ENIGMA TOOLBOX

Sara Larivière, Boris Bernhardt

Feb 21, 2023

GETTING STARTED

CHAPTER

ONE

100+ ENIGMA-DERIVED STATISTICAL MAPS

The **ENIGMA TOOLBOX** provides a centralized, and continuously updated, repository of meta-analytical *case-control comparisons* across a wide range of disorders. As many ENIGMA groups have moved beyond meta-analysis towards 'mega'-analysis of subject-level data, the **ENIGMA TOOLBOX** also includes an example of subject-level *example data* from an individual site that have been processed according to ENIGMA protocols. It should be noted, however, that subject-level or raw imaging data are not openly available for dissemination to the scientific community. Interested scientists are encouraged to visit the ENIGMA website to learn more about current projects, joining and contributing to an active Working Group, or proposing a new group.

CHAPTER

TWO

COMPATIBLE WITH ANY NEUROIMAGING DATASETS AND SOFTWARE

To increase generalizability and usability, every function within the **ENIGMA TOOLBOX** is also compatible with typical neuroimaging datasets parcellated according to the Desikan-Killiany, Glasser, and Schaefer parcellations.

To simplify things, we provide tutorials on how to (*i*) *import vertexwise and/or parcellated data*, (*ii*) *parcellate vertexwise cortical data*, (*iii*) *map parcellated data to the surface*, and (*iv*) *export data results*. Import/export file formats include: .txt/.csv, FreeSurfer/"curv". mgh/.mgz, GIfTI/.gii, and CIfTI/.dscalar.nii/.dtseries.nii. Cortical and subcortical surfaces are also available in FreeSurfer/surface, GIfTI/.gii, .vtk, and .obj formats, allowing cross-software compatibility.

CHAPTER

THREE

CORTICAL AND SUBCORTICAL VISUALIZATION TOOLS

Tired of displaying your surface findings in tables? Look no further! The **ENIGMA TOOLBOX** has got you covered! Check out our *visualization tools* to project cortical and subcortical data results to the surface and generate publication-ready figures.

CHAPTER

FOUR

PREPROCESSED MICRO- AND MACROSCALE DATA

The emergence of open databases yields new opportunities to link human gene expression, cytoarchitecture, and connectome data. As part of the **ENIGMA TOOLBOX**, we have generated and made available a range of preprocessed and parcellated data, including: (*i*) *transcriptomic data* from the Allen Human Brain Atlas, (*ii*) *cytoarchitectural data* from the BigBrain Project, (*iii*) a *digitized parcellation* of the von Economo and Koskinas cytoarchitectonic map, and (*iv*) *functional and structural connectivity data* from the Human Connectome Project.

CHAPTER

FIVE

MULTISCALE ANALYTICAL WORKFLOWS

The **ENIGMA Toolbox** comprises two neural scales for the contextualization of findings: (*i*) using microscale properties, namely *gene expression* and *cytoarchitecture*, and (*ii*) using macroscale network models, such as *regional hub susceptibility analysis* and *disease epicenter mapping*. Moreover, our Toolbox includes non-parametric *spatial permutation models* to assess statistical significance while preserving the spatial autocorrelation of parcellated brain maps. From these comprehensive workflows, users can gain a deeper understanding of the molecular, cellular, and macroscale network organization of the healthy and diseased brains.

CHAPTER

SIX

STEP-BY-STEP TUTORIALS

The **ENIGMA TOOLBOX** includes ready-to-use and easy-to-follow code snippets for every functionality and analytical workflow! Owing to its comprehensive tutorials, detailed functionality descriptions, and visual reports, the **ENIGMA TOOLBOX** is easy to use for researchers and clinicians without extensive programming expertise.

CHAPTER

SEVEN

DEVELOPMENT AND GETTING INVOLVED

Should you have any problems, questions, or suggestions about the **ENIGMA TOOLBOX**, please do not hesitate to post them to our mailing list! Are you interested in collaborating or sharing your ENIGMA-related data/codes/tools? Noice! Make sure you familiarize yourself with our contributing guidelines first and then discuss your ideas on our Github issues and pull request.

7.1 Installation

ENIGMA TOOLBOX is available in Python and Matlab!

Python

ENIGMA TOOLBOX has the following dependencies:

- numpy
- matplotlib
- vtk
- nibabel
- pillow
- pandas
- scipy
- scikit-learn
- nilearn

The **ENIGMA TOOLBOX** can be directly downloaded from Github as follows:

```
git clone https://github.com/MICA-MNI/ENIGMA.git
cd ENIGMA
python setup.py install
```

Matlab

ENIGMA TOOLBOX was tested with Matlab R2017b.

To install the Matlab toolbox simply download and unzip the GitHub toolbox (slow) or run the following command in your terminal (fast):

```
git clone https://github.com/MICA-MNI/ENIGMA.git
```

Once you have the toolbox on your computer, simply run the following command in Matlab:

```
addpath(genpath('/path/to/ENIGMA/matlab/'))
```

7.2 Usage notes

The **ENIGMA TOOLBOX** includes a set of commonly used tools accompanied by validated, ready-to-use, and easy-to-follow code snippets. These code snippets will guide you through each analysis step, starting from loading ENIGMA-derived datasets to visualizing findings from advanced secondary analyses.

To accommodate a larger scientific audience, we present code snippets in both *Python* and *Matlab*. Tutorials in the **ENIGMA TOOLBOX** are also compatible with individual site data or data derived from a mega-analysis (**mega**) as well as data obtained from a meta-analytical studies (**meta**). Each available option is presented in their respective tabs as shown below; if tabs do not specify **meta** or **mega**, then the code is generalizable to both dataset options.

Python | meta

Matlab | meta

If you have **meta**-analysis data (*e.g.*, summary statistics)
If you have individual site or **mega**-analysis data

Python | mega

Matlab | mega

Some tutorials may have specific prerequisite steps; if that is the case, we provided the links to the steps you must complete first as follows:

Prerequisites
Install me!

7.3 What's new?

7.3.1 v2.0.0 (July 26, 2022)

One year of bug fixing and brand spanking new BigBrain moments

```
[FIX ] Bug fixes all throughout                    |  @saratheriver
[DOC ] Updated documentations                      |  @saratheriver
[NEW ] New BigBrain moments                        |  @caseypaquola
```

7.3.2 v1.1.3 (July 28, 2021)

Typo in `economo_koskinas_spider` matlab function.

```
[FIX ] Fixed typo in economo_koskinas_spider        |  @saratheriver
[DOC ] Updated citation                             |  @saratheriver
```

7.3.3 v1.1.2 (June 30, 2021)

Replaced dashes with underscores in summary statistics.

```
[FIX ] Removed dashes in sumstats                   |  @saratheriver
[DOC ] Update documentations                        |  @saratheriver
```

7.3.4 v1.1.1 (April 12, 2021)

New import/export features for CIfTI files. Improved documentation.

```
[NEW ] New import/export module (CIfTI)             |  @NicoleEic, @saratheriver
[DOC ] Update documentations                        |  @saratheriver
```

7.3.5 v1.1.0 (March 15, 2021)

New import/export features allowing compatibility with other dataset formats and neuroimaging softwares. Improved documentation.

```
[NEW ] New import/export module                     |  @saratheriver
[FIX ] Fix spin permutation index error             |  @saratheriver
[DOC ] Update documentations                        |  @saratheriver
```

7.3.6 v1.0.3 (February 25, 2021)

OCD subcortical volume measures were not loading when using `load_summary_stats('ocd')`; this issue is now resolved.

```
[ENH ] Fix OCD sctx summary stats loading issue     |  @saratheriver
[DOC ] Update documentations                        |  @saratheriver
```

7.3.7 v1.0.2 (February 23, 2021)

Bug fixes in the `fetch_ahba` function. Users who wish to use this function (or have used it in a previous version) must update their **ENIGMA Toolbox** to this release; `fetch_ahba` from previous releases will now error.

```
[ENH ] Fix python version mislabeling               |  @saratheriver
[ENH ] Fix bug and replace stable gene lists        |  @saratheriver
[DOC ] Update documentations                        |  @saratheriver
```

7.3.8 v1.0.1 (January 11, 2021)

Bug fixes and improvement

```
[ENH ] Enhance cross-disorder module                  |  @saratheriver
[ENH ] Fix bug in surface visualization               |  @saratheriver
```

7.3.9 v1.0.0 (December 22, 2020)

Initial release

```
[DOC ] Citation for ENIGMA Toolbox preprint           |  @saratheriver
[NEW ] Pip package available                          |  @saratheriver
```

7.3.10 v0.1.2 (December 20, 2020)

Bug fixes and improvement - still in pre-release

```
[DOC ] Revise documentations                          |  @saratheriver
[NEW ] Generalizability to several parcellations      |  @saratheriver
[NEW ] Additional parcellations for HCP connectivity  |  @saratheriver
[NEW ] Additional parcellations for AHBA              |  @saratheriver
[NEW ] Additional parcellations for histology module  |  @saratheriver
[NEW ] Cross-disorder module                          |  @boyongpark, @saratheriver
[ENH ] Fix bug in fetch_ahba                          |  @saratheriver
[ENH ] Reference updates                              |  @saratheriver
```

7.3.11 v0.1.1 (November 14, 2020)

Bug fixes and improvement following beta testing - still in pre-release

```
[TES ] Toolbox testing                                |  @boyngpark, @royj23,␣
↪@caseypaquola, @YezhouWang, @sofievalk
[DOC ] Revise documentations                          |  @saratheriver
[ENH ] Fix minor code bugs here and there             |  @saratheriver
[ENH ] Remove unstable genes across donors            |  @saratheriver
[NEW ] BigBrain contextualization module              |  @caseypaquola, @saratheriver
[NEW ] Cytoarchitectonics contextualization module    |  @caseypaquola, @saratheriver
[NEW ] HCP subcortico-subcortical connectivity        |  @saratheriver
[ENH ] References for new modules                     |  @saratheriver
[ENH ] Update API for new functions                   |  @saratheriver
```

7.3.12 v0.1.0 (October 16, 2020)

Pre-release of the **ENIGMA TOOLBOX**: it's the start of a beautiful thing!

```
[DOC ] Write and revise various documentations       |  @saratheriver
[STY ] Stylistic updates to ReadTheDocs              |  @saratheriver
[NEW ] Load example data                             |  @saratheriver
[NEW ] Load summary statistics                       |  @saratheriver
[NEW ] Load connectivity data                        |  @saratheriver
[NEW ] Hub susceptibility model                      |  @saratheriver
[NEW ] Spin permutation tests                        |  @saratheriver
[NEW ] Epicenter mapping                             |  @saratheriver
[NEW ] Fetch gene expression data                    |  @saratheriver
[NEW ] Extract disease-related gene expression maps  |  @saratheriver
[NEW ] Surface data visualization tools              |  @saratheriver
```

7.4 Summary statistics

This page contains descriptions and examples to load case-control datasets from several ENIGMA Working Groups. These ENIGMA summary statistics contain the following data: **effect sizes for case-control differences** (d_icv), **standard error** (se_icv), **lower bound of the confidence interval** (low_ci_icv), **upper bound of the confidence interval** (up_ci_icv), **number of controls** (n_controls), **number of patiens** (n_patients), **observed p-values** (pobs), **false discovery rate (FDR)-corrected p-value** (fdr_p).

ENIGMA's standardized protocols for data processing, quality assurance, and meta-analysis of individual subject data were conducted at each site. For site-level meta-analysis, all research centres within a given specialized Working Group tested for case *vs*. control differences using multiple linear regressions, where diagnosis (*e.g.*, healthy controls vs. individuals with epilepsy) was the predictor of interest, and subcortical volume, cortical thickness, or surface area of a given brain region was the outcome measure. Case-control differences were computed across all regions using either Cohen's *d* effect sizes or *t*-values, after adjusting for different combinations of age, sex, site/scan/dataset, intracranial volume, IQ (see below for disease-specific models).

Can't find the data you're searching for?

Let us know what's missing and we'll try and fetch that data for you and implement it in our toolbox. Get in touch with us here. If you have locally stored summary statistics on your computer, check out our tutorials on *how to import data* and accordingly take advantage of all the **ENIGMA TOOLBOX** functions.

---

* *indicates case-control tables used in the code snippets.*

7.4.1 22q11.2 deletion syndrome

Available summary statistics tables

From Sun et al., 2020, Mol Psychiatry | **age, sex, data set/site, and ICV* correction; FDR correction available**
**only for surface area measures*

CortThick_case_vs_controls
CortSurf_case_vs_controls
CortThick_psychP_vs_psychN (+/- psychosis)
CortSurf_psychP_vs_psychN (+/- psychosis)

From Ching et al., 2020, Am J Psychiatry | **age, age^2, sex, scan site, and ICV correction; FDR correction available**

SubVol_case_vs_controls
SubVol_case_vs_controls_AD (A-D deletion)
SubVol_case_vs_controls_AB (A-B deletion)
SubVol_AB_vs_AD
SubVol_psychP_vs_psychN (+/- psychosis)

Python | meta

```
>>> from enigmatoolbox.datasets import load_summary_stats

>>> # Load summary statistics for ENIGMA-22q
>>> sum_stats = load_summary_stats('22q')

>>> # Get case-control cortical thickness and surface area tables
>>> CT = sum_stats['CortThick_case_vs_controls']
>>> SA = sum_stats['CortSurf_case_vs_controls']

>>> # Extract Cohen's d values
>>> CT_d = CT['d_icv']
>>> SA_d = SA['d_icv']
```

Matlab | meta

```
% Load summary statistics for ENIGMA-22q
sum_stats = load_summary_stats('22q');

% Get case-control cortical thickness and surface area tables
CT = sum_stats.CortThick_case_vs_controls;
SA = sum_stats.CortSurf_case_vs_controls;

% Extract Cohen's d values
CT_d = CT.d_icv;
SA_d = SA.d_icv;
```

7.4.2 Attention deficit hyperactivity disorder

Available summary statistics tables

From Hoogman et al., 2019, Am J Psychiatry | **mega-analysis; age, sex, and ICV* correction; FDR correction available**

**only for surface area measures*

ALL AGES
CortThick_case_vs_controls_allages
CortSurf_case_vs_controls_allages

ADULTS (*age 22-63 years*)

CortThick_case_vs_controls_adult
CortSurf_case_vs_controls_adult

ADOLESCENTS (*age 15-21 years*)
CortThick_case_vs_controls_adolescent
CortSurf_case_vs_controls_adolescent

CHILDREN (*age 4-14 years*)
CortThick_case_vs_controls_pediatric
CortSurf_case_vs_controls_pediatric

From Hoogman et al., 2017, Lancet Psychiatry | **mega-analysis; age, sex, ICV, and site correction; p<0.0156 for FDR correction at q=0.05; mean [(left+right)/2] region of interest volume**

ALL AGES
SubVol_case_vs_controls_allages

ADULTS (*age22 years*)
SubVol_case_vs_controls_adult

ADOLESCENTS (*age 15-21 years*)
SubVol_case_vs_controls_adolescent

CHILDREN (*age14 years*)
SubVol_case_vs_controls_pediatric

Python | meta

```
>>> from enigmatoolbox.datasets import load_summary_stats

>>> # Load summary statistics for ENIGMA-ADHD
>>> sum_stats = load_summary_stats('adhd')

>>> # Get case-control cortical thickness and surface area tables
>>> CT = sum_stats['CortThick_case_vs_controls_adult']
>>> SA = sum_stats['CortSurf_case_vs_controls_adult']

>>> # Extract Cohen's d values
>>> CT_d = CT['d_icv']
>>> SA_d = SA['d_icv']
```

