UNIVERSITY OF GOTHENBURG

SAHLGRENSKA ACADEMY

Region-based analysis of magnetic resonance brain images: integrating statistical analysis and visualization.

Yu-Ping Hsu

Essay/Thesis:	30 hp
Program and/or course:	RFA900 V21 självständigt arbete i radiofysik
Level:	Second Cycle
Term/year:	Spring 2021
Supervisor:	Rolf A. Heckemann
Examiner:	Magnus Båth
Report no:	

Abstract

Essay/Thesis:	30 hp
Program and/or course:	RFA900 V21 självständigt arbete i radiofysik
Level:	Second Cycle
Term/year:	Spring 2021
Supervisor:	Rolf A. Heckemann
Examiner:	Magnus Båth
Report no:	
Keywords:	Neuroimage, Brain segmentation, Statistical analysis, and visualization

Purpose:	The purpose of the project was to develop a processing toolchain that takes MR brain images and their anatomical segmentations as input and produces image visualizations, data visualizations, and statistical analyses in a unified manner for the users who work with neuroimaging and quantitative brain anatomy, including pathological anatomy. A secondary aim was to package and release the software publicly as open source.
Theory:	The purpose of data analysis is to inspect, clean, transform and model raw data in order to discover salient information which can be used to build knowledge. Image visualization is defined as the process of converting data, such as pixel values or voxel values, into two- or three-dimensional graphical representations. Data visualization presents data in a graphical format. In the field of quantitative brain anatomy, data analysis, image visualization, and data visualization take place in separate systems. Integrating data analysis, image visualization, and data visualization in a platform makes it possible for users to analyze region-based MR brain images more efficiently.
Method:	To develop a toolchain that meets practitioner's needs, another master student and my supervisor were consulted. Functions were developed iteratively based on feedback. By combining the developed functions, a workflow for region-based MRI brain image analysis was designed. Documentation and examples for all functions were written using R Markdown. Flowcharts for developed functions were drawn.
Result:	An R package implementing image visualization, data visualization and statistical analysis functions was developed, including functions and a web app for region-based analysis of MR brain images. Functions were combined into a toolchain; data analysis and image visualization were integrated on the same platform. A workflow for region-based MR brain images analysis was designed.
Conclusion:	The workflow design, programmed functions, and unified analysis platform for MR brain images are unique and novel contributions achieved in this master's project.

Sammanfattning

Dataanalys innebär att inspektera, rengöra, transformera och modellera rådata för att upptäcka viktig information som kan användas för att bygga kunskap och stödja beslutsfattande. Bildvisualisering innebär att konvertera data som pixelvärden eller voxelvärden till två- eller tre-dimensionell grafisk representation. Datavisualisering består i att presentera data i ett grafiskt format. Inom området av kvantitativ hjärnanatomi sker för närvarande dataanalys, bildvisualisering och datavisualisering vanligtvis i separata system. Genom att integrera statistisk analys och bildvisualisering i en plattform blir det möjligt för användare att analysera regionbaserade MR-hjärnbilder mer effektivt.

Syftet med projektet var att utveckla en verktygskedja som använder MR-hjärnbilder och deras anatomiska segmenteringar som indata och producerar bildvisualiseringar, datavisualiseringar och statistiska analyser i en och samma plattform för användare som arbetar med Neuroradiologi och kvantitativ hjärnanatomi, inklusive patologisk anatomi. Ett sekundärt mål var att packa funktionerna och publicera dem som öppen programvara.

Under utvecklingen av R-paketet som innehåller funktionerna för bildvisualisering, datavisualisering och statistisk analys var en annan mastersstudent och min handledare mina kravställare. Grundläggande modellering upprättades genom att diskutera med ovanstående personer för att fastställa deras behov. Funktioner utvecklades iterativt baserat på återkoppling från ovan nämnda personer. Genom att kombinera de utvecklade funktionerna designades ett arbetsflöde för analys av regionbaserade MR-hjärnbilder. Dokumentation och exempel för alla funktioner skrevs med R Markdown. Flödesscheman för de utvecklade funktionerna ritades.

Ett R-paket och en webapplikation för regionbaserad analys av MR-hjärnbilder utvecklades. Utvecklade funktioner kombinerades till en verktygskedja; dataanalys och bildvisualisering integrerades i en och samma plattform. Ett arbetsflöde för regionbaserad MR-hjärnbildsanalys designades.

Designen av arbetsflödet, den utvecklade programvaran och den enhetliga analysplattformen för MR-hjärnbilder är ett nytt och unikt bidrag som uppnåtts i detta självständiga arbete.

Table of contents

1. Introduction

Thanks to the rapid progress of science and technology in the field of medical imaging and the increased interest in machine learning, more medical images are now available in the medical field. Magnetic resonance imaging (MRI) is the best-suited modality for structural brain imaging because the images show exquisite contrast between different soft tissues, enabling the distinction between physically and functionally distinct brain regions.

The interpretation of images currently relies on experts, visual review based on their professional experience. However, with the increase in the number of medical images, the workload of radiologists continues to increase. Fully automatic image interpretation will remain unattainable in the near future. A more direct and promising method to support radiologists is computer assisted analysis of MR brain images in parallel to visual review. Quantitative measures based on the automatic analysis, for example volumetry of anatomical regions, can then aid the radiologist's interpretation.

1.1. Data analysis

The purpose of data analysis is to inspect, clean, transform and model raw data in order to discover salient information which can be used to build knowledge [1]. In this project, MR brain images and their segmentations have been analyzed using the approaches listed in this section.

1.1.1. Volumetric analysis

The intention behind volumetric analysis is to detect and quantify characteristic features of brain diseases in an objective and automatic fashion [2]. For example, many studies have shown that the hippocampal volume of patients with Alzheimer's disease is significantly reduced [3].

1.1.2. Z-score

By itself, the volume measurement of an anatomical region in an individual is not meaningful. Only by comparison with a normative reference does it become informative. Biological and measurement variability need to be taken into account; therefore, the normative reference should be derived from a reasonably large sample of measurements from individuals with known characteristics (for example healthy adults). The sample will follow a certain distribution, which in the case of brain volumetry is approximately normal in most cases. To carry out the comparison, the central tendency of this distribution needs to be modelled, as well as the dispersion. A suitable measure (assuming normality) for the central tendency is the arithmetic mean μ; for the dispersion, it is the standard deviation σ.

A raw score x, which in this case is the volume of a brain region, is converted into a standard score (Z-score) using Equation (1).

$$\mathrm{Z} = \frac{x - \mu}{\sigma} \tag{1}$$

Intuitively, the Z-score represents whether a measurement is normal (between –2 and 2) or abnormal (deviating more strongly from the reference).

1.1.3. Asymmetry

In healthy adults, the right hemisphere of the brain is slightly heavier than the left hemisphere, but the left hemisphere has more grey matter than the right hemisphere. Normal asymmetry characteristically follows a pattern described as torque [4]. In the study of obsessive-compulsive disorder in the child/adolescent group, there is a significant difference in the volume asymmetry of the thalamus and the pale bulb [5].

Brain asymmetry index A_r is calculated using Equation (2).

$$A_r = \frac{2|V_R - V_L|}{V_R + V_L} \tag{2}$$

where:

V_R is the volume of the right brain.

V_L is the volume of the left brain.

A higher asymmetry index indicates a higher degree of asymmetry in the specific structure.

1.1.4. Similarity

Dice coefficients and Jaccard indices can be used to measure the similarity between two segmentation labels. The Jaccard index, also known as the Jaccard similarity coefficient, is defined as the size of the intersection divided by the size of the union of the sample sets. Its value range is between 0 and 1. A Jaccard index of 0.7 means two samples are 70% similar. Assuming that the target area predicted by model 1 is A and the target area predicted by model 2 is B, then the Jaccard index can be calculated using the Equation (3).

$$\frac{\text{The number of pixels in the intersection of A and B}}{\text{Number of pixels of A } + \text{Number of pixels of B} - \text{The number of pixels in the overlapping area of A and B}} \tag{3}$$

The value range for Dice coefficient is also between 0 and 1. A Dice coefficient value of 1 means segmentation in the two the samples are a perfect match. Dice coefficient can be calculated using the Equation (4).

$$\frac{2 \times \text{The number of pixels in the intersection of A and B}}{\text{Number of pixels of A } + \text{Number of pixels of B}} \tag{4}$$

1.2. Image visualization

Image visualization is defined as the process of converting data, such as pixel values, voxel values, and numerical measurements into two- or three-dimensional graphical representations. A typical visualization of the three-dimensional MR data shows a two-dimensional section in one of three standard orientations (axial (transverse), coronal, or sagittal). To appreciate the third dimension, the reader scrolls through a stack of such images. Most software programs for viewing also offer various levels of advanced visualization, for example selecting a non-standard orientation, thick-slab rendering (viewing a limited number of sections at once), volumetric surface rendering (generating a three-dimensional impression on a two-dimensional display), and animated modes.

1.2.1. Viewing segmentations

When an image has been segmented, the resulting label image can be viewed using the same display modes as the source image. Combined viewing modes are, however, more suitable for understanding and evaluating segmentations in the spatial context of the source image. A simple combined viewing mode superimposes the segmentation labels on the source image. Users will turn the overlay on and off repeatedly to appreciate the spatial relationship between the source image and the segmentation labels. More advanced software allows the user to assign a transparency value to the overlay, regulated via a graphical slider element.

An even more useful mode for viewing segmentations (depending on user preference) can be a contour display. Instead of superimposing a whole label, only its contour or outline are overlaid. This mode enables assessment of, for example, accuracy and level of detail with which the structure in question is delineated. With the help of segmentation contours, users will be able to see where the segmenter or the algorithm placed the boundaries of the region in question within the anatomical context, and whether they agree with the anatomical interpretation that this placement represents. Figure 1 shows an example of segmentation contour.

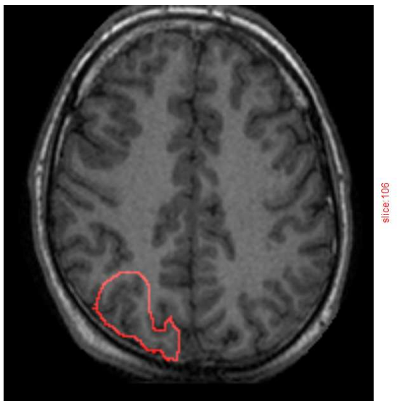


Figure 1: A contour for OL lateral remainder occipital lobe R.

1.2.2. Segmentation comparison

A viewing task of special interest in this work is comparing pairs of segmentations of the same region. A user may want to assess the quality of an automatically generated label with reference to a trusted reference label. If, for example, one segmentation label is larger or smaller than the other, the user should be able to tell whether the over- or under-estimation is localized or whether the boundary disagreement occurs around the region as a whole. To my knowledge, no viewing software currently offers features to support this type of task.

1.3. Data visualization

Data visualization presents data in a graphical format [6]. A data visualization method that takes a prominent role in this project is the violin plot. A violin plot shows the data distribution and its probability density. The width of the curve corresponds to the density of data points in each region. The

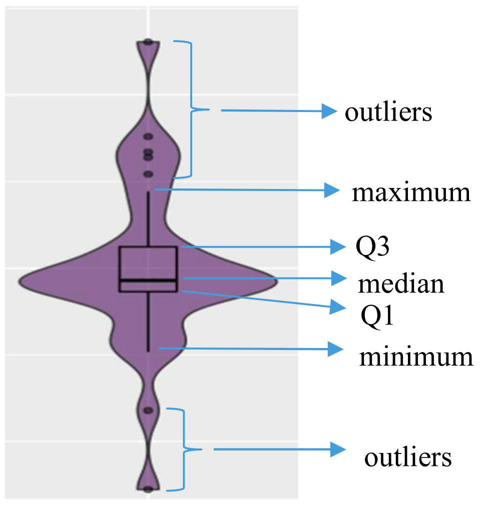


Figure 2: Violin plot with a boxplot

violin plot is mainly used to show the distribution shape of data. A violin plot can be advantageously combined with box plot. The example in Figure 2 shows the maximum, minimum, median, upper and lower quartiles of a dataset, as well as the outliers. The two ends of the box are the first quartile (covering 25% of the robust range of the data) (Q1) and the third quartile (covering 75% of the robust range) (Q3). The middle line of the box is the median, which represents the boundary between the top and bottom 50% of the robust range. The thin black lines ("whiskers") extending from the two ends of the box represent robust range of the data. The upper end point of the solid line above the box is the robust maximum value, and the solid line under the box shape is the robust minimum value. Circles outside the solid line indicate outliers.

At present, data analysis, image visualization, and data visualization take place in separate systems. Users need to discover functions in various software packages to analyze data, visualize images, and visualize data. Different platforms may require different data formats. Conversion or preparation of data for another platform often requires a significant effort. In the process of data transmission between different platforms, data may be lost or distorted. This can cause problems in the analysis. If the data analysis, image visualization, and data visualization could be integrated in one and the same platform, region-based MR brain image analysis would be made more efficient, and the workload of image analysts would be reduced.

1.4. Aim

The purpose of the project was to develop a processing toolchain that takes MR brain images and their anatomical segmentations as input and produces image visualizations, data visualizations, and statistical analyses in a unified manner for the users who work with neuroimaging and quantitative brain anatomy, including pathological anatomy. A secondary aim was to package and release the software publicly as open source.

1.5. Structure of this report

The structure of this report is as follows:

The material section briefly describes the material that was used for development of the functions, and the method section consists of a first-person narrative account of the development. The result section contains a brief description of each function and the relevant theory. A workflow chart is also provided. The contextualization and choices/limitations for this project are presented in the discussion section. Appendix 1 includes the documentation of the package. Flowcharts for functions are included in Appendix 2.

2. Material and method

2.1. Material

2.1.1. R language

The R programming language and software environment was designed by Ross Ihaka and Robert Gentleman of the University of Auckland in New Zealand and is now maintained by the R Core Team at the R Foundation [7]. It is an open-source statistical programming language that is often used to develop statistical and data analysis software systems. Its functions can be expanded by installing packages. In recent years, due to big data, data science, and the popularity of artificial intelligence, R has become widely used by statisticians and data scientists. Moreover, matrix analysis speed for the R language is comparable to the free software GNU Octave and the commercial software MATLAB. R provides the requisites for building a toolchain that integrates statistical analysis and image visualization in one platform. According to the TIOBE index for March 2021 indicating the popularity of programming languages, the R language ranks 13th. Competing programming languages on the same index are Python (2), MATLAB (15th), SAS (22nd), and Julia (34th).

2.1.2. R packages

An R package is a collection of R functions, compiled code, and sample data, typically for a single purpose or type of endeavour. R packages are stored in a directory named "library" in the R language environment. CRAN, an acronym for the "Comprehensive R Archive Network", is a network of ftp and web servers around the world that store identical, up-to-date versions of code and documentation for R. There are 10,000+ packages in CRAN which can be downloaded and installed in R. R developers can benefit from the work of others by downloading packages and either using its functions as they are, or to develop their own improvements and extensions.

The packages which I have used for developing functions are as below.

2.1.2.1. EBImage

EBImage is a package for image processing and analysis. It can be used to process objects in matrices or arrays.

2.1.2.2. misc3d

The misc3d package contains functions for processing 3D data.

2.1.2.3. neurobase

neurobase is a package which enables reading and writing of image data in the NIfTI format (Neuroimaging Informatics Technology Initiative), an open file format, having an .nii or .nii.gz file extension, widely used in neuroimaging research. The package defines a class 'nifti' and offers a variety of functions for interacting with objects of this type.

The neurobase package depends on the oro.nifti package, which provides functions to read and write images with nifti format.

2.1.2.4. rgl

The rgl package is a tool for rendering three-dimensional graphics.

2.1.2.5. tidyverse

The tidyverse is a collection of R packages that together implement a modern approach to data analysis work. This project made use of dplyr, magrittr, plyr, purr, tibble, tidyr, ggplot2, and shiny, in particular. The latter two deserve a more detailed explanation.

ggplot2 is a powerful graphics visualization R package which uses a layer concept to create graphics. Additional layers can be overlaid on already existing objects of a drawing. For example, a boxplot can be overlaid on a violin plot to visualize the relation of the two graphs [8]. In addition, the ggplot2 package provides a considerable degree of flexibility so that the output by this project can be easily adapted to user requirements.

The ggplot2 package provides a large number of default color schemes and manual color matching options. The official instructions point out that the viridis palette in the ggplot2 package provides colour maps that are perceptually uniform in both color and black-and-white. The viridis palette is designed to be recognized by viewers with color blindness [9]. For this reason, it was selected for this project.

