
• 15 Surface formats and mesh tools.....	39
• 16 Command line rendering and provenance.....	41
• 17 Migration and troubleshooting.....	44
• Appendix References and acknowledgments.....	47
• Appendix Shared tutorials.....	48

1 Installation and launch

BrainNet Viewer visualizes brain surfaces, networks, mapped volumes and ROIs in MATLAB Desktop, Python Desktop and Docker/Web. This manual explains the shared workflow and the deployment and behavior differences needed to use each edition.

MATLAB installation

MATLAB Desktop requires an installed, activated MATLAB and a graphics-capable session. Windows R2024b/R2025a and macOS R2024a are supported starting environments; consult the installation guide for version-specific requirements. Independent NIfTI/GIfTI readers do not require SPM. Optional numerical GPU acceleration can require MATLAB toolboxes and compatible hardware. MATLAB Runtime-only installation and a standalone executable are not supplied.

Set MATLAB Current Folder to the installed software root, the directory containing the top-level BrainNetViewer.m, then run:

```
app = BrainNetViewer;
```

The launcher adds `src/matlab/BrainNetViewer` itself. From elsewhere, add only the installed software root to MATLAB's path first. Do not use `addpath(genpath(...))`, mix different software versions, or assume that a different installed BrainNetViewer is the intended installation. In MATLAB, which `BrainNetViewer -all` helps diagnose a path collision. No installer is supplied by this chapter.

Python installation

Obtain the supplied wheel and verify its version and identity against the accompanying installation notes. Create a new virtual environment rather than copying one from another computer. Keep the package's dependency constraints unchanged. The following Windows example uses PowerShell; replace the wheel path with your actual file:

```
py -3.10 -m venv .venv-bnv

$bnvPython = (Resolve-Path .venv-bnv/Scripts/python.exe).Path

& $bnvPython -m pip install --upgrade pip

$bnvWheel = (Resolve-Path brainnet_viewer-2.0.0b1-py3-none-any.whl).Path

$bnvWheelUrl = & $bnvPython -c "from pathlib import Path; import sys;
print(Path(sys.argv[1]).resolve().as_uri())" $bnvWheel

& $bnvPython -m pip install "brainnet-viewer[desktop] @ $bnvWheelUrl"

& $bnvPython -m pip check

& $bnvPython -m brainnet_viewer
```

On macOS, select the Python version appropriate for the installation notes and run:

```
python3 -m venv .venv-bnv

source .venv-bnv/bin/activate

python -m pip install --upgrade pip

BNV_WHEEL="/path/to/brainnet_viewer-2.0.0b1-py3-none-any.whl"

BNV_WHEEL_URL="$(python -c 'from pathlib import Path; import sys;
print(Path(sys.argv[1]).resolve().as_uri())' "$BNV_WHEEL")"
```

```
python -m pip install "brainnet-viewer[desktop] @ $BNV_WHEEL_URL"

python -m pip check

brainnet-viewer
```

Use the supplied `brainnet_viewer-2.0.0b1` wheel and install its desktop extra in a fresh environment. Dependency downloads require network access unless supplied separately. If installation fails, retain the complete error rather than changing fixed version constraints; the installation guide includes proxy troubleshooting. Launch with the environment interpreter or graphical launcher.

Docker Web installation

Install and start Docker Desktop on Windows/macOS, or Docker Engine with Compose on Linux. Windows uses the WSL 2 backend and Linux containers. No MATLAB licence or local Python environment is required.

Download and extract `BrainNet-Viewer-Web-2.0.0-beta.1.zip`. Open a terminal in the folder containing `compose.web.release.yml`. Run these commands to download the image and start the service in the background:

```
docker compose -f compose.web.release.yml pull

docker compose -f compose.web.release.yml up -d
```

Open <http://127.0.0.1:1031/index.html>. Keep Docker running; the terminal may be closed. Use `docker compose -f compose.web.release.yml logs --tail 80` to diagnose startup errors. The supplied configuration is local-only, CPU-based and has one shared session.

Add Files uploads local inputs; bundled resources are also selectable. Export Figure downloads the image and provenance JSON separately. Save Scene downloads a portable ZIP, which Open Scene and Apply restore. Save the Scene before stopping or updating Docker; unsaved working data is temporary. See [docs/web/en/INSTALLATION.md](#) for platform instructions, stopping, updates and troubleshooting.

Find the resource library

The MATLAB distribution supplies `Data/SurfTemplate`, `Data/ExampleFiles` and `Data/Tutorials` inside its software tree. `AtlasNetworks` holds matched node/edge files. AAL90 auxiliary inputs are in `resources/atlasses/AAL90` from the archive root; Python and Docker provide them under `atlasses/AAL90` in the package resource library. Web exposes bundled inputs in library selectors. To locate Python resources:

```
python -c "from pathlib import Path; import brainnet_viewer;
print(Path(brainnet_viewer.__file__).parent / 'resources')"
```

File pickers open the relevant bundled library where configured. Copy data into a separate working folder before changing it; do not edit installed resource files. The resource notices provide source credits and citations. Software installation and data redistribution are separate questions.

2 Workspace and first visualization

What should appear

The English main window opens with its startup image. Use Data Sources -> Visualization Settings -> Apply -> interact/export. Select Add Files or a role-specific input, then Validate and Load. The shared Apply button commits settings and renders directly; Render Scene explicitly redraws. Web uploads selected local files to its container or uses bundled library inputs.

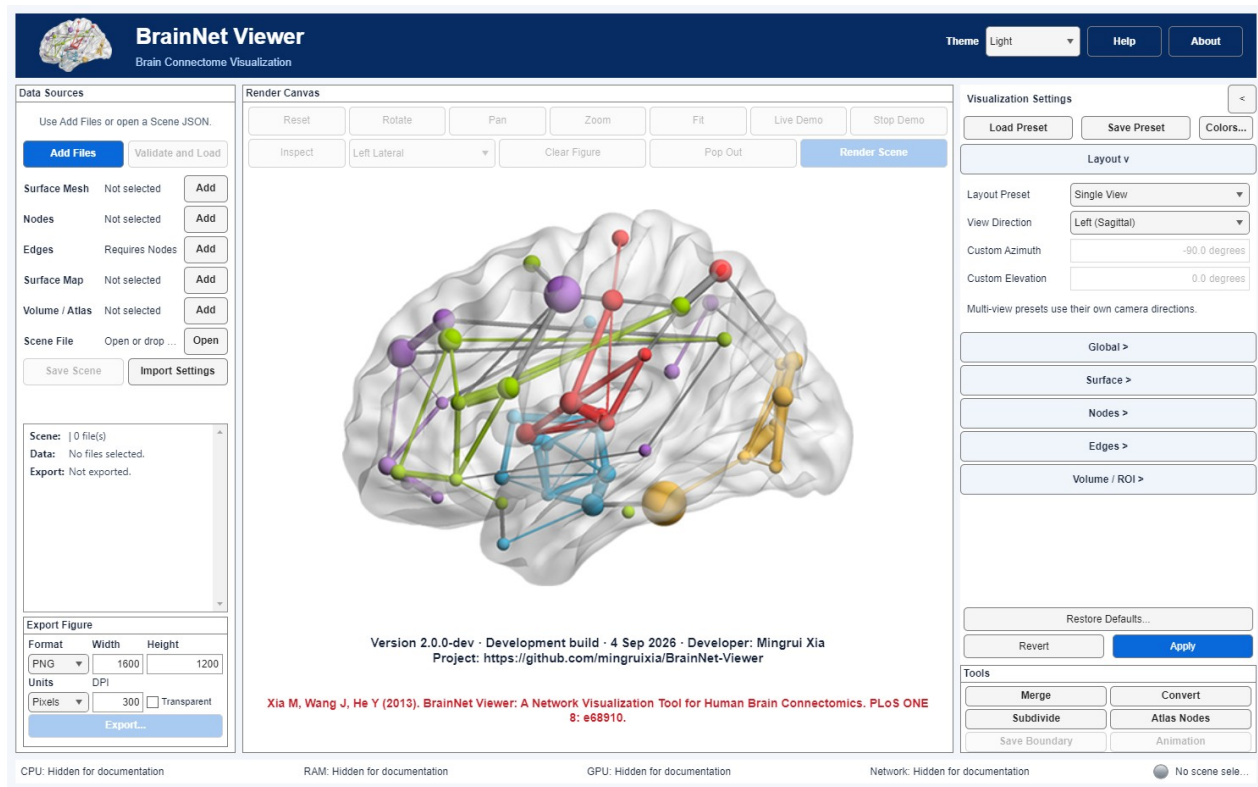


Figure 1. The English startup window identifies the data, render and settings areas before files are loaded. MATLAB illustration.

Start with the real ICBM152 surface exercise, not an arbitrary template found in a data directory. For file roles, coordinate cautions and data-root behavior, continue to Chapter 3. For saving an analysis, use Chapter 4.

First surface workflow

1. Choose Surface Mesh Add and select Data/SurfTemplate/BrainMesh_ICBM152.nv.
2. Validate and Load; review the file path and counts.
3. Choose Layout and an opaque gray Uniform surface.
4. Apply commits and renders directly.
5. Interact with the intended view and export only after reviewing the result.

Collapse Visualization Settings to enlarge the drawing area. Python opens one settings section at a time without forcing it to the top. Rotate, Pan and Zoom target the selected subplot. Light/Dark changes the interface, not the scientific canvas background.

Before sharing an image or package

Check the installed version and resource credits. Figure provenance may include private file paths or titles; review it before sharing. Save a Scene before stopping the Web container because uploads and unsaved session data are temporary.

3 Data formats and coordinates

Choose a file's role, not just its extension. In Data Sources, Add Files can classify candidates; use the individual slots when classification is ambiguous. Validate and Load reads and checks the selected combination before a figure is interpreted. In MATLAB, the default desktop data root is src/matlab/BrainNetViewer, not the Scene JSON's folder.

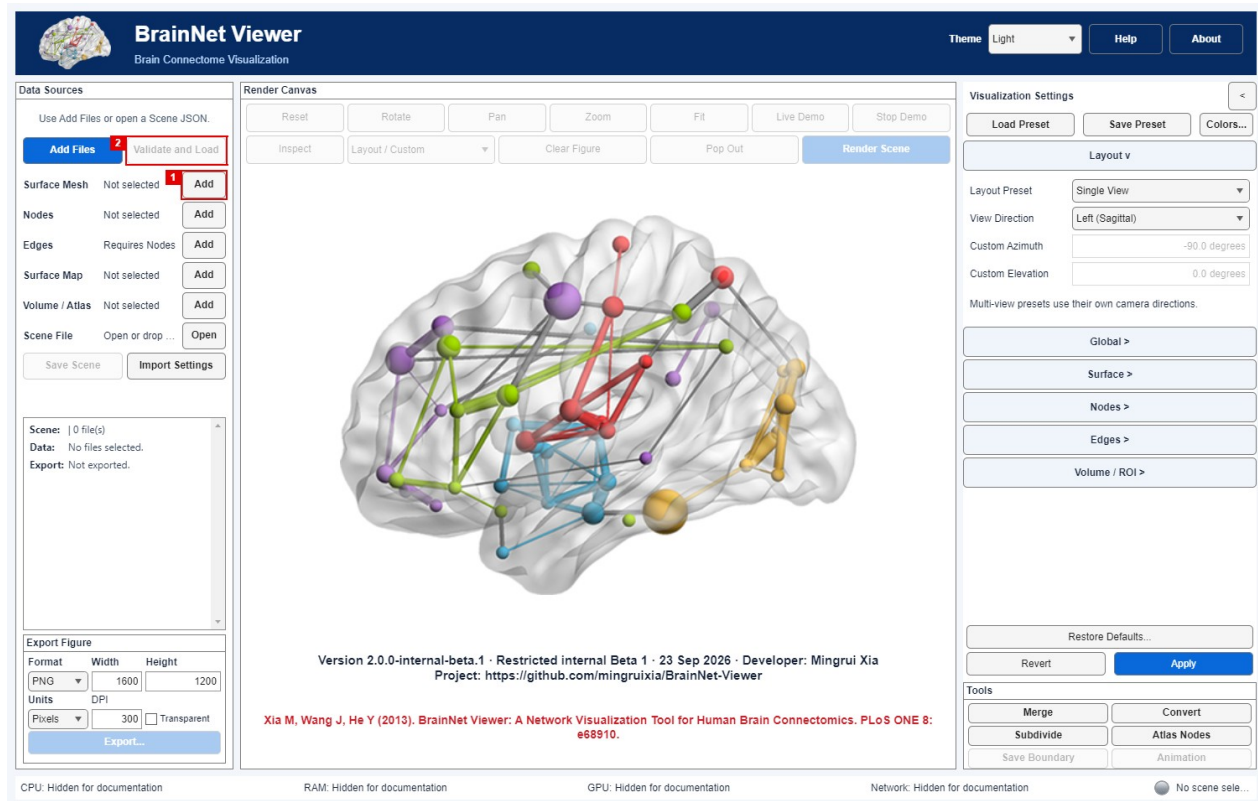


Figure 2. Step 1 selects Surface Mesh Add; step 2 validates and loads the selected inputs. MATLAB illustration.

Role in Data Sources	Typical input	Key correspondence rule
Surface Mesh	.nv, .gii/.gii.gz, .pial, .mesh, .g, MNI/BIC .obj, .mz3/.mz3.gz	Geometry supplies XYZ and faces; a readable surface alone does not define MNI/subject space. See Chapter 15.
Nodes	BrainNet .node	Each row is a marker with coordinates and display fields. Preserve row order when pairing with an edge matrix; illustrative columns are not automatically measured quantities. See Chapter 8.
Edges	BrainNet .edge	Matrix rows/columns must correspond to node rows; directedness and threshold semantics depend on settings. See Chapter 9.
Surface Map	Original-order per-vertex scalar text	One value per mesh vertex in the same vertex order . Equal length does not prove atlas correspondence; see Chapter 7.
Volume	NIfTI-1/2 .nii/.nii.gz or supported paired .hdr/.img	Header/affine and statistic metadata matter. A readable volume need not be registered to a separate surface; see Chapter 10 and Chapter 11.
Scene File	Modern .scene.json	An analysis reference, not a mesh or volume; its files are resolved under the configured data root. See Chapter 4.

Choose a useful input combination

You do not need to load every role for every figure. A surface alone shows geometry; nodes alone show spatial markers; adding an edge matrix to nodes shows a network without an anatomical mesh. Combining a surface with nodes or with nodes and edges places those elements over anatomy. A scalar surface map or a volume-derived surface map requires a matching surface. The resulting map can also be shown with nodes and edges. To draw atlas or statistical ROIs, load a suitable volume and choose the ROI Volume Type; a surface makes their anatomical context easier to judge. An edge file without its matching nodes cannot establish endpoint locations.

The nine examples below illustrate the different input combinations. For an analysis, start with the fewest data roles needed and add one file at a time. After each addition, inspect the Scene Summary and check that the layers are spatially consistent. Colors and camera positions can be adjusted independently of the input data.