Matlab | meta

```
% Load summary statistics for ENIGMA-ADHD
sum_stats = load_summary_stats('adhd');

% Get case-control cortical thickness and surface area tables
CT = sum_stats.CortThick_case_vs_controls_adult;
SA = sum_stats.CortSurf_case_vs_controls_adult;

% Extract Cohen's d values
CT_d = CT.d_icv;
```

(continues on next page)

(continued from previous page)

```
SA_d = SA.d_icv;
```

7.4.3 Autism spectrum disorder

Available summary statistics tables

From van Rooij et al., 2018, Am J Psychiatry | **age, sex, IQ, and ICV* correction; FDR correction available (uncorrected p-values not provided); mean* [(left+right)/ 2)] region of interest volume**

**only for subcortical volume measures*

CortThick_case_vs_controls_meta_analysis

CortThick_case_vs_controls_mega_analysis

SubVol_case_vs_controls_meta_analysis

Python | meta

```
>>> from enigmatoolbox.datasets import load_summary_stats

>>> # Load summary statistics for ENIGMA-Autism
>>> sum_stats = load_summary_stats('asd')

>>> # Get case-control cortical thickness table
>>> CT = sum_stats['CortThick_case_vs_controls_meta_analysis']

>>> # Extract Cohen's d values
>>> CT_d = CT['d_icv']
```

Matlab | meta

```
% Load summary statistics for ENIGMA-Autism
sum_stats = load_summary_stats('asd');

% Get case-control cortical thickness table
CT = sum_stats.CortThick_case_vs_controls_meta_analysis;

% Extract Cohen's d values
CT_d = CT.d_icv;
```

7.4.4 Bipolar disorder

Available summary statistics tables

From Hibar al., 2018, Mol Psychiatry | **age, sex, and ICV* correction; FDR correction available**
**only for surface area measures*

ADULTS (*age25 years*)
CortThick_case_vs_controls_adult
CortSurf_case_vs_controls_adult
CortThick_typeI_vs_typeII_adult
CortSurf_typeI_vs_typeII_adult

ADOLESCENTS/YOUNG ADULTS (*age<25 years*)
CortThick_case_vs_controls_adolescent
CortSurf_case_vs_controls_adolescent
CortThick_typeI_vs_typeII_adolescent
CortSurf_typeI_vs_typeII_adolescent

From Hibar al., 2016, Mol Psychiatry | **age, sex, and ICV correction; p<4.91E-3 for FDR correction at q=0.05; mean [(left+right)/2] region of interest volume**

SubVol_case_vs_controls_typeI
SubVol_case_vs_controls_typeII
SubVol_typeII_vs_typeI

Python | meta

```
>>> from enigmatoolbox.datasets import load_summary_stats

>>> # Load summary statistics for ENIGMA-BD
>>> sum_stats = load_summary_stats('bipolar')

>>> # Get case-control surface area table
>>> CT = sum_stats['CortThick_case_vs_controls_adult']
>>> SA = sum_stats['CortSurf_case_vs_controls_adult']

>>> # Extract Cohen's d values
>>> CT_d = CT['d_icv']
>>> SA_d = SA['d_icv']
```

Matlab | meta

```
% Load summary statistics for ENIGMA-BD
sum_stats = load_summary_stats('bipolar');

% Get case-control surface area table
CT = sum_stats.CortThick_case_vs_controls_adult;
SA = sum_stats.CortSurf_case_vs_controls_adult;

% Extract Cohen's d values
CT_d = CT.d_icv;
SA_d = SA.d_icv;
```

7.4.5 Epilepsy

Available summary statistics tables

From Whelan al., 2018, Brain **| age, sex, and ICV correction; Bonferroni correction p<1.49E-4; FDR correction also available**

CortThick_case_vs_controls_allepilepsy
SubVol_case_vs_controls_allepilepsy
CortThick_case_vs_controls_gge
SubVol_case_vs_controls_gge
CortThick_case_vs_controls_ltle
SubVol_case_vs_controls_ltle
CortThick_case_vs_controls_rtle
SubVol_case_vs_controls_rtle
CortThick_case_vs_controls_allotherepilepsy
SubVol_case_vs_controls_allotherepilepsy

Python | meta

```
>>> from enigmatoolbox.datasets import load_summary_stats

>>> # Load summary statistics for ENIGMA-Epilepsy
>>> sum_stats = load_summary_stats('epilepsy')

>>> # Get case-control subcortical volume and cortical thickness tables
>>> SV = sum_stats['SubVol_case_vs_controls_ltle']
>>> CT = sum_stats['CortThick_case_vs_controls_ltle']

>>> # Extract Cohen's d values
>>> SV_d = SV['d_icv']
>>> CT_d = CT['d_icv']
```

Matlab | meta

```
% Load summary statistics for ENIGMA-Epilepsy
sum_stats = load_summary_stats('epilepsy');

% Get case-control subcortical volume and cortical thickness tables
SV = sum_stats.SubVol_case_vs_controls_ltle;
CT = sum_stats.CortThick_case_vs_controls_ltle;

% Extract Cohen's d values
SV_d = SV.d_icv;
CT_d = CT.d_icv;
```

7.4.6 Major depressive disorder

Available summary statistics tables

From Schmaal et al., 2017, Mol Psychiatry | **age, sex, and scan site correction; FDR correction available**

ADULTS (*age>21 years*)
CortThick_case_vs_controls_adult
CortSurf_case_vs_controls_adult
CortThick_case_vs_controls_adult_firstepisode
CortSurf_case_vs_controls_adult_firstepisode
CortThick_case_vs_controls_adult_recurrent
CortSurf_case_vs_controls_adult_recurrent
CortThick_firstepisode_vs_recurrent_adult
CortSurf_firstepisode_vs_recurrent_adult
CortThick_case_vs_controls_adult_early (age of onset21 years)
CortSurf_case_vs_controls_adult_early (age of onset21 years)
CortThick_case_vs_controls_adult_late (age of onset>21 years)
CortSurf_case_vs_controls_adult_late (age of onset>21 years)
CortThick_early_vs_late_adult
CortSurf_early_vs_late_adult

ADOLESCENTS (*age21 years*)
CortThick_case_vs_controls_adolescent
CortSurf_case_vs_controls_adolescent
CortThick_case_vs_controls_adolescent_firstepisode
CortSurf_case_vs_controls_adolescent_firstepisode
CortThick_case_vs_controls_adolescent_recurrent
CortSurf_case_vs_controls_adolescent_recurrent
CortThick_firstepisode_vs_recurrent_adolescent
CortSurf_firstepisode_vs_recurrent_adolescent

From Schmaal et al., 2016, Mol Psychiatry | **age, sex, ICV, and scanner differences correction; Bonferroni correction p<5.6E-3; mean [(left+right)/2] region of interest volume**

SubVol_case_vs_controls
SubVol_case_vs_controls_late (age of onset>21 years)
SubVol_case_vs_controls_early (age of onset21 years)
SubVol_late_vs_early
SubVol_case_vs_controls_firstepisode
SubVol_case_vs_controls_recurrent
SubVol_recurrrent_vs_firstepisode

Python | meta

```
>>> from enigmatoolbox.datasets import load_summary_stats

>>> # Load summary statistics for ENIGMA-MDD
>>> sum_stats = load_summary_stats('depression')
```

(continues on next page)

(continued from previous page)

```
>>> # Get case-control cortical thickness and surface area tables
>>> CT = sum_stats['CortThick_case_vs_controls_adult']
>>> SA = sum_stats['CortSurf_case_vs_controls_adult']

>>> # Extract Cohen's d values
>>> CT_d = CT['d_icv']
>>> SA_d = SA['d_icv']
```

Matlab | meta

```
% Load summary statistics for ENIGMA-MDD
sum_stats = load_summary_stats('depression');

% Get case-control cortical thickness and surface area tables
SV = sum_stats.SubVol_case_vs_controls_adult;
CT = sum_stats.CortThick_case_vs_controls_adult;
SA = sum_stats.CortSurf_case_vs_controls_adult;

% Extract Cohen's d values
SV_d = SV.d_icv;
CT_d = CT.d_icv;
SA_d = SA.d_icv;
```

7.4.7 Obsessive-compulsive disorder

Available summary statistics tables

From Boedhoe et al., 2018, Am J Psychiatry | **age, sex, scan site, and ICV* correction; FDR correction available**
**only for surface area measures*

ADULTS (*age18 years*)
CortThick_case_vs_controls_adult
CortSurf_case_vs_controls_adult
CortThick_medicatedcase_vs_controls_adult
CortSurf_medicatedcase_vs_controls_adult

PEDIATRIC (*age<18 years*)
CortThick_case_vs_controls_pediatric
CortSurf_case_vs_controls_pediatric
CortThick_medicatedcase_vs_controls_pediatric
CortSurf_medicatedcase_vs_controls_pediatric

From Boedhoe et al., 2017, Am J Psychiatry | **age, sex, scan site, and ICV correction; Bonferroni correction p<5.6E-3; mean [(left+right)/2] region of interest volume**

ADULTS (*age18 years*)
SubVol_case_vs_controls_adult

SubVol_medicatedcase_vs_controls_adult
SubVol_unmedicatedcase_vs_controls_adult
SubVol_medicatedcase_vs_unmedicated_adult
SubVol_case_vs_controls_adult_late (age of onset18 years)
SubVol_case_vs_controls_adult_early (age of onset<18 years)
SubVol_late_vs_early_adult
SubVol_case_vs_controls_adult_depression (as comorbidity)
SubVol_case_vs_controls_adult_nodepression
SubVol_depression_vs_nodepression_adult
SubVol_case_vs_controls_adult_anxiety (as comorbidity)
SubVol_case_vs_controls_adult_noanxiety
SubVol_anxiety_vs_noanxiety_adult

PEDIATRIC (*age<18 years*)
SubVol_case_vs_controls_pediatric
SubVol_medicatedcase_vs_controls_pediatric
SubVol_unmedicatedcase_vs_controls_pediatric
SubVol_medicatedcase_vs_unmedicated_pediatric

Python | meta

```
>>> from enigmatoolbox.datasets import load_summary_stats

>>> # Load summary statistics for ENIGMA-OCD
>>> sum_stats = load_summary_stats('ocd')

>>> # Get case-control cortical thickness and surface area tables
>>> CT = sum_stats['CortThick_case_vs_controls_adult']
>>> SA = sum_stats['CortSurf_case_vs_controls_adult']

>>> # Extract Cohen's d values
>>> CT_d = CT['d_icv']
>>> SA_d = SA['d_icv']
```

Matlab | meta

```
% Load summary statistics for ENIGMA-OCD
sum_stats = load_summary_stats('ocd');

% Get case-control cortical thickness and surface area tables
CT = sum_stats.CortThick_case_vs_controls_adult;
SA = sum_stats.CortSurf_case_vs_controls_adult;

% Extract Cohen's d values
CT_d = CT.d_icv;
SA_d = SA.d_icv;
```

7.4.8 Schizophrenia

Available summary statistics tables

From van Erp et al., 2018, Biol Psychiatry | **age and sex correction; FDR correction available**

CortThick_case_vs_controls

CortSurf_case_vs_controls

From van Erp et al., 2016, Mol Psychiatry | **age, sex, scan site, and ICV correction; Bonferroni correction p<5.6E-3**

SubVol_case_vs_controls

SubVol_case_vs_controls_mean (mean [(left+right)/ 2)] region of interest volume)

Python | meta

```
>>> from enigmatoolbox.datasets import load_summary_stats

>>> # Load summary statistics for ENIGMA-Schizophrenia
>>> sum_stats = load_summary_stats('schizophrenia')

>>> # Get case-control cortical thickness and surface area tables
>>> CT = sum_stats['CortThick_case_vs_controls']
>>> SA = sum_stats['CortSurf_case_vs_controls']

>>> # Extract Cohen's d values
>>> CT_d = CT['d_icv']
>>> SA_d = SA['d_icv']
```

Matlab | meta

```
% Load summary statistics for ENIGMA-schizophrenia
sum_stats = load_summary_stats('schizophrenia');

% Get case-control cortical thickness and surface area tables
CT = sum_stats.CortThick_case_vs_controls;
SA = sum_stats.CortSurf_case_vs_controls;

% Extract Cohen's d values
CT_d = CT.d_icv;
SA_d = SA.d_icv;
```

7.5 Individual site data

This page contains descriptions and examples to load our example data, re-order subcortical labels, and z-score values!

7.5.1 Load example data

This is an example dataset that includes 10 healthy controls (7 females, age±SD=33.3±8.8 years) and 10 individuals with epilepsy (7 females, age±SD=39.8±14.8 years).

Covariates | As per ENIGMA-Epilepsy protocol, covariate information includes **SubjID** (subjectID), **Dx** (diagnosis, 0=controls, 1=patients), **SDx** (sub-diagnosis, 0=controls, 1=non-lesional patients, 2=genetic generalized epilepsy (IGE/GGE) patients, 3=left TLE, 4=right TLE), **Age** (in years), **Sex** (1=males, 2=females), **Handedness** (1=right, 2=left), **AO** (age at onset in years, patients only), **DURILL** (duration of illness in years, patients only), and **ICV** (intracranial volume).

Subcortical volume | Subcortical grey matter volumes regroup data from 12 subcortical regions, bilateral hippocampus, and bilateral ventricles.

Cortical thickness | Cortical thickness was measured at each vertex as the Euclidean distance between white and pial surfaces, and subsequently averaged within each of the Desikan-Killiany parcels.

Cortical surface area | The cortical surface area of every Desikan-Killiany parcel is also provided as part of ENIGMA imaging protocols; this morphological measure is defined by the sum of the area of each of the triangles within the parcel.

Python | mega

```
>>> from enigmatoolbox.datasets import load_example_data

>>> # Load all example data from an individual site
>>> cov, metr1_SubVol, metr2_CortThick, metr3_CortSurf = load_example_data()
```

Matlab | mega

```
% Load all example data from an individual site
[cov, metr1_SubVol, metr2_CortThick, metr3_CortSurf] = load_example_data();
```

7.5.2 Re-order subcortical regions

The column order from ENIGMA-derived subcortical volume matrices does not match the order in the connectivity matrices nor the pre-requisite for our subcortical visualization tools. But, hey, don't you worry! To re-order ENIGMA-derived subcortical volume data, you may use our `reorder_sctx()` function, which will re-order the columns of the subcortical volume dataset accordingly (*i.e.*, alphabetically, with all left hemisphere structures first followed by all right hemisphere structures).

Python | mega

```
>>> from enigmatoolbox.utils.useful import reorder_sctx

>>> # Re-order the subcortical data alphabetically and by hemisphere
>>> metr1_SubVol_r = reorder_sctx(metr1_SubVol)
```

Matlab | mega

```
% Re-order the subcortical data alphabetically and by hemisphere
metr1_SubVol_r = reorder_sctx(metr1_SubVol);
```

7.5.3 Z-score data

With our example data loaded and re-ordered, we can then z-score data in patients, relative to controls, so that lower values correspond to greater atrophy. For simplicity, we also compute the mean z-score across all left TLE patients. These mean, z-scored vectors will be used in subsequent tutorials as measures of subcortical and cortical atrophy!

Prerequisites
Re-order subcortical data

Python | mega

```
>>> from enigmatoolbox.utils.useful import zscore_matrix

>>> # Z-score patients' data relative to controls (lower z-score = more atrophy)
>>> group = cov['Dx'].to_list()
>>> controlCode = 0
>>> SV_z = zscore_matrix(metr1_SubVol_r.iloc[:, 1:-1], group, controlCode)
>>> CT_z = zscore_matrix(metr2_CortThick.iloc[:, 1:-5], group, controlCode)
>>> SA_z = zscore_matrix(metr3_CortSurf.iloc[:, 1:-5], group, controlCode)

>>> # Mean z-score values across individuals with from a specific group (e.g., left␣
↪TLE, that is SDx == 3)
>>> SV_z_mean = SV_z.iloc[cov[cov['SDx'] == 3].index, :].mean(axis=0)
>>> CT_z_mean = CT_z.iloc[cov[cov['SDx'] == 3].index, :].mean(axis=0)
>>> SA_z_mean = SA_z.iloc[cov[cov['SDx'] == 3].index, :].mean(axis=0)
```

Matlab | mega

```
% Z-score patients' data relative to controls (lower z-score = more atrophy)
group        = cov.Dx;
controlCode  = 0;
SV_z         = zscore_matrix(metr1_SubVol_r(:, 2:end-1), group, controlCode);
CT_z         = zscore_matrix(metr2_CortThick(:, 2:end-5), group, controlCode);
SA_z         = zscore_matrix(metr3_CortSurf(:, 2:end-5), group, controlCode);

% Mean z-score values across individuals with from a specific group (e.g., left TLE,␣
↪that is SDx == 3)
SV_z_mean    = array2table(mean(SV_z{find(cov.SDx == 3), :}, 1), ...
                           'VariableNames', SV_z.Properties.VariableNames);
CT_z_mean    = array2table(mean(CT_z{find(cov.SDx == 3), :}, 1), ...
                           'VariableNames', CT_z.Properties.VariableNames);
SA_z_mean    = array2table(mean(SA_z{find(cov.SDx == 3), :}, 1), ...
                           'VariableNames', SA_z.Properties.VariableNames);
```

7.6 Cross-disorder effect

This page contains descriptions and examples to perform cross-disorder analyses to explore brain structural abnormalities that are common or different across disorders.

7.6.1 Principal component analysis

To yield novel insights into brain structural abnormalities that are common or different across disorders, we can explore shared and disease-specific morphometric signatures by applying a principal component analysis (PCA) to any combination of disease-specific summary statistics (or other imported data), resulting in shared latent components that can be used for further analysis.

Python | meta

```
>>> from enigmatoolbox.cross_disorder import cross_disorder_effect
>>> from enigmatoolbox.plotting import plot_cortical, plot_subcortical
>>> from enigmatoolbox.utils import parcel_to_surface
>>> import matplotlib.pyplot as plt

>>> # Extract shared disorder effects
>>> components, variance, names = cross_disorder_effect()

>>> # Visualize cortical and subcortical eigenvalues in scree plots
>>> fig, ax = plt.subplots(1, 2, figsize=(14, 6))
>>> for ii, jj in enumerate(components):
>>>     ax[ii].plot(variance[jj], lw=2, color='#A8221C', zorder=1)
>>>     ax[ii].scatter(range(variance[jj].size), variance[jj], s=78, color='#A8221C',
>>>                    linewidth=1.5, edgecolor='w', zorder=3)
>>>     ax[ii].set_xlabel('Components')

>>>     if ii == 0:
>>>         ax[ii].set_ylabel('Cortical eigenvalues')
>>>     else:
>>>         ax[ii].set_ylabel('Subcortical eigenvalues')

>>>     ax[ii].spines['top'].set_visible(False)
>>>     ax[ii].spines['right'].set_visible(False)

>>> fig.tight_layout()
>>> plt.show()

>>> # Visualize the first cortical and subcortical components on the surface brains
>>> plot_cortical(parcel_to_surface(components['cortex'][:, 0], 'aparc_fsa5'), color_
↪range=(-0.5, 0.5),
...               cmap='RdBu_r', color_bar=True, size=(800, 400))

>>> plot_subcortical(components['subcortex'][:, 0], color_range=(-0.5, 0.5),
...                  cmap='RdBu_r', color_bar=True, size=(800, 400))
```

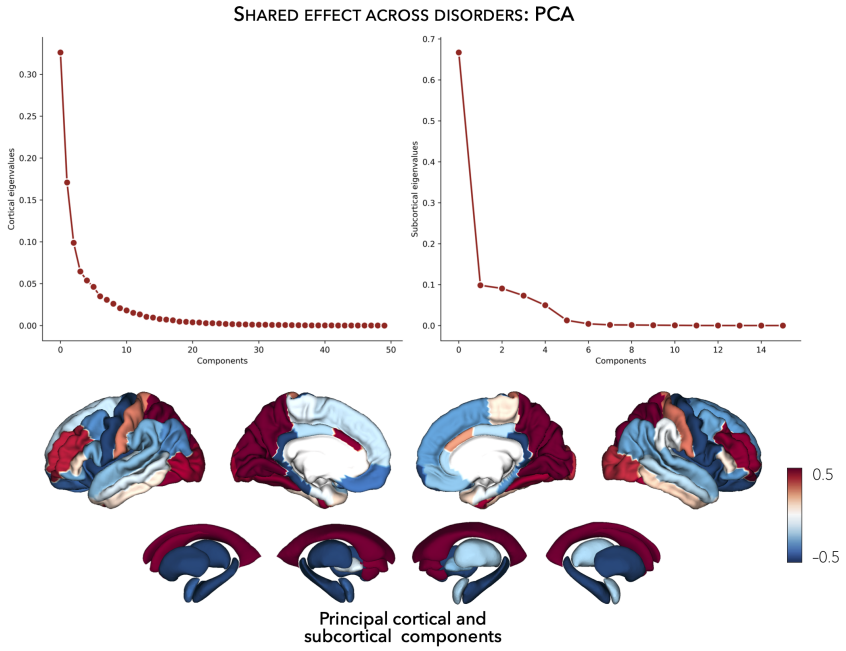
Matlab | meta

```
% Extract shared disorder effects
[components, variance, ~, names] = cross_disorder_effect();

% Visualize cortical and subcortical eigenvalues in scree plots
fns  = fieldnames(components);
```

(continues on next page)

(continued from previous page)

```
f = figure,
set(gcf,'color','w');
set(gcf,'units','normalized','position',[0 0 .75 0.3])

for ii = 1:numel(fieldnames(components))
     axs = subplot(1, 2, ii); hold on
     s = scatter(1:size(components.(fns{ii}), 2), variance.(fns{ii}), 128, [0.66 0.13␣
↪0.11], 'filled');
     s.LineWidth = 1.5; s.MarkerEdgeColor = 'w';

     plot(1:size(components.(fns{ii}), 2), variance.(fns{ii}), 'linewidth', 2, 'color
↪', [0.66 0.13 0.11])
     xlabel('Components')

     if ii == 1
          ylabel('Cortical eigenvalues')
     else
          ylabel('Subcortical eigenvalues')
     end
end

% Visualize the first cortical and subcortical components on the surface brains
f = figure,
    plot_cortical(parcel_to_surface(components.cortex(:, 1)), 'color_range', [-0.5 0.
↪5], 'cmap', 'RdBu_r')

f = figure,
    plot_subcortical(components.subcortex(:, 1), 'color_range', [-0.5 0.5], 'cmap',
↪'RdBu_r')
```



7.6.2 Cross-correlation

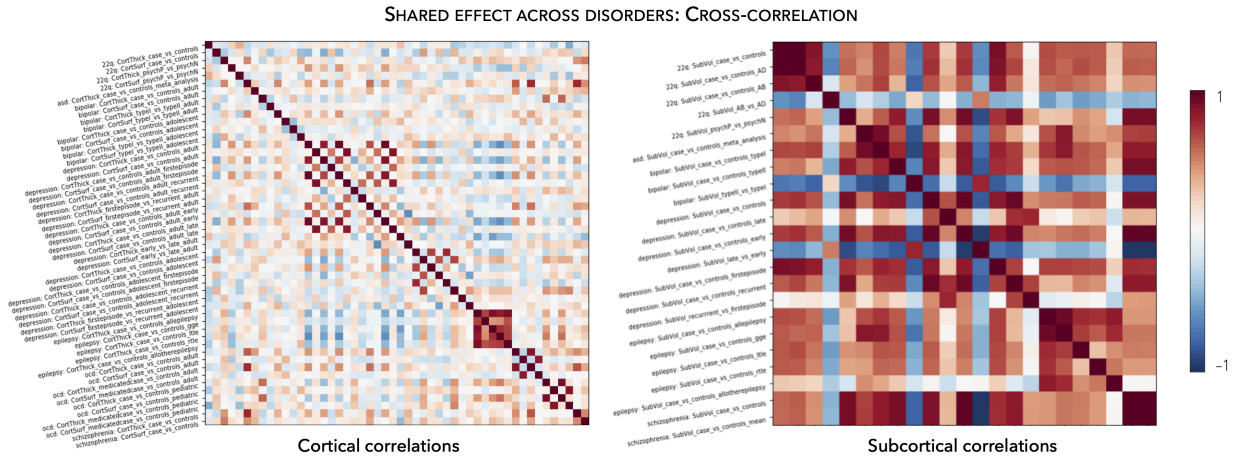
We can also explore shared and disease-specific morphometric signatures by systematically cross-correlating patterns of brain structural abnormalities with any combination of summary statistics (or other pre-loaded ENIGMA-type data), resulting in a correlation matrix

Python | meta

```
>>> from enigmatoolbox.cross_disorder import cross_disorder_effect
>>> from nilearn import plotting

>>> # Extract shared disorder effects
>>> correlation_matrix, names = cross_disorder_effect(method='correlation')

>>> # Plot correlation matrices
>>> plotting.plot_matrix(correlation_matrix['cortex'], figure=(12, 8), labels=names[
↪'cortex'], vmax=1,
...                      vmin=-1, cmap='RdBu_r', auto_fit=False)

>>> plotting.plot_matrix(correlation_matrix['subcortex'], figure=(12, 8),␣
↪labels=names['subcortex'], vmax=1,
...                      vmin=-1, cmap='RdBu_r', auto_fit=False)
```

Matlab | meta

```
% Extract shared disorder effects
[~, ~, correlation_matrix, names] = cross_disorder_effect('method', 'correlation');

% Plot correlation matrices
f = figure('units','normalized','outerposition',[0 0 .65 1]),
    imagesc(correlation_matrix.cortex, [-1 1])
    axis square;
    colormap(RdBu_r);
    colorbar;
    set(gca, 'YTick', 1:1:size(correlation_matrix.cortex, 1), ...
        'YTickLabel', strrep(names.cortex, '_', ' '), 'XTick', 1:1:size(correlation_
↪matrix.cortex, 1), ...
        'XTickLabel', strrep(names.cortex, '_', ' '), 'XTickLabelRotation', 45)

f = figure('units','normalized','outerposition',[0 0 .65 1]),
    imagesc(correlation_matrix.subcortex, [-1 1])
    axis square;
    colormap(RdBu_r);
    colorbar;
    set(gca, 'YTick', 1:1:size(correlation_matrix.subcortex, 1), ...
        'YTickLabel', strrep(names.subcortex, '_', ' '), 'XTick',␣
↪1:1:size(correlation_matrix.subcortex, 1), ...
        'XTickLabel', strrep(names.subcortex, '_', ' '), 'XTickLabelRotation', 45)
```



7.7 Import vertexwise or parcellated data

This page contains descriptions and examples to import your own data, stored as different file format types. Compatible import file formats are: *.txt/.csv*, *FreeSurfer/"curv"*, *.mgh/.mgz*, *GIfTI/.gii*, *CIfTI/.dscalar.nii/.dtseries.nii*

As an example, we will import vertexwise cortical thickness data sampled on the Conte69 surface template (available within the **ENIGMA Toolbox**). The examples below can be easily modified to import any vertexwise or parcellated data by simply changing the path to the data and the filenames. Here, we imported data from ENIGMA's import_export directory.

7.7.1 .txt / .csv

If your data are stored as text (.txt) or comma-separated values (.csv) files, then you may use the `dlmread()` (*Matlab*) or `np.loadtxt()` (*Python*) functions to load your data.

Table please

Should you prefer to upload your data as a table (*e.g.*, when importing ENIGMA-type summary statistics stored in your local computer), then you may use the `readtable()` (*Matlab*) or `pd.readcsv()` (*Python*) functions.

Python

```
>>> import enigmatoolbox
>>> import numpy as np
>>> import os

>>> # Define path to your data
>>> dpath = os.path.join(os.path.dirname(enigmatoolbox.__file__) + "/datasets/import_
↪export/")

>>> # Import data stored as .txt / .csv
>>> CT = []
>>> for _, h in enumerate(['lh', 'rh']):
>>>     CT = np.append(CT, np.loadtxt(dpath + '{}.conte69_32k_thickness.txt'.
↪format(h)))
```

Matlab

```
% Define path to your data
dpath = what('import_export'); dpath = data_path.path;

% Import data stored as .txt / .csv
data_lh = dlmread([dpath '/lh.conte69_32k_thickness.txt']);
data_rh = dlmread([dpath '/rh.conte69_32k_thickness.txt']);
CT = [data_lh data_rh];
```

7.7.2 FreeSurfer / "curv"

If your data are stored as FreeSurfer "curv" format (*e.g.*, ?h.thickness), then you may use the `read_curv()` (*Matlab*) or `nib.freesurfer.io.read_morph_data()` (*Python*) functions to load your data. You can get the Matlab function from here.

Python

```
>>> import enigmatoolbox
>>> import nibabel as nib
>>> import numpy as np
>>> import os

>>> # Define path to your data
>>> dpath = os.path.join(os.path.dirname(enigmatoolbox.__file__) + "/datasets/import_
↪export/")

>>> # Import data stored as FreeSurfer "curv"
>>> CT = []
>>> for _, h in enumerate(['lh', 'rh']):
>>>     CT = np.append(CT, nib.freesurfer.io.read_morph_data(dpath + '{}.conte69_32k_
↪thickness'.format(h)))
```

Matlab

```
% Define path to your data
dpath = what('import_export'); dpath = dpath.path;

% Import data stored as FreeSurfer "curv"
data_lh = read_curv([dpath '/lh.conte69_32k_thickness']);
data_rh = read_curv([dpath '/rh.conte69_32k_thickness']);
CT = [data_lh; data_rh].';
```

7.7.3 .mgh / .mgz

If your data are stored as .mgh or .mgz formats, then you may use the `load_mgh()` (*Matlab*) or `nib.load` (*Python*) functions to load your data. You can get the Matlab function from here.

Python

```
>>> import enigmatoolbox
>>> import nibabel as nib
>>> import numpy as np
>>> import os

>>> # Define path to your data
>>> dpath = os.path.join(os.path.dirname(enigmatoolbox.__file__) + "/datasets/import_
↪export/")

>>> # Import data stored as .mgh / .mgz
>>> CT = []
>>> for _, h in enumerate(['lh', 'rh']):
>>>     CT = np.append(CT, nib.load(dpath + '{}.conte69_32k_thickness.mgh'.format(h)).
↪get_fdata().squeeze())
```

Matlab

```
% Define path to your data
dpath = what('import_export'); dpath = dpath.path;

% Import data stored as .mgh / .mgz
data_lh = load_mgh([dpath '/lh.conte69_32k_thickness.mgh']);
data_rh = load_mgh([dpath '/rh.conte69_32k_thickness.mgh']);
CT = [data_lh; data_rh].';
```

7.7.4 GIfTI / .gii

If your data are stored as GIfTI/.gii format, then you may use the `gifti()` (*Matlab*) or `nib.load` (*Python*) functions to load your data. You can get the Matlab function from here.

Python

```
>>> import enigmatoolbox
>>> import nibabel as nib
>>> import numpy as np
>>> import os

>>> # Define path to your data
>>> dpath = os.path.join(os.path.dirname(enigmatoolbox.__file__) + "/datasets/import_
↪export/")

>>> # Import data stored as GIfTI / .gii
>>> CT = []
>>> for _, h in enumerate(['lh', 'rh']):
>>>     CT = np.append(CT, nib.load(dpath + '{}.conte69_32k_thickness.gii'.format(h)).
↪darrays[0].data)
```

Matlab

```
% Define path to your data
dpath = what('import_export'); dpath = dpath.path;

% Import data stored as GIfTI / .gii
data_lh = gifti([dpath '/lh.conte69_32k_thickness.gii']);
data_rh = gifti([dpath '/rh.conte69_32k_thickness.gii']);
CT = [data_lh.cdata; data_rh.cdata].';
```

7.7.5 CIfTI / .dscalar.nii / .dtseries.nii

If your data are stored as CIfTI/.dscalar.nii/dtseries.nii format, then you may use the `cifti_read()` (*Matlab*) or `nib.load` (*Python*) functions to load your data. You can get the Matlab function from here.

Python

```
>>> import enigmatoolbox
>>> import nibabel as nib
>>> import numpy as np
>>> import os

>>> # Define path to your data
>>> dpath = os.path.join(os.path.dirname(enigmatoolbox.__file__) + "/
↪datasets/import_export/")

>>> # Import data stored as CIfTI / .dscalar.nii / .dtseries.nii
>>> CT = []
>>> for _, h in enumerate(['lh', 'rh']):
>>>     CT = np.append(CT, np.asarray(nib.load(dpath + '{}.conte69_32k_
↪thickness.dscalar.nii'.format(h)).get_data()))
```

Matlab

```
% Define path to your data
dpath = what('import_export'); dpath = dpath.path;

% Import data stored as CIfTI / .dscalar.nii / .dtseries.nii
data_lh = cifti_read([dpath '/lh.conte69_32k_thickness.dscalar.nii']);
data_rh = cifti_read([dpath '/rh.conte69_32k_thickness.dscalar.nii']);
CT = [data_lh.cdata; data_rh.cdata].';
```

7.8 Vertexwise parcellated data

This page contains descriptions and examples to (*i*) parcellate vertexwise data (sampled on fsaverage5 or conte69), allowing users to take advantage of every function within the **ENIGMA TOOLBOX**, and (*ii*) map parcellated data back to vertexwise space, allowing surface visualization of data and cross-software compatibility.

7.8.1 Vertexwise → parcellated data

As an example, we parcellate vertexwise cortical thickness data imported in the *previous tutorial* according to the Schaefer-200 atlas. Users can use this function to parcellate (fsaverage5 or conte69) vertexwise data using different parcellations, including: aparc_fsa5, glasser_fsa5, schaefer_100_fsa5, schaefer_200_fsa5, schaefer_300_fsa5, schaefer_400_fsa5, aparc_conte69, glasser_conte69, schaefer_100_conte69, schaefer_200_conte69, schaefer_300_conte69, schaefer_400_conte69.

Python

```
>>> from enigmatoolbox.utils.parcellation import surface_to_parcel

>>> # Parcellate vertexwise data
>>> CT_schaefer_200 = surface_to_parcel(CT, 'schaefer_200_conte69')
```

Matlab

```
% Parcellate vertexwise data
CT_schaefer_200 = surface_to_parcel(CT, 'schaefer_200_conte69');
```

7.8.2 Parcellated → vertexwise data

Mapping parcellated data to the surface has never been easier! Our `parcel_to_surface()` function works with ENIGMA- and non-ENIGMA datasets. This function is especially useful for visualizing parcellated data on cortical surface templates using our *visualization tools* or other popular neuroimaging softwares (check out our tutorials on *how to export data*).

As an example, we map the parcellated data from the *above tutorial* back to the Conte69 surface.

Don't like conte69? Relax, we got you covered!

The same approach can be used to map parcellated data to the fsaverage5 surface template; simply replace every 'conte69' with 'fsa5'!

Python

```
>>> from enigmatoolbox.utils.parcellation import parcel_to_surface

>>> # Map parcellated data to the surface
>>> CT_schaefer_200_c69 = parcel_to_surface(CT_schaefer_200, 'schaefer_200_conte69')
```

Matlab

```
% Map parcellated data to the surface
CT_schaefer_200_c69 = parcel_to_surface(CT_schaefer_200, 'schaefer_200_conte69');
```

7.9 Export data results

This page contains descriptions and examples to export data results across a range of file formats compatible with several neuroimaging software. These include: *.txt/.csv*, *FreeSurfer/"curv"*, *.mgh/.mgz*, *GIfTI/.gii*

As an example, we will export the vertexwise data from the *previous tutorial*. The examples below can be easily modified to export any vertexwise (all formats) or parcellated (.txt/.csv format) data results by simply changing the data array to be saved, the data export path, and the filenames. Here, we exported cortical thickness data to the ENIGMA's import_export datasets directory.

What about exporting brain surfaces?

Exported data results can be mapped to the surface templates using several other neuroimaging software. Cortical and subcortical surfaces are available here in different file formats (FreeSurfer/surface, GIfTI/.gii, .vtk, .obj).

7.9.1 .txt / .csv

If you want to export your data as text (.txt) or comma-separated values files, then you may use the `writetable()` (*Matlab*) or `np.savetxt()` (*Python*) functions. If your input data are in a table or dataframe, then you may remove the `array2table()` function (*Matlab*) or use the `.to_csv()` function (*Python*) functions.

Python

```
>>> import enigmatoolbox
>>> import numpy as np
>>> import os

>>> # Specify data to be exported
>>> data = CT_schaefer_200_c69

>>> # Define output path and filenames
>>> dpath = os.path.join(os.path.dirname(enigmatoolbox.__file__) + "/datasets/import_
↪export/")
>>> fname_lh = 'lh.schaefer_200_c69_thickness.txt'
>>> fname_rh = 'rh.schaefer_200_c69_thickness.txt'

>>> # Export data as .txt / .csv
>>> np.savetxt(dpath + fname_lh, data[:len(data)//2])
>>> np.savetxt(dpath + fname_rh, data[len(data)//2:])
```

Matlab

```
% Specify data to be exported
data = CT_schaefer_200_c69

% Define output path and filenames
dpath = what('import_export'); dpath = dpath.path;
fname_lh = 'lh.schaefer_200_c69_thickness.txt'
fname_rh = 'rh.schaefer_200_c69_thickness.txt'
```

(continues on next page)

(continued from previous page)

```
% Export data as .txt / .csv
writetable(array2table(data(1:end/2)), [dpath, fname_lh], 'WriteVariableNames', 0)
writetable(array2table(data(end/2+1:end)), [dpath, fname_rh], 'WriteVariableNames', 0)
```

7.9.2 FreeSurfer / "curv"

If you want to export your data as FreeSurfer "curv" format (*e.g.*, ?h.thickness), then you may use the `write_curv()` (*Matlab*) or `nib.freesurfer.io.write_morph_data()` (*Python*) functions to load your data. You can get the Matlab function from here.

What is this nfaces business?

In the `write_curv()` and `nib.freesurfer.io.write_morph_data()` functions, users are required to specify the number of faces (or triangles) in the associated surface file. To simplify things, we built a simple function, `nfaces()`, in which users can specify the surface name ('*conte69*', '*fsa5*') and the hemisphere ('*lh*', '*rh*', '*both*') to obtain the appropriate number of faces.

Python

```
>>> import enigmatoolbox
>>> from enigmatoolbox.datasets import nfaces
>>> import nibabel as nib
>>> import os

>>> # Specify data to be exported
>>> data = CT_schaefer_200_c69

>>> # Define output path and filenames
>>> dpath = os.path.join(os.path.dirname(enigmatoolbox.__file__) + "/datasets/import_
↪export/")
>>> fname_lh = 'lh.schaefer_200_c69_thickness'
>>> fname_rh = 'rh.schaefer_200_c69_thickness'

>>> # Export data as FreeSurfer "curv"
>>> nib.freesurfer.io.write_morph_data(dpath + fname_lh, data[:len(data)//2], nfaces(
↪'conte69', 'lh'))
>>> nib.freesurfer.io.write_morph_data(dpath + fname_rh, data[len(data)//2:], nfaces(
↪'conte69', 'rh'))
```

Matlab

```
% Specify data to be exported
data = CT_schaefer_200_c69

% Define output path and filenames
dpath = what('import_export'); dpath = dpath.path;
fname_lh = 'lh.schaefer_200_c69_thickness'
fname_rh = 'rh.schaefer_200_c69_thickness'
```

(continues on next page)

(continued from previous page)

```
% Export data as FreeSurfer "curv"
write_curv([dpath, fname_lh], data(1:end/2), nfaces('conte69', 'lh'));
write_curv([dpath fname_rh], data(end/2+1:end), nfaces('conte69', 'rh'));
```

7.9.3 .mgh / .mgz

If you want to export your data as .mgh or .mgz formats, then you may use the `load_mgh()` (*Matlab*) or `nib.freesurfer.mghformat.MGHImage()` (*Python*) functions to load your data. You can get the Matlab function from here.

What is this getaffine business?

In the `save_mgh()` and `nib.freesurfer.mghformat.MGHImage()` functions, users are required to specify a vox2ras transform matrix. To simplify things, we built a simple function, `getaffine()`, in which users can specify the surface name ('*conte69*', '*fsa5*') and the hemisphere ('*lh*', '*rh*', '*both*') to obtain the appropriate transform.

Python

```
>>> import enigmatoolbox
>>> from enigmatoolbox.datasets import getaffine
>>> import nibabel as nib
>>> import numpy as np
>>> import os

>>> # Specify data to be exported
>>> data = CT_schaefer_200_c69

>>> # Define output path and filenames
>>> dpath = os.path.join(os.path.dirname(enigmatoolbox.__file__) + "/datasets/import_
↪export/")
>>> fname_lh = 'lh.schaefer_200_c69_thickness.mgh'
>>> fname_rh = 'rh.schaefer_200_c69_thickness.mgh'

>>> # Export data as .mgh / .mgz
>>> nib.freesurfer.mghformat.MGHImage(np.float32(data[:len(data)//2]),
...                                   getaffine('conte69', 'lh')).to_filename(dpath +␣
↪fname_lh)
>>> nib.freesurfer.mghformat.MGHImage(np.float32(data[len(data)//2:]),
...                                   getaffine('conte69', 'lh')).to_filename(dpath +␣
↪fname_rh)
```

Matlab

```
% Specify data to be exported
data = CT_schaefer_200_c69

% Define output path and filenames
dpath = what('import_export'); dpath = dpath.path;
fname_lh = 'lh.schaefer_200_c69_thickness.mgh'
fname_rh = 'rh.schaefer_200_c69_thickness.mgh'
```

(continues on next page)

(continued from previous page)

```
% Export data as .mgh / .mgz
save_mgh(data(1:end/2), [dpath, fname_lh], getaffine('conte69', 'lh'));
save_mgh(data(end/2+1:end), [dpath, fname_rh], getaffine('conte69', 'rh'));
```

7.9.4 GIfTI / .gii

If you want to export your data as GIfTI/.gii format, then you may use the `savegifti()` (*Matlab*) or `nib.save` (*Python*) functions to load your data. You can get the Matlab function from here.

Script name change

To avoid confusion with *Matlab's* function `save()`, we renamed the GIfTI Toolbox's save function as `savegifti()`.

Python

```
>>> import enigmatoolbox
>>> import nibabel as nib
>>> import os

>>> # Specify data to be exported
>>> data = CT_schaefer_200_c69

>>> # Convert left and right hemisphere data to GIfTI image
>>> data_lh = nib.gifti.gifti.GiftiImage()
>>> data_lh.add_gifti_data_array(nib.gifti.gifti.GiftiDataArray(data=data[:len(data)//
↪2]))
>>> data_rh = nib.gifti.gifti.GiftiImage()
>>> data_rh.add_gifti_data_array(nib.gifti.gifti.GiftiDataArray(data=data[len(data)//
↪2:]))

>>> # Define output path and filenames
>>> dpath = os.path.join(os.path.dirname(enigmatoolbox.__file__) + "/datasets/import_
↪export/")
>>> fname_lh = 'lh.schaefer_200_c69_thickness.gii'
>>> fname_rh = 'rh.schaefer_200_c69_thickness.gii'

>>> # Export data as GIfTI / .gii
>>> nib.save(data_lh, dpath + fname_lh)
>>> nib.save(data_rh, dpath + fname_rh)
```

Matlab

```
% Specify data to be exported
data = gifti(CT_schaefer_200_c69);
data_lh = data; data_lh.cdata = data_lh.cdata(1:end/2);
data_rh = data; data_rh.cdata = data_rh.cdata(end/2+1:end);

% Define output path and filenames
```

(continues on next page)

(continued from previous page)

```
dpath = what('import_export'); dpath = dpath.path;
fname_lh = 'lh.schaefer_200_c69_thickness.gii'
fname_rh = 'rh.schaefer_200_c69_thickness.gii'

% Export data as GIfTI / .gii
savegifti(data_lh, [dpath, fname_lh], 'Base64Binary');
savegifti(data_rh, [dpath, fname_rh], 'Base64Binary');
```

7.9.5 CIfTI / .dscalar.nii / .dtseries.nii

If you want to export your data as CIfTI/.dscalar.nii/.dtseries.nii format, then you may use the `write_cifti()` (*Matlab* and *Python*) functions to load your data. The *Matlab* function, however, relies on this toolbox right here.

Python

```
>>> from enigmatoolbox.datasets import write_cifti
>>> import os

>>> # Specify data to be exported
>>> data = CT_schaefer_200_c69;

>>> # Define output path and filenames
>>> dpath = os.path.join(os.path.dirname(enigmatoolbox.__file__) + "/datasets/import_
↪export/")
>>> fname_lh = 'lh.schaefer_200_c69_thickness.dscalar.nii'
>>> fname_rh = 'rh.schaefer_200_c69_thickness.dscalar.nii'

>>> # Export data as CIfTI / .dscalar.nii / .dtseries.nii
>>> write_cifti(data[:len(data)//2], dpath=dpath, fname=fname_lh, labels=None,␣
↪surface_name='conte69', hemi='lh')
>>> write_cifti(data[len(data)//2:], dpath=dpath, fname=fname_rh, labels=None,␣
↪surface_name='conte69', hemi='rh')
```

Matlab

```
% Specify data to be exported
data = CT_schaefer_200_c69;

% Define output path and filenames
dpath = what('import_export'); dpath = dpath.path;
fname_lh = 'lh.schaefer_200_c69_thickness.dscalar.nii'
fname_rh = 'rh.schaefer_200_c69_thickness.dscalar.nii'

% Export data as CIfTI / .