2.1.2.6. shiny

Shiny is a package used to develop interactive web applications in R. Shiny makes it possible to create appealing presentations that interact with existing R packages such as ggplot2 and tidyverse.

In addition, the tidyverse defines the tibble datatype, a variant of the classical dataframe type with added functionality that makes it convenient to work with.

2.1.3. Atlas

The Hammers Atlas Database (www.brain-development.org) is a publicly shared resource centered around MR brain images and manual segmentations. These data are free to use for academic purposes, non-commercial research, and teaching. The data used in this project is the Adult Individual Atlases. The database contains 30 individual manually drawn, carefully verified segmentation label sets (95 regions each), created by experts on T1-weighted 3D magnetic resonance brain scans of 30 healthy adults [10]–[13].

In addition, automatic segmentations of the same images are also available; these have been generated using MAPER, software that segments magnetic resonance brain images into anatomical regions. MAPER, is developed by Heckemann et al. [14].

2.2. Method

To develop an R package which contains image visualizations, data visualizations and statistical analysis functions working as a toolchain, the data processing functions and working characteristics of the R language must be studied. I started my study by reading a book called R for Data Science [15]. I then examined the neurobase package to find out what kind of data I was going to process and how to process it. Due to the fact that I did not know much about region-based MRI brain image analysis, the function development was difficult at first. After consulting with my supervisor, I decided to talk with potential users of the package in order to understand their needs. Another master student, who was working on a project on brain morphometry in Parkinson's disease and my supervisor became my consultants for package development. Having discussed and understood their needs, I researched available packages which were suitable for developing functions. I surveyed existing software suites, such as convert3D [16], NiftySeg [17] and MIRTK [18], to see if any of them met the requirements. However, I had difficulty installing MIRTK and NiftySeg on my computer.

The packages mentioned in section 2.1.2. were used to develop the functions included in the toolchain. The neurobase package was used to read NIfTI files. The plyr package was used to split NIfTI files into matrices. After that, data were predominantly stored as tibble objects. Functions in the dplyr package were used to process data. The pipe function in the magrittr package was used to achieve high readability of the code, and the map series functions in the purrr package were used for improving efficiency. Functions in readr package were used to convert the processed data into wide table(s) that were easy to read.

I encountered some difficulties in dealing with the users' requirements concerning image visualization. CRAN contains many packages that can be used for image visualization. After installing several packages, I found that most image processing packages cannot be used to process data in matrix or array formats. After doing a lot of research, I found functions in EBImage downloaded from the Bioconductor website (http://www.bioconductor.org/) that could be used directly on objects with matrix or array formats. I therefore chose to develop two 2D image visualization functions. The rgl and misc3d packages were used to enable 3D image visualization requirements according to the users' requirements.

While working on the development of data visualization and statistical analysis functions, I decided to adopt an interactive data presentation method after discussing with my supervisor. The shiny package was chosen. Although this package provides ready-made components for data display, charts, and models on the web, I found that these did not meet the requirements. Therefore, I spent some time taking online courses on advanced use of the shiny package. In my interactive application, readr package functions load the CSV files (comma-separated values) generated by the previously developed functions. The ggplot2 package was used for graphics visualization. The viridis palette in the ggplot2 package was also used in the application in order to enable colorblind-friendly graphics visualization.

MR files and manual segmented files from Hammers Atlas Database as well as auto-segmented files were used to test the functions. The sample images used in this project are 3D T1-weighted magnetic resonance images acquired at 1.5 Tesla. Acquisition details are of minor importance in the context of the project, as the software places no restrictions on the modality or sequence, as long as the images represent a 3D object volumetrically and are available in, or can be converted to, NIfTI format. The manual segmentation labels in the sample have been produced by experts following rigidly defined and validated public protocols [10]–[13] . The automatic segmentation labels have been prepared by the supervisor using MAPER software [14] on the basis of the manual segmentation labels in a leave-one-out cross-comparison setup; ie., each atlas image was used in turn as the target image and segmented using the other 29 atlases as training data.

For some functions, such as regcount, labelnaming and bsimilarity, I compared the results with outputs from Convert3D to ensure that the calculations were correct. After the completion of a function (including code description), I uploaded it to github (https://github.com/yupingikub/IKUB) for the student consultant and my supervisor to download and test. All functions were developed iteratively based on feedback. Figure 3 shows the procedure for the iterative development. After confirming that the developed functions met the requirements, the function description and function flowcharts were created. Documentation for the package were written using R Markdown. It contains examples for every function.

The workflow design, programmed functions, and unified analysis platform for MR brain images are unique and novel contributions achieved in this master's project.

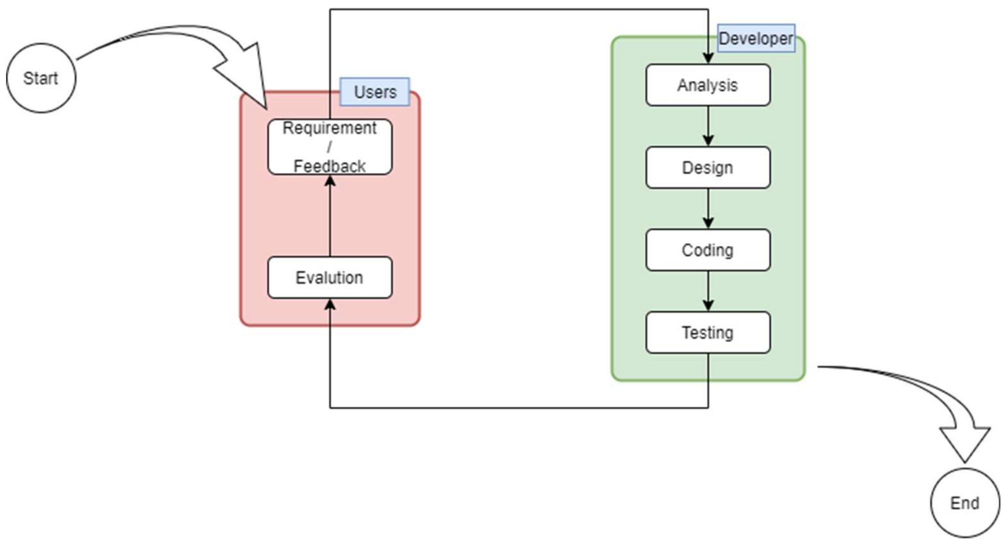


Figure 3: Iterative development

3. Result

An R package, a shiny application and a workflow were developed for providing tools that take MR brain images and their anatomical segmentations as input and produces image visualizations, data visualizations, and statistical analyses. According to the stakeholders' requirements, the developed R package named IKUB and the shiny application implement the following functions:

1. Calculate total counts and total volumes for all regions for one supplied NIfTI file.
2. Calculate volumes for all regions with label names for one or more supplied NIfTI files.
3. Calculate mean and standard deviation for all regions for a user-defined reference group.
4. Calculate Z-scores of all regions for one or more NifTI-files, using the reference group.
5. Calculate Dice coefficients and Jaccard indices for a supplied NIfTI file pair.
6. Calculate asymmetry index/indices for one or more supplied NIfTI files.
7. Extract numbers of slices for one or more supplied label numbers.
8. Display label segmentation differences in 2D and 3D view.
9. Display anatomical segmentation contours in 2D view.
10. Display anatomical surface in 3D view.
11. Generate analytical data.
12. Plot generated data as violin plots and box plots in the shiny app.

3.1. Workflow

As part of this project, a workflow was designed to process from NIfTI files to different outputs. The workflow shown in Figure 4 is a proposed procedure. Most of the functions in this package can be used independently. Users can design their own workflows to suit their needs.

All functions in the designed workflow are briefly described in this section. For detailed function usages, please refer to the documentation in appendix 1. A selection of use cases of the workflow (Figure 4) are described in the following.

The author made use of the Hammers Atlas Database, © Imperial College of Science, Technology and Medicine 2007. All rights reserved. The resource is available from www.brain-development.org [10]–[13]. In the demonstrations of the segcomparison function and the segcomparison3D function, NIfTI files generated by MAPER were also used.

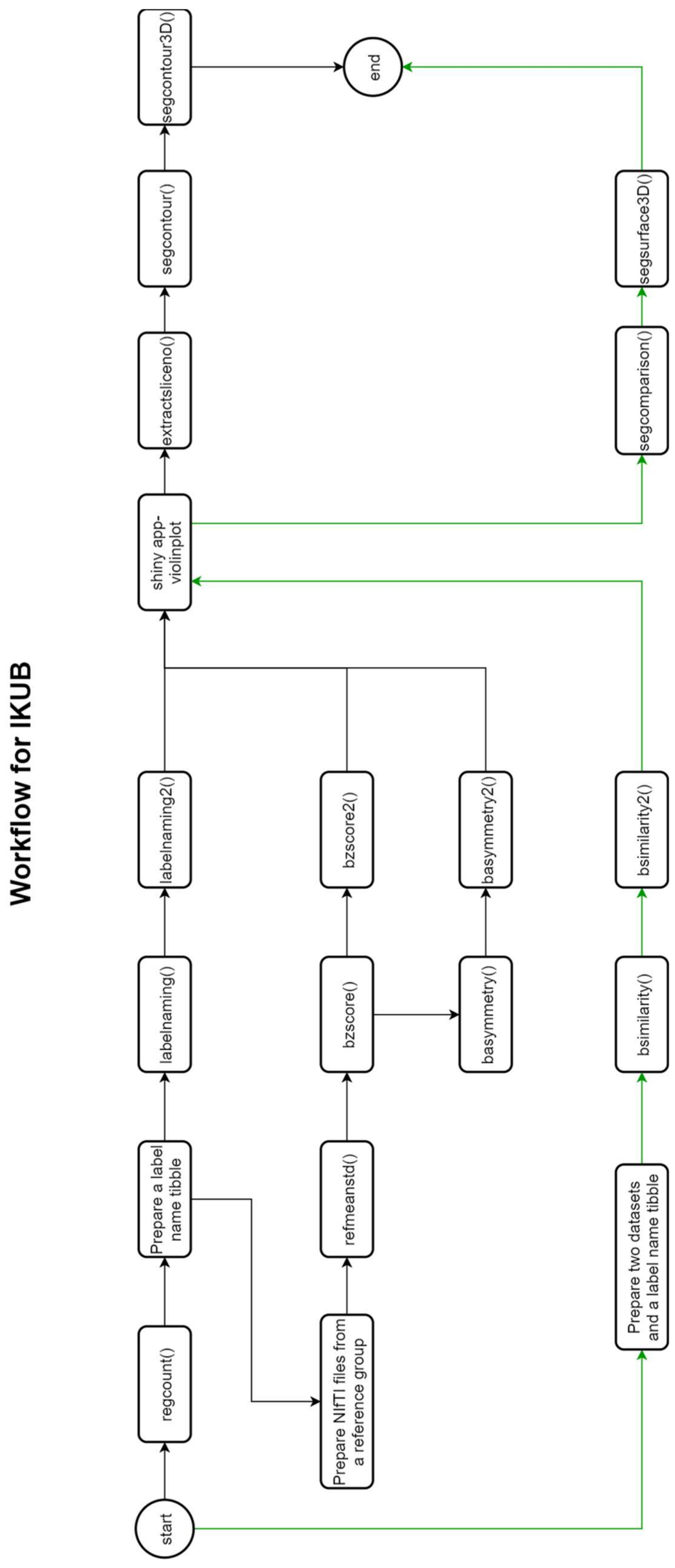


Figure 4: A suggested workflow which contains four paths.

3.1.1. The first path

The red line in Figure 5 shows the first workflow path. The functions used in the first path are described in this subsection. The first path is designed for the brain volumetric analysis.

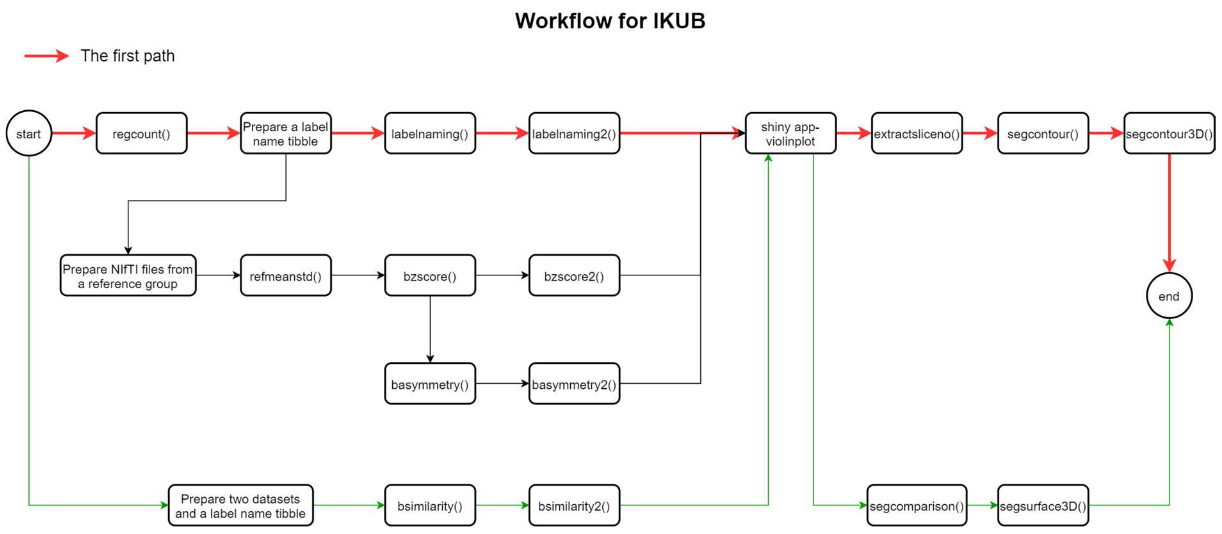


Figure 5: The first path of the designed workflow.

3.1.1.1. *regcount function*

The first path is designed so that users begin with the regcount function to obtain the statistics for all label numbers which are included in a supplied NIfTI-file. The function returns total volumes and total counts for all regions in the R console. Output from the regcount function is shown in R_Output 1.

R_Output 1: File being analyzed is a01-seg.nii.gz. Calculated output from regcount function is as follows:

```
regcount('a01-seg.nii.gz')

## # A tibble: 96 x 4
##    filename       Label_No  counts   vol_mm3
##    <chr>             <int>   <int>     <dbl>
##  1 a01-seg.nii.gz        0 4248688 3495212.
##  2 a01-seg.nii.gz        1    2154    1772.
##  3 a01-seg.nii.gz        2    1873    1541.
##  4 a01-seg.nii.gz        3    1524    1254.
##  5 a01-seg.nii.gz        4    1543    1269.
##  6 a01-seg.nii.gz        5    9845    8099.
##  7 a01-seg.nii.gz        6    9010    7412.
##  8 a01-seg.nii.gz        7    5789    4762.
##  9 a01-seg.nii.gz        8    4325    3558.
## 10 a01-seg.nii.gz        9    5627    4629.
## # ... with 86 more rows
```

3.1.1.2. *labelnaming function*

The regcount function can be used to calculate total counts and volume for all regions for a supplied NIfTI file. Meanwhile, the output returned from the regcount function contains only statistics for all label numbers. The second step of this path is to prepare a label name tibble which has correct format and unique columns. Then, users input the label name tibble and one or more NIfTI-files into the labelnaming function which returns volumes for all regions with label names in the R console. Output from labelnaming is shown in R_Output 2.

R_Output 2: Input 30 NIfTI files including in Hammers Atlas Database into labelnaming function. The function returns total volumes for all regions with label numbers and label names for supplied input.

```
filename<-c(paste('a0', 1:9, '-seg.nii.gz', sep=''), paste('a', 10:29, '-seg.nii.gz', sep='
'))
labelnaming(filename)

## # A tibble: 95 x 32
##    Label_No Label_name        `a01-seg.nii.gz` `a02-seg.nii.gz` `a03-seg.nii.gz`
##       <int> <chr>                        <dbl>            <dbl>            <dbl>
##  1        1 TL hippocampus R             1772.            1952.            1999.
##  2        2 TL hippocampus L             1541.            1949.            2037.
##  3        3 TL amygdala R                1254.            1411.            1212.
##  4        4 TL amygdala L                1269.            1375.            1374.
##  5        5 TL anterior temp~            8099.            7276.            8510.
##  6        6 TL anterior temp~            7412.            4835.            7732.
##  7        7 TL anterior temp~            4762.            3121.            5066.
##  8        8 TL anterior temp~            3558.            1840.            4553.
##  9        9 TL parahippocamp~            4629.            4864.            4517.
## 10       10 TL parahippocamp~            4414.            4882.            4744.
## # ... with 85 more rows, and 27 more variables: a04-seg.nii.gz <dbl>,
## #   a05-seg.nii.gz <dbl>, a06-seg.nii.gz <dbl>, a07-seg.nii.gz <dbl>,
## #   a08-seg.nii.gz <dbl>, a09-seg.nii.gz <dbl>, a10-seg.nii.gz <dbl>,
## #   a11-seg.nii.gz <dbl>, a12-seg.nii.gz <dbl>, a13-seg.nii.gz <dbl>,
## #   a14-seg.nii.gz <dbl>, a15-seg.nii.gz <dbl>, a16-seg.nii.gz <dbl>,
## #   a17-seg.nii.gz <dbl>, a18-seg.nii.gz <dbl>, a19-seg.nii.gz <dbl>,
## #   a20-seg.nii.gz <dbl>, a21-seg.nii.gz <dbl>, a22-seg.nii.gz <dbl>,
## #   a23-seg.nii.gz <dbl>, a24-seg.nii.gz <dbl>, a25-seg.nii.gz <dbl>,
## #   a26-seg.nii.gz <dbl>, a27-seg.nii.gz <dbl>, a28-seg.nii.gz <dbl>,
## #   a29-seg.nii.gz <dbl>, a30-seg.nii.gz <dbl>
```

3.1.1.3. labelnaming2 function

The labelnaming2 function is used to generate data for brain segmentation volume analysis. The function works similarly to the labelnaming function but prepares the output for further analysis in the shiny app. Users should save labelnaming2 output as a CSV file.

3.1.1.4. Violin plot-shiny app

The next step is to upload the CSV file to the online shiny app to visualize data. The link to the online shiny app is https://alisonhyp.shinyapps.io/shiny_violinplot/

A violin plot combined with a box plot can show many details. Sometimes it may provide too much distracting information. Therefore, the interactive app is designed so that users can select or deselect certain options to display the data flexibly. Users can also manually assign a title, label names for x- and y-axis for the figure. By selecting the plot option – annotate outlier, the app will display which data points are outliers. Figure 6 shows output from the interactive shiny app.

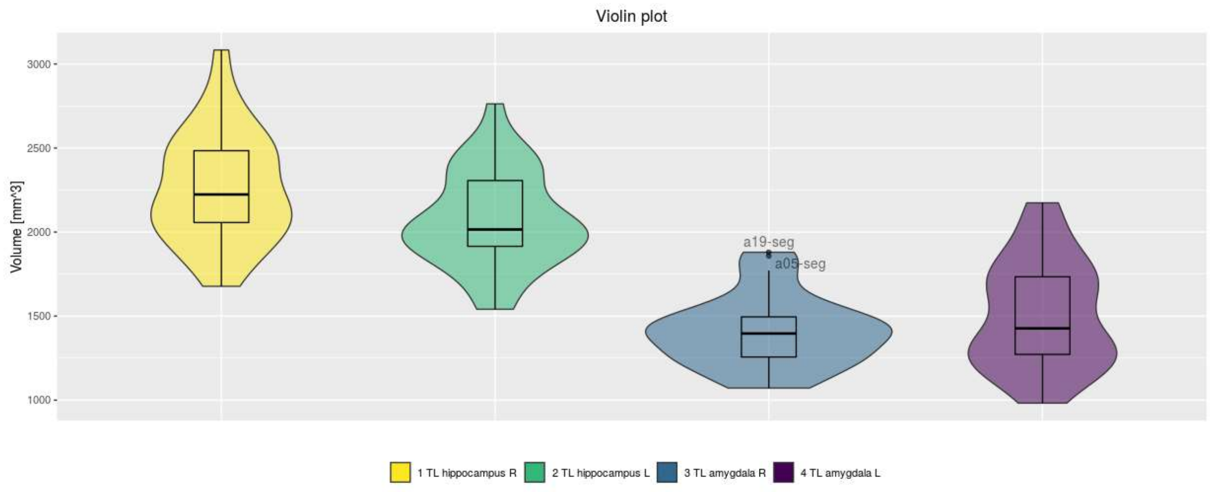


Figure 6: Shiny app data visualization created from the output of the labelnaming2 function. Four regions are selected to construct a combined violin-box plot. Two outliers, a19-seg and a05-seg, in the right amygdala region are highlighted.

3.1.1.5. extractsliceno function

In statistics, an outlier is a value in the dataset that is strongly different from the other values. Outliers in statistical data should be handled carefully. They can occur at random, but sometimes they are caused by systematic processes that interfere with the processes under study. By following up on outliers, users may be able to detect and safeguard against, or eliminate, such confounding processes.

On this workflow path, users can use the shiny app to detect outliers, then use the extractsliceno function to extract slice numbers. Users can input a NIfTI-filename and one or more target label numbers into the extractsliceno function. The function will return the numbers of the slices that include the supplied target label number(s) for the supplied NIfTI file. Figure 6 shows that two data points, a19-seg and a05-seg, are outliers in the right amygdala region. To determine in which slices this region occurs, users apply the extractsliceno function to extract slice numbers for the target label number. Output is shown in R_Output 3.

R_Output 3: Uses extractsliceno function to obtain slice numbers including TL amydala R in transverse plane for the NIfTI file a19-seg.nii.gz.

```
extractsliceno('a19-seg.nii.gz', 3)

##  [1] 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
64
```

3.1.1.6. segcontour function

The segcontour function is designed to display anatomical contours for one or more label numbers in a medical MR image. According to output from the extractsliceno function, users can use the segcountour function to examine the contours in these slices. Users can study these images and try to find out the reasons for data deviation. Figure 7 shows the output from the segcontour function.

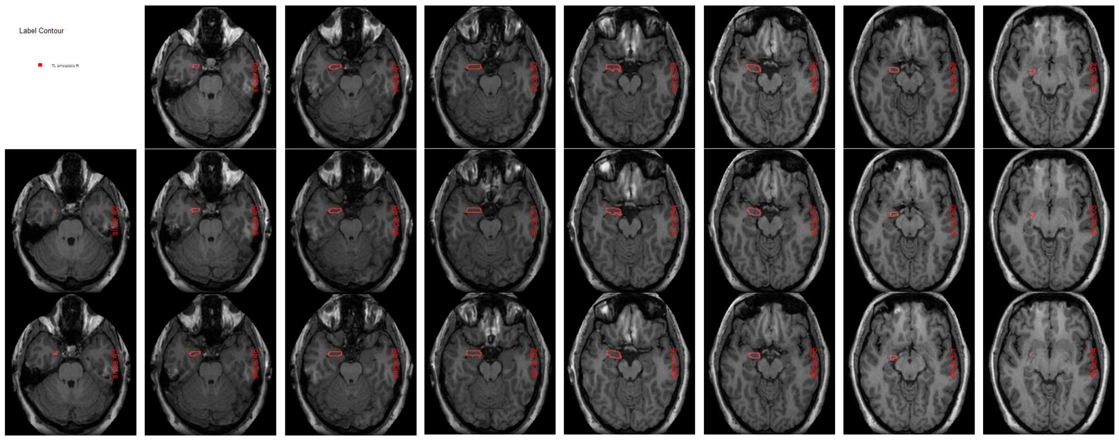


Figure 7: Using output from extractsliceno as slice_no input. Input for region-based NIfTI file and MR NIfTI file are a19-seg.nii.gz and a19.nii.gz. Target label number is 3. Images in transverse plane are shown.

3.1.1.7. segsurface3D function

The segsurface3D function can display anatomical segmentation contours of a medical MR image in a 3D view for one or more supplied label numbers. The 3D view can be rotated by dragging the mouse and zoomed in or out by scrolling the mouse scroll wheel. The 3D image provides rough depictions of segmented regions.

Figure 8 shows output from segcontour3D function.

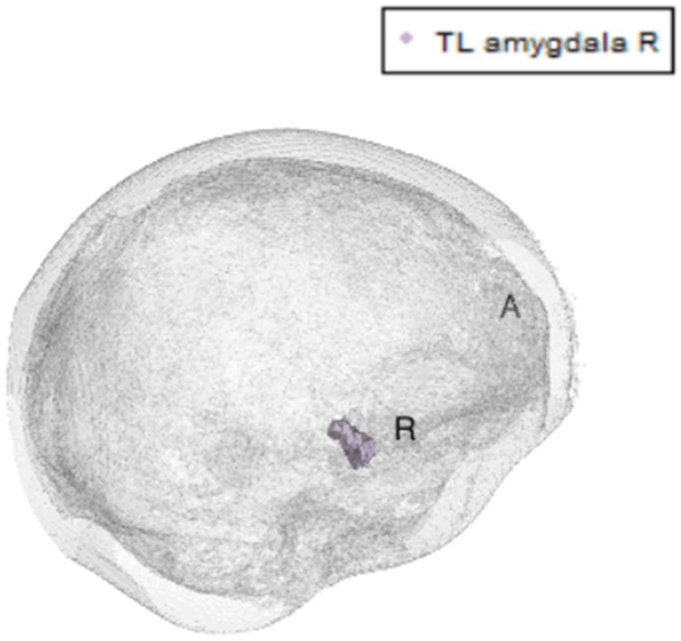


Figure 8: 3D view for region TL amygdala R in NIfTI file a19-seg.nii.gz.

3.1.2. The second path

The second path of the workflow is shown in Figure 9. The second path is also designed for Z-score analysis. The procedure from the start point to prepare a label name tibble has been mentioned earlier, so the description begins with preparing NIfTI files from a reference group.

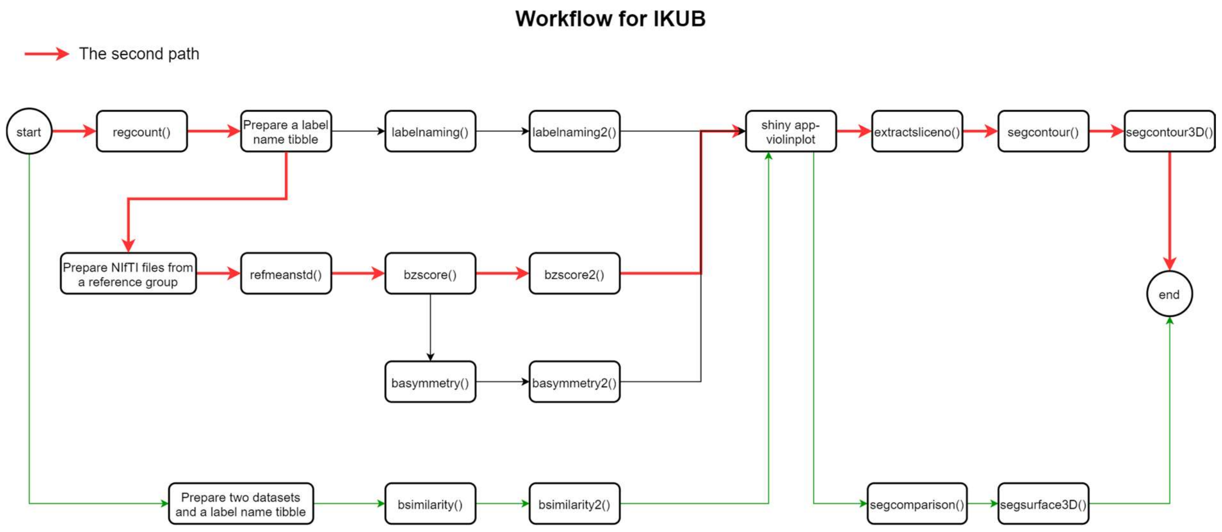


Figure 9: The second path of the workflow is marked with red color.

3.1.2.1. refmeanstd and bzscore function

Output from the labelnaming function provides a general description of the volume in each region. To calculate the Z-scores for supplied NIfTI files, the refmeanstd function is first used to calculate the mean and standard deviation for all regions in a reference group. A result calculated by the refmeanstd function is shown in R_Output 4. The bzscore function use Equation (1) to calculate Z-scores. Output from the refmeanstd function is used as one of the inputs for the bzscore function. The bzscore function returns Z-scores of all regions for one or more NIfTI files. R_Output 5 shows the result returned from the bzscore function.

R_Output 4: Using 30 NIfTI files in Hammers Atlas Database as input for the refmeanstd function to calculate standard deviation and mean values.

```
reflabels<-c(paste('a0', 1:9, '-seg.nii.gz', sep=''), paste('a', 10:29, '-
seg.nii.gz', sep=''))
refstats <- refmeanstd(reflabels)

## # A tibble: 96 x 3
##    Label_No    mean     std
##  *    <int>   <dbl>   <dbl>
##  1        0 3535557. 248326.
##  2        1    2261.    328.
##  3        2    2072.    285.
##  4        3    1410.    209.
##  5        4    1509.    317.
##  6        5    7274.   1311.
##  7        6    6814.   1038.
##  8        7    3933.    899.
##  9        8    3654.   1020.
## 10        9    4704.    679.
## # ... with 86 more rows
```

R_Output 5: Using output from the refmeanstd function as input for the bzscore function, which returns Z-scores for all regions for a01-seg.nii.gz and a02-seg.nii.gz.

```
bzscore(c('a01-seg.nii.gz', 'a02-seg.nii.gz'), labelnametibble, refstats)

## # A tibble: 95 x 4
##    Label_No Label_name                               `a01-seg.nii.gz` `a02-seg.nii.gz`
##       <int> <chr>                                               <dbl>            <dbl>
##  1        1 TL hippocampus R                                  -1.49            -0.942
##  2        2 TL hippocampus L                                  -1.86            -0.432
##  3        3 TL amygdala R                                     -0.747            0.00529
##  4        4 TL amygdala L                                     -0.757           -0.425
##  5        5 TL anterior temporal lobe medial ~                 0.630            0.00206
##  6        6 TL anterior temporal lobe medial ~                 0.577           -1.91
##  7        7 TL anterior temporal lobe lateral~                 0.922           -0.903
##  8        8 TL anterior temporal lobe lateral~                -0.0939          -1.78
##  9        9 TL parahippocampal and ambient gy~                -0.110            0.236
## 10       10 TL parahippocampal and ambient gy~                -0.731           -0.0597
## # ... with 85 more rows
```

3.1.2.2. bzscore2 function

The bzscore2 function is used to generate standard score data for all regions for supplied NIfTI-files. The function works like the bzscore function. The output from the bzscore2 function must be exported in CSV format for further analysis in the shiny app-violinplot.

In the next step, the CSV file is uploaded to the shiny app to visualize the data. The result is shown in Figure 10.

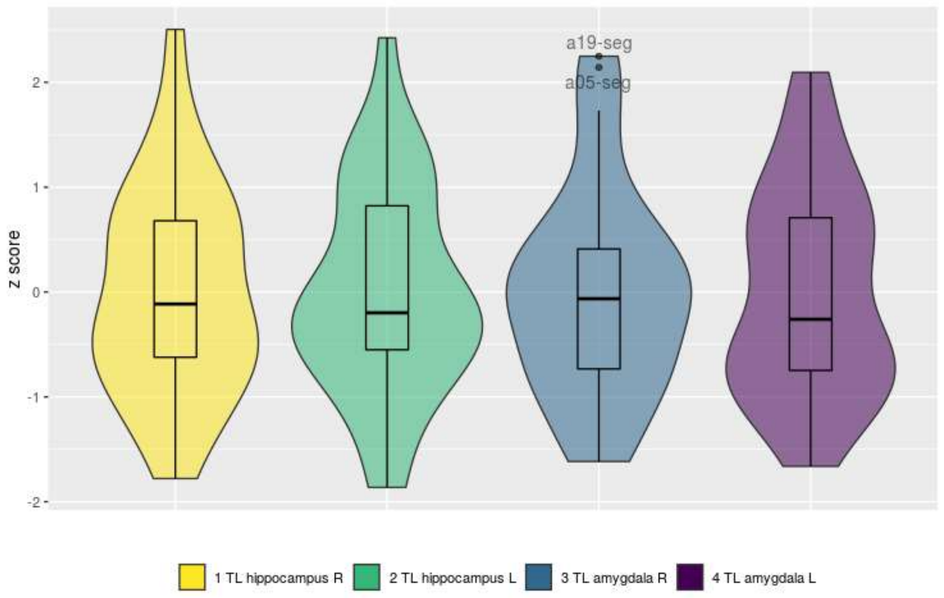


Figure 10: Violin plots with included boxplots as produced by the shiny app. Two outliers, a19-seg and a05-seg, for the right amygdala are annotated.

As in Path one, the segcontour and segsurface3D functions can be used to display one or more specific label numbers for a selected NIfTI file.

3.1.3. The third path

Figure 11 shows the third path of the design workflow. This path is designed for users who are interested in anatomical asymmetry of the brain.

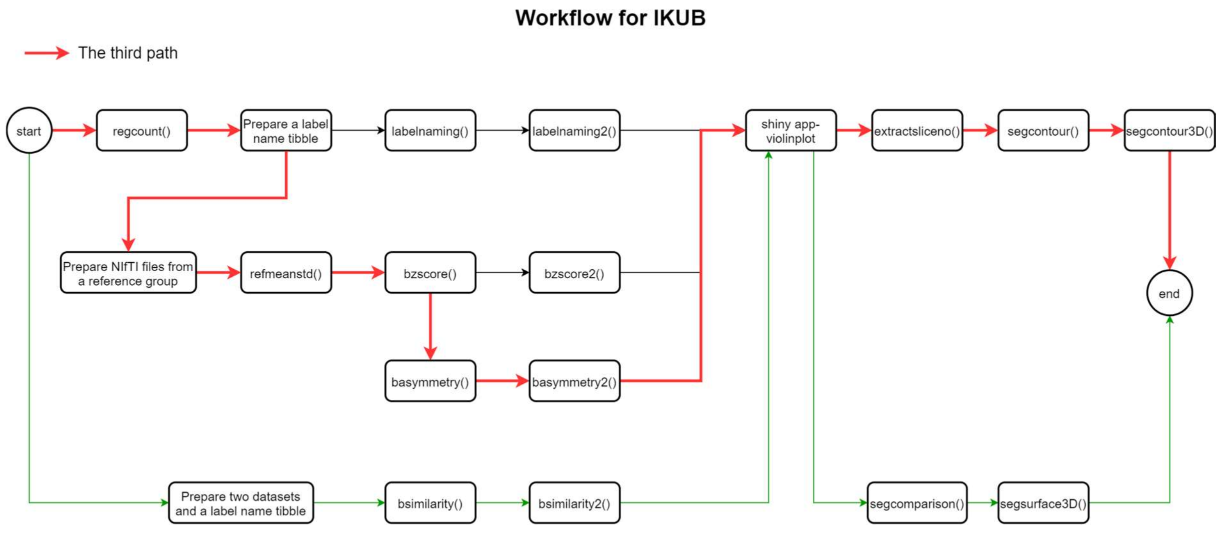


Figure 11: Workflow path for brain asymmetry analysis

Concerning the first part of the procedure and the bzscore function, please refer to the sections above. The description of the functions in the third path begins with the basymmetry function.

3.1.3.1. basymmetry function

The basymmetry function uses Equation (2) to calculate asymmetry indices for paired regions in one or more supplied NIfTI-files. Output_R 6 presents asymmetry indices for the left and right regions for two NIfTI files.

R_Output 6: Brain region asymmetry calculated by the basymmetry function. Input files are a01-seg.nii.gz and a02-seg.nii.gz.

```
basymmetry(c('a01-seg.nii.gz','a02-seg.nii.gz'),labelnametibble, 2)

## # A tibble: 46 x 3
##    item                              `a01-seg.nii.gz` `a02-seg.nii.gz`
##    <chr>                                        <dbl>            <dbl>
##  1 caudate nucleus                            0.0229           0.0089
##  2 cerebellum                                 0.0224           0.0638
##  3 CG anterior cingulate gyrus                0.508            0.0510
##  4 CG posterior cingulate gyrus               0.0297           0.0146
##  5 FL anterior orbital gyrus                  0.00912          0.159
##  6 FL inferior frontal gyrus                  0.191            0.0147
##  7 FL lateral orbital gyrus                   0.145            0.245
##  8 FL medial orbital gyrus                    0.111            0.0197
##  9 FL middle frontal gyrus                    0.114            0.0864
## 10 FL posterior orbital gyrus                 0.0792           0.188
## # ... with 36 more rows
```

3.1.3.2. *basymmetry2 function*

Users can use the basymmetry2 function to generate and export data to CSV format for further analysis in shiny app-violinplot.

Upload the CSV file to the shiny app for further analysis. The data visualization result from the shiny app is shown in Figure 12.

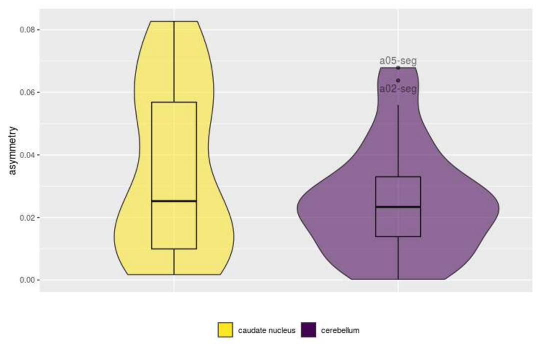


Figure 12: Asymmetry index plotted by the shiny app. Caudate nucleus and cerebellum are selected. Two outliers are annotated for the cerebellum region.

Users can use the segcontour and segsurface3D functions to perform further analysis. Suppose users are interested asymmetry in cerebellum in a05-seg.nii.gz. Users can use the extractsliceno function to obtain slices numbers for cerebellum R and cerebellum L, then input the output from the extractsliceno function to the segcontour function. The output from segcontour function is shown in Figure 13. The output from segsurface3D is shown in Figure 14.

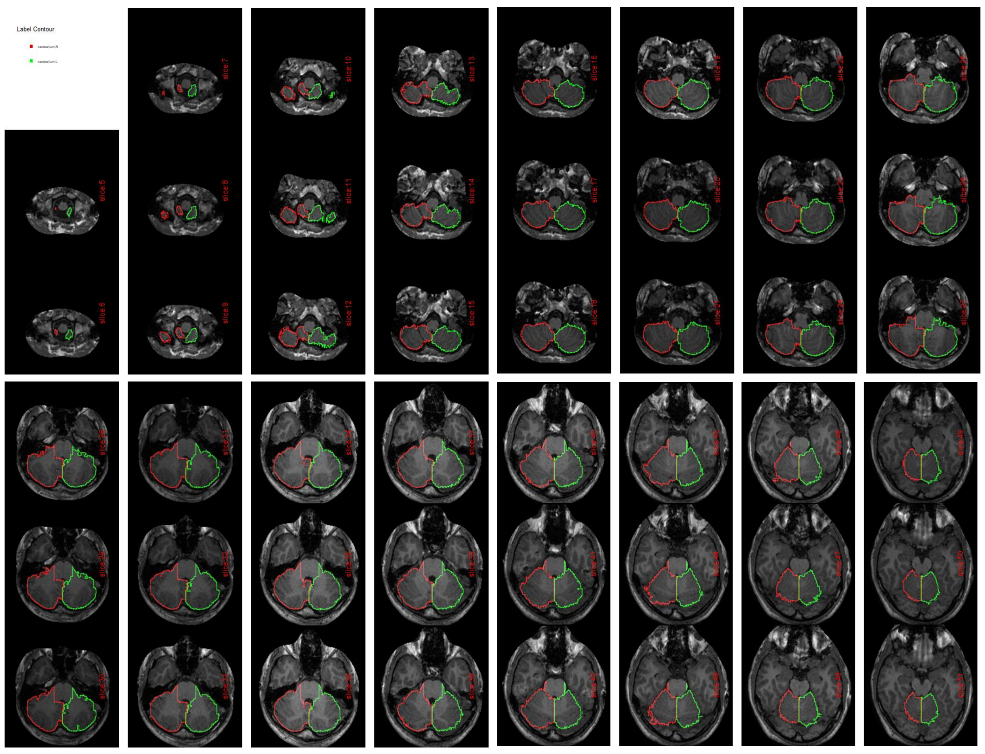


Figure 13: Contour in red color represents cerebellum R and contour in green color represents cerebellum L. cerebellum R and cerebellum L are included in slice number from 3 to 73. Only slice number from 5 to 51 are displayed.

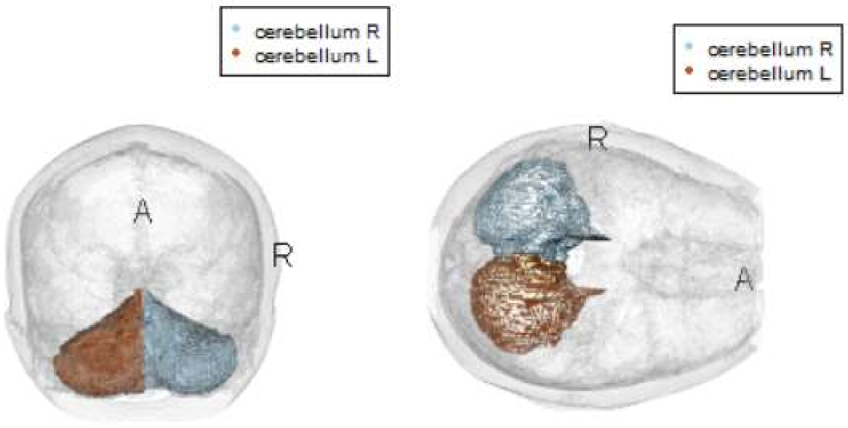


Figure 14: Displays 3D view for cerebellum R and cerebellum L for a05-seg.nii.gz.

3.1.4. The fourth path

The fourth path shown in Figure 15 is designed for comparing pairs of segmentation label images. This would be used to assess, for example, the level of agreement between an automatic segmentation result and a trusted reference. The user inputs a pair of NIfTI-files generated from the same source image.

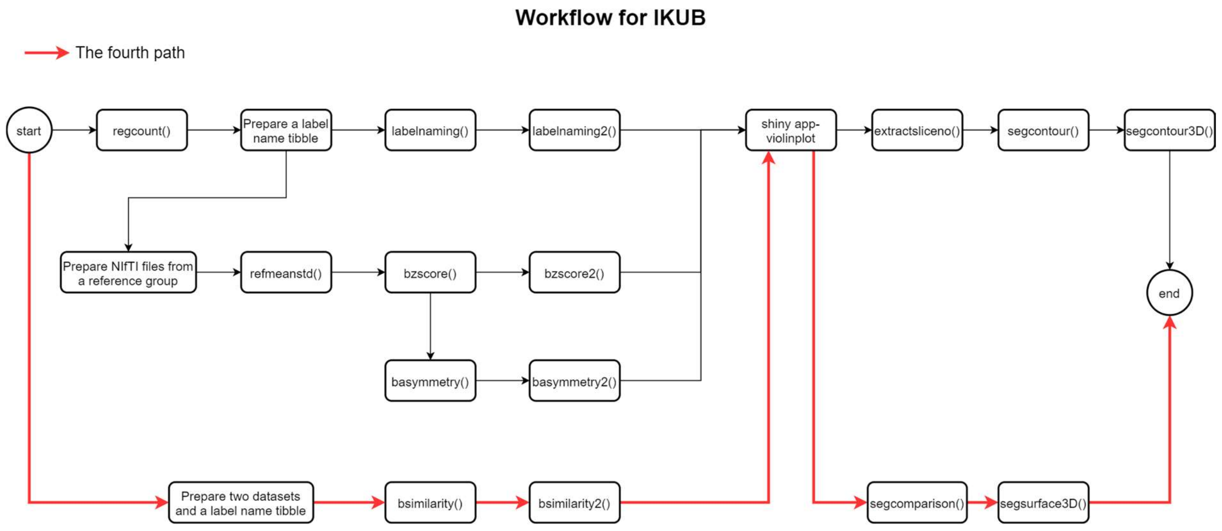


Figure 15: Shows the fourth path which is marked in red color for the workflow.

3.1.4.1. bsimilarity function

The bsimilarity function is used to calculate Dice coefficients and Jaccard indices for a supplied file pair.

R_Output 7 as below contains the output from the bsimilarity function.

R_Output 7: a01-seg.nii.gz from Hammers Atlas Database and a1.nii.gz generated by MAPER are combined to a file pair. The bsimilarity function returns Dice coefficient and Jaccard index for all regions in the file pairs.

```
bsimilarity('a01-seg.nii.gz', 'a1.nii.gz', labelnametibble)

## # A tibble: 95 x 4
##    Label_No Label_name                             dice.coefficient jaccard.index
##       <int> <chr>                                             <dbl>         <dbl>
##  1        1 TL hippocampus R                                  0.844         0.731
##  2        2 TL hippocampus L                                  0.85          0.74
##  3        3 TL amygdala R                                     0.788         0.65
##  4        4 TL amygdala L                                     0.833         0.714
##  5        5 TL anterior temporal lobe medial par~             0.872         0.773
##  6        6 TL anterior temporal lobe medial par~             0.84          0.725
##  7        7 TL anterior temporal lobe lateral pa~             0.748         0.598
##  8        8 TL anterior temporal lobe lateral pa~             0.695         0.533
##  9        9 TL parahippocampal and ambient gyrus~             0.86          0.754
## 10       10 TL parahippocampal and ambient gyrus~             0.867         0.765
## # ... with 85 more rows
```

3.1.4.2. bsimilarity2 function

The bsimilarity function calculates Dice coefficients and Jaccard indices for one file pair. To compare one or more file pairs, users invoke the bsimilarity2 function. bsmilarity2 returns Jaccard indices for all regions for all supplied NIfTI file pairs. The generated output must be exported to CSV format in order to be made available for further analysis in shiny app-violinplot.

The next step is to upload the csv file to the shiny app. The output from the shiny app is shown in Figure 16.

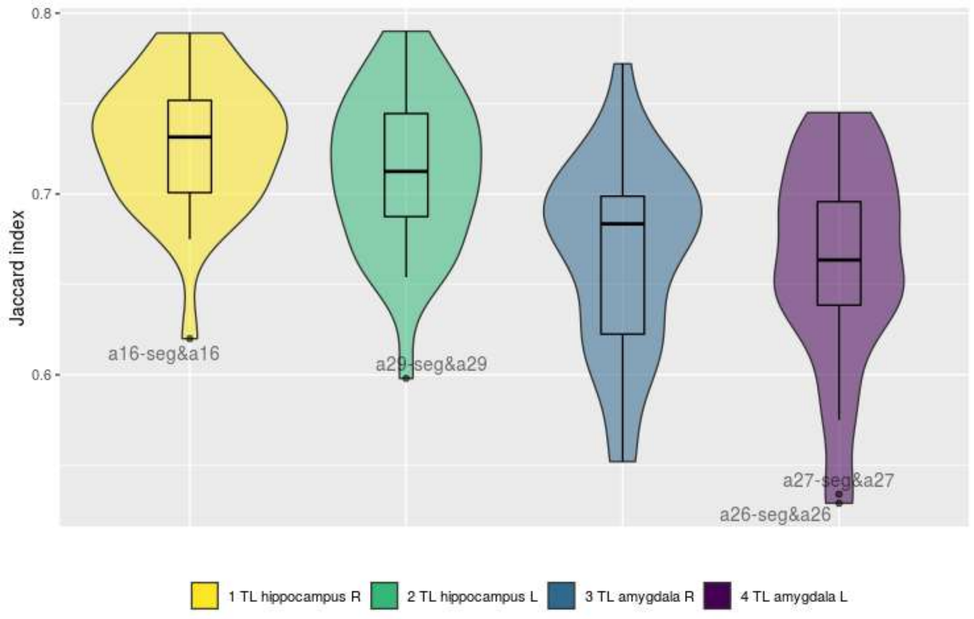


Figure 16: Violin plots with included boxplots, showing Jaccard indices for four regions and 30 participants.

3.1.4.3. segcomparison function

After having used the shiny app to analyze the csv file from the bsimilarity2 function, users may be interested in the similarity of a specific region. The segcomparison function displays segmentation differences between two NIfTI files with regard to a label, identified by number. Voxels labelled as the region in filename_seg1 only are displayed in green, those labelled as the region in filename_seg2 only are displayed in blue, and labelled as the region in both files are displayed in red.

Suppose that users are interested in TL hippocampus R in the file pair (a16-seg.nii.gz and a16.nii.gz). The segcomparison function can be used to display segmentation differences. The output from the segcomparison function is shown in Figure 17.

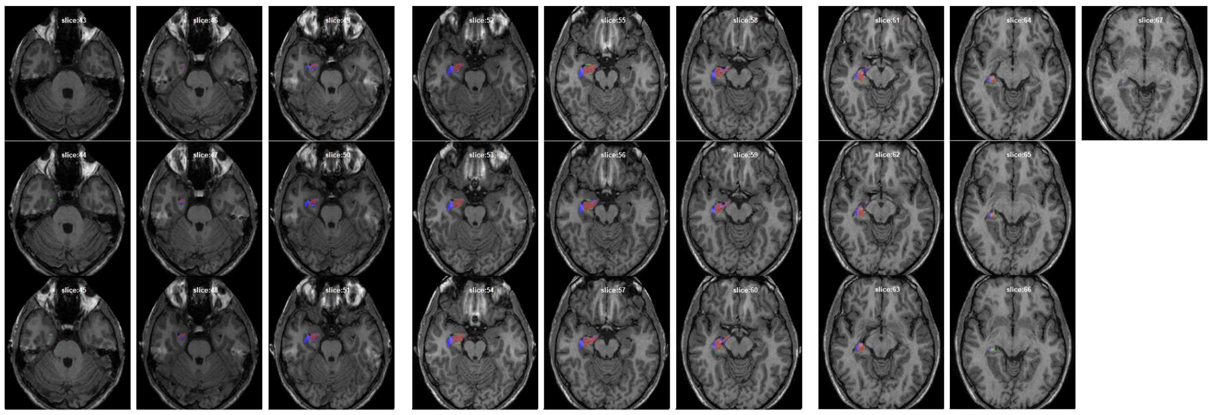


Figure 17: Displays segmentation differences in all slices including region TL hippocampus R for the file pair. Images display in the transverse plane.

3.1.4.4. segsurface3D function

The IKUB package includes statistical and visualization functions for brain segmentation images. With these functions, users can obtain information such as counts, volumes, and Z-scores for all regions, asymmetry indices for whole brain or all regions. Users can also evaluate the similarity for a pair of segmentations generated from the same source. By uploading generated csv files to shiny app, data visualization and statistical analysis can be accomplished. The designed workflow can be used to process analysis for region-based MR brain images.

Figure 18 shows the output from segsurface3D function. The same file pair and label number as in subsection 3.1.4.3 are used.

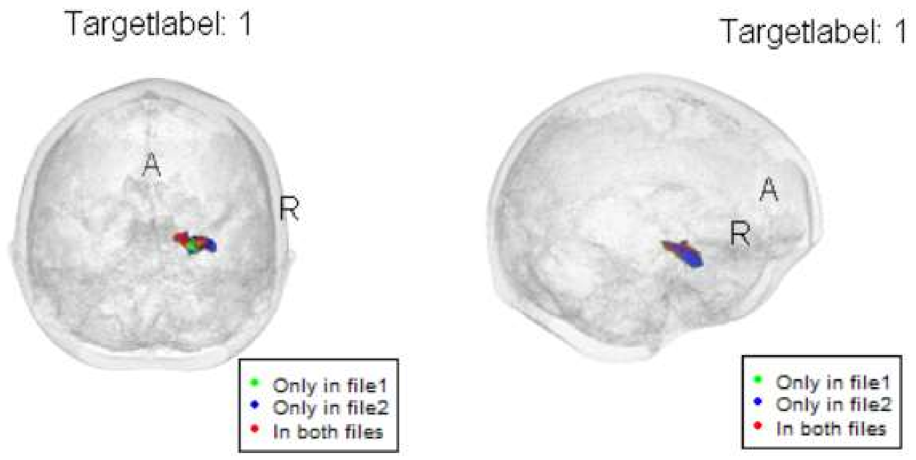


Figure 18: Segmentation differences in 3D view. Target label is TL hippocampus R.

4. Discussion

The software developed in this project comprises the IKUB package and the shiny web app. The IKUB package includes statistical and visualization functions for brain segmentation images. With these functions, users can obtain information such as counts, volumes, Z-scores, and asymmetry scores for all regions, asymmetry indices for whole brain or all regions. Users can also evaluate the similarity of a pair of segmentations generated from the same source. By uploading generated csv files to the shiny app, data visualization and statistical analysis can be accomplished. The designed workflow can be used to process analysis for region-based MR brain images.

4.1. Contextualization

The development addresses the following needs that no other software covers:

4.1.1. Integration platform

This development integrates data analysis, image visualization, and data visualization in one platform. Users do not need to transfer data between platforms for different analysis tasks. The output from one function can become the input for another function, while the data stays on the same platform. In terms of analyzing the standard score (Z-score), the output from the refmeanstd function becomes one of inputs for the bzscore2 function. A csv file produced by the bzscore2 function becomes input to the shiny app. Users obtain a data visualization as result of this processing chain. The processing chain is shown in Figure 19.

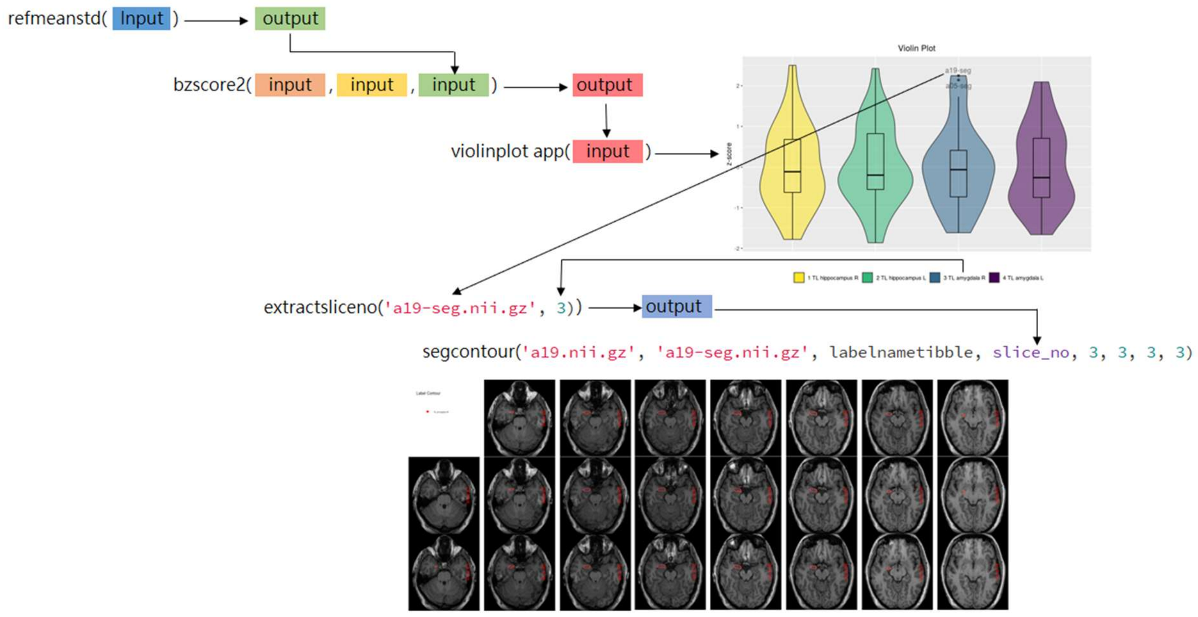

refmeanstd(Input) → output
bzscore2(input , input , input) → output
violinplot app(input) →
Violin Plot
extractsliceno('a19-seg.nii.gz', 3)) → output
segcontour('a19.nii.gz', 'a19-seg.nii.gz', labelnametibble, slice_no, 3, 3, 3, 3)

Figure 19: Sample processing chain, going from the data input via the refmeanstd function to the data visualization output from the violin plot shiny app.

4.1.2. Easy install

After having installed R and RStudio, users only need to install the IKUB package and load the package as a library. The build tools are integrated in R and the installation process is automated.

4.1.3. Flexible functions

The developed functions have a high degree of functional flexibility. Users can supply different arguments for functions to obtain the desired result. Take the volume calculation as an example. The commands -dup and -lstat in Convert3D are used to calculate the total counts and volumes for all regions for one NIfTI file. The labelnaming function in the package can be used to perform the same calculations on multiple files. Output from Convert3D provides only label numbers, but output from the labelnaming function includes both the label numbers and label names. Having label names in output can provide more direct information to users. Most of the functions in the package have been developed so that users can use multiple elements for each argument - this is one of the advantages over other software.

4.1.4. Documentation and workflow

The documentation for the package contains the description of the functions and arguments. It also provides examples for functions which are used in the workflow. Documentation was written using R Markdown and can be knitted to pdf, word, and html formats. The documentation in html format has a floating table of contents. This can help users quickly find the function descriptions they need. By reading examples, the users can rapidly learn how to use the functions.

A workflow was designed as part of the development. Users can analyze data, visualize images, and visualize data according to the suggested workflow to increase efficiency.

4.2. Choices/limitations

4.2.1. Small number of consulted users

At present, there is no standard workflow in the field of brain segmentation data analysis. The workflow that was designed in this project is based on discussions with another master student and my supervisor. Because of the small amount of user-feedback, the workflow may only be applicable to a small number of users. As the package is now released, it can be hoped that other researchers will download it, install it, use it, and provide feedback on their experience. This will enable me to improve the workflow and make it applicable to a larger number of users.

4.2.2. Why Jaccard indices?

The Jaccard index is similar to the Dice coefficient, but it provides better geometrical description. Suppose there are two segmentation results: label X, which we trust, and label Y, which we want to test.

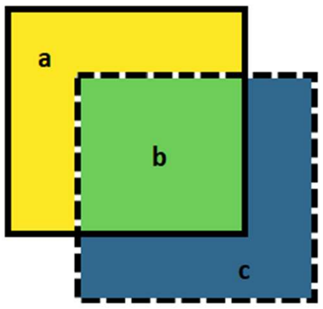


Figure 20: The black box represents the segmentation result which we trust, and the dash line box represents the segmentation result which we are testing. The yellow region in the figure represents pixels that only exist in label X, the blue region represents pixels that only exist in label Y, and the green region represents pixels that exist in both labels.

According to Equation (4), the Dice coefficient is derived using Figure 20 producing the following formula.

$$Dice\ coefficient = \frac{2 \times b}{2 \times b + a + c} = \frac{2 \times b}{(a+b)+(b+c)}$$

and Jaccard index can be derived using Figure 20 and Equation (3) producing the following formula.

$$Jaccard\ index = \frac{b}{a+b+c}$$

The Jaccard index divides the region of interest into a, b, and c and expresses the proportion of b to the total region of interest. With increasing b, the Jaccard index also increases. Dice index is the proportion of the intersection to the average of the two label volumes. To me, the derivation of Jaccard index is more intuitive in terms of geometric description than Dice coefficient.

Similarity refers to the distance between two samples. The other reason that the Jaccard index is chosen is that the Dice coefficient does not satisfy the triangle inequality. The triangle inequality describes the following observation, valid in Euclidean systems: Suppose there are three sets A={a}, B={a}, and AB={a, b}.

Dice distance d_D is defined as the following equation:

$$d_D = 1 - \frac{2 \times \text{The intersection of A and B}}{\text{Number of A } + \text{Number of B}} \tag{5}$$

$$d_D(A,B) = 1 - \frac{2 \times 0}{a+b} = 1$$

The distance between set A and set B is 1.

$$\because \text{Image of real segmentation and image which is predicted by the system have the same size.}$$

$$\therefore a = b$$

$$d_D(A,AB) = 1 - \frac{2 \times a}{a+a+b} = 1 - \frac{2 \times a}{2a+b} = [a = b\,] = \frac{1}{3}$$

$$d_D(B,AB) = 1 - \frac{2 \times b}{b+a+b} = 1 - \frac{2 \times b}{a+2b} = [a = b\,] = \frac{1}{3}$$

The distance between the third set and each of the other are one-third.

According to the triangle inequality, the sum of the lengths of any two sides must be greater than or equal to the length of the remaining side.

$$\because\ d_D(A,B) \nleq d_D(A,AB) + d_D(B,AB)$$

$\therefore$ The Dice coefficient does not satisfy triangle inequality.

The Jaccard distance d_J is defined as the following equation:

$$d_J = 1 - \frac{The\ intersection\ of\ A\ and\ B}{The\ union\ of\ A\ and\ B} = 1 - \frac{Dice\ coefficient}{2 - Dice\ coefficient}$$

So that,

$$d_J(A,B) = 1 - \frac{0}{2-0} = 1$$

$$d_J(A,AB) = 1 - \frac{2/3}{2-2/3} = \frac{1}{2}$$

$$d_J(B,AB) = 1 - \frac{2/3}{2-2/3} = \frac{1}{2}$$

$$d_J(A,B) \leqq d_J(A,AB) + d_J(B,AB)$$

The Jaccard distance satisfies triangle inequality.

4.3. Future work

The IKUB package and a shiny app were developed in this project. The advantage of the interactive shiny app is that users are able to use the developed functions without installing R or having knowledge of R. If all functions were to be redeveloped in the form of shiny app, it will attract more users.

In addition, developing new functions and adjusting the workflow according to the feedback from the users who download the IKUB package in order to improve the performance of the package. User feedback may also include ideas for new functionality. As the user base expands, I will seek to address their needs by developing additional functions. The functions on the IKUB package should be regularly maintained. Regular maintenance will ensure that all functions in the IKUB package work properly and are up-to-date.

Gathering systematic results on runtime performance was outside the scope of the project. The runtime performance analysis will be included in the future work in order to improve the processing efficiency.

5. Conclusion

A unified processing toolchain in R that takes MR brain images and their segmentations as input and produces image visualizations, data visualizations, and statistical analyses was developed in a unified manner. The developed functions included in the resulting toolchain were assembled as the IKUB package and released as open-source software. The complete source code can be download from Github (https://github.com/yupingikub/IKUB). The documentation contains descriptions and examples for the functions. A suggested workflow was also designed and included in the documentation. For the future function maintenance, the flowcharts of the functions are also drawn.

There is currently no standard workflow for brain image segmentation analysis. The designed workflow is a step towards standardizing. Future work for this project includes developing new functions and adjusting the workflow according to the feedback from the users who download the IKUB package in order to improve the performance of the package. In addition, maintaining the functions in the package in a timely manner ensures that the package is up-to-date.

The workflow design, programmed functions, and unified analysis platform for MR brain images are unique and novel contributions achieved in this master's project.

References

[1] B. Selene Xia and P. Gong, “Review of business intelligence through data analysis,” *Benchmarking Int. J.*, vol. 21, no. 2, pp. 300–311, Jan. 2014, doi: 10.1108/BIJ-08-2012-0050.

[2] Dr. Nakamur, Dr. Leverenz, and Dr. Wang, “Making the Most of Brain Imaging Through Quantitative Volumetric Analysis,” Cleveland Clinic, Jan. 2018. [Online]. Available: https://consultqd.clevelandclinic.org/making-the-most-of-brain-imaging-through-quantitative-volumetric-analysis/

[3] A. Vijayakumar and A. Vijayakumar, “Comparison of Hippocampal Volume in Dementia Subtypes,” *ISRN Radiol.*, vol. 2013, p. 174524, Dec. 2012, doi: 10.5402/2013/174524.

[4] P. I. Yakovlev and P. Rakic, “Patterns of decussation of bulbar pyramids and distribution of pyramidal tracts on two sides of the spinal cord.,” *Trans Am Neurol Assoc*, no. 91, pp. 366–367, 1966.

[5] X.-Z. Kong *et al.*, “Mapping Cortical and Subcortical Asymmetry in Obsessive-Compulsive Disorder: Findings From the ENIGMA Consortium,” *Biol. Psychiatry*, vol. 87, no. 12, pp. 1022–1034, Jun. 2020, doi: 10.1016/j.biopsych.2019.04.022.

[6] “Data Visualization,” *Statistical Analysis Software*. https://www.sas.com/en_us/insights/big-data/data-visualization.html

[7] “R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.” https://www.R-project.org/

[8] H. Wickham, D. Navarro, and T. L. Pedersen, “ggplot2: Elegant Graphics for Data Analysis.” https://github.com/hadley/ggplot2-book

[9] “Viridis colour scales from viridisLite.” https://ggplot2.tidyverse.org/reference/scale_viridis.html

[10] A. Hammers *et al.*, “Three-dimensional maximum probability atlas of the human brain, with particular reference to the temporal lobe,” *Hum. Brain Mapp.*, vol. 19, no. 4, pp. 224–247, Aug. 2003, doi: 10.1002/hbm.10123.

[11] I. S. Gousias *et al.*, “Automatic segmentation of brain MRIs of 2-year-olds into 83 regions of interest,” *NeuroImage*, vol. 40, no. 2, pp. 672–684, Apr. 2008, doi: 10.1016/j.neuroimage.2007.11.034.

[12] I. Faillenot, R. A. Heckemann, M. Frot, and A. Hammers, “Macroanatomy and 3D probabilistic atlas of the human insula,” *NeuroImage*, vol. 150, pp. 88–98, Apr. 2017, doi: 10.1016/j.neuroimage.2017.01.073.

[13] H. M. Wild, R. A. Heckemann, C. Studholme, and A. Hammers, “Gyri of the human parietal lobe: Volumes, spatial extents, automatic labelling, and probabilistic atlases,” *PLOS ONE*, vol. 12, no. 8, p. e0180866, Aug. 2017, doi: 10.1371/journal.pone.0180866.

[14] R. A. Heckemann, S. Keihaninejad, P. Aljabar, D. Rueckert, J. V. Hajnal, and A. Hammers, “Improving intersubject image registration using tissue-class information benefits robustness and accuracy of multi-atlas based anatomical segmentation,” *NeuroImage*, vol. 51, no. 1, pp. 221–227, May 2010, doi: 10.1016/j.neuroimage.2010.01.072.

[15] H. Wickham and G. Grolemund, *R for Data Science*. 2017. [Online]. Available: https://r4ds.had.co.nz/index.html

[16] *Convert3D*. [Online]. Available: http://www.itksnap.org/pmwiki/pmwiki.php?n=Downloads.C3D

[17] *NiftySeg*. [Online]. Available: https://github.com/KCL-BMEIS/NiftySeg

[18] *MIRTK*. [Online]. Available: https://github.com/BioMedIA/MIRTK

6. Acknowledgement

I dedicated this R package to my beloved father-in-law, Ivan Krizsan, who was born in Budapest, Hungary. Thanks to my supervisor for his patience and help. Thanks to Rebecca Henrysson for providing suggestions on package development. Finally, I appreciate the support from my family and friends.

Appendix 1 - IKUB Documentation

IKUB-DOCUMENTATION

Yu-Ping Hsu

2021-04-25

The 'IKUB' package includes functions that interact with MR brain images of class 'NIfTI', implemented by package 'oro.nifti', and their segmentations as input in order to produce image visualizations, data visualizations and statistical analyses.

The author made use of the Hammers Atlas Database, © Imperial College of Science, Technology and Medicine 2007. All rights reserved. The resource is available from www.brain-development.org [1]–[4]. In the demonstrations of function `segcomparison()` and `segcomparison3D()`, NIfTI files generated by MAPER [5] were used in order to compare segmentation regions differences. **Due to copyright issues, all examples in this documentation do not return output. Users can download the atlas from the above link and use the relevant functions to return the results. This package should not be used for clinical diagnosis.**

The suggested workflow for using developed functions in this package is attached at the end of this document for users' reference.

The packages and versions of packages used in this package are as follows.

```
dplyr (1.0.5)
EBImage (4.32.0)
graphics (4.0.4)
grDevices (4.0.4)
magrittr (2.0.1)
misc3d (0.9-0)
neurobase (1.29.0)
plyr (1.8.6)
purrr (0.3.4)
rgl (0.105.22)
stats (4.0.4)
tibble (3.1.1)
tidyr (1.1.3)
```

Users can install the development version of IKUB from GitHub with:

```r
# Install the devtools package
install.packages('devtools')
# Install the IKUB package
devtools::install_github("yupingikub/IKUB", build_vignettes = TRUE, force=
TRUE)
```

Please install the following packages before using the IKUB package:

```r
# Install packages from CRAN
install.packages(c('tidyverse', 'oro.nifti', 'neurobase', 'rgl', 'misc3d')
# Installation of EBImage package
install.packages("BiocManager")
BiocManager::install("EBImage")
```

`basymmetry()`

Brain asymmetry index A_r is calculated using the following equation:

$$A_r = \frac{2|V_R - V_L|}{V_R + V_L}$$

where:

V_R is the volume of the right brain region.

V_L is the volume of the left brain region.

The function calculates asymmetry index/indices for one or more supplied NIfTI-files. It takes as input a label name tibble with correct format and unique label numbers and label names. Option one returns one or more asymmetry indices for whole brain for supplied NIfTI-file(s). Option two returns asymmetry indices for the left and right regions for supplied NIfTI-file(s).

Usage:

```
basymmetry(filename, labelnametibble, option)
```

Arguments:

- filename: One or more NIfTI-files.
- labelnametibble: A label name tibble with correct format and unique columns. First column of the label name tibble should contain an integer label number and second column should contain a label name string. Label numbers and label names must be unique. The trailing characters of label names must be R or L.
- option: 1: Returns asymmetry indices for whole brain. 2: Returns asymmetry indices for the left and right regions. Default is 1.

```
# option=1
basymmetry(c('a01-seg.nii.gz','a02-seg.nii.gz'),labelnametibble, 1)

# option=2
basymmetry(c('a01-seg.nii.gz','a02-seg.nii.gz'),labelnametibble, 2)
```

`basymmetry2()`

Takes a label name tibble with correct format and unique label numbers and label names. First column of the label name tibble should be an integer label number and second column should be a label name string. The trailing characters of label names must be R or L. The function returns asymmetry index/indices for one or more supplied NIfTI-files. The generated output must be exported to CSV format in order to be made available for further analysis in shiny app-violinplot.

Usage:

```
basymmetry2(filename, labelnametibble)
```

Arguments:

- filename: One or more NIfTI-files.

- labelnametibble: A label name tibble with correct format and unique columns. First column of the label name tibble should contain an integer label number and second column should contain a label name string. Label numbers and label names must be unique. The trailing characters of label names must be R or L.

```
filename <- c(paste('a0', 1:9, '-seg.nii.gz', sep=''), paste('a', 10:30, '
-seg.nii.gz', sep=''))
output <- basymmetry2(filename, labelnametibble)
# Gets a temporary directory to store the CSV file in.
path <- tempdir()
# Name of the CSV file that the output will be stored in.
file_n <- 'asymmetry'
file_p <- file.path(path, paste0(file_n, ".csv"))
# Exports the output from basymmetry2 function to a CSV file.
write.csv(output, file_p, row.names = FALSE, na = "NA")
```

`bsimilarity()`

Calculates Dice coefficients and Jaccard indices for a supplied NIfTI file pair. Only regions which are in the label name tibble, target labels will be included in the result.

Usage:

```
bsimilarity(filename1, filename2, labelnametibble)
```

Arguments:

- filename1: A filename for one NIfTI-file.
- filename2: A filename for one NIfTI-file.
- labelnametibble: A label name tibble with correct format and unique columns. First column of the label name tibble should contain an integer label number and second column should contain a label name string. Label numbers and label names must be unique.

```
bsimilarity('a01-seg.nii.gz', 'a1.nii.gz', labelnametibble)
```

`bsimilarity2()`

The function is used to generate Jaccard indices for all regions for one or more file pairs. The generated output must be exported to CSV format in order to be made available for further analysis in shiny app-violinplot.

Usage:

```
bsimilarity2(filename1, filename2, labelnametibble)
```

Arguments:

- filename1: One or more NIfTI-files.
- filename2: One or more NIfTI-files.
- labelnametibble: A label name tibble with correct format and unique columns. First column of the label name tibble should contain an integer label number and second column should contain a label name string. Label numbers and label names must be unique.

```r
filename1 <- c(paste('a0', 1:9, '-seg.nii.gz', sep=''), paste('a', 10:30,
'-seg.nii.gz', sep=''))
filename2 <- c(paste('a', 1:30, '.nii.gz', sep=''))
output <- bsimilarity2(filename1, filename2, labelnametibble)
# Gets a temporary directory to store the CSV file in.
path <- tempdir()
# Name of the CSV file that the output will be stored in.
file_n <- 'Jaccardindex'
file_p <- file.path(path, paste0(file_n, ".csv"))
# Exports the output from bsimilarity2 function to a CSV file.
write.csv(output, file_p, row.names = FALSE, na = "NA")
```

`bzscore()`

Converts a raw score x into a Standard score (Z-score) using the following equation:

$$z = \frac{x - \mu}{\sigma}$$

where:

μ is the mean of the reference group.

σ is the standard deviation of the reference group.

Calculates Z-scores of all regions. Inputs one or more NIfTI-files from one or more individuals, for which Z-scores are to be calculated, a label name tibble and a tibble including mean values and standard deviations from the reference group for one or more individuals. Returns Z-scores with label names for regions which are in the label name tibble.

Usage:

```r
bzscore(filename, labelnametibble, refstats)
```

Arguments:

- filename: One or more NIfTI-files for individuals from which Z-scores are to be calculated.
- labelnametibble: A label name tibble.
- refstats: A tibble which was previously calculated by the refmeanstd function.

```r
# Restore refstats tibble saving in computer
refstats <- readRDS("refstats.rds")
bzscore(c('a01-seg.nii.gz', 'a02-seg.nii.gz'), labelnametibble, refstats)
```

`bzscore2()`

Calculates Z-scores of all regions. Inputs one or more NIfTI-files from one or more individuals, for which Z-scores are to be calculated, a label name tibble and a tibble including mean values and standard deviations from the reference group for one or more individuals. Generates output including Z-scores with label names for regions which are in the label name tibble. The generated output must be exported to CSV format in order to be made available for further analysis in shiny app-violinplot.

Usage:

```
bzscore2(filename, labelnametibble, refstats)
```

Arguments:

- filename: One or more NIfTI-files for individuals from which Z-scores are to be calculated
- labelnametibble: A label name tibble.
- refstats: A tibble which was previously calculated by the refmeanstd function.

```
filename <- c(paste('a0', 1:9, '-seg.nii.gz', sep=''), paste('a', 10:30, '
-seg.nii.gz', sep=''))
output <- bzscore2(filename, labelnametibble, refstats)
# Gets a temporary directory to store the CSV file in.
path <- tempdir()
# Name of the CSV file that the output will be stored in.
file_n <- 'zscore'
file_p <- file.path(path, paste0(file_n, ".csv"))
# Exports the output from bzscore2 function to a CSV file.
write.csv(output, file_p, row.names = FALSE, na = "NA")
```

`check_label()`

Checks if Label_No and Label_name columns exist. Checks if label number is an integer and if there are any duplicate label numbers or label names in the tibble. First column of the label name tibble must contain an integer label number and the second column must contain a label name string. Label numbers and label names must be unique. Returns an error message or a label name tibble.

`check_label_RGB()`

Checks if Label_No, Label_name, R, G and B columns exist. Checks if label number is an integer and if there are any duplicate label numbers or label names in the tibble. First column of the label name tibble must contain an integer label number and the second column must contain a label name string. Label numbers and label names must be unique. Returns an error message or a label name tibble.

`element_count()`

Calculates total number of counts in both images and counts for overlap. Inputs one slice number and one label number. The function calculates counts for total number of pixels in both images, pixels for overlap, pixels only in image1 and pixels only in image2 in the given slice for the given label number. This function is primarily used by other functions.

`extractsliceno()`

Extracts slice numbers for one or more supplied label numbers and anatomical plane. Users input one NIfTI-file and one or more target label numbers and anatomical plane. The function returns slice numbers which include supplied target label(s) for supplied anatomical plane.

Usage:

```
extractsliceno(filename, target_label, anatomical_plane)
```

Arguments:

- filename: One NIfTI-file.
- target_lable: One target label number.
- anatomical_plane: 1 for coronal plane, 2 for sagittal plane and 3 for transverse plane. Default is 3.

```
extractsliceno('a01-seg.nii.gz', 15:16)
```

`files_element()`

Inputs a file pair. The function returns total number of counts in both images and counts for overlap, counts only in image1, counts only in image2 for given filenames. This function is primarily used by other functions.

`labelnaming()`

Calculates volumes for all regions with label name. User inputs one or more NIfTI-files and a label name tibble. Returns total volumes, with label names, for all regions in the supplied NIfTI-file(s). A label name tibble must have correct format and unique columns. First column of the label name tibble must contain an integer label number and second column must contain a label name string. Label numbers and label names must be unique. Only regions which are in the label name tibble will be included in the result.

Usage:

```
labelname(filename,labelnametibble)
```

Arguments:

- filename: One or more NIfTI-files.
- labelnametibble: A label name tibble.

```
labelnaming(c('a01-seg.nii.gz', 'a02-seg.nii.gz'), labelnametibble)
```

`labelnaming2()`

Calculates volumes for all regions with label names. User inputs one or more NIfTI-files and a label name tibble. Saves total volumes, with label names, for all regions in the supplied NIfTI-file in a csv file in the working directory. A label name tibble must have correct format and unique columns. First column of the label name tibble must contain an integer label number and second column must contain a label name string. Label numbers and label names must be unique. Only regions which are in the label name tibble will be included in the result. The generated output must be exported to CSV format in order to be made available for further analysis in shiny app-violinplot.

Usage:

```
labelname2(filename,labelnametibble)
```

Arguments:

- filename: One or more NIfTI-files.

- labelnametibble: A label name tibble.

```
filename <- c(paste('a0', 1:9, '-seg.nii.gz', sep=''), paste('a', 10:30, '
-seg.nii.gz', sep=''))
output <- labelnaming(filename, labelnametibble)
# Gets a temporary directory to store the CSV file in.
path <- tempdir()
# Name of the CSV file that the output will be stored in.
file_n <- 'brainvolume'
file_p <- file.path(path, paste0(file_n, ".csv"))
# Exports the output from labelnaming2 function to a CSV file.
write.csv(output, file_p, row.names = FALSE, na = "NA")
```

`refmeanstd()`

Calculates mean and standard deviation for all regions in a reference group. Reference group must contain at least two NIfTI-files. Returns the mean values and standard deviations for all regions.

Usage:

```
refmeanstd(filename)
```

Arguments:

- filename: At least two NIfTI-file filenames.

Generating the refstats tibble from a certain reference group and saving it will allow for reuse across sessions that uses the same reference group which will save time and effort.

```
reflabels<-c(paste('a0', 1:9, '-seg.nii.gz', sep=''), paste('a', 10:29, '-
seg.nii.gz', sep='') )
refstats <- refmeanstd(reflabels)
head(refstats,6)

# Gets a temporary directory to store the RDS file in.
path <- tempdir()
# Name of the RDS file that the output will be stored in.
file_n <- 'refstats'
file_p <- file.path(path, paste0(file_n, ".rds"))
# Use function saveRDS() to save refstats tibble in an RDS file.
saveRDS(refstats, file=file_p)
```

`regcount()`

Calculates total counts and total volumes for all regions in the supplied NIfTI-file.

Usage:

```
regcount(filename)
```

Arguments:

- filename: A NIfTI-file filename.

```
regcount('a01-seg.nii.gz')
```

`segcomparison()`

Displays label segmentation differences in supplied MR image for the supplied label number. Region only present in filename_seg1 is displayed in green color. Region only present in filename_seg2 is displayed in blue color. Region present in both files is displayed in red color.

Usage:

```
segcomparison(filename_MR ,filename_seg1, filename_seg2, Label_No, anatomi
cal_plane, rowno, colno)
```

Arguments:

- filename_MR: A NIfTI-file of a MR image.
- filename_seg1: First NIfTI-file of segmentation.
- filename_seg1: Second NIfTI-file of segmentation.
- Label_no: One label number for which region is to be displayed.
- anatomical_plane: 1 for coronal plane, 2 for sagittal plane and 3 for transverse plane. Default is 3.
- rowno: Number of rows for showing images. Default is 1.
- colno: Number of columns for showing images. Default is 1.

```
segcomparison('a01.nii.gz' ,'a01-seg.nii.gz', 'a1.nii.gz', 60)
```

`segcomparison3D()`

Displays label segmentation differences in supplied MR image for supplied label number in a 3D view. Region only present in filename_seg1 is displayed in green color. Region only present in filename_seg2 is displayed in blue color. Region present in both files is displayed in red color. 3D view can be rotated by dragging the mouse.

`segcomparison3D()` requires the rgl and misc3d packages.

Usage:

```
segcomparison3D(filename_MR ,filename_seg1, filename_seg2, targetlabel)
```

Arguments:

- filename_MR: A NIfTI-file of an MR image.
- filename_seg1: First NIfTI-file of segmentation (displays in green color).
- filename_seg1: Second NIfTI-file of segmentation (displays in blue color).
- targetlabel: One label number for which region is to be displayed.

```
segcomparison3D('a01.nii.gz' ,'a01-seg.nii.gz', 'a1.nii.gz', 60)
```

`segcontour()`

Displays anatomical segmentation contours of a medical MR image with one or more slices. User inputs one NIfTI-file of segmentation, one NIfTI-file of MR image, a labelnametibble with correct format and unique input columns, one or more numbers of target-slices, one or more target-labels, for which anatomical segementations are to be displayed, an anatomical plane and the number of rows and columns for combining multiple plots into one overall graph. Parameters for the anatomical plane are 1 for coronal plane, 2 for sagittal plane and 3 for transverse plane. Returns image(s) with segmentation contour(s) for every slice with legend. Only contours which are in the label name tibble and input label numbers will be displayed in the result.

`segcontour()` requires the EBImage package for image processing.

Usage:

```
segcontour(filename_MR, filename_seg, labelnametibble, slice_no, target_la
bel, anatomical_plane, rowno, colno)
```

Arguments:

- filename_MR: A NIfTI-file of an MR image.
- filename_seg: A NIfTI-file of segmentation.
- labelnametibble: A label name tibble with correct format and unique columns. First column of the label name tibble must contain an integer label number and second column must contain a label name string. Label numbers and label names must be unique.
- slice_no: One or more slice numbers for which label contours are to be displayed.
- target_label: One or more label numbers for which label contours are to be displayed.
- anatomical_plane: 1 for coronal plane, 2 for sagittal plane and 3 for transverse plane. Default is 3.
- rowno: Number of rows for showing images. Default is 1.
- colno: Number of columns for showing images. Default is 1.

```
segcontour('a01.nii.gz', 'a01-seg.nii.gz', labelnametibble, 57:59, 17:18,
1, 2, 2)
```

`segsurface3D()`

Displays anatomical segmentation surface of a medical MR image in a 3D view for the supplied label numbers. 3D view can be rotated by dragging the mouse. Returns 3D view with segmentations and a legend for label surfaces with label names. Only surfaces which are in the label name tibble and input label numbers will be displayed in the result. The different segments will have the color from the R, G, B columns in the labelnametibble.

`segsurface3D()` requires the packages rgl and misc3d.

Usage:

```
segsurface3D(filename_MR, filename_seg, labelnametibble, targetlabel)
```

Arguments:

- filename_MR: A NIfTI-file of an MR image.
- filename_seg: A NIfTI-file of segmentation.
- labelnametibble: A label name tibble with correct format and unique columns. First column of the label name tibble should contain an integer label number and second column should contain a label name string. Label numbers and label names must be unique. Three columns, R, G, B, should be included in the labelnametibble.
- targetlabel: One or more label numbers for which label contours are to be shown.

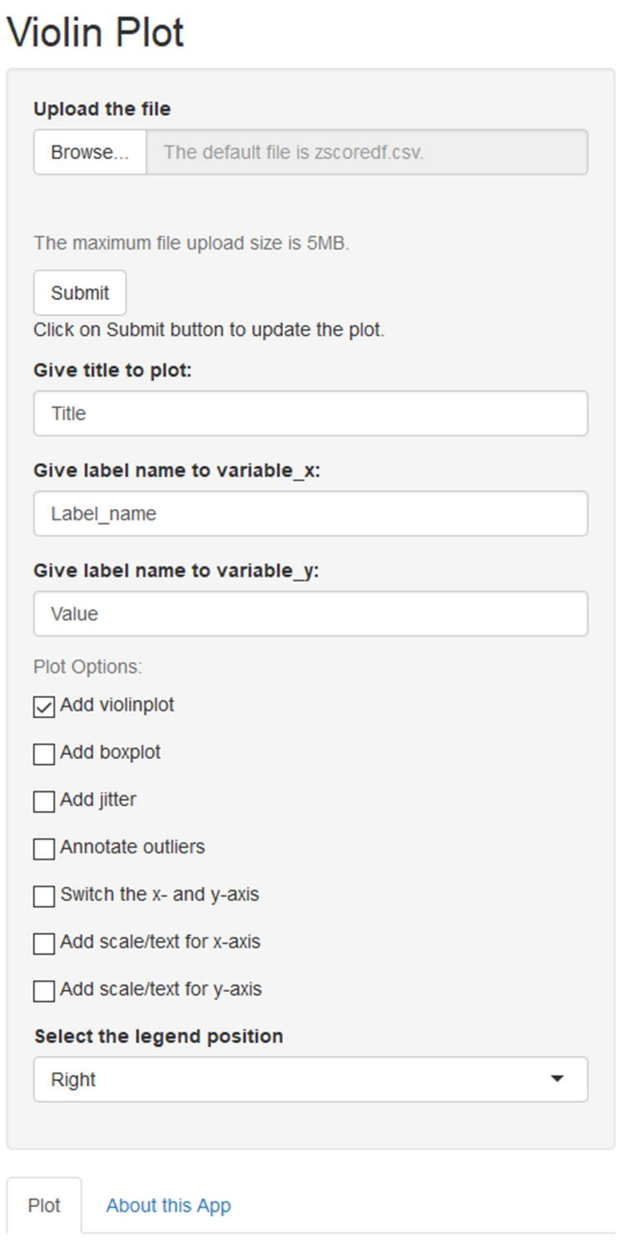
```
segsurface3D('a01.nii.gz', 'a01-seg.nii.gz', labelnametibble, c(5,6))
```

`slice_label()`

Extracts label numbers using slices. Inputs one slice number. Returns a tibble with two columns, which are slice number and label number. This function is primarily used by other functions.

`shiny_violinplot app`

```
knitr::include_app("https://alisonhyp.shinyapps.io/shiny_violinplot/", hei
ght = "400px")
```


Violin Plot

Upload the file

Browse... The default file is zscoredf.csv.

The maximum file upload size is 5MB.

Submit

Click on Submit button to update the plot.

Give title to plot: Title

Give label name to variable_x: Label_name

Give label name to variable_y: Value

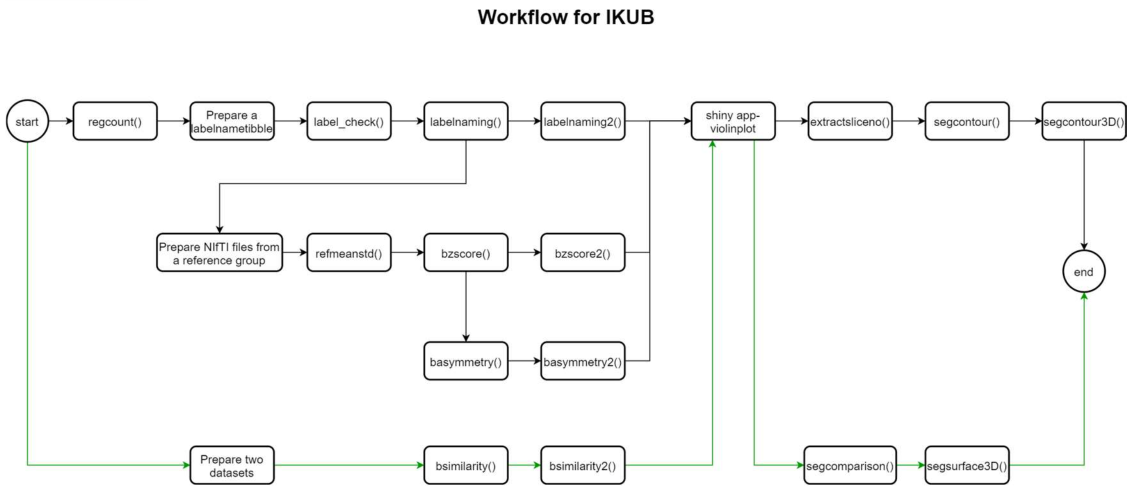
Plot Options:

- [x] Add violinplot
- [ ] Add boxplot
- [ ] Add jitter
- [ ] Annotate outliers
- [ ] Switch the x- and y-axis
- [ ] Add scale/text for x-axis
- [ ] Add scale/text for y-axis

Select the legend position: Right

Shiny_violinplot is a shiny app which is designed to analyze data generated by functions `labelnaming2()`, `bzscore2()`, `bsimilarity2()` or `basymmetry2()`. Users use these functions to generate csv files and upload files to app with link https://alisonhyp.shinyapps.io/shiny_violinplot/. The maximum file upload size is 5MB. Items in `Select item(s)` are unique elements which are extracted from data in the variable_x column in the input csv file. `Select item(s)` allows for selecting multiple input. The app will plot a violinplot after users submit the selection. User can click on checkboxes to activate or deactivate plot options. The position of the legend can be also selected by users. Users can define their own titles, label names for x- and y-axis.

Workflow



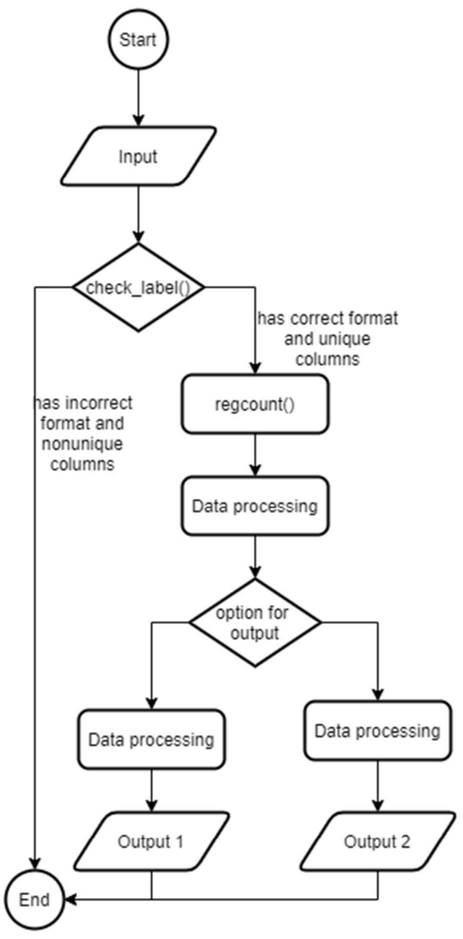
Workflow for IKUB

References

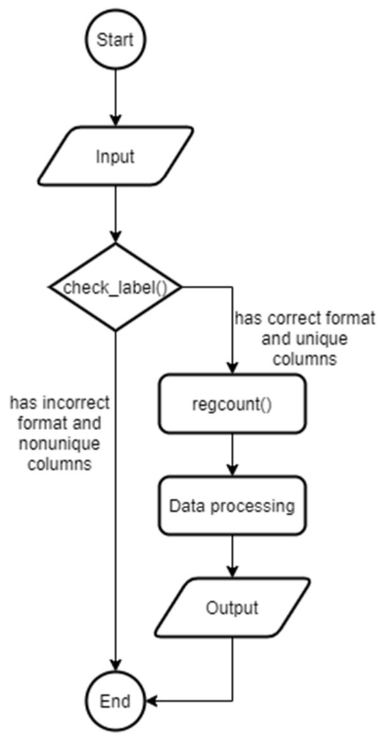
1. A. Hammers et al., "Three-dimensional maximum probability atlas of the human brain, with particular reference to the temporal lobe," Hum. Brain Mapp., vol. 19, no. 4, pp. 224–247, Aug. 2003, doi: 10.1002/hbm.10123.
2. I. S. Gousias et al., "Automatic segmentation of brain MRIs of 2-year-olds into 83 regions of interest," NeuroImage, vol. 40, no. 2, pp. 672–684, Apr. 2008, doi: 10.1016/j.neuroimage.2007.11.034.
3. I. Faillenot, R. A. Heckemann, M. Frot, and A. Hammers, "Macroanatomy and 3D probabilistic atlas of the human insula," NeuroImage, vol. 150, pp. 88–98, Apr. 2017, doi: 10.1016/j.neuroimage.2017.01.073.
4. H. M. Wild, R. A. Heckemann, C. Studholme, and A. Hammers, "Gyri of the human parietal lobe: Volumes, spatial extents, automatic labelling, and probabilistic atlases," PLOS ONE, vol. 12, no. 8, p. e0180866, Aug. 2017, doi: 10.1371/journal.pone.0180866.
5. R. A. Heckemann, S. Keihaninejad, P. Aljabar, D. Rueckert, J. V. Hajnal, and A. Hammers, "Improving intersubject image registration using tissue-class information benefits robustness and accuracy of multi-atlas based anatomical segmentation," NeuroImage, vol. 51, no. 1, pp. 221–227, May 2010, doi: 10.1016/j.neuroimage.2010.01.072.

Appendix 2 - Flowchart of function

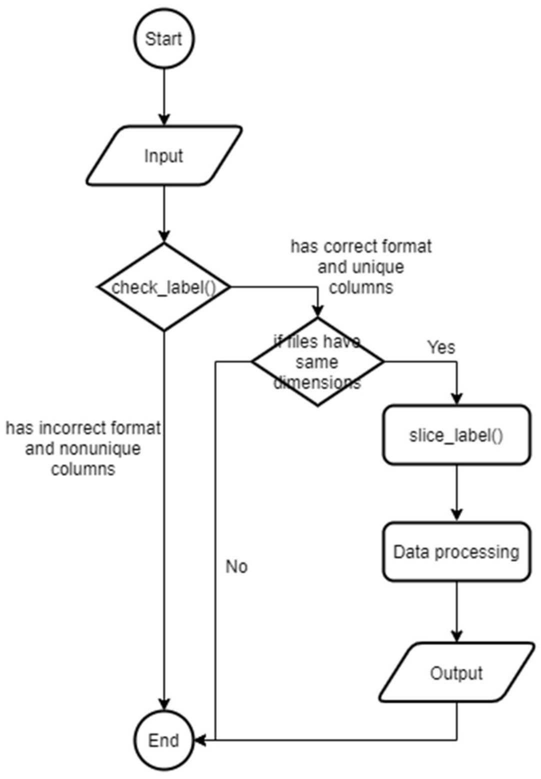
basymmetry()



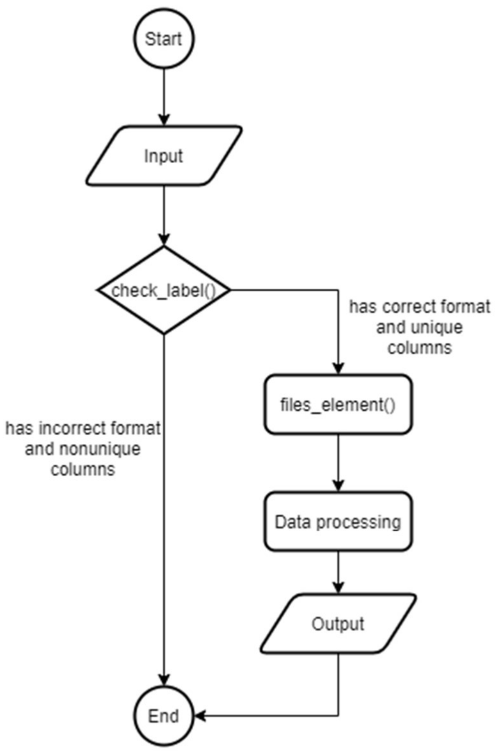
basymmetry2()


Start
Input
check_label()
has correct format and unique columns
has incorrect format and nonunique columns
regcount()
Data processing
Output
End

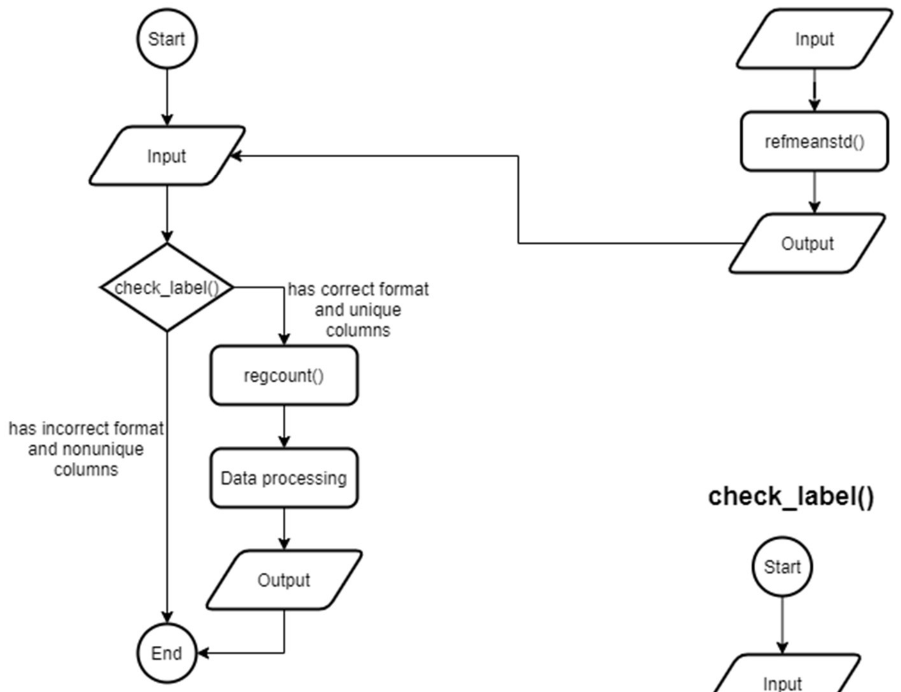
bsimilarity()


Start
Input
check_label()
has correct format and unique columns
has incorrect format and nonunique columns
if files have same dimensions
Yes
No
slice_label()
Data processing
Output
End

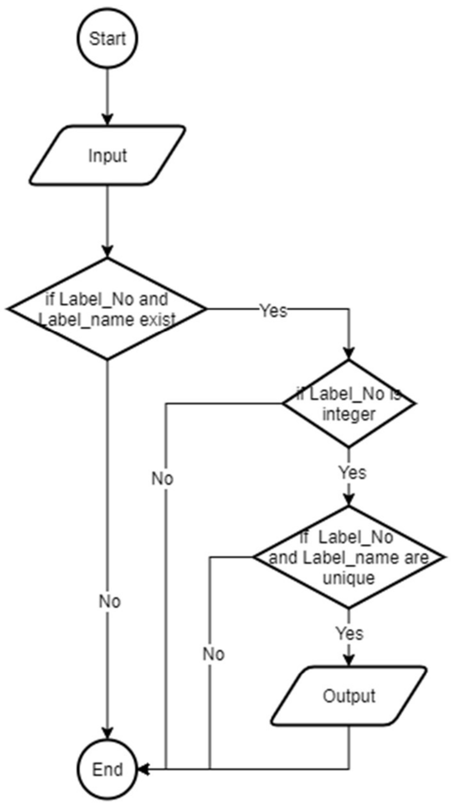
bsimilarity2()


Start
Input
check_label()
has correct format and unique columns
has incorrect format and nonunique columns
files_element()
Data processing
Output
End

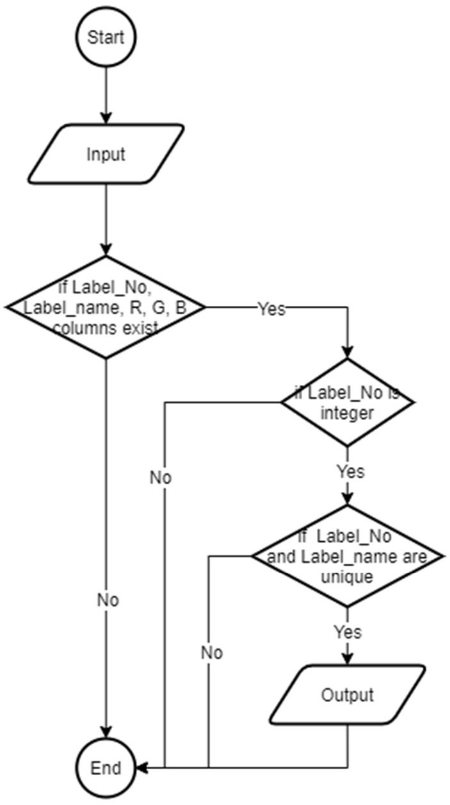
bzscore()/bzscore2()



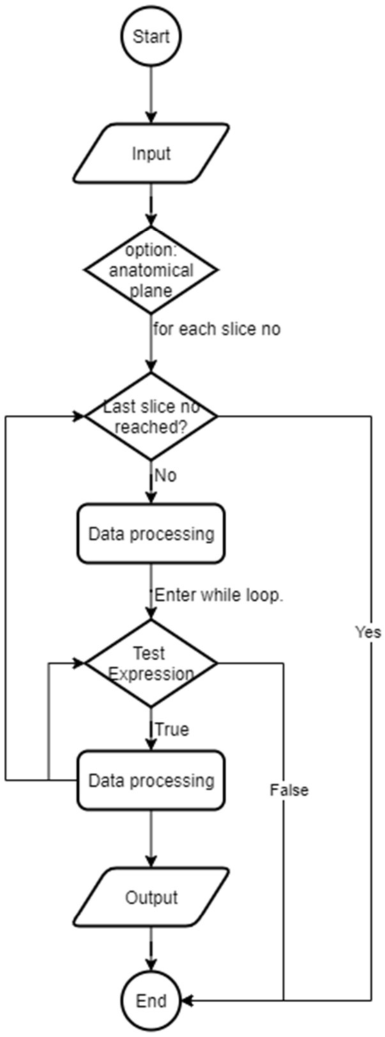
check_label()


Start
Input
if Label_No and Label_name exist
Yes
No
if Label_No is integer
Yes
No
if Label_No and Label_name are unique
Yes
No
Output
End

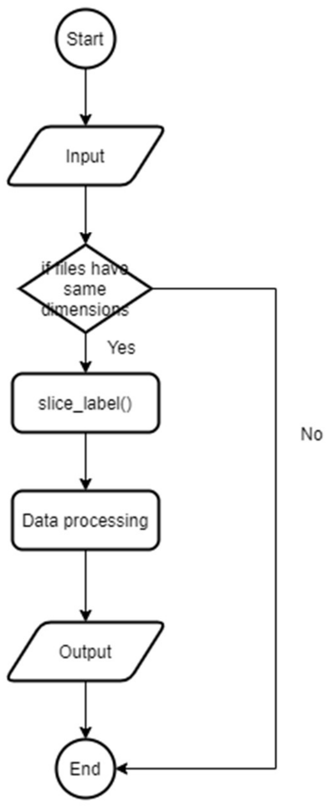
check_label()


Start
Input
if Label_No, Label_name, R, G, B columns exist
Yes
if Label_No is integer
Yes
if Label_No and Label_name are unique
Yes
Output
No
No
No
End

extractsliceno()

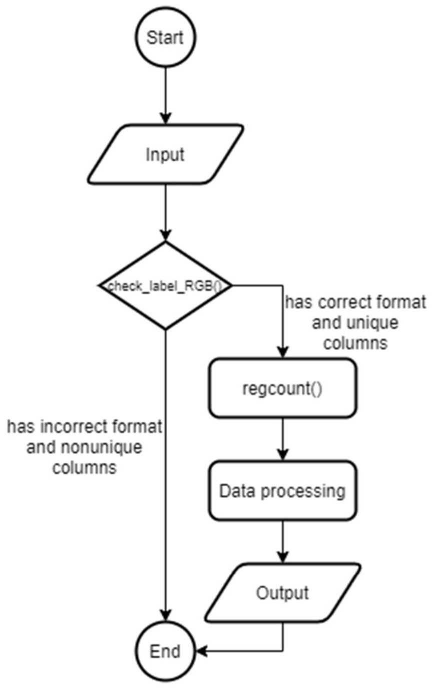


files_element()

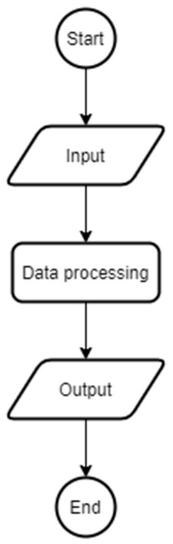


labelnaming()/

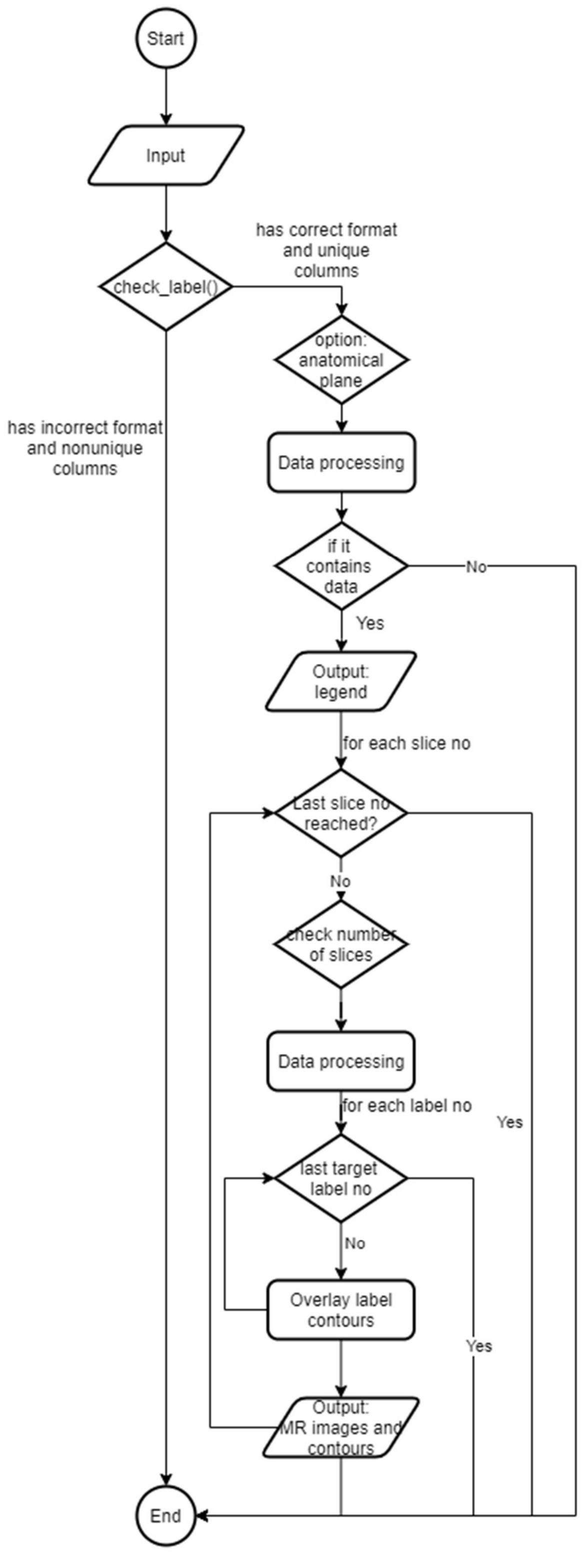
labelnaming2()


Start
Input
check_label_RGB()
has correct format and unique columns
has incorrect format and nonunique columns
regcount()
Data processing
Output
End

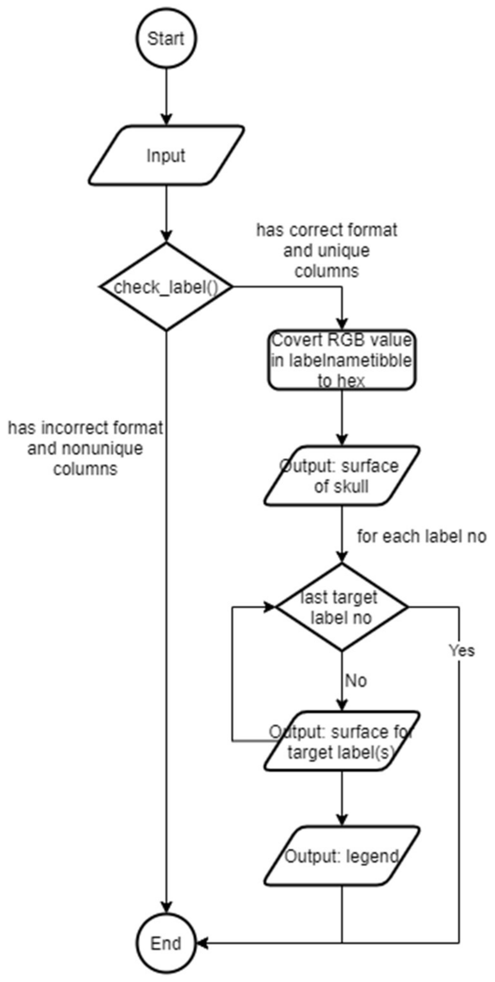
regcount()



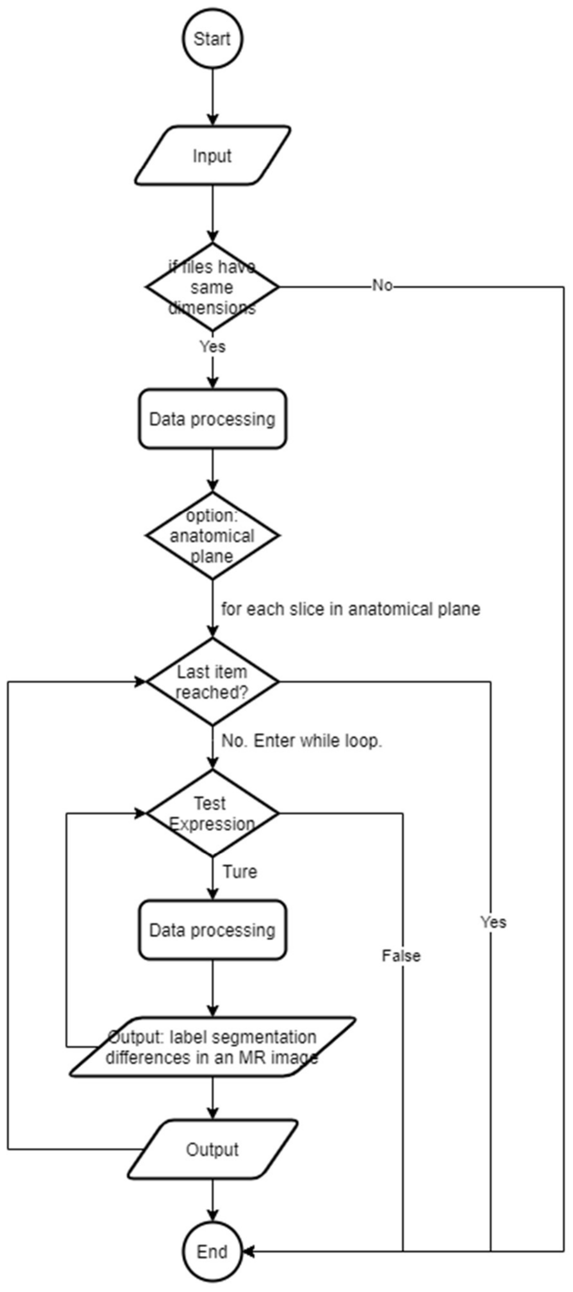
segcontour()



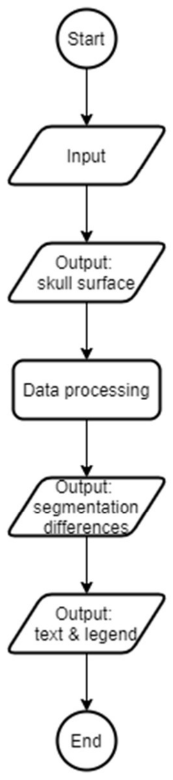
segsurface3D()


Start
Input
check_label()
has correct format and unique columns
has incorrect format and nonunique columns
Covert RGB value in labelnametibble to hex
Output: surface of skull
for each label no
last target label no
Yes
No
Output: surface for target label(s)
Output: legend
End

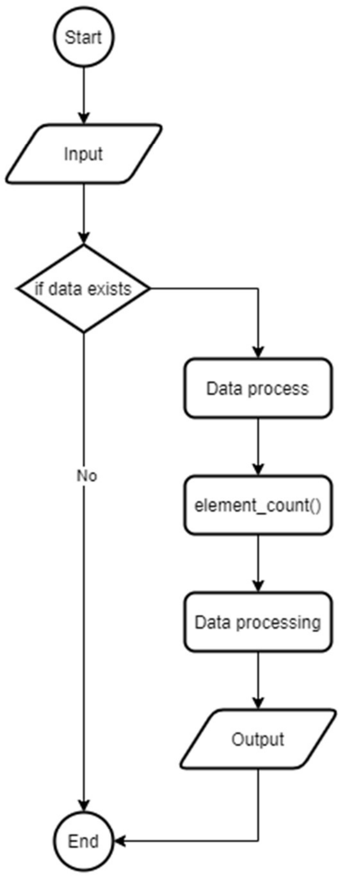
segcomparison()



segcomparison3D()



slice_label


Start
Input
if data exists
No
Data process
element_count()
Data processing
Output
End