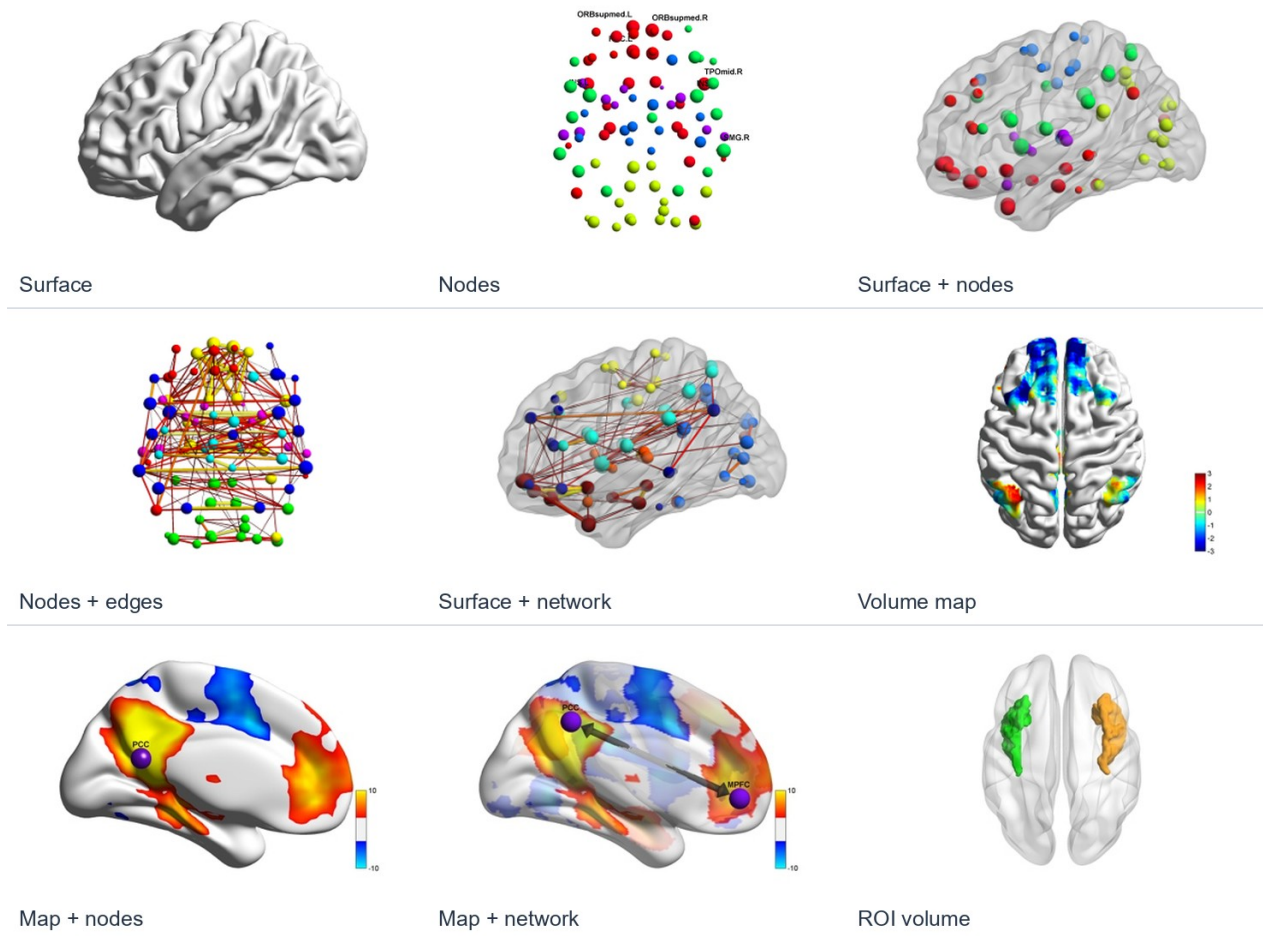


Figure 3. Data-role combinations for surface, node, edge and volume visualization. MATLAB illustration.

Prepare text-based inputs

The BrainNet .nv format begins with an integer vertex count, then that many XYZ coordinate triplets, an integer triangle count, and that many triplets of **one-based** vertex indices. The number of coordinates and faces must agree with the declared counts. A # comment may document provenance, but it does not change coordinates or supply a registration transform. Mesh utilities in Chapter 15 can create a new .nv from supported formats after a review step.

A .node file has one node per line: X Y Z color_or_module size label. The first five fields must be finite numbers; the final label may contain spaces. A label consisting of - means no visible label. Column 4 may encode a continuous color value or a discrete module ID, depending on the Nodes color setting; column 5 controls mapped or raw radius. The following three rows are a **format example only**, not atlas coordinates:

```
0 0 0 1 2 Region_A
10 0 0 2 3 Region_B
0 10 0 2 1 Region_C
```

The matching .edge file is a whitespace-delimited 3×3 numeric matrix in precisely the same node order. For example, the first row below describes values from Region_A to Regions A, B and C:

```
0 0.4 0
0.4 0 -0.2
0 -0.2 0
```

Zero denotes an absent edge. In undirected mode, only eligible upper-triangle entries are drawn; in directed mode, ordered off-diagonal entries can be drawn separately. A file may contain # comments. Do not treat this illustrative matrix as a real functional-connectivity estimate. For a per-vertex text Surface Map, provide exactly one scalar (finite values; Python also accepts NaN as missing) per vertex in the mesh's original vertex order; a matching count without a trustworthy vertex correspondence is insufficient.

Atlas families are not interchangeable node orders or coordinate spaces. Check the package's file inventory, attribution and each sample's node count before pairing any .node with an .edge. Do not infer that a numerical atlas ID is a statistical effect size.

The independent modern NIfTI/GIfTI readers do **not** require SPM. Plain Analyze 7.5 .hdr/.img lacking NIfTI magic is not accepted. GIfTI external binary arrays are supported within the reader's validation rules; keep companions together when moving data. For surface formats, .obj means **MNI/BIC**, not Wavefront; .mesh means the supported BrainVISA encoding, not arbitrary Medit data. FreeSurfer .pial geometry is kept in stored surface coordinates; it is not automatically scanner-RAS or MNI. BrainVISA geometry is kept as stored rather than applying the old implicit XYZ inversion. GIfTI transforms/metadata may be recorded without applying registration. Format support does not imply that embedded MZ3 scalars/RGBA are automatically used for display. Consult the detailed matrix and Chapter 15 for explicit rejections.

Outside-root files and portability

MATLAB: For a file outside the App's configured root, the desktop asks Stage and Add or Cancel. Approval makes a verified copy under user-data/imports/<hash>/ and stages supported companions, leaving the original untouched. It does not transfer rights to publish the copy. A programmatic selectInputPath requires explicit external-import approval; the CLI does not silently stage arbitrary absolute paths. A portable Scene keeps relative references and hashes, but not the bytes of the input files. Preserve their relative directory structure beneath the destination data root. See Chapter 4.

Python data and Scene portability

Python accepts selected external files and saves a Scene with companion assets beside its JSON. Preserve this asset structure when moving the Scene. The Python CLI defaults to the JSON directory as its data root; MATLAB resolves references against its configured root. Same file count or a matching hash does not establish anatomical correspondence.

AtlasNetworks contains AAL90 and fourteen matched GroupFC pairs: AAL116, AAL3v2, Brodmann82, Brainnetome246, Dos160, Fair34, Glasser360, HOA112, Power264, Shen278 and Schaefer100/200/400/1000Yeo7. GroupFC node sizes encode absolute connection strength; colors encode Yeo modules for Schaefer or data-derived communities otherwise. Edges are group-level resting-state examples from healthy adults, not normative estimates. Match filenames and row order. AAL90 retains its supplied binary/weighted examples; its IDs differ from AAL3v2.

Scientific alignment check

Before combining inputs, identify species/template or subject, XYZ convention and handedness, physical units, surface vertex order, volume affine, and atlas/statistic meaning. Volume mapping may use the NIfTI affine to sample voxels, but **does not register two datasets**. Inspect assumes compatible MNI-like coordinates for its AAL/Brodmann lookup and does not perform an affine registration. A file hash establishes identity, not spatial alignment, license clearance or statistical validity. If any of these are unknown, stop anatomical interpretation and document the uncertainty. The real AAL vertex-map lesson and T-map lesson show exact-file examples with their limits.

4 Scenes and presets

Choose the right file

Need	Use	What it contains
Reopen a complete visualization	Scene JSON	Input references and hashes, applied visualization settings, and supported camera/export session state; not the data bytes
Reuse a look on different data	Visualization Preset	Display settings only; no input files, camera or export destination
Recover a trusted old project	Legacy configuration import	A migration starting point that requires field-by-field review, not a lossless modern Scene

Save and reopen a Scene

MATLAB: Load and validate the inputs, configure the scene, and select Apply to commit and render. Save the Scene JSON from the main window. The Scene records paths relative to its configured data root where possible and verifies referenced inputs by hash. It does not embed mesh, node, edge or volume data. Keep the inputs and their compatible directory structure with the JSON when moving a project. If files are outside the configured root, follow the explicit import/staging workflow in Chapter 1; selecting an external file does not silently grant redistribution rights.

When reopening, review resolved paths, hashes, file roles, spatial alignment and the rendered result. A missing or changed input is a reason to investigate, not to bypass validation. Scene replay can restore supported camera and export preferences, but an old Scene may not express every newer option. Scene Workflows documents the detailed schema, path and command-line behavior.

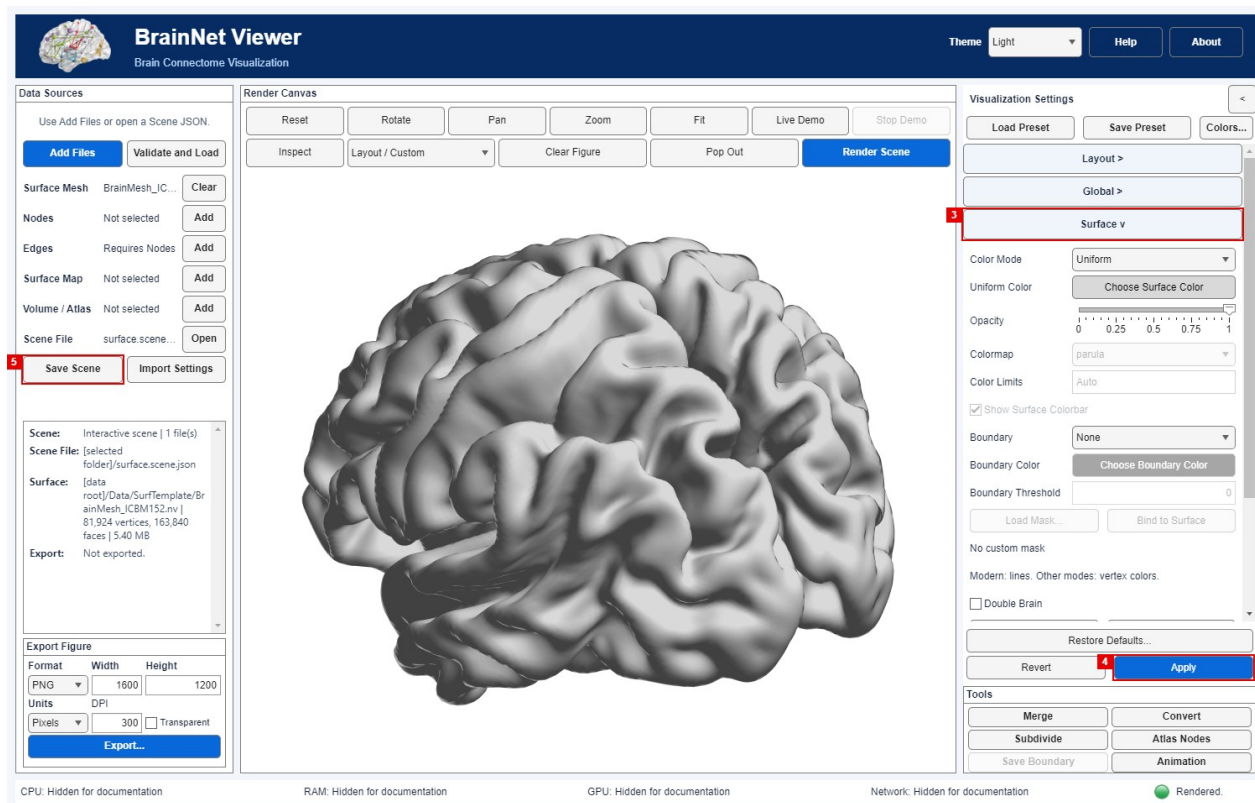


Figure 4. A saved Scene reopened with an opaque gray surface. Step 3 opens Surface settings, step 4 applies and renders, and step 5 saves a Scene. MATLAB illustration.

Apply a visualization preset

MATLAB: Choose a preset after loading the intended data. Preset load replaces the visualization settings sections; it does not bring in files or preserve pending edits and camera state. Review the resulting controls, then use Render Scene for a visible redraw. Save a new Scene JSON if the input-plus-display combination should be reproducible. See Visualization Presets for the supported fields and reset behavior.

Python Scenes and presets

Python saves a Scene JSON with copied companion assets beside it; keep both. Docker/Web Save Scene downloads a portable ZIP containing the Scene and its assets. Reopen the ZIP with Open Scene and Apply. A preset contains appearance settings, not data bytes or the session camera. Python/Web Save Preset can capture pending panel edits; Apply first to save the intended rendered configuration.

Cross-edition Scene/preset replay requires review of schema, input paths and edition-specific metadata; matching filenames alone do not guarantee interchange. Trusted EC MAT import is the documented MATLAB conversion route, not a general Python MAT importer.

Import a legacy configuration

Use a legacy MAT configuration only when its source is trusted. The migration translates what it can, reports fields requiring review, and may change the rendered appearance. Inspect file roles, paths, layout, thresholds, colors and export options; compare a representative image before adopting the result. Review the translated settings before use. Migration and troubleshooting lists current limits.

Batch and provenance

For repeatable non-interactive rendering, use `BrainNetRender` with a reviewed Scene. The Scene JSON, input checks and export provenance help reconstruct a figure but do not replace recording data origin, atlas version, coordinate convention, software build and any manual decisions. Chapter 16 covers batch use. Before sharing any Scene, inspect paths, metadata and sidecars for private local information; sharing the JSON never authorizes sharing its referenced data.

5 Layout and camera controls

Use **Visualization Settings** → **Layout** to choose a view arrangement and initial camera direction. The shared **Apply** commits changes and renders; **Revert** discards pending edits. Saved Scenes and presets retain applied layout settings, but presets do not carry the current camera.



Figure 5. Overview with eight views for a surface that can be split into hemispheres. MATLAB illustration.

Control	New-settings default	Effect and when to use it
Layout Preset	Single View	Five choices: Single View; Overview (6 / 8 Views); Lateral / Medial; Lateral / Medial / Ventral; Lateral / Medial / Dorsal. Multi-view presets set their own pane directions and use a shared scale; they are not independent copies of the single-view direction control.
View Direction	Left (Sagittal)	Single-view directions: Left, Superior, Anterior, Custom, Right, Inferior, Posterior, Isometric. In multi-view, the preset's own directions take precedence. A named view is a camera orientation, not proof of anatomical L/R.
Custom Azimuth	no explicit value	Enabled for Custom direction; numeric degrees, -360 to 360 . Save and replay the applied value in Scene/preset.
Custom Elevation	no explicit value	Enabled for Custom direction; numeric degrees, -90 to 90 .

Read a multi-view figure

Single View is best for adjusting one camera or inspecting one selected point. The lateral/medial arrangement places the outside and inside of each separable hemisphere in four panes. The ventral variant adds inferior views; the dorsal variant adds a superior overview. **Overview (6 / 8 Views)** provides a broader orientation survey: a safely separable two-hemisphere mesh can use eight panes, while a single or inseparable surface uses six complete-data views. Choosing a multi-view preset is not the same as loading multiple independent datasets: every pane represents the same Scene from another camera.

The layout gallery illustrates the available viewing arrangements. Verify the dataset coordinate convention before interpreting orientation labels. At the intended export size, check that labels and thin edges remain legible in the smaller panes.

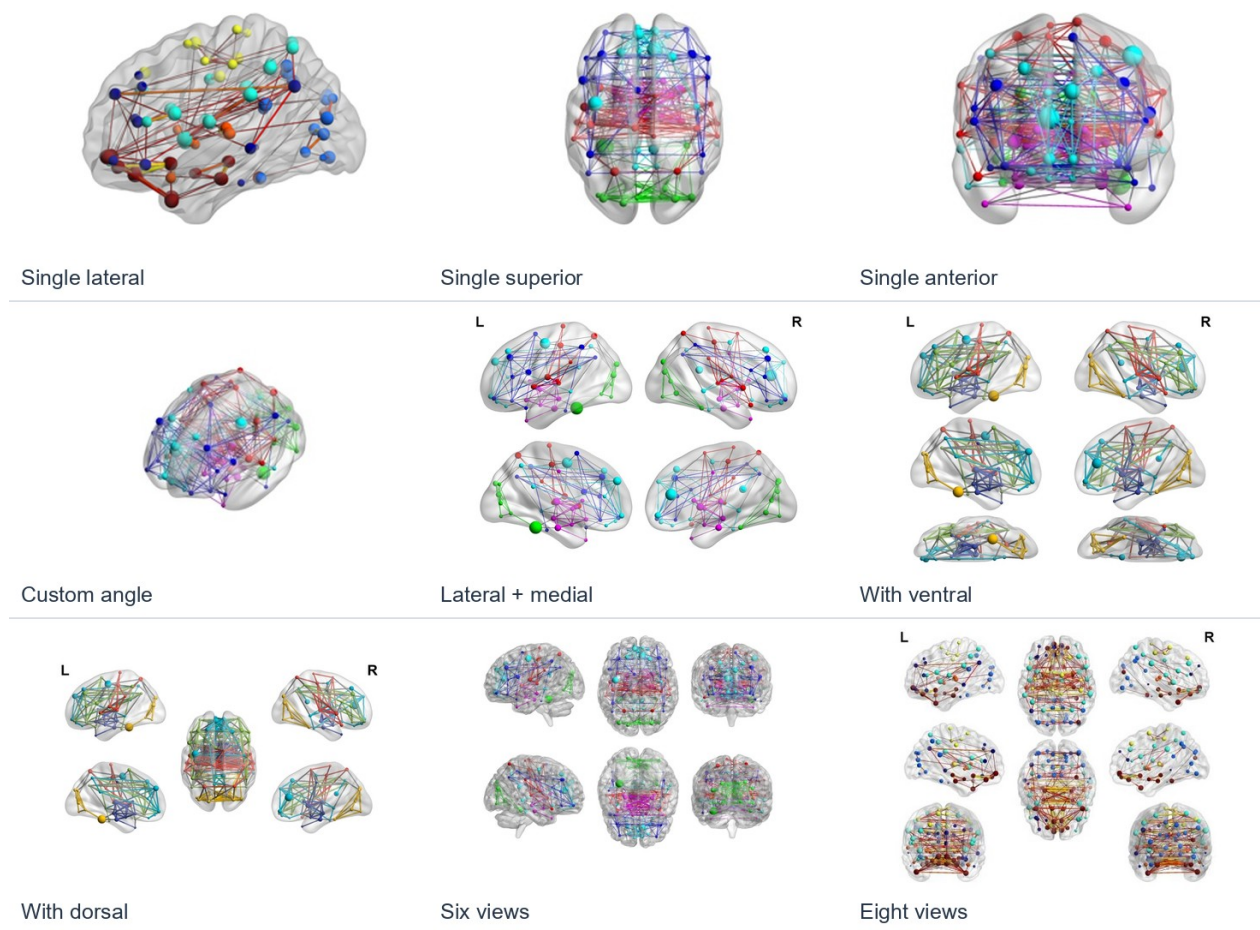


Figure 6. Viewing arrangements and relative positions of the brain panes. MATLAB illustration.

Overview (6 / 8 Views) uses eight split views only when a surface can be safely partitioned into nonempty, vertex-disjoint X-side groups. Otherwise it shows six complete-data views and does not invent a missing hemisphere. This conservative rule does not segment arbitrary anatomy or verify registration. A saved eight-view camera is reset with a notice if replay resolves to six views, and conversely; see Layout 2.

After rendering, use the canvas controls to Rotate, Pan, Zoom, Fit, or Reset. Fit recenters and frames visible geometry without changing input coordinates; Reset restores the selected layout's initial camera. Resizing preserves the current camera rather than silently fitting it. MATLAB Pop Out opens an independent inspection copy; later movement there does not replace the main canvas camera. Save Scene records a valid modern main-canvas camera; Save Preset does not. See camera/replay details and window sizing.

Double Brain (described in Chapter 7) is a separate single-view display option, not a sixth layout. It is restricted to Left, Superior or Anterior direction with a surface and no scalar-map/volume input. The pane badges governed by **Show Orientation Labels** are described in Chapter 6; they identify coordinate halves unless an input-bound anatomical declaration has been explicitly verified.

Multi-view readability depends on mesh shape, labels and display scaling. Check the final figure at its intended export size.

Python view behavior

Python Pop Out moves the original canvas into a separate window and returns it when that window closes; it is not an independent inspection copy. Multi-view interaction affects only the selected brain. View Direction is disabled for multi-view layouts. Reset restores layout cameras; Fit frames geometry without changing input coordinates.

6 Global appearance and rendering

Use **Visualization Settings** → **Global** for the canvas background, shading, lighting and rendering detail. **Apply** commits and renders. A Scene or preset may contain different explicit values; the table describes new settings, not a promise to overwrite loaded configurations.

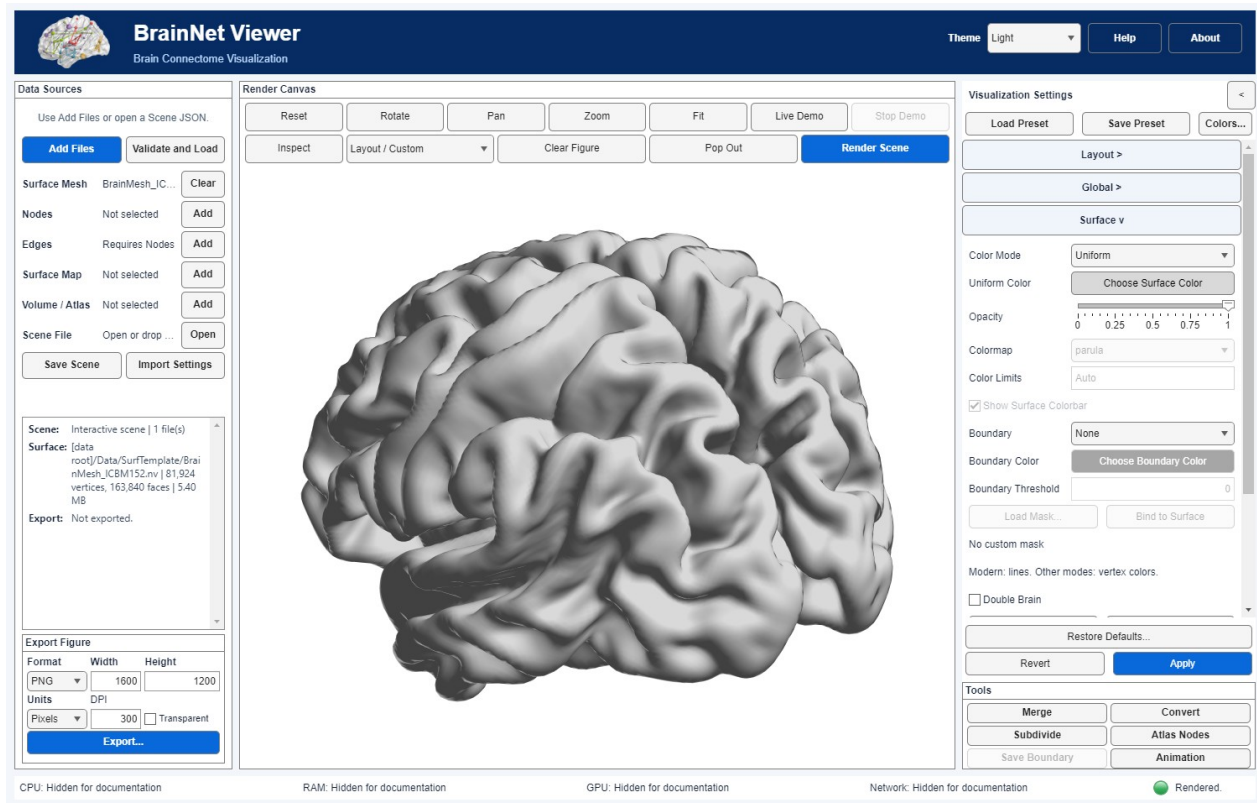


Figure 7. An opaque gray ICBM152 surface with Surface settings visible. MATLAB illustration.

Control	New-settings default	Meaning / constraints
Background Color → Choose Color	white [1 1 1]	Sets the scientific canvas background, not the Light/Dark interface theme. App export set to current follows the rendered canvas color.
Surface Shading	Interpolated	Flat, Faceted or Interpolated surface face/vertex shading. Faceted draws black mesh outlines and may cost more on dense meshes. This is display shading, not mesh smoothing.
Lighting Model	Phong	None, Flat, Gouraud or Phong lighting for shaded geometry. Appearance depends on material, view and graphics backend.
Light Direction	Right	Headlight, Right or Left. Camera/light behavior should be inspected after manual rotation; the choice does not change data coordinates.
Surface Material	Dull	Dull, Shiny, Metal or Custom. Custom enables the three strength fields below; presets follow MATLAB's material behavior.
Ambient / Diffuse / Specular Strength	[0.5 0.5 0.5] stored	Each accepts 0–1 and is editable for Custom material. Values affect lighting response, not underlying scalar data.

Control	New-settings default	Meaning / constraints
Render Detail	Adaptive (modern)	Adaptive uses load-aware renderer policy. Legacy Low/Moderate/High use node sphere detail 10/50/100 and edge-cylinder sides 5/10/20; High can be costly. The surface topology is unchanged.
Show Orientation Labels	off	Multi-view pane badges are $X \leq 0$ / $X > 0$ unless the Scene has an explicit, input-bound anatomical-X declaration; only then can they say L/R. They label panes, not the screen's left/right edge. Single View and six complete-data Overview have no half-space badges.

Compare appearance settings

Material changes the balance of ambient, diffuse and specular reflections. **Dull** gives a relatively subdued shape; **Shiny** and **Metal** emphasize highlights. A Custom material lets you adjust the three strengths rather than replacing the input mesh. Surface Shading controls whether neighboring vertex colors are interpolated and whether triangle boundaries are visible; use Faceted to inspect mesh edges, not to infer an anatomical boundary. The Lighting Model controls how illumination is calculated, while Light Direction chooses where it comes from. These are separate decisions: an interpolated color map can still look flat under weak lighting, and changing illumination never changes a statistic in the source map.

The appearance gallery compares material, shading and light direction on one geometry. Set the palette and statistical range first, choose lighting and material, then check legibility at the intended export size. Use Render Detail conservatively on dense networks: higher sphere and cylinder tessellation increases geometry without adding measurements or changing surface topology.

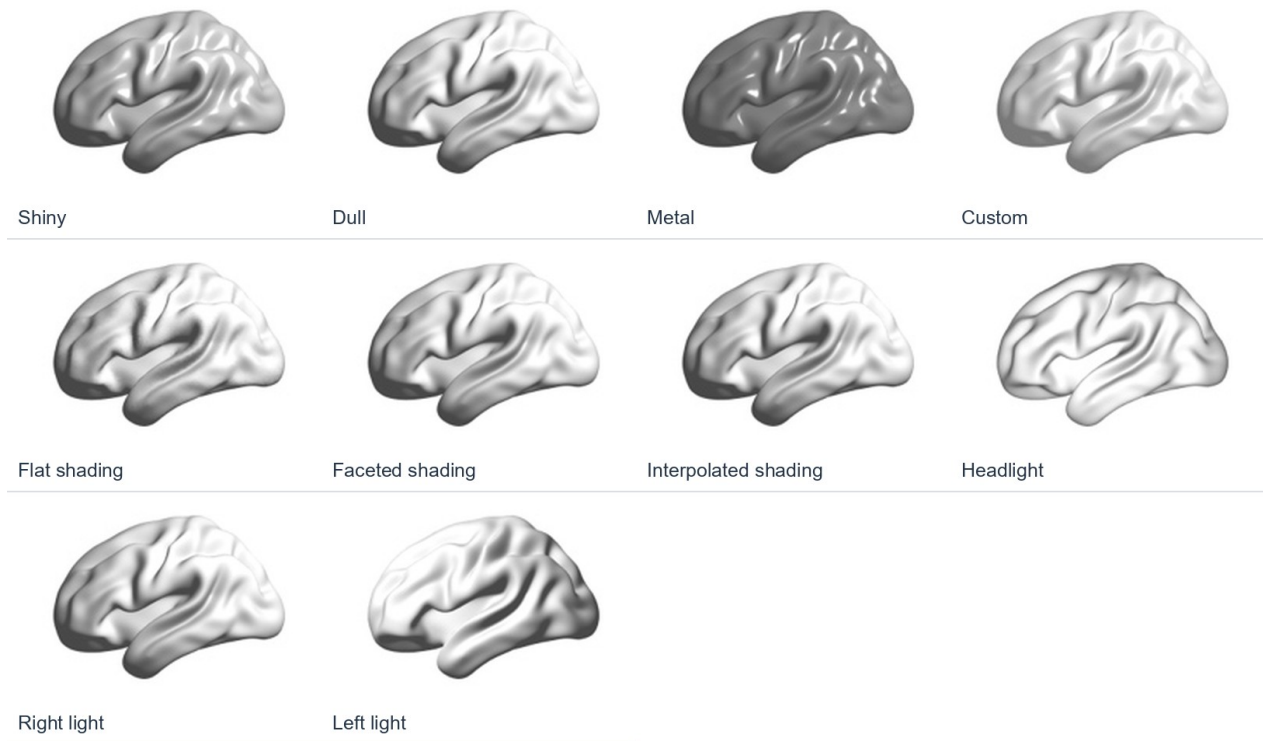


Figure 8. Surface material, shading and light-direction examples. MATLAB illustration.

Theme in the top bar changes UI colors only. For export, **current** preserves the actual scientific canvas background; transparent raster requests may be rejected rather than silently flattened. See backgrounds. Lighting/material specifics and the exact legacy geometry table are in Material Settings and Render Detail.

MATLAB orientation labels: The App does not infer anatomy from a filename or species. To show anatomical L/R, first verify coordinate units, shared registration, the $X=0$ midline and positive-X direction, then attach the declaration to hashed Scene inputs using the documented Scene/API workflow. Merely ticking Show Orientation Labels without that declaration shows coordinate halves. Check label contrast on the display and in the exported figure.

Restore Defaults resets display settings while keeping the current canvas visible. Click Apply or Render Scene to redraw; export is paused until the retained preview is replaced. It does not return to the welcome page.

Edition specific appearance

Python uses the VTK graphics backend. Material and lighting options express similar display intentions but need not produce pixel-identical output to MATLAB. Python pane labels must be interpreted using the verified input coordinates; the MATLAB input-bound anatomical declaration workflow is not a general registration tool. Numerical preparation in Python uses CPU paths; do not infer MATLAB GPU support from a GPU model in the status bar.

7 Surface appearance and maps

Use **Visualization Settings** → **Surface** to set the surface color, opacity, boundaries and vertex colors. **Apply** commits and renders. A loaded Scene or preset keeps its explicit values unless you edit them.

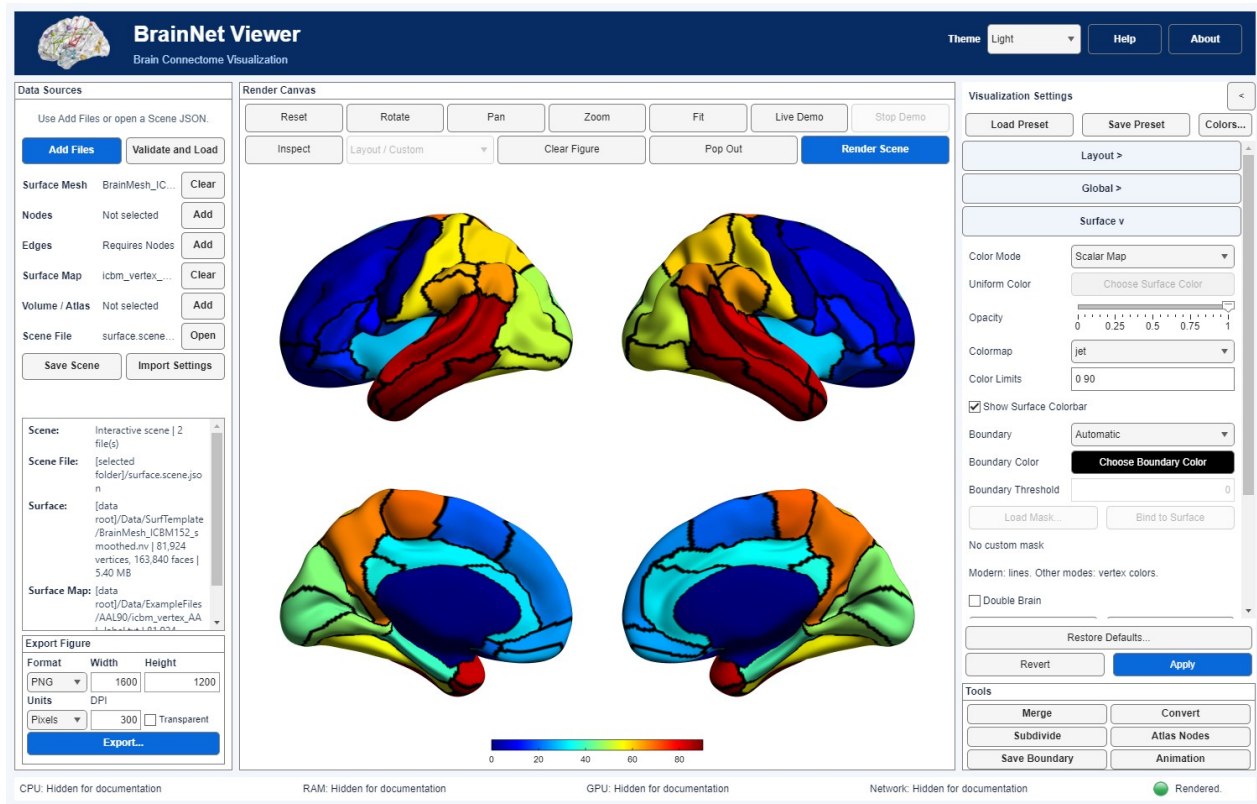


Figure 9. AAL vertex labels with automatic boundaries in four lateral and medial views. MATLAB illustration.

Control	New-settings default	Meaning / enabling rule
Color Mode	Uniform	Uniform uses one RGB surface color; Scalar Map colors vertices from a loaded surface map or supported volume-to-surface mapping; Vertex RGB uses an explicitly bound original-order RGB table. Choose the mode that matches the data, not merely a desired palette.
Uniform Color	[0.95 0.95 0.95]	Active only for Uniform; Choose Surface Color changes the single color.
Opacity	Settings default 0.3	Slider range 0 (transparent) to 1 (opaque). A new surface-only or surface-plus-map/volume Scene instead starts at 1 ; explicit Scene/preset values are preserved. This display opacity does not change input values.
Colormap	Parula	Active for a loaded scalar surface map. Choose a named palette or Custom (Colors...) . Volume-to-surface coloring uses the Volume panel's separate colormap; Vertex RGB bypasses scalar palettes.
Color Limits	Auto	For Scalar Map, enter Auto or two numeric endpoints (for example -2 2). Fixed limits control color scaling, not vertex-value filtering. Recheck them when switching datasets.

Control	New-settings default	Meaning / enabling rule
Show Surface Colorbar	on	Shows one continuous colorbar when a scalar surface map or volume-to-surface map is displayed; Uniform and Vertex RGB have no scalar bar. Existing Scenes/presets retain an explicit off setting. Atlas ROI ID legends are separate categorical keys.
Boundary	None	Automatic draws class-transition lines; Threshold marks vertices at a strict value threshold; Custom Mask uses an original-order binary mask. Boundary operations need the relevant map except Custom Mask.
Boundary Color	black [0 0 0]	Active when a boundary mode is selected; changes display color only.
Boundary Threshold	0	Numeric field used only by Threshold; its classification is value > threshold, not >=.
Load Mask... / Bind to Surface	no mask	Available for Custom Mask. Import or bind a binary table only after verifying original vertex order; a matching count alone cannot prove anatomical correspondence. The status line reports binding state.
Double Brain	off	Displays a second copy of one surface in Single View, Left/Superior/Anterior only; optional node/edge data follow explicit row ordering. No surface-map/volume input. This is not arbitrary two-subject composition.
Load RGB... / Bind RGB	no RGB table	Text is N×3 finite RGB in [0,1] for one brain, or 2N×3 for Double Brain, in original surface vertex order. Binding is required; it does not independently verify initial correspondence. The status line reports binding state.

Double Brain and boundary examples

Double Brain copies **one** surface for a side-by-side network comparison. If a node file contains N rows, rows 1 through $\text{floor}(N/2)$ belong to the first display and the remaining rows to the second. For example, a 180-row file may place 90 matched regions in each copy. Its edge file is then 180×180: the upper-left and lower-right 90×90 blocks describe within-copy connections, and the two off-diagonal blocks describe connections between copies. The program does not discover which rows correspond to the same regions or verify a longitudinal, group or subject interpretation. Prepare and review the ordering before turning on Double Brain; an ordinary 90×90 edge matrix is not the paired network for a 180-row node file.

The boundary gallery compares Double Brain, categorical atlas outlines and continuous mapped color. A region outline is a display boundary, not a new ROI volume. Continuous colors require a scale and must not be interpreted as category IDs. Automatic, Threshold and Custom Mask boundaries follow the rules above. Recheck visibility when changing surface opacity, map range or export resolution.

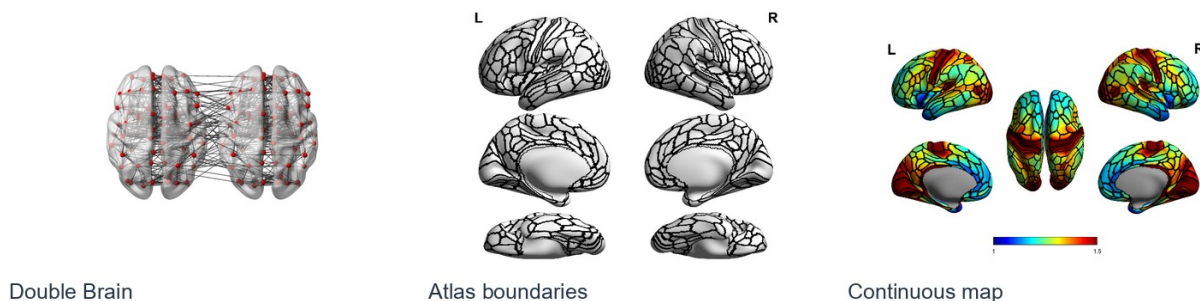


Figure 10. Double Brain, atlas boundaries and continuous surface coloring. MATLAB illustration.

Automatic in the current panel is the modern line-style boundary option; an older Legacy Automatic setting can be read for compatibility but is not a separate current panel choice. Threshold and Custom Mask use vertex colors rather than geometric isolines. See boundary algorithms and mask safety. **Vertex RGB** preserves the original mesh while active and can cost more for large surfaces; it is not a scalar map or statistical threshold. See RGB format/binding and Double Brain ordering.

The **Volume** section controls volume mapping algorithm, sign/threshold handling and ROI drawing; these are separate from Surface's scalar palette and are described in Chapter 10. A visually plausible map is not evidence that the mesh and volume are registered. Confirm coordinate spaces before anatomical interpretation.

For an AAL vertex-label exercise with Automatic boundaries and binary-mask export, see shared tutorial 05 in Appendix Shared tutorials. Atlas IDs are categories, not effect sizes. Follow the edition-specific mesh pairing and retain the resource credits.

Inspect on the rendered canvas

MATLAB: Click Inspect, then pick a displayed surface vertex or node. The on-canvas label reports source index, available value/module, assumed-MNI XYZ, Brodmann Area and AAL region. Another pick replaces it; clicking Inspect again clears it. Edges, arrows, ROI meshes and fibers are not pick targets. Surface picks use the nearest displayed source vertex rather than interpolating the mouse-hit position.

Python Inspect

Click Inspect and pick a displayed surface vertex. Python reports the source vertex/value, assumed-MNI XYZ and Brodmann Area/AAL region in a compact annotation above the canvas. Another pick replaces it; clicking Inspect again clears it. Python node, edge and ROI picking are not supplied by this surface Inspect route. Neither edition registers or affinely transforms inputs for this lookup.

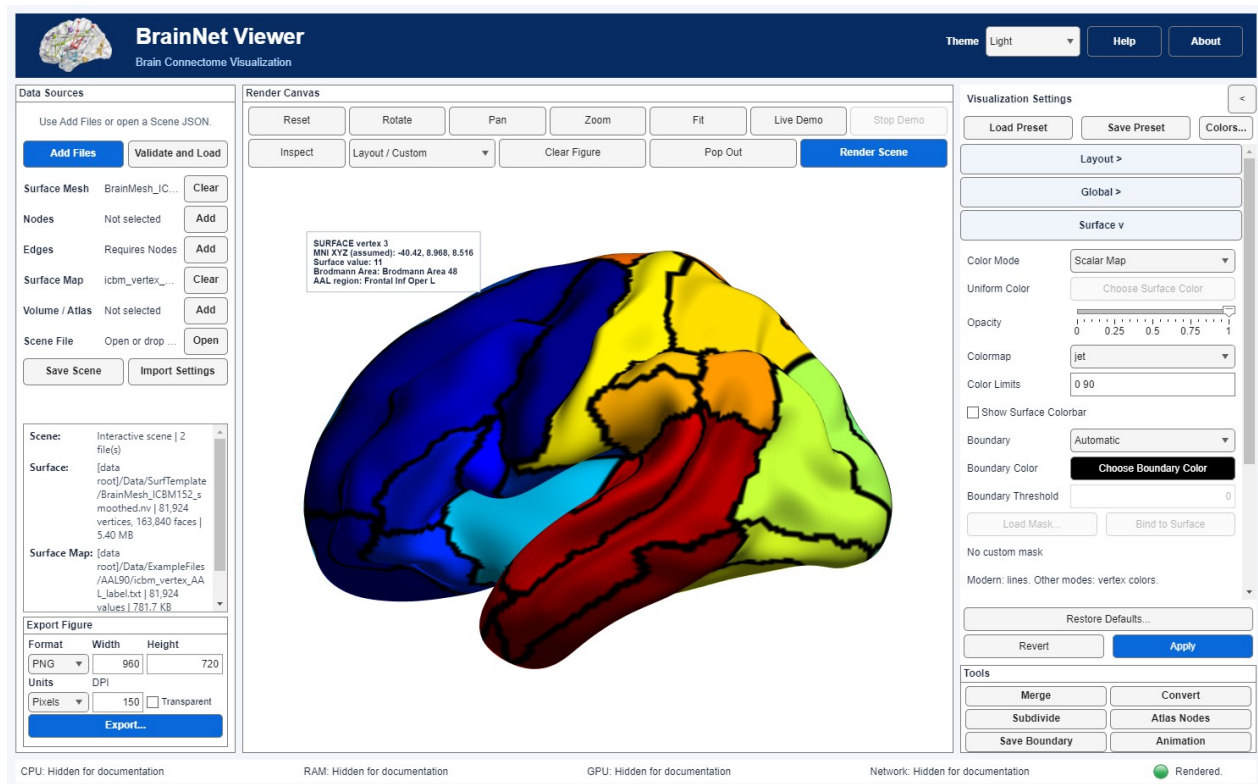


Figure 11. An Inspect label on the rendered surface reports Brodmann and AAL regions. MATLAB illustration.

Inspect assumes coordinates are already in the relevant MNI space. It does not register data or apply transforms. Native-space, animal or unregistered inputs may give plausible but incorrect labels. Verify coordinates before anatomical interpretation; see Inspect limits.

The AAL vertex-map exercise demonstrates this lookup with the bundled AAL/Brodmann resources. A successful lookup does not prove spatial registration.

8 Nodes

Use **Visualization Settings** → **Nodes** to control marker visibility, labels, size and color. In a .node row, columns 1–3 are XYZ coordinates, **column 4 supplies module/color values**, column 5 supplies node-size values, and column 6 is a label. Verify units and anatomical space rather than inferring them from the extension. The shared **Apply** commits and renders; **Revert** discards pending edits.

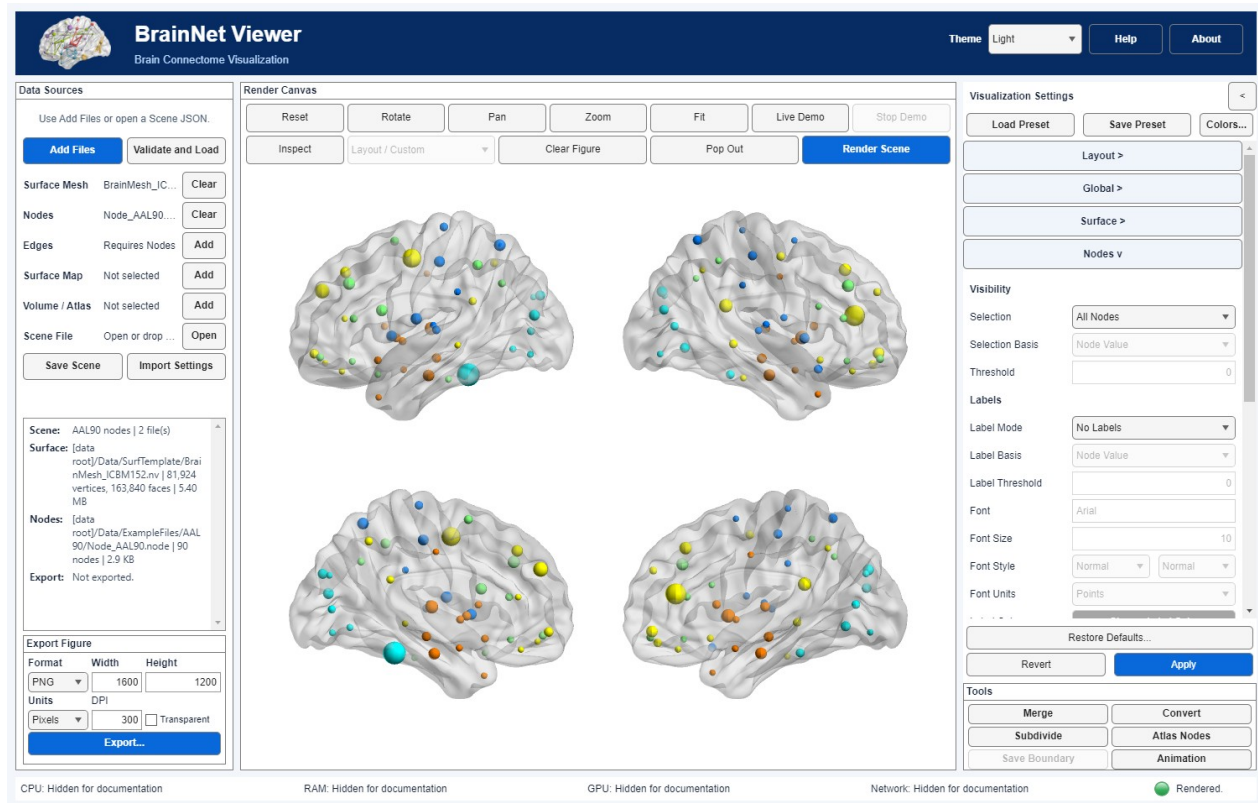


Figure 12. The bundled AAL90 nodes on ICBM152 in four views; colors use node column 4 and mapped sizes use column 5. MATLAB illustration.

Visibility and labels

Control	New-settings default	Current rule
Selection	All Nodes	Threshold keeps nodes with original Selection Basis > Threshold ; equality is excluded. Connected Nodes keeps either endpoint of an edge retained after edge filtering; no selected edges means no Connected nodes.
Selection Basis / Threshold	Node Value / 0	Enabled for Threshold. Node Value reads column 5; Module reads column 4. Module still uses a numeric > comparison, not equality to a chosen ID. Mapped radii and Scale do not affect this test.
Label Mode	No Labels	All Labels, No Labels, or Threshold. Labels appear only on visible nodes with nonzero rendered radius; a label exactly - is suppressed, but a hyphenated name remains valid.
Label Basis / Label Threshold	Node Value / 0	Enabled for Threshold labels. Uses the original column 5 or 4 value with strict >; independent of node-selection threshold.
Font / Font Size	Arial / 10	Font name and a positive numeric size (editor limit >0 to 100). Font Units defaults to Points; other offered units are Pixels, Normalized, Inches and Centimeters.

Control	New-settings default	Current rule
Font Style / Label Color / Keep Labels in Front	Normal + Normal / black / off	Weight is Normal or Bold; angle Normal or Italic. The front option changes display order, not data coordinates; it cannot guarantee collision-free dense labels.

Node size

Control	New-settings default	Current rule
Size Mode	Mapped	Uniform uses Base Size; Mapped normalizes source values to radii; Raw uses unnormalized source values; Threshold uses either Mapped or Raw radius rules then replaces radii strictly below Size Threshold with 1. Equality remains.
Size Source	Node Value	Column 5 by default, or Module/column 4 as a modern extension. Inactive in Uniform. This choice does not alter color mode.
Base Size / Scale	3 / 1	Base Size matters in Uniform; positive Scale multiplies the resulting radius exactly once in all modes. Radius is in input-coordinate units, not screen pixels.
Size Threshold: Mapped/Raw, value	Mapped / 0	Enabled only for Threshold. Comparison occurs before Scale; a below-threshold radius becomes 1 before scaling. This threshold changes size, not visibility.

MATLAB numeric size rules: Mapped sizes follow the legacy normalization (including its log compression for large ranges); constant input gives radius 4 before Scale. Raw negative/invalid final radii are rejected rather than silently made positive. A zero final radius omits that sphere and its label but does not delete the input node or its edges. See numeric size rules.

Node color

Control	New-settings default	Current rule
Color Mode	Uniform	Uniform uses one Color; Mapped samples a palette from column 4 ; Modules assigns colors by sorted unique column-4 module IDs; Threshold uses two colors split by column 4. Column 5 is not the default color source.
Color / Colormap / Color Limits	RGB [37 89 152]/255 / Jet / Auto	Color button is active in Uniform. Palette is active for Mapped or Modules and can be custom through Colors...; fixed low high limits apply to Mapped, clamping outside values. Auto uses the full input column-4 range, not just visible nodes.
Color Threshold / Low (<=) / High (>)	0 / [37 89 152]/255 / red	In Threshold, column-4 values <= threshold use Low, values > use High. Mapped limits/palette do not change these two fixed colors.

For Modules, the palette comes from the **Nodes** colormap; same-module edge coloring can inherit it when Edges → Node Module is selected. A custom palette is not the same as the explicit, geometry-bound Vertex RGB surface table. See node color semantics, selection/labels and label placement.

Dense labels can overlap even when **Keep Labels in Front** is enabled. Inspect their placement at the intended export size and simplify labels or the view when necessary.

9 Edges

Use **Visualization Settings** → **Edges** to control connectivity selection, direction, size, color and opacity. An .edge matrix must follow the loaded .node row order. Values, coordinate distances and directed relationships have meaning only in that input's declared scientific context; neither edge lengths nor arrowheads imply white-matter tracts. The shared **Apply** commits and renders; **Revert** discards pending edits.

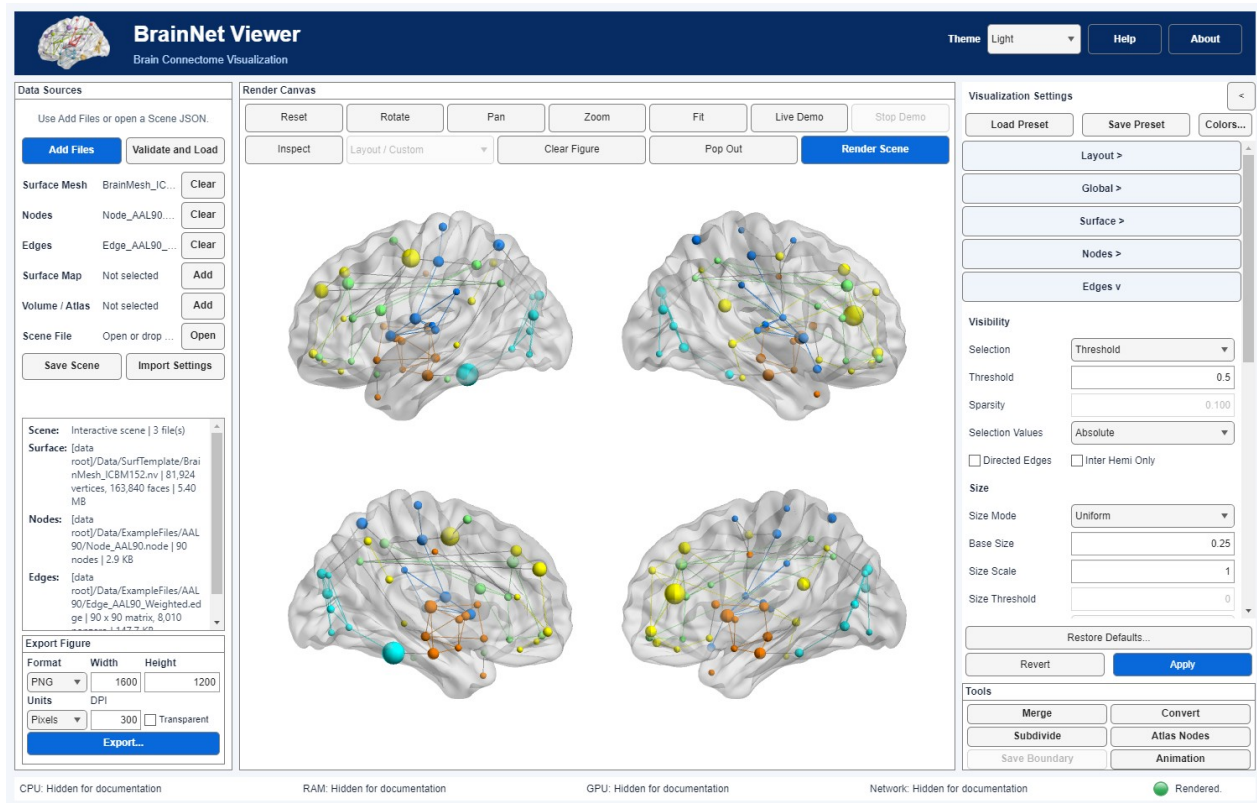


Figure 13. The bundled AAL90 weighted network in four views, showing undirected connections at threshold 0.5. MATLAB illustration.

Visibility and direction

Control	New-settings default	Current rule
Selection	All Edges	All keeps nonzero eligible off-diagonal edges. Threshold keeps selection values $\geq$ Threshold. Sparsity (Top Count) ranks nonzero eligible candidates and retains a deterministic count.
Threshold / Sparsity	0 / 0.1	Threshold is used only in Threshold mode. Sparsity fraction is 0–1 for Top Count; 0 selects no edges. Selection does not rewrite source weights.
Selection Values	Absolute	Signed or Absolute changes ranking/thresholding only; source weights are not rewritten. Other size/color/opacity signs have separate controls.
Directed Edges	off	Off uses nonzero upper-triangle candidates; on uses nonzero off-diagonal ordered pairs and draws arrowheads. The matrix is not automatically symmetrized.
Inter Hemi Only	off	After selection, retains opposite-X-side or midline endpoints by node $X=0$. This is a coordinate rule, not anatomical hemisphere detection.

Sparsity (Top Count) selects a deterministic count of eligible nonzero edges. The current UI provides All, Threshold and Sparsity (Top Count), not a separate Legacy Threshold mode. Connected Nodes in Chapter 8 follows the finally selected edge pairs. Review tie handling when comparing editions.

Size and arrowheads

Control	New-settings default	Current rule
Size Mode	Mapped	Uniform uses Base Size; Mapped maps retained weights to radii 0.4–6 (constant set: 2); Raw clamps to 0.2–6; Threshold first tests original sign-adjusted weights $\geq$ Size Threshold, then sizes qualifying edges by Mapped or Raw basis. Below threshold uses radius 0.4, not removal.
Base Size / Size Scale	1 / 0.7	Base Size is used in Uniform. Positive Size Scale multiplies radii once in all modes. These are radii in input-coordinate units, not pixel widths.
Size Threshold / Threshold Basis	0 / Mapped	Enabled only in Threshold. Basis is Mapped or Raw; the threshold comparison precedes normalization and scaling.
Size Values	Absolute	Signed or Absolute is independent of Selection Values and color/opacity signs. Negative invalid final radii are rejected rather than silently made positive.
Arrowhead Shape: Length / Width	1 / 0.7	Enabled for Directed Edges. Multipliers are 0.25–3; length is capped at 35% of edge length. The edge tube ends at the arrowhead base to avoid overlap. These settings change geometry, not matrix direction or weight.

Threshold+Raw has a different lower-radius rule from ordinary Raw; see full radius calculation. A zero-radius edge may be absent from geometry without changing its selection/Connected semantics.

Color and opacity

Control	New-settings default	Current rule
Color Mode	Uniform	Uniform uses one Color. Mapped uses a 64-entry palette. Threshold uses Low when value $\leq$ Color Threshold and High when $>$; Length uses Short when original-node Euclidean distance $<$ Length Threshold and Long at $\geq$. Node Module compares both endpoints; Custom Matrix uses an explicitly bound node-order color-index matrix.
Color / Colormap / Color Limits	[1 0.9 0.73] / Jet / Auto	The Color button is for Uniform. Mapped/Custom Matrix use the selected named colormap or Custom (Colors...); fixed low high limits apply only to Mapped. Auto uses retained color weights after selected sign handling. Equal limits use the last palette color.
Color Threshold / Length Threshold	0 / 75	Threshold coloring uses original edge weights with its own sign choice; equality is Low. Length uses input XYZ coordinate units, not tract/geodesic length; equality is Long.
Threshold Values / Mapped Values	Absolute / Absolute	Separate Signed/Absolute choices for their respective color modes. They do not change Selection Values, radii or opacity.
Low / High colors; Inter-module Color	gray [.38 .38 .38] / [1 .9 .73]; gray [.5 .5 .5]	Low/High are used for Threshold or Length (buttons become Short/Long). Node Module uses the Nodes module palette for same-module edges and Inter-module Color for different modules. Select Nodes $\rightarrow$ Modules to make visible node colors match.
Load Colors... / Bind to Nodes	no matrix	Custom Matrix only. Confirm matrix dimensions and original node order; the binding cannot independently establish scientific correspondence.
Opacity Mode / Value / Range	Uniform / 0.7 / [0.1 0.9]	Uniform uses one 0–1 opacity. Mapped maps selected weights to two output opacity endpoints in [0,1]; constant weights use the upper endpoint. A transparent edge is not deselected.
Opacity Values	Absolute	Signed/Absolute for Mapped opacity only, independent of selection, size and color.

The Node Module palette is derived from the Nodes colormap; the Edges colormap selector does not override it. In MATLAB, this mode supports at most 20 distinct modules and rejects excess categories rather than recycling colors. Custom Matrix colors require explicit binding and file-format review. Color threshold/length endpoints are detailed in Edge Colors; mapped opacity in Edge Opacity. Changing a visual option does not rewrite the .edge file or establish registration.

Before publishing, check dense or directed networks at the intended export size; an edge that is selected may still be hard to see because of its radius, opacity or overlap. Graphics backends need not produce identical pixels.

10 Volume mapping and statistics

Use **Visualization Settings** → **Volume** to map a 3-D NIFTI onto a surface. A readable NIFTI is not automatically aligned with a separately loaded surface. Mapping transforms surface coordinates by the volume affine to sample voxels; it does **not** perform image registration. Verify the surface/volume coordinate spaces and units before interpreting locations. **Apply** commits settings and renders; **Revert** discards pending changes.

In the Volume panel, **Threshold Mode**, **Statistical Threshold**, **P Value**, **Cluster Size** and **Voxel Connectivity** appear above the **Surface Mapping** section because they also govern statistical ROIs. Controls enable according to Volume Type and Threshold Mode; ROI P Value uses Cluster Size, while Surface Map P Value does not.

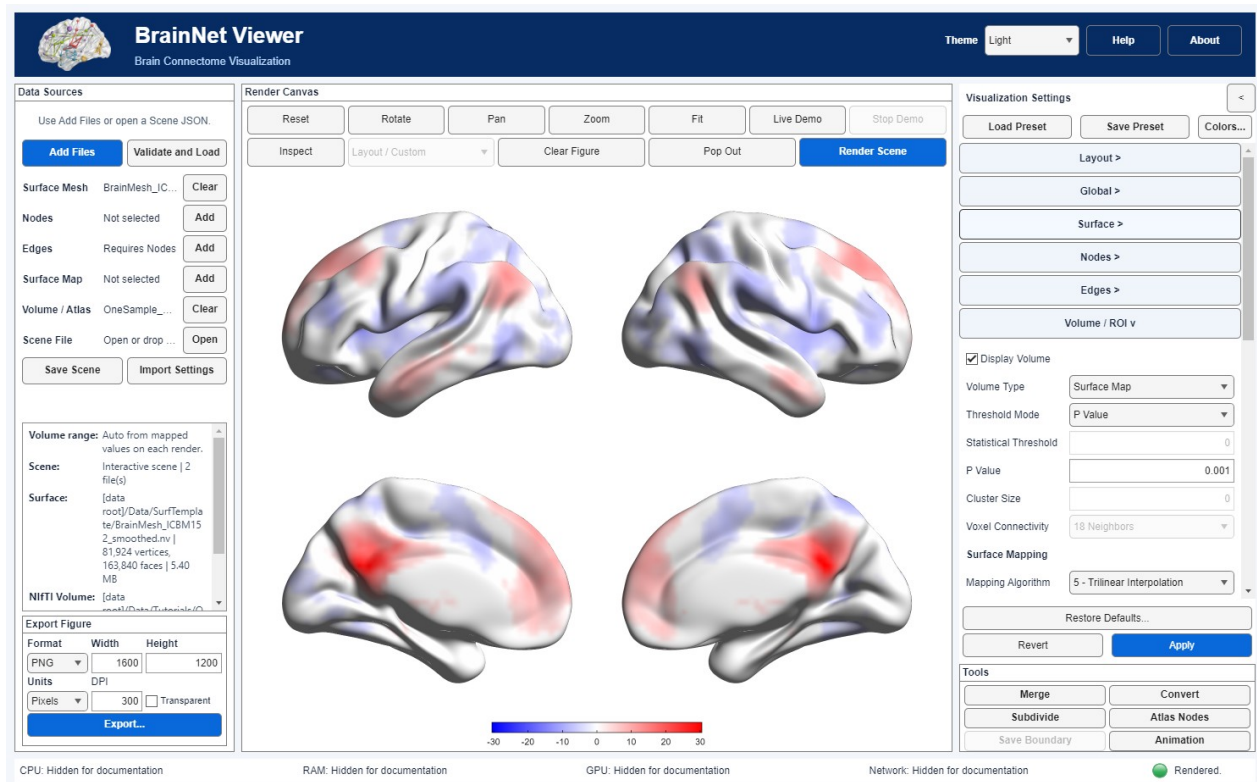


Figure 14. A signed statistical map displayed on four surface views with the P-value controls visible. MATLAB illustration.

Control	New-settings default	Current rule
Display Volume	off	Enables the chosen Volume Type. A newly loaded file does not by itself establish scientific suitability.
Volume Type	Surface Map	Surface Map maps a 3-D volume onto a loaded surface; ROIs (Atlas / Statistical) extracts labelled meshes and uses the Chapter 11 controls.
Mapping Algorithm	2	Used for Surface Map only. The eight algorithms below have different numerical meanings, not a simple low/high-quality order.
Threshold Mode	None	None keeps values; Value keeps finite values with $\text{abs}(\text{value}) \geq \text{abs}(\text{Statistical Threshold})$; Cluster adds a minimum connected-component size. P Value converts supported NIFTI statistical metadata into a voxelwise cutoff. In ROI mode it also applies Cluster Size; see Chapter 11.

Control	New-settings default	Current rule
Statistical Threshold / P Value / Cluster Size / Voxel Connectivity	0 / 0.05 / 0 / 18 neighbors	Value and Cluster use the numeric threshold; Cluster also uses voxel count and 6/18/26 connectivity. P Value requires $0 < P < 1$ and valid statistic metadata. The original volume is not rewritten.

For Value/Cluster, zero is background, nonfinite voxels are excluded, equality at the absolute-value threshold is retained, and Cluster removes components **smaller** than Cluster Size. The 6/18/26 selection defines statistical connectivity; it is not the separate 26-connected safety check used in ROI smoothing. Cluster filtering can change the values available for later mapping and the resulting color range. See statistical ROI rules.

For P Value, the reader uses the NIfTI description from SPM, REST or DPABI (SPM{T_[20]}, REST{F_[3,20]}, DPABI{R_[20]}, or a Z token); a standard NIfTI CORREL/TTEST/FTEST/ZSCORE intent and its parameters are a fallback. Missing or invalid degrees of freedom cause an error, never a silent numeric-threshold fallback. T, R and Z use **two-sided** uncorrected P; F uses the **upper tail** and excludes negative F values. The resulting statistic cutoff, degrees of freedom, tail and metadata source are recorded in render preparation/provenance. P filtering occurs **before** mapping; interpolation or neighborhood smoothing can subsequently change vertex values, so a displayed vertex value is not itself a voxelwise P-value test. There is no family-wise/FDR correction or spatial registration.

Mapping algorithm (Surface Map)

No. / menu label	Calculation and boundary caution
1 · Nearest Voxel	Round the transformed surface position and sample that voxel. Invalid/out-of-volume positions receive zero.
2 · Average Surface Neighbors	Take algorithm 1 values first, then average each vertex with its surface-mesh neighbors and itself. This is a surface-adjacency average, not a 3×3×3 voxel mean; mesh topology affects the result.
3 · Average 3-by-3 Voxel Neighborhood	Apply a 3×3×3 mean filter to the volume, then nearest-voxel sample it. Volume boundary handling follows the filter implementation.
4 · Gaussian Voxel Smoothing	Smooth the volume with a Gaussian kernel, then nearest-voxel sample it. MATLAB uses smooth3; Python uses its numerical implementation. Kernel/boundary details are edition-specific, and nearby labels or signs can mix.
5 · Trilinear Interpolation	Interpolate the eight enclosing voxel values at an interior position; outside the supported interior returns zero.
6 · Maximum 3-by-3 Voxel Neighborhood	At a valid interior rounded position, take the maximum of 27 voxels.
7 · Minimum 3-by-3 Voxel Neighborhood	At a valid interior rounded position, take the minimum of 27 voxels.
8 · Extremum 3-by-3 Voxel Neighborhood	Choose the 27-voxel sample with greatest absolute magnitude and retain its original sign .

The mapping gallery illustrates how the eight algorithms display the same input. It is not an accuracy ranking. For a discrete atlas, start with Nearest Voxel: averaging and interpolation can create values that are not source labels. For continuous statistics, algorithms 2 and 5 can give smoother displays; they respectively average sampled mesh neighbors and interpolate the volume at each vertex. Record the chosen method and compare the result with the source map before interpreting a location.

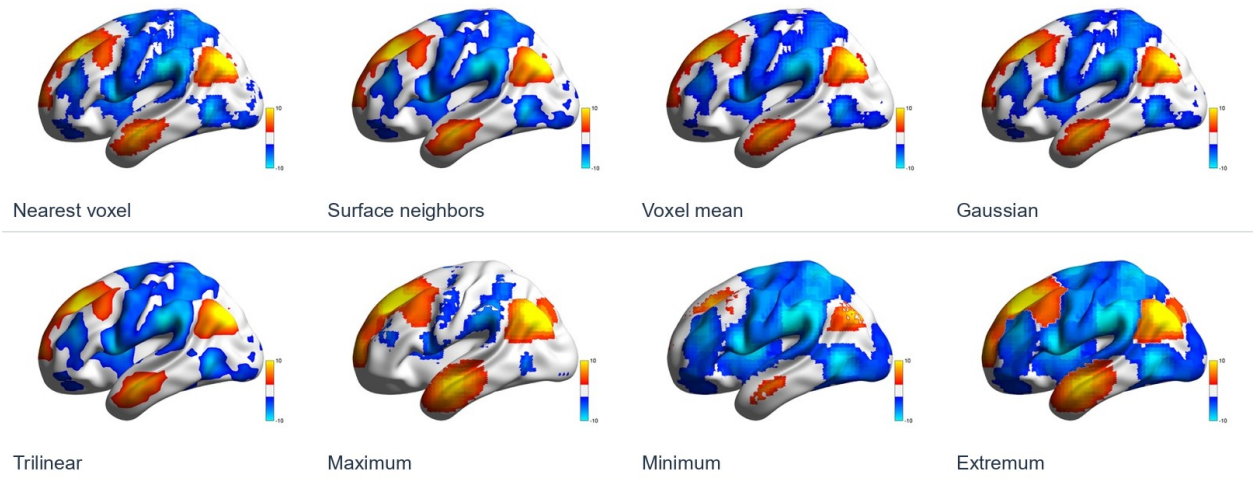


Figure 15. Eight volume-to-surface mapping algorithms applied to the same input. MATLAB illustration.

For sign and color-range choices, **Display Values** changes the signed color axis but does not remove negative or positive vertex values. If a map contains values from -3 to $+6$, Both mode uses a symmetric -6 to $+6$ axis; a $+3$ and a -3 therefore have comparable distance from the center even though $+6$ is the observed positive maximum. Positive Only or Negative Only uses the corresponding half of the palette. Change the relevant bounds only after checking the actual sampled value distribution and the displayed colorbar; do not interpret white/null color as a formal nonsignificant voxel unless the thresholding and adjustment rules warrant that interpretation.

MATLAB: CPU supports all eight. The current GPU mapping path supports algorithms 1 and 5 only; an explicit GPU request for the other algorithms falls back to CPU with a recorded reason. Auto chooses GPU for algorithm 5 on sufficiently large surfaces when a supported GPU is available; otherwise it uses CPU. This is an execution policy, not a guarantee of speedup on every dataset. Python numerical mapping uses CPU implementations; graphics display uses VTK. Corresponding algorithm names do not guarantee identical smoothing kernels, boundary handling or sampled values between editions.

Ranges and sign display

Control	New-settings default	Current rule
Positive Range / Negative Range	Auto / Auto	Enter Auto or minimum and maximum (either endpoint may be Auto). Manual bounds persist when the volume file changes and are identified in Scene Summary; recheck them for each new dataset. Auto Ranges only resets the pending fields—click Apply to commit and render.
Colormap	Blue-White-Red	One 256-entry palette: its lower half represents negative values and upper half positive values. A custom RGB palette can be set with Colors.... The Surface panel's scalar-map palette is separate.
Display Values	Positive & Negative	Both, Positive Only or Negative Only changes the color axis , not which vertex values exist . Both uses a symmetric $[-M, +M]$ axis around zero even when the data extrema differ; one-sided modes use the respective palette half. Values outside a chosen range clamp to endpoint colors.
Adjust for Null / Null Color	on / $[0.95 \ 0.95 \ 0.95]$	When on, the null-side palette region receives Null Color; when off, finite values use the continuous palette. Null Color is editable only when adjustment is on. This changes display, not sampled values.

Both-mode symmetry can reduce contrast for the smaller-magnitude sign. Do not infer absence of an effect merely because one side is pale; inspect the numeric data and actual colorbar. The illustrated example uses a resting-state example map from a published study, but its alignment to a chosen surface still needs independent verification. See signed-axis details.

Multiple-comparison correction is **not available**. Check the source map's statistic, degrees of freedom, coordinate alignment and your analysis-specific inference policy before interpretation.

11 ROI drawing

Set **Display Volume** on and choose **Volume Type** → **ROIs (Atlas / Statistical)** to draw atlas or statistical regions. **Apply** commits and renders; **Revert** discards pending edits. ROI identifiers are labels, not continuous quantitative values.

The shared threshold controls are above **Surface Mapping** in the Volume panel, not inside that surface-only section. In ROI mode, P Value and Cluster Size are enabled together; Statistical Threshold is used by Value/Cluster instead.

ROI controls

Control	New-settings default	Current rule
Threshold Mode	None	None preserves positive integer atlas IDs. Value and Cluster construct connected regions with $\text{abs}(\text{value}) \geq \text{abs}(\text{Statistical Threshold})$; Cluster also removes components smaller than Cluster Size. P Value converts supported NIFTI T/F/R/Z metadata into a statistic cutoff, then applies Cluster Size before assigning ROI IDs.
Statistical Threshold / P Value / Cluster Size / Voxel Connectivity	0 / 0.05 / 0 / 18 neighbors	Value/Cluster use the numeric threshold; P Value requires $0 < P < 1$ and valid statistic metadata. T/R/Z are two-sided; F is upper-tail only. P Value and Cluster use Cluster Size in voxels, not mm^3 . Zero and nonfinite voxels are excluded; choose 6, 18 or 26 connectivity.
Draw All ROIs	on	Draws every valid positive ID. While on, ROI Selection is disabled.
ROI Selection	empty	Turn Draw All off to enter positive IDs separated by spaces or commas. Typing a nonempty list also turns Draw All off. Atlas mode uses source IDs; statistical Value/Cluster/P Value modes use newly assigned, consecutive component IDs, which can change with thresholds, Cluster Size or connectivity.
ROI Color Mode	Label Index	Uniform uses one ROI Color; Label Index cycles the selected categorical ROI Palette by ID; Custom requires an explicit ROI_ID R G B table. Color modes do not redefine region membership.
ROI Palette	HSV	Label Index can choose any named palette, including Viridis, Plasma, Inferno, Magma and Cividis. Its 256 colors cycle by positive ROI ID; this is categorical indexing, not a continuous quantitative colorbar. HSV and Lines generally separate categories better than sequential gradients.
Show ROI ID Legend	off	Optional ROI ID/color key in the first rendered view; at most 16 entries are shown, with an overflow count. It is not a value colorbar.
ROI Color / Load ROI Colors... / Custom table	[0.8 0.2 0.2] / no table	ROI Color applies in Uniform. For Custom, enter or load rows ROI_ID R G B, unique positive integer IDs and finite RGB in [0,1]. Every selected ID present in the volume needs a color; missing rows cause an error rather than a fallback.
ROI Smoothing / Smoothing Kernel	Gaussian / 3	None, Box or Gaussian. Kernel is in voxels; 1 disables smoothing, and larger effective odd kernels soften boundaries. This is display mesh smoothing, not a change to the stored volume or scientific cluster IDs.

ROI P Value uses this sequence: convert P to a statistic cutoff, discard subthreshold/nonfinite voxels, remove components smaller than Cluster Size, then number surviving components. Positive and negative T/R/Z voxels in contact can form one ROI; F excludes negative values. The cutoff, degrees of freedom, metadata source, removed clusters and resulting IDs are recorded in preparation provenance. Missing/contradictory metadata raises an error. This is an uncorrected voxelwise threshold; Cluster Size does not by itself establish cluster-level significance, and no multiple-comparison correction or registration is performed. An atlas should normally use None, not P Value.

ROI meshes use a 0.5 isosurface and a virtual zero halo to close regions touching volume borders. Smoothing checks each original 26-connected component and conservatively falls back to the whole original ROI if one would disappear. This does not guarantee volume, thin-branch or full topology preservation. In MATLAB, active smoothing with kernel >1 requires Image Processing Toolbox (bwconncomp); without it choose None or kernel 1. See ROI geometry, custom colors and statistical numbering. Recheck a Custom table whenever statistical filtering changes IDs. Automatic atlas/statistic classification is not provided.

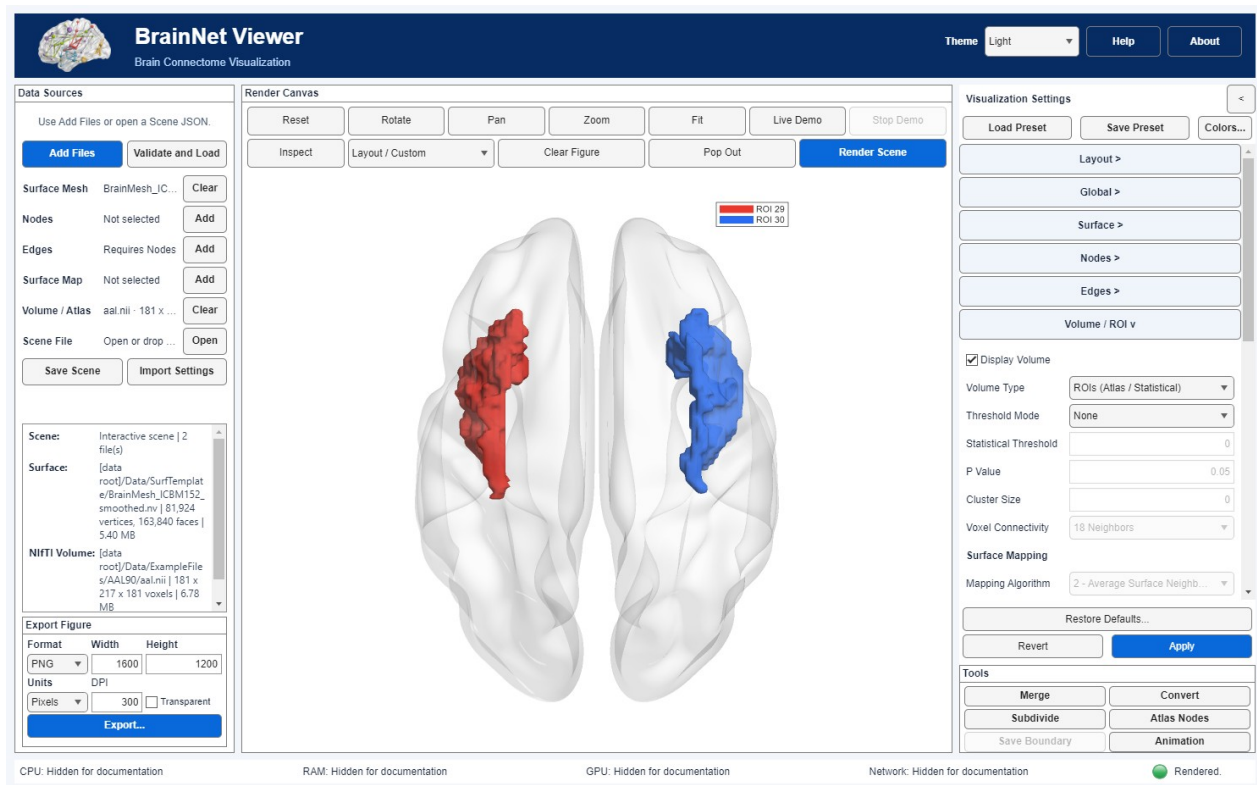


Figure 16. Bilateral AAL insula regions displayed above a translucent surface. MATLAB illustration.

Python ROI and null display

Python ROI processing uses its Python numerical dependencies rather than MATLAB Image Processing Toolbox. Kernel smoothing is for display; component safeguards are not a guarantee of full topology or volume preservation. A hemisphere-only surface restricts displayed ROIs to the corresponding hemisphere. Statistical component IDs can change with threshold, cluster size or connectivity.

Adjust for Null in Python uses the original surface color for null/zero mapped values. MATLAB retains an editable Null Color control for volume mapping. Zero/missing values in vertex scalar maps also retain the original surface color; a null color is a display convention, not by itself statistical significance.

12 Colormaps and presets

Use custom colormaps to control display colors and visualization presets to reuse settings across datasets. A colormap changes appearance, not numerical values, thresholds or registration. A preset transfers display settings, not data files.

Custom RGB colormaps

Open **Colors...** below **Load Preset / Save Preset** in the right-hand Visualization Settings panel. The editor has four independent **Target** choices: **Surface**, **Nodes**, **Edges** and **Volume**. Volume uses one signed colormap, not separate positive and negative palettes. Named choices now include Viridis, Plasma, Inferno, Magma and Cividis; these are also available directly from each target's colormap selector. Choose a named **Preset** and **Count** (2–4096 rows), then **Generate**; **Resample** interpolates the current table to Count, and **Reverse** reverses its row order. The Red/Green/Blue table accepts finite components in [0,1]; a valid edit updates the preview. **Import...** reads RGB TXT/JSON; **Export...** writes the current target's table only, without applying it. Existing export files require an overwrite decision.

Use in Settings stages the four maps in the right-hand panel and closes the editor. **Cancel**/closing discards editor changes. A merely viewed named map stays named; modifying it produces an embedded numeric map. Choose a display mode that consumes that target's colormap, then press the single panel-bottom **Apply**: it commits settings and renders. **Revert** discards pending panel edits. A separate Render Scene click is unnecessary after a successful Apply.

	Red	Green	Blue
1	0.1725	0.4824	0.7137
2	0.6706	0.8510	0.9137
3	0.9686	0.9686	0.9686
4	0.9922	0.6824	0.3804
5	0.8431	0.0980	0.1098

5 RGB rows. Use in Settings, then choose Apply to render.

Figure 17. The custom Volume colormap editor before the colors are applied to the scene. MATLAB illustration.

Surface Scalar, Node Mapped/Modules, Edge Mapped, and Volume Surface Map are examples of applicable consumers. In **Edge Node Module** mode, same-module edges inherit the **Node** module palette, not the Edge colormap. Node and Edge fixed threshold colors do not use the continuous palette. For **Volume Display Values = Positive & Negative**, the one map is split at its midpoint between negative and positive values; review the resulting signed colorbar and limits. The ROI **Label Index** palette is a separate categorical selector in Volume settings, not an editor target. Numeric palettes are RGB-interpolated, not guaranteed perceptually uniform or accessible. See the node, edge, volume and ROI chapters before interpreting colors scientifically.

MATLAB RGB JSON uses kind brainnet-colormap, schema 1.0 and a colors array. Python exports a plain RGB array and accepts supported rgb/colors objects; the edition schemas are not assumed identical. A reviewed three-column RGB text table is a simpler exchange route. RGB components must be finite and in [0,1]; importing a table changes colors, not scientific data. Treat legacy EC MAT configuration as trusted input.

Save and load a display preset

1. Load a dataset and edit Visualization Settings. Press **Apply** to commit and render.
2. MATLAB Save Preset captures applied settings. Python Save Preset can capture current panel values including pending edits. Apply first to save the intended rendered configuration. Neither preset packages input files, session camera or export-panel preferences.
3. With another dataset, select Load Preset and review the resulting controls. MATLAB confirmation replaces pending display settings and clears the old render/camera; Python stages normalized settings for review. Apply commits and renders. Recheck data-specific ranges and thresholds.
4. Recheck data-specific color limits, thresholds, units, selected ROI IDs and anatomy before exporting. A role-compatibility warning is not proof of spatial registration or scientific validity.

Save Scene is different: it records one Scene's input references, settings and (when saved from the modern App) session camera/export preferences; MATLAB inputs are referenced, whereas Python saves companion assets beside the JSON. Invalid, unknown-version or explicitly input/camera-bearing preset JSON is rejected. Cancellation and failed validation do not replace the current settings. Fiber rendering remains unavailable; Surface Map and statistical ROI P-value presets require supported statistic metadata in the selected NIfTI. Preset names/titles may contain private text; review before sharing.

The real-data illustrated exercise shows custom Volume RGB staging, Apply and preset replay. Check the resulting colorbar and limits against the loaded data before reusing a preset in a figure.

13 Figure export

Export requires a currently rendered Scene; a loaded but unrendered Scene cannot be exported. Export is an output operation, not a change to the scientific data or applied settings.

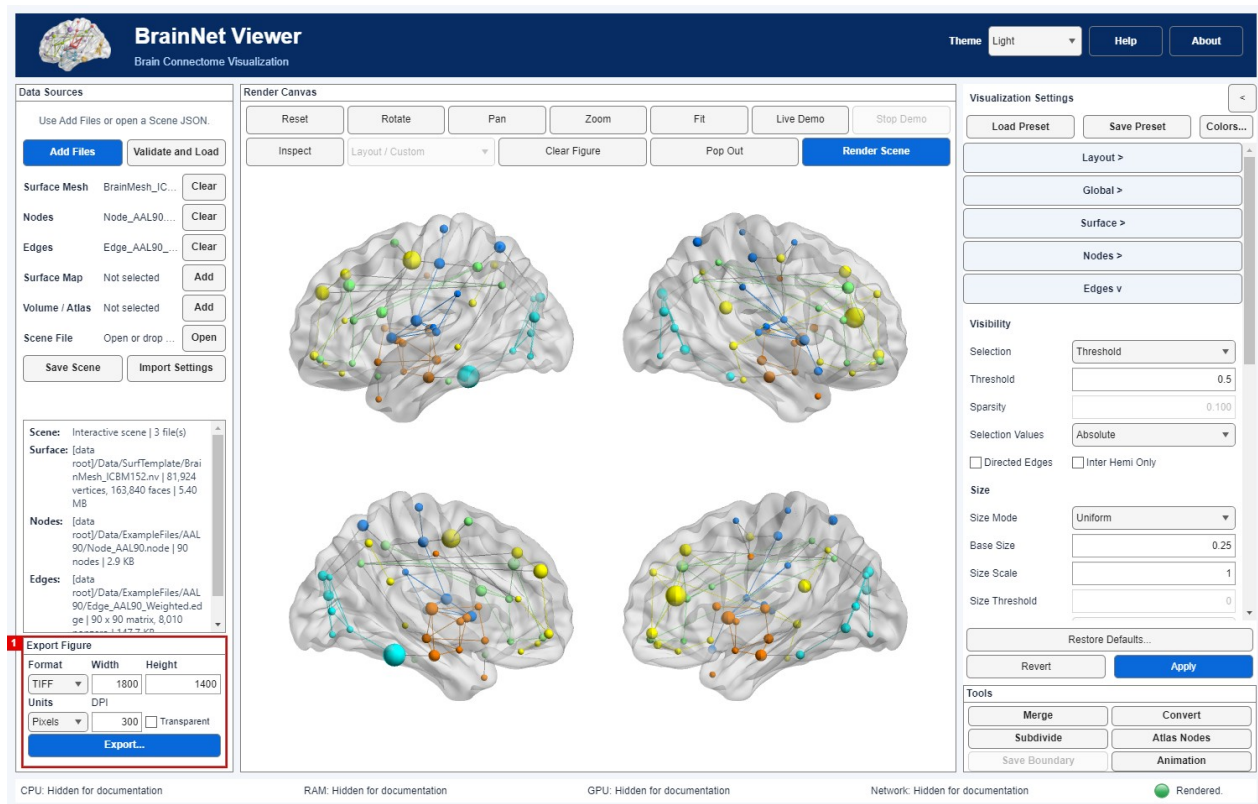


Figure 18. The red frame identifies Export Figure settings, including output format, dimensions and resolution. MATLAB illustration.

Export a still figure

After rendering, use Export Figure below Data Sources. Select format, dimensions, units, DPI and Transparent. Desktop opens a destination chooser. Web downloads the image and provenance JSON separately to the browser download location; allow multiple downloads if prompted. The compact panel has no separate Export Background or Size Mode selector.

Control	Current choices and meaning
Format	PNG, JPEG, TIFF, BMP, PDF or SVG. Raster formats produce pixels; PDF/SVG do not guarantee editable vector 3-D content.
DPI, Width, Height, Units	Positive DPI and dimensions; units include Pixels, Millimeters/Millimetres, Centimeters/Centimetres, Inches and Points. Physical size is converted using DPI.
Content handling	No separate Content control in the compact main panel. PDF/SVG can include rasterized 3-D geometry; Python transparent PDF/SVG rasterizes the complete scene and annotations.
Transparent	Transparent requests supported alpha output. With it off, use the canvas background. JPEG/BMP cannot be transparent; review actual output alpha rather than assuming it.

The file chooser may replace the selected output; choose a new path if an existing file must be preserved. A successful export writes the figure **and** a neighboring `.provenance.json` sidecar with Scene/renderer/export information. Review that JSON and any absolute input paths or titles before sharing. Output preferences are part of an App-saved Scene session, but loading a display preset does not change them. The interface Light/Dark theme is not the figure background.

Python export and recovery

Both editions offer pixel and physical sizes including millimetres; MATLAB labels use Millimeters/Centimeters and Python uses Millimetres/Centimetres. Pixel export uses the requested dimensions; physical dimensions convert using DPI. A Python request is limited to 16–16000 pixels per dimension, 64 million pixels and integer DPI 1–2400.

Python opaque PDF/SVG contains raster brain geometry with supported vector annotations; SVG text can be glyph outlines. Transparent PDF/SVG uses a full RGBA raster. These files are not pure-vector editable brain meshes. JPEG/BMP cannot be transparent. Review saved dimensions, alpha, labels and the provenance sidecar.

After export or a reported error, the screen window, camera, DPI and colorbar fonts should return to the onscreen state. Close Figure Saved and review the main canvas. Preserve an existing image by choosing a new filename; do not assume a dialog grants permission to overwrite unrelated files.

14 Camera animation

Render first and adjust the canvas camera, then choose Export Animation in Tools. The animation dialog exports the current static scene into a new folder; it does not animate changing connectivity or edit the main canvas/camera. Its defaults are 960 × 720 pixels, 5 seconds, 24 fps, world Z axis, 0°→360°, and Safe Fit on. Available formats are MP4 when the edition's encoder is available, Motion JPEG AVI, and PNG frame sequence. Choose Export... and a new folder name under an existing parent; existing bundle paths are never overwritten.

Export the CURRENT rendered scene into a NEW folder. Your canvas and camera remain unchanged.

Format	PNG ▼
Width (pixels)	640
Height (pixels)	480
Duration (seconds)	1
Frame rate (fps)	4
World rotation axis	Z ▼
Start offset (degrees)	0
End offset (degrees)	360

☒ Safe Fit: keep geometry inside every rotating view

A full turn omits the duplicate endpoint. PNG exports one image per frame.

Export... Close

Figure 19. Camera-animation controls for a PNG frame-sequence example. MATLAB illustration.

Safe Fit centers and orthographically frames geometry across the orbit on a private rendering copy; disable it only if preserving current framing matters more than possible cropping. Angles are offsets from the current camera around the chosen world X/Y/Z axis. A full turn omits its duplicated endpoint. Frame count is $\text{round}(\text{duration} \times \text{fps})$, so effective duration may differ slightly. Export speed is not playback speed; no GPU video-encoder claim is made. The dialog's **Cancel Export** takes effect after the current frame/codec operation; partial output is cleaned up.

The new folder contains movie.mp4 or movie.avi, or frame-000001.png etc., plus preview.png and provenance.json. The latter can contain Scene references, input metadata, cameras and per-frame records; source inputs are **not** copied. Limits: 128–3840 pixels per dimension, 1–60 fps, 2–1800 frames, at most 120 requested seconds, one billion total frame-pixels, and a nonzero orbit no greater than 360°. Video dimensions must be even; there is no transparent movie or audio. Codec availability and pixel-identical results across platforms are not guaranteed. See the animation guide for API and verification details.

Animation encoding by edition

MATLAB uses VideoWriter profiles. Python uses FFmpeg with available libx264 or MJPEG encoders; desktop dependencies supply the documented imageio-ffmpeg route. PNG frame sequences are an alternative when a movie encoder is unavailable. Encoder availability is not a promise of GPU video encoding. Animation renders a static Scene with a moving camera, not changing connectivity.

15 Surface formats and mesh tools

A successfully read mesh is **not** proof of MNI registration, matching units, compatible handedness, or permission to redistribute it. The modern readers preserve raw geometry rather than silently applying a dataset-specific transform.

Load and review a surface

In **Data Sources** → **Surface Mesh**, select a supported mesh; then **Validate and Load**, configure the display, and **Apply** to render. The same independent readers serve the desktop, Scene JSON and BrainNetRender. Surface support includes .nv, FreeSurfer .pial, BrainVISA .mesh, BYU/MOVIE.BYU .g, MNI/BIC .obj, MZ3 .mz3/.mz3.gz, and GIFTI .gii/.gii.gz, within each reader's restrictions. .obj means the MNI/BIC polygon format here, **not** generic Wavefront OBJ. .mesh does not imply Medit format. .pial is the advertised FreeSurfer desktop suffix; do not assume arbitrary FreeSurfer suffixes or compressed .pial.gz work. See the FreeSurfer, BrainVISA, BYU, MNI/BIC and MZ3 guides for exact encodings and rejection rules.

Coordinates and face winding are kept as returned by each reader; disk indices become one-based internally. BrainVISA .mesh **does not** receive the original BrainNet's implicit XYZ inversion. FreeSurfer surface RAS is not automatically scanner RAS/MNI. GIFTI transform metadata is recorded, not automatically applied by the mesh utilities. MZ3 embedded RGBA/scalars are validated but not automatically promoted to display settings or Surface Map. Verify space and geometry alignment before combining a mesh with nodes, volume, atlases or other surfaces. A single unsplittable mesh should use applicable full-surface layouts; two-hemisphere layouts cannot be assumed from its filename.

Tools that write new data

Tool in Tools	Inputs and operation	Output / caution
Merge Surfaces	Automatically review 2–64 meshes, order, bounds, counts and SHA-256. Space, units and note are optional; no verification checkbox is required.	Concatenates coordinates/faces into a new ASCII .nv; does not register, weld, smooth, repair or transfer color/scalar attributes.
Convert Surface to NV	Review one supported mesh and optional space/unit declarations.	New .nv preserving reader-returned geometry; no implicit legacy .mesh coordinate inversion or attribute transfer.
Subdivide Surface to NV	Review one mesh; choose 1–3 linear midpoint levels (4× faces per level).	New denser .nv; no smoothing, curvature preservation or automatic node/map correspondence.
Generate Nodes from Atlas	Review a discrete integer NIFTI label atlas, selected region IDs, affine and declared space/units.	New .node; each region uses its unweighted voxel-centre centroid, which need not lie within a concave/disconnected region. No registration or name lookup.

Each tool has a review step, checks source/dependency fingerprints again before publication, writes a **new** destination without replacing an existing one, and leaves the active Scene unchanged. Mesh utilities use default combined limits of five million vertices and ten million triangles; read/allocation may still require substantial RAM. The .nv/.node output includes a # JSON provenance comment, potentially with absolute source paths. Do not publish it without inspecting privacy and licensing. Declarations in comments do **not** assign a Scene coordinate space or perform transforms. Load the new file explicitly through Data Sources; for a portable Scene, place it within the configured data root or use an appropriate root. See merge, convert, subdivide, and atlas nodes for exact review fields, formats and limits.

Join 2 to 64 surfaces in list order. Coordinates and triangle winding are retained; no registration, vertex welding or color/map transfer.

[data root]\Data\SurfTemplate\BrainMesh_ICBM152Left_smoothed.nv

[data root]\Data\SurfTemplate\BrainMesh_ICBM152Right_smoothed.nv

Add Files...

Remove

Move Up

Move Down

INPUT REVIEW — raw coordinates, no transforms applied
81924 vertices | 163840 triangles

1. [data root]\Data\SurfTemplate\BrainMesh_ICBM152Left_smoothed.nv
40962 vertices [1:40962] | 81920 faces [1:81920]
Bounds XYZ: [-62.777, -0.750334] / [-101.378, 65.7245] / [-43.754, 74.9444]
SHA-256: 0e21eee007d7aa1e7379f47b65d272c885eee8ec4d6809ca1078cb4af0780fea

2. [data root]\Data\SurfTemplate\BrainMesh_ICBM152Right_smoothed.nv
40962 vertices [40963:81924] | 81920 faces [81921:163840]
Bounds XYZ: [0.750334, 62.777] / [-101.378, 65.7245] / [-43.754, 74.9444]
SHA-256: 2422ff0724a6dea9c326f6b866c007ba39eab5ec289bdcf87dcfb854104b441a

Space (optional)

For example: subject-01 native scanner space; blank is unknown

Units (optional)

For example: mm; no unit conversion

Note (optional)

Optional provenance note

Inputs reviewed. Ready to save; space, units and note are optional. No registration is applied.

Save New .nv...

Close

Figure 20. Merge Surfaces automatically reviews the selected inputs. Space, units and note are optional. MATLAB illustration.

These geometry operations are not a scientifically validated surface-registration pipeline. Verify anatomy and resource permissions for each dataset before using or distributing derived files.

Python tools and declarations

Python Merge and Convert do not require the optional space/unit/note declaration fields shown in the MATLAB Merge illustration. Add inputs to trigger review, check geometry and write a new destination. This optional mesh metadata rule does not apply to Python Generate Nodes from Atlas: confirm known coordinate space and units before publishing atlas nodes.

Atlas nodes use unweighted voxel-center centroids transformed by the atlas affine. They are not guaranteed to lie inside concave or disconnected parcels. The tool does not compute FC, assign measured connection strength or perform modularity analysis; constant module/size fields are user choices, not measured network quantities. Tool provenance comments can contain source paths in either edition.

16 Command line rendering and provenance

MATLAB command-line rendering uses BrainNetRender without opening the App; a MATLAB license and working graphics environment are required. Python uses brainnet-render with off-screen VTK graphics support. Docker/Web provides a separate browser workflow; it does not require these commands for normal use.

Prepare a reproducible Scene

MATLAB Scene preparation: In the desktop, select files, Validate and Load, adjust Visualization Settings, and Apply (which commits and renders). Use Save Scene to create Scene JSON. It records applied settings, relative input references/hashes, coordinate/redistribution metadata, and—when saved by the modern App—optional camera and export-panel state. It does not embed source data, accept unapplied edits, certify spatial registration or grant file-sharing rights. The data root is an explicit directory: references resolve relative to that root, not relative to the Scene JSON's folder. If external files are selected in the App, approval stages verified copies under its configured root; a manually written Scene or CLI run does not silently copy outside data.

To render one image without opening the App:

```
record = BrainNetRender("D:/results/scene.json", "D:/results/figure.png", ...  
    DataRoot="D:/project-data", ComputeBackend="cpu", ...  
    PerformanceMode="quality", Width=2000, Height=1500, ...  
    Resolution=300, Overwrite=false);
```

The command reads and validates input files, prepares data, renders a hidden modern figure and writes the image plus figure.png.provenance.json. Output extensions are PNG, JPEG, TIFF, BMP, PDF or SVG; complex PDF/SVG scenes may contain rasterized 3-D content. The output file and sidcar are protected from overwrite by default. Use a distinct output name for every run or explicitly choose `Overwrite=true` after checking the targets. A simple MATLAB loop may call `BrainNetRender` over different Scene/output paths; there is no claim of an independent job scheduler or automatic dataset discovery.

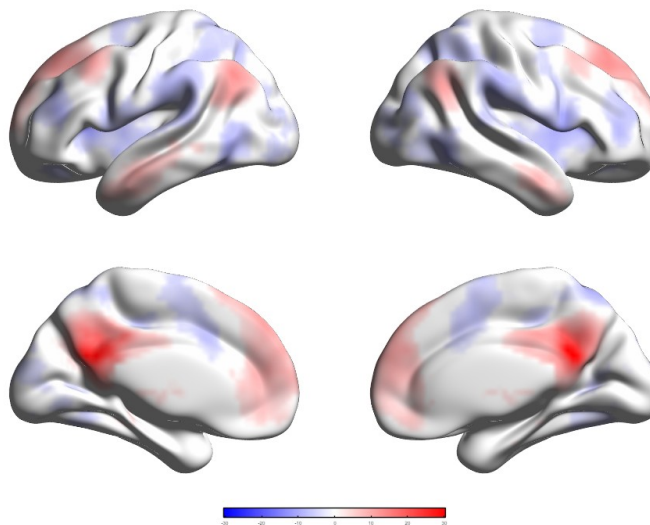


Figure 21. A four-view figure rendered by BrainNetRender from a real-data Scene JSON. MATLAB illustration.

Options to review per run

Option	Current behavior
DataRoot	Defaults to the installed BrainNetViewer folder; set it explicitly for portable project data.
ComputeBackend	auto (default), cpu, gpu; GPU applies only where individual operations support it.
PerformanceMode	quality (default), interactive, auto. For publication runs, explicitly request quality.
Width, Height, Resolution, Units	Specify both dimensions or neither. Otherwise saved App export preferences are used, falling back to Scene image defaults. Explicit dimensions without Units use pixels. DPI and physical units are interpreted by the exporter.
BackgroundColor, ContentType	Explicit choices override saved preferences. JPEG/BMP cannot use transparent none; raster formats reject vector content. Auto/PDF/SVG do not promise pure vectors.
LegacyConfig	Optional trusted legacy EC MAT conversion before rendering; not a full old-GUI replay or a sandbox for untrusted MAT objects.
Planner	Optional reusable <code>brainnet.render.RenderPlanner</code> object; within one MATLAB process it can reuse preparation work when keys match. It does not prove GPU speedup or identical pixels.

Scene loading checks input identities/hashes; a changed or missing file should stop instead of silently substituting data. The Scene's optional camera is restored only when its content signature remains compatible; otherwise the layout default is used. For comparisons across machines/versions, review settings, numeric fingerprints, renderer/version and camera records in the provenance JSON; do not require pixel-identical image hashes across graphics drivers. **The sidecar may contain local paths, titles and sensitive metadata.** Review it and input-resource licenses before publication. Nothing is uploaded automatically. For reusable appearance without input paths, use a visualization preset rather than Scene JSON; see Chapter 12.

Shared tutorial 10 uses your own same-edition Scene and explicit output settings. Review sidecar privacy and cross-platform rendering differences. For Docker deployment, follow the installation chapter and the separate Docker/Web installation guide.

Python command line rendering

Activate the installed Python environment. Create the output parent directory beforehand. Relative Scene assets resolve against `--data-root`, defaulting to the Scene directory. Supply units explicitly: Python can retain saved physical units when dimensions are supplied without `--units`, unlike the MATLAB explicit-dimension pixel fallback.

```
brainnet-render --scene "D:/project-data/scene.json" `
  --data-root "D:/project-data" --output "D:/results/figure.png" `
  --width 800 --height 600 --units Pixels --dpi 200 --transparent
```

Use `--opaque` instead of `--transparent` for an opaque figure. `--overwrite` explicitly replaces the image and sidecar; without it outputs are protected. `--reset-view` ignores the saved camera while retaining settings. Units are Pixels, Millimetres, Centimetres, Inches or Points. There is no `ComputeBackend` option in this Python entry point.

Success prints a JSON result. Handled failures report BrainNet render failed on stderr and exit with status 1. Review warnings and camera-restoration status. Restoration requires compatible content signatures, with default-layout fallback when unsupported or unverified. A loop can call the CLI over explicit inputs; it is not a preprocessing or job-discovery service.

Python CLI provenance records the Scene filename/hash rather than its absolute Scene path. Review actual tool/input metadata before public sharing; this is not blanket anonymization. Retain configuration/input hashes and graphics environment, not just image hashes, for cross-machine comparisons.

17 Migration and troubleshooting

BrainNet Viewer provides MATLAB Desktop, Python Desktop and Docker/Web workflows. A MATLAB Runtime-only package is not supplied. This chapter explains configuration migration, interaction controls and troubleshooting.

Moving from BrainNet Viewer 1.7/1.8

Older workflow	Current route	What to check
GUIDE main window / Load File	BrainNetViewer; Data Sources → Validate and Load	Use the project launcher and an explicit data root. The App asks before staging an outside-root file; Scene JSON does not bundle its input files.
BrainNet_Option.fig and OK/Apply	Right-hand Visualization Settings → shared Apply	Apply commits and renders . Revert drops pending edits; canvas Reset changes only the camera; Restore Defaults replaces display settings after confirmation.
Save/Load Option MAT	Save/Load Preset for display reuse; Import Settings for trusted EC MAT	Presets omit inputs/camera/export-panel preferences. An EC import is a reviewed conversion, not guaranteed lossless reproduction; retain the original MAT and inspect <code>fields_requiring_review</code> .
Save or reopen an analysis	Save Scene / Open Scene JSON	Scene records input references/hashes and applied settings, with optional modern camera/export state. Keep the corresponding data-root structure; verify hashes and spatial alignment.
Redraw / Clear Figure	Render Scene for explicit redraw; Clear Figure for view-only clearing	Clearing the figure preserves inputs/settings; a per-role Clear removes that input.
Get position	In-canvas Inspect	Queries AAL/Brodmann using assumed MNI-like coordinates; it does not register or transform the input. Verify provenance before citing regions.
Merge Mesh / generate coordinates	Merge Surfaces, Convert Surface to NV, Subdivide Surface to NV, Generate Nodes from Atlas	New reviewed files, no silent source overwrite or registration. Conversion is geometry-only; the old <code>.mesh</code> implicit coordinate inversion is not retained.
Movie / print / script	Export Animation, Export Figure, BrainNetRender	Animation is a static-scene camera orbit. Print from an exported image/PDF externally; the modern CLI is not a drop-in <code>BrainNet_MapCfg</code> replacement.

Locate familiar menus and interaction tools

The desktop follows one sequence: load files, configure, render, interact and export. Data Sources selects and validates the inputs; Visualization Settings controls appearance; the central canvas displays the result. Help and About are in the header. Separate tool windows provide mesh and atlas-node operations.

After rendering, use the canvas Rotate, Pan and Zoom tools to adjust individual views. Fit frames the visible geometry; Reset returns to the layout's initial camera. View Direction supplies Left, Superior, Anterior and opposite/other directions for Single View; select the desired direction and render again. MATLAB Inspect replaces the old Get Position datatip workflow and clears its compact on-canvas label when clicked again. Live Demo and Stop Demo control an on-screen camera orbit, while Export Animation makes a separately configured movie or PNG frame sequence.

Native printing and MATLAB Plot Edit are not provided. Export at the required pixel/physical size, then print or annotate in an external application. Use a visualization preset for reusable appearance or a Scene for reproducible inputs and applied settings. BrainNetRender is the MATLAB Scene-based command. Previous 1.x releases remain separately available; do not mix their paths or configuration formats with the 2.0 installation.

Modern Settings schema is 1.22.0. Older Scene/preset/EC values may migrate, but their appearance can change because node/edge threshold endpoints, modules, signed volume color handling, ROI numbering/smoothing and other semantics were clarified or repaired. Recheck science-critical settings with a small dataset that independently varies .node columns 4 and 5 and includes values exactly at thresholds. Do not use visual similarity alone as proof of parity. For source-specific rules see Nodes, Edges, Volume, ROI/Inspect and Colormaps.



Figure 22. The current main window brings Data Sources, render controls, Visualization Settings and export into one workspace. MATLAB illustration.

Current boundaries, not silent promises

- The independent NIfTI reader supports NIfTI-1/2 single files and paired headers/images, including supported gzip variants; **pure Analyze 7.5** without NIfTI magic is rejected. MNI/BIC .obj is not Wavefront OBJ. FreeSurfer .pial, BrainVISA .mesh, MZ3 and GIFTI have narrower reader-specific rules. Loading a geometry does not establish coordinate registration.
- ROI P Value converts supported statistical NIfTI metadata but does not infer whether an input is an atlas or statistic map, correct multiple comparisons or prove cluster-level significance. Fiber-tract rendering, arbitrary Plot Edit, GIF/EPS/EMF export, native print and Runtime-only packaging are not supplied in this public beta. Docker/Web uses one shared session, software rendering and temporary working storage; it is not an authenticated multi-user server.
- Follow edition-specific installation and compatibility requirements. Python Desktop runs in an independent environment; Docker requires a Linux container engine with the matching processor architecture. Untested desktop configurations, including Intel Mac, Linux and additional monitor/scaling combinations, can require further validation. Save portable Scenes and exported files before changing environments.
- GPU acceleration is operation-specific, not a promise that all rendering or video encoding runs on GPU. Complex PDF/SVG scenes may be rasterized; JPEG/BMP have no transparent background. PNG/TIFF transparency is conditional on an actual alpha-bearing export. Hashes and provenance identify data/configuration, not anatomical validity, publication rights or pixel identity across systems.

If something goes wrong

Symptom	First check / safe action
The launcher is not found or a different BrainNet starts	Use the current project root containing BrainNetViewer.m, add that root to the MATLAB path if necessary, and remove conflicting path entries from the session; do not alter the original software folder. See getting started.
External input cannot become a portable Scene	Respond to the App's Stage and Add prompt, or deliberately launch with an appropriate data root. Do not hand-edit absolute paths into portable Scene references. Review copied companions and permissions.
Scene/preset JSON fails to open, or files/hashes do not match	Use Open Scene JSON for Scene files and Load Preset for *.preset.json; these schemas are not interchangeable. Check version/required fields, data-root-relative paths, companion files and the original hashes. Do not suppress validation by changing a hash.
New file selected but the old image remains / nothing exports	Validate and Load, then Apply or explicit Render Scene. Export requires a current render. Review loaded preset settings and retained volume limits; follow the edition-specific preset behavior in Chapter 12.
Wrong side, scale, or atlas region / empty split view	Check the actual coordinate frame, orientation, units, surface-volume alignment and hemisphere separability. The App does not register data. Use a supported whole-surface layout for an inseparable mesh; treat Inspect's AAL/Brodmann result as provisional.
ROI P Value metadata error, unsupported file, or custom color error	Verify NIfTI statistic type/degrees of freedom; do not treat an atlas as a statistic map. Otherwise choose a compatible reader/file or provide every selected ROI ID in the custom color table; see ROI rules.
GPU, transparency or animation export fails	Try the documented CPU backend where relevant; verify output format, real alpha support, video codec availability, even video dimensions, frame/size limits and a new animation folder. Do not overwrite or publish partial output.

For an unresolved reproducible failure, record the edition/build and MATLAB or Python version, data roles and non-sensitive dimensions, exact error, minimal steps, compute backend and output format. Keep participant data and absolute provenance paths out of public reports unless authorized. Verify limitations against the actual package before sharing results.

Python installation and reporting failures

Confirm the Python executable, version and package import path in the intended virtual environment, then run pip check. Do not add source PYTHONPATH or relax pinned dependency constraints to hide installation errors. Capture the complete error and minimal steps.

Check Scene versus preset, asset companions, root and hashes for reopening failures. Check graphic drivers for off-screen rendering errors. Report operating system, edition/build, interpreter or MATLAB version, GPU, monitor resolution/scaling, format and a complete-window screenshot when relevant. Do not publish sensitive inputs, credentials or local paths.

Appendix References and acknowledgments

The references below identify the principal software and atlas sources. Cite the actual atlas release, derivative and analysis method used in your study, rather than treating this general list as a substitute for dataset-specific attribution. The 2013 article is the primary software reference for BrainNet Viewer.

Software and atlas references

- Xia M, Wang J, He Y (2013). BrainNet Viewer: A Network Visualization Tool for Human Brain Connectomics. PLoS ONE 8: e68910.
- Brodmann K (1909). Vergleichende Lokalisationslehre der Großhirnrinde. Barth, Leipzig.
- Dosenbach NUF, Nardos B, Cohen AL, et al. (2010). Prediction of individual brain maturity using fMRI. Science 329: 1358–1361.
- Fair DA, Cohen AL, Power JD, et al. (2009). Functional brain networks develop from a local to distributed organization. PLoS Computational Biology 5: e1000381.
- Power JD, Cohen AL, Nelson SM, et al. (2011). Functional network organization of the human brain. Neuron 72: 665–678.
- Shattuck DW, Mirza M, Adisetiyo V, et al. (2008). Construction of a 3D probabilistic atlas of human cortical structures. NeuroImage 39: 1064–1080.
- Smith SM, Jenkinson M, Woolrich MW, et al. (2004). Advances in functional and structural MR image analysis and implementation as FSL. NeuroImage 23 Suppl 1: S208–S219.
- Tzourio-Mazoyer N, Landeau B, Papathanassiou D, et al. (2002). Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain. NeuroImage 15: 273–289.

Distributed atlas example sources

- AAL3v2: <https://www.gin.cnrs.fr/fr/outils/aal/>
- Glasser360: https://afni.nimh.nih.gov/pub/dist/atlas/MNI_HCP/MNI_Glasser_HCP_2019_v1.0/
- Power264: <https://github.com/brainspaces/power264>
- Shen278: https://www.nitrc.org/frs/download.php/5785/shenetal_neuroimage2013_funcatlas.zip
- Schaefer100/200/400Yeo7: https://github.com/ThomasYeoLab/CBIG/tree/master/stable_projects/brain_parcellation/Schaefer2018_LocalGlobal/Parcellations/MNI

Use the paired example files with their unchanged node order. Cite the actual atlas source and derivative used; example connectivity is for visualization, not a normative connectivity reference.

Acknowledgments

We acknowledge Prof. Alan C. Evans for the ICBM152 surface and the BrainVISA contributors for Colin27 geometry. Distributed templates, atlases, examples and derivatives retain their file-specific provider credits and notices.

BrainNet Viewer acknowledges assistance from Gaolang Gong, Ni Shu, Zhilei Xu, Jiaying Zhang, Yapei Xie, Alan C. Evans, Xindi Wang, Jinhui Wang, Chaogan Yan, Zhengjia Dai, Teng Xie, Qixiang Lin, Miao Cao and Jin Liu.

Contact Mingrui Xia at mxia@bnu.edu.cn or mingruixia@gmail.com.

Appendix Shared tutorials

These thirteen tutorials combine MATLAB, Python Desktop and Docker/Web workflows. Follow the edition-labelled instructions for installation, inputs, settings and output. Links are relative to this manual: retain the repository documentation structure when moving the file.

[Shared tutorial index](#)

[MATLAB and Python topic coverage](#)

[01 Installation and first launch](#)

[02 First surface visualization](#)

[03 Scenes and presets](#)

[04 Nodes modules and connectivity](#)

[05 Vertex maps boundaries and Inspect](#)

[06 Statistical volume to surface mapping](#)

[07 Bilateral AAL insula ROIs](#)

[08 Layouts and figure export](#)

[09 Custom colors and presets](#)

[10 Command line Scene replay](#)

[11 Surface and atlas tools](#)

[12 Camera animation](#)

[13 Performance and reproducibility](#)

Illustrations identify their edition; the shared lessons use the same scientific inputs with edition-specific controls. No prebuilt Scene files are supplied. Save your own Scene during the exercises.