Moving Towards Reproducible and Programmatic Generation of Neuroimaging Visualizations

Sidhant Chopra *
Department of Psychology
Yale University
CT, USA, 06511
sidhant.chopra@yale.com

Loïc Labache
Department of Psychology
Yale University
CT, USA, 06511

Elvisha Dhamala
Department of Psychology
Yale University
CT, USA, 06511

Edwina R Orchard
Department of Psychology
Yale University
CT, USA, 06511

Avram Holmes
Department of Psychology
Yale University
CT, USA, 06511

September 10, 2022

Neuroimaging visualization forms the centerpiece of interpretation and communication of scientific results, as well as data quality control. Often, these images and figures are produced by manually changing settings on Graphical User Interfaces (GUI). There now exist many well-documented code-based brain visualization tools that allow users to programmatically generate publication-ready figures directly within R, Python and MATLAB environments. Compared to figures generated using GUIs, programmatic figures are more replicable, flexible, interactive, and integrated with the scientific process. Here we outline the advantages of using programmatic neuroimaging visualization tools, provide a list of available tools across programming environments and include examples of voxel, vertex, region-of-interest and edge-level visualizations.

***K*eywords** Neuroimaging visualization · Reproducibility · Programmatic figures · Open Science · R · Python · Brain Visualization

1 Introduction

The visualization of neuroimaging data is one of the primary ways in which we evaluate data quality, interpret results, and communicate findings. These visualizations are commonly produced using graphical user interface-based (GUI) tools where individual images are opened and, within each instance, display settings are manually changed until the desired output is reached. In large part, the choice to use GUI-based software has been driven by a perception of convenience, flexibility, and accessibility. However, there now exist many

*Corresponding author

code-based software packages which are well-documented and often do not require high-level knowledge of programming, making them more accessible to the neuroimaging community (Table 1). These tools are flexible and allow for the generation of reproducible, high-quality, and publication-ready brain visualizations in only a few lines of code (Figure 1), especially within the R, Python and MATLAB environments. Here we review the rational for the wide-spread adoption of code-generated visualizations by highlighting major advantages in replicability, flexibility, and integration over GUI based tools. We also providing didactic examples of visualizations and associated code as a point of reference.

2 Replicability

In recent years, there have been multiple large-scale efforts empirically demonstrating the lack of reproducibility of findings from neuroimaging data (Poldrack et al., 2017). One common solution proposed for achieving robust and reliable discoveries has been to encourage scientific output which can be transparently evaluated and independently replicated. In practice, this typically entails openly sharing detailed methods, materials, code, and data. While there is a trend towards increasing transparency and code sharing of neuroimaging analyses, the sharing of code used to generate figures such as brain renderings and spatial maps has been relatively neglected. This gap in reproducibility is partly driven by the fact that brain figures are often created using a manual process that involves tinkering with sliders, buttons, and overlays on a GUI, concluding with a screenshot and sometimes beautification in image processing software like Illustrator, Photoshop or Inkscape. Such a process inherently makes neuroimaging visualizations difficult, if not impossible to replicate at times, even by the authors themselves.

Visualization scripts should reflect a core feature of open science. Given that brain figures regularly form the centerpiece of interpretation within papers, conference presentations, or news reports, making sure they can be reliably regenerated is crucial for knowledge generation and dissemination. By writing and sharing code used to generate brain visualizations, a direct and traceable link is established between the underlying data and the corresponding scientific figure. While this code doesn't necessarily reflect the validity or accuracy of the scientific finding, it allows for reproducibility, instilling transparency and robustness, while demonstrating a desire to further scientific knowledge. Some even consider publishing figures which cannot be replicated as closer to advertising, rather than science (Steel, 2013).

While some GUI-based tools have historically offered command-line access to generate replicable visualizations, they can lack both the flexibility to easily generate publication ready figures and the benefits, such as iteration, provided by your preferred programming environment. Likewise, other GUI-based tools offer replicability in the form of automatically generated batch scripts or in-built terminals, which are often idiosyncratic and lack documentation to make them easily usable or reproducible by those not familiar with the specific software.

3 Flexibility

Being able to exactly replicate your figures via code has marked advantages beyond open science practices. In particular, the ability to reprogram inputs (such as statistical maps) and settings (such as color schemes, thresholds, and visual orientations) can streamline your entire scientific workflow. Changing inputs and settings via code allows for the easy production of multiple figures, such as those resulting from multiple analyses which require similar visualizations. A simple for-loop or copying and pasting the code with altered input and/or settings-of-interest can be a powerful method for exploring visualization options or rapidly creating multi-panel figures. Likewise, an arduous request from a reviewer or collaborator to alter the image processing or analysis becomes less of a burden when the associated figures can be re-generated with a few lines of code, as opposed to re-pasting and re-illustrating them manually. Having a code-base with

modifiable inputs can mean that the generation of visualizations requires less time, energy and effort than image and instance specific GUI-based generation. This also makes it easier to generate consistent figures across subsequent projects. Keep in mind that the gains of writing code for your figures are cumulative, and in addition to improving your programming, you start to build a code-base for figure generation that you can continue to reuse and share throughout your scientific career.

Precise controls via code over visualization settings, such as color schemes, legend placement and camera angles, can provide you with much greater flexibility over visualizations. Nonetheless, part of the appeal of GUI-based tools is that the presets for such settings can provide a useful starting point and reduce the decision burden on novice users. However, similar presets are often available in the form of default settings across most code-based packages, negating the need for the user to manually enter each and every choice required for creating an image. Most code-based tools also come with documentation, with R-packages on the *CRAN* or *Neuroconductor* (Muschelli et al., 2019) repositories strictly requiring detailed guidance. Recent packages have started to include detailed beginner-friendly documentation in GitHub repositories, or even entire papers (e.g., Pham, Muschelli, & Mejia, 2022; Mowinckel & Vidal-Piñeiro, 2020; Huntenburg et at., 2017; Schäfer & Ecker, 2020) which provide examples of figures that can be used as starting points or templates for new users. As the popularity of code-sharing for figure increases, there will be a cornucopia of templates that can be used as the basis for new figures.

While brain visualizations are often thought of as the end results of analyses, they also form a vital part of quality control for imaging data. Tools to automatically detect artefacts, de-noise the data and generate derivatives are becoming more robust, but we are not yet at the stage where visualizing the data during processing is no longer necessary. Nonetheless, when working with large datasets such as Human Connectome Project (Van Essen et al., 2013) or UK BioBank (Sudlow et al., 2015), it is unfeasible to use traditional GUI-based tools to visually examine the data. The time it takes to open a single file and achieve the desired visualization settings vastly compounds when working with large datasets. Knowing how to programmatically generate brain visualizations can allow you to iterate your visualization code over each image of a large datasets making checking the quality of each data processing step achievable. The visual outputs of each iteration can be complied into accessible documents which can be easily scrolled, with more advanced usage allowing for the creation of interactive HTML reports, similar to those created by standardized data processing tools like *fmriprep* (Esteban et al., 2019). Increasing capacity to conduct visual quality control on larger datasets will improve the identification of processing errors and result in more reliable and valid findings.

4 Integrative and Interactive Reporting

Often in neuroimaging studies, programming languages such as R, Python and MATLAB are used for stastical analysis and generating non-brain figures, but the brain figures are outsourced to separate GUI's based tools such as *FSLeyes*, *Freeview* or *ITK-snap*. Switching from your analysis environment to a GUI-based visualization process can be a cumbersome deviation from the scientific workflow. This can make debugging errors more difficult, as you have to regularly switch program to visually examine the results of any modifications or adjustments to prior analyses. Using the brain visualization tools that already exist within your chosen programming environment can provide instant visual feedback on the impact of modifications to processing or analysis.

Increasingly popular software such as *R Markdown*, *Quarto* and *Jupyter Notebook* allow for the mixing of prose and code in a single script, resulting in fully reproducible and publication ready papers. By using code-based tools available within your preferred environment, brain visualizations can be directly integrated and embedded within a paper or report. For instance, a fully reproducible version of the current paper can be found on GitHub. Some journals that publish neuroimaging studies are moving towards allowing the

submission of reproducible manuscripts, including reproducible figures (e.g. *e-Life*, *Aperture Neuro*), with other journals like *F1000Research* and *GigaScience* even allowing on-demand re-running of code linked to the associated article using 'compute capsules' from the cloud-based platform Code Ocean (Code Ocean, 2021).

Neuroimaging data are often spatially 3D and can have multiple time points, adding a 4th dimension (e.g., functional imaging data). Thus, communicating findings or evaluating quality using static 2D slices is challenging, and may not be the best representation of the data, or the associted interpretations. While well-curated 3D renderings can help with spatial localisation (see Madan, 2015; Pernet & Madan, 2019), in the end, static images can only provide an incomplete representation of the data, and forces researchers to choose the "best" angle or slice to show, which often involves compromising one result to emphasize another. An added advantage of some of the code-based tools is that you can generate 'rich' media like interactive figures or animations, which allow users to zoom, rotate and scroll through slices. Interacting with a figure in this way can improve scientific communication of findings. Linking to or even embedding these videos or interactive figures in papers can greatly enhance the communication of findings and make your paper more engaging for the reader. Such rich brain visualizations lend themselves to being embedded or shared on science communication mediums beyond academic papers - such as presentations, websites and social media - all of which can promote the communication of your research with peers and reach larger audiences (Li and Xie, 2020). This last point is becoming increasingly salient as promoting science on social media has become a core medium for spreading discoveries, science communication to the public, and even a primary avenue for employment opportunities for early-career researchers (Baker, 2015; Lee, 2019). Overall, public engagement is one of the cornerstones of science, and the images we create are at the center of the process.

Table 1. Examples of code-based neuroimaging visualizations tools that can be accessed directly within R, MATLAB and Python environments.

	Voxel	Vertex	ROI	Edge	Streamlines
R					
ANTsR	+	+	+		
brainconn				+	
brainR	+		+		
ciftitools	+	+	+*		
fsbrain	+	+	+*		
ggseg			+		
neurobase	+				
oro.nifti	+				
Python					
ANTsPy	+	+	+		
brainiak	+				
Brainplotlib		+	+*		
Brainspace/surfplot		+	+*		
DIPY	+				+
ENIGMA TOOLBOX			+		
FSLeyes	+	+	+		+
ggseg			+		
graphpype				+	

	Voxel	Vertex	ROI	Edge	Streamlines
MMVT		+	+	+	
MNE	+	+	+		
mrivis	+				
NaNSlice	+				
netneurotools		+	+*		
netplotbrain			+	+	
nilearn	+	+	+	+	
niwidget	+	+			+
Pycortex	+	+	+*		
pySurfer		+	+*		
surfice	+	+	+	+	+
Visbrain	+	+	+	+	
MATLAB					
BrainNetViewer	+		+	+	
Brainspace		+	+*		
Brainstorm	+	+	+		
bspmview	+		+		
CandlabCore	+		+		
ECoG/fMRI Vis toolbox		+	+*		
ENIGMA TOOLBOX			+		
FieldTrip	+	+			
Lead-DBS	+		+		
mni2fs		+			
mrtools	+	+	+*		
plotSurfaceROI-Boundary		+	+*		
Vistasoft	+	+	+		+

Note: The tools listed contain functionality required to generate (at least close-to) publication-ready neuroimaging figures via user-entered code within R, MATLAB and Python environments. This list does not include cross-platform general purpose visualization software.
* Cortex only

5 Examples of Neuroimaging Visualization Packages available in *R and Python*

The following two figures provide examples of voxel, vertex, ROI and edge-level figures generated within R (Figure 1) and Python (Figure 2) using open source, well documented and beginner-friendly packages. These are not an exhaustive representation of packages available for visualizing brain data in R and Python (See Table 1), rather, the figures aim to give the reader a sense of the available options, and an entry point to using them. All code used to compile the figures, as well as the contents of each panel are provided an accompanying online repository.

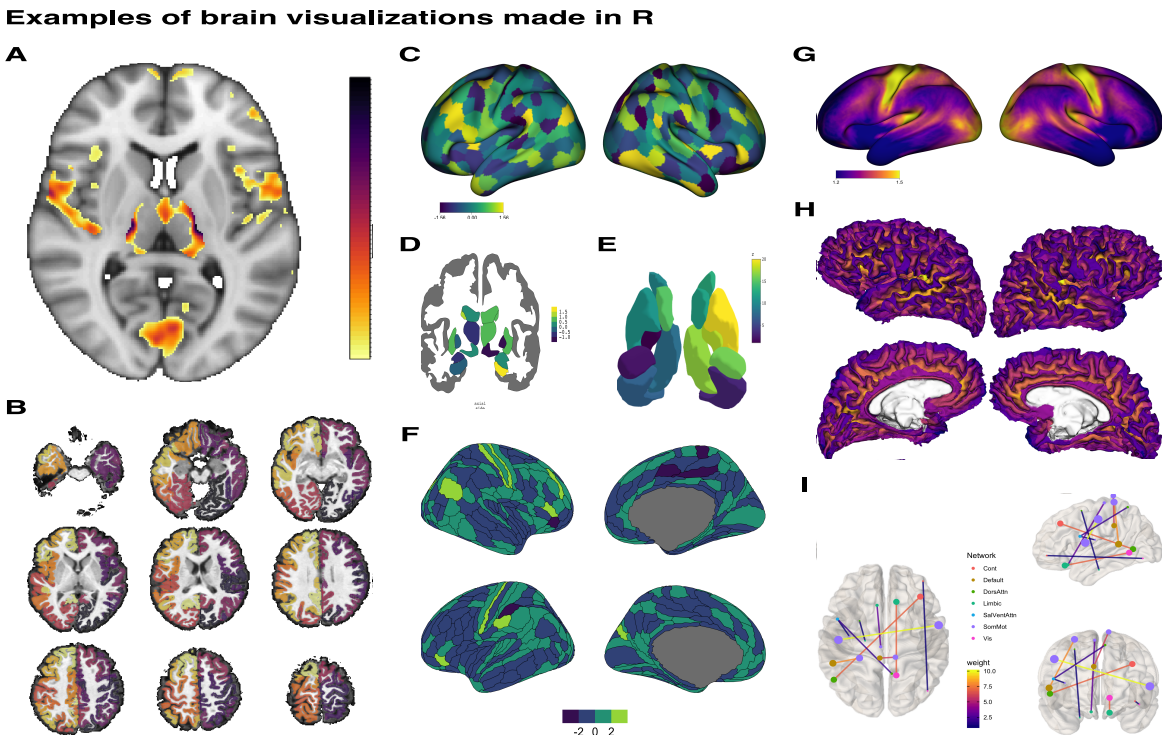

Figure 1. Examples of brain imaging visualizations made using R. A) Voxel-level statistical map thresholded and overlaid over a T1-weighted template image, with a single axial slice shown. Made using the the `ortho2` function from the `neurobase` package. B) A voxel-level cortical parcellation overlaid on a individual T1-weighted image, shown in 9-slice axial orientation. Made using the `overlay` function from the `neurobase` package. C) A CIFTI format surface ROI atlas with a corresponding statistic assigned to each region, with both hemispheres displayed on a inflated template surface in lateral view. Made using the `view_xifti_surface` from the `ciftiTools` package. D) A coronal cross-sectional rendering of subcortical structures where a value has been assigned to each region. Made using the `aseg` atlas from the `ggseg` package. E) A 3D rendering of 9 bilateral subcortical regions where a value has been assigned to each region. Made using the `aseg` atlas from the `ggseg3d` package. F) Medial and lateral views of a ROI atlas displayed on inflated cortical surface where a value has been assigned to each region. Made using the `glasser` atlas from the `ggsegGlasser` package, which was plotted using `ggseg`. G) Lateral view of a CIFTI format vertex-level data displayed on a inflated template surface. Made using the `view_xifti_surface` function from the `ciftiTools` package. H) Medial and lateral views of vertex-level data displayed on a individuals white matter surface. Made using the `vis.subject.morph.standard` function from the `fsbrain` package. I) A weighted and undirected graph plotted on top, left and front views of a schematic outline of a brain in MNI coordinate space. Made using the `brainconn` function from the `brainconn` package. All code used to compile this figure, as well as the contents of each panel are provided in an accompanying online repository.

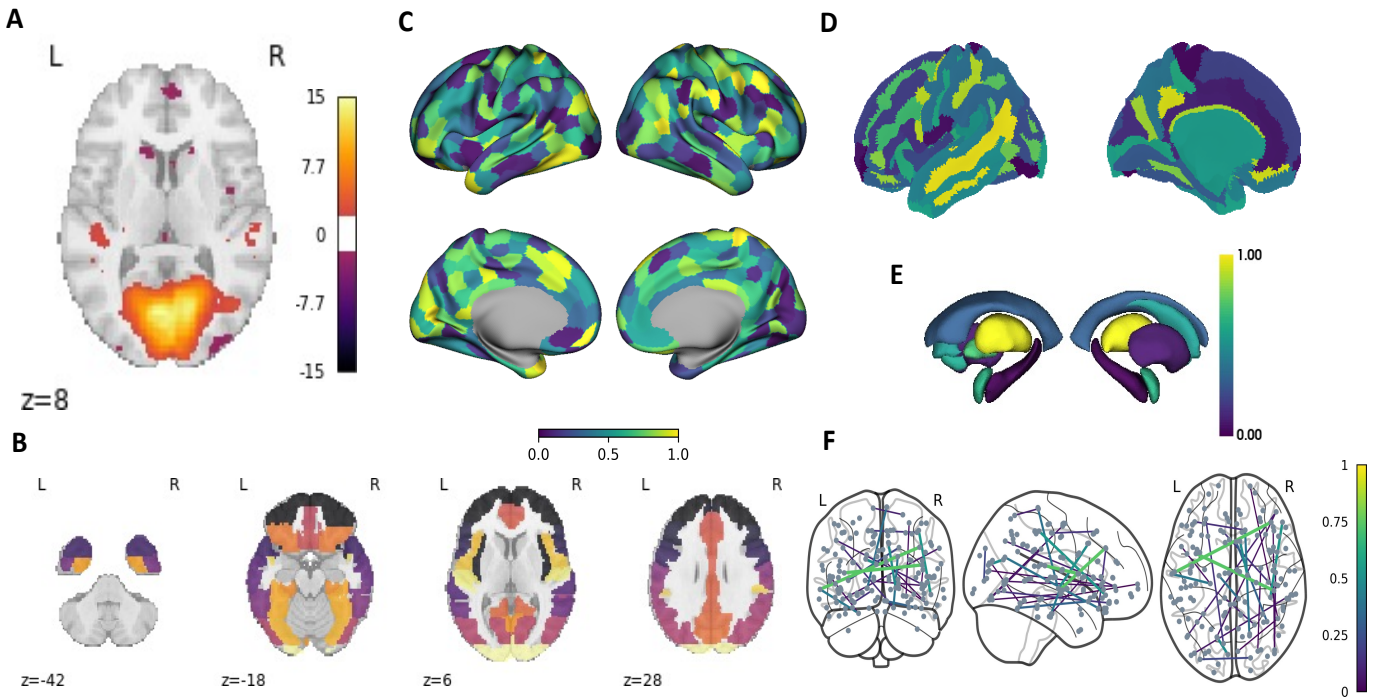

Figure 2. Examples of brain imaging visualizations made using Python. A) Voxel-based statistical map thresholded and overlaid over a T1-weighted template data, with a single axial slice shown. Made using the `plotting.plot_stat_map` function from `nilearn`. B) Voxel-level cortical parcellation overlaid on T1-weighted MRI data shown in four axial slices. Made using `plotting.plot_roi` function from `nilearn`. C) Medial and lateral views of an cortex-wide ROI atlas, displayed on an inflated template surface where a statistical value has been randomly assigned to each ROI. Made using `Plot` function from `surfplot`. D) Lateral and medial views of vertex-level data displayed in a 3D rendering on an individual's white matter surface. Made using plotting.view_surf function in from `nilearn`. 3D rendering of 16 subcortical structures from the the Deiskan-Killiany atlas, where a statistical value has been randomly assigned to each region. Made using the `plot_subcortical` function from the `ENIGMA TOOLBOX`. A weighted and undirected graph plotted on front, right, and top views of a schematic outline of a brain in MNI coordinate space. Made using plotting.plot_connectome function in `nilearn`. All code used to generate the contents of each panel, and compile this figure are provided in an accompanying online repository.

6 Limitations and Functionality Gaps

While many code-based tools are well documented and do not require a strong knowledge of programming, there can still be a steeper learning curve for new users, compared to using a GUI. This is especially true for the purpose of a publication-ready figure, were fine adjustments to visual auxiliary such as legend placement, font size and multi-panel figure positioning are needed. While most code-based tools offer some control over these finer steps, there are differences between them in feature availability and usability. Relatedly, while some interactive image viewers can be opened within an integrated development environment like R-Studio (e.g. Muschelli, 2016), for quick and interactive viewing of single images, GUI tool can be faster and more practical.

Often cerebellar and brain-stem regions are not well represented in software (e.g. Figure 1), potentially mirroring the cortico-centric sentiment that has prevailed in human neuroimaging research (Chin, Chang, & Holmes, 2022). Likewise, custom non-cortical atlases such as non-standard subcortical atlas schemes are not yet straightforward, and usually require multiple functions and packages to visualize. Finally, some neuroimaging derived data types, such as streamlines resulting from DWI-based tractography, are still not well represented in code-based visualization tools.

As can be seen in Table 1, there are usually multiple packages within each programming environment which are capable of visualizing each data type. While this provide choice for advanced users, it can also lead to confusion for novice users who may not be familiar with the nuanced differences between tools. Future work should continue to consolidate brain visualization methods into unified beginner-friendly code-based tools which are capable of plotting multiple data types.

7 References

Poldrack, R. A., Baker, C. I., Durnez, J., Gorgolewski, K. J., Matthews, P. M., Munafò, M. R., ... & Yarkoni, T. (2017). Scanning the horizon: towards transparent and reproducible neuroimaging research. Nature reviews neuroscience, 18(2), 115-126.

Steel, G. (2013, September 3). Publishing research without data is simply advertising, not science. Open Knowledge Foundation. `https://blog.okfn.org/2013/09/03/publishing-research-without-data-is-simply-advertising-not-science/`

The Comprehensive R Archive Network. (n.d.). Retrieved April 21, 2022, from `https://cran.r-project.org/`

Muschelli, J., Gherman, A., Fortin, J. P., Avants, B., Whitcher, B., Clayden, J. D., ... & Crainiceanu, C. M. (2019). Neuroconductor: an R platform for medical imaging analysis. *Biostatistics*, *20*(2), 218-239.

Pham, D., Muschelli, J., & Mejia, A. (2022). ciftiTools: A package for reading, writing, visualizing, and manipulating CIFTI files in R. *NeuroImage*, 118877.

Mowinckel, A. M., & Vidal-Piñeiro, D. (2020). Visualization of brain statistics with R packages ggseg and ggseg3d. *Advances in Methods and Practices in Psychological Science*, *3*(4), 466-483.

Huntenburg, J., Abraham, A., Loula, J., Liem, F., Dadi, K., & Varoquaux, G. (2017). Loading and plotting of cortical surface representations in Nilearn. *Research Ideas and Outcomes*, *3*, e12342.

Wickham H. (2016). *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York. ISBN 978-3-319-24277-4, `https://ggplot2.tidyverse.org`.

Muschelli, J. (2016). *papayar*. GitHub repository. `https://github.com/muschellij2/papayar`

David C. Van Essen, Stephen M. Smith, Deanna M. Barch, Timothy E.J. Behrens, Essa Yacoub, Kamil Ugurbil, for the WU-Minn HCP Consortium. (2013). The WU-Minn Human Connectome Project: An overview. NeuroImage 80(2013):62-79.

Sudlow, C., Gallacher, J., Allen, N., Beral, V., Burton, P., Danesh, J., ... & Collins, R. (2015). UK biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. PLoS medicine, 12(3), e1001779.

Esteban, O., Markiewicz, C. J., Blair, R. W., Moodie, C. A., Isik, A. I., Erramuzpe, A., ... & Gorgolewski, K. J. (2019). fMRIPrep: a robust preprocessing pipeline for functional MRI. Nature methods, 16(1), 111-116.

Pernet, C., & Madan, C. R. (2019). Data visualization for inference in tomographic brain imaging.

Code Ocean. (2021, October 27). Computational Research Platform on. Retrieved April 21, 2022, from `https://codeocean.com/`

Madan, C. R. (2015). Creating 3D visualizations of MRI data: A brief guide. F1000Research, 4.

Baker M. (2015). Social media: A network boost. Nature 518: 263–265. 10.1038/nj7538-263a

Lee J.-S. (2019). How to use Twitter to further your research career. Nature. 10.1038/d41586-019-00535-w

Li, Y., & Xie, Y. (2020). Is a picture worth a thousand words? An empirical study of image content and social media engagement. Journal of Marketing Research, 57(1), 1-19.

Schäfer, T., & Ecker, C. (2020). fsbrain: an R package for the visualization of structural neuroimaging data. bioRxiv.

Chin, R., Chang, S. W., & Holmes, A. J. (2022). Beyond cortex: The evolution of the human brain. *Psychological Review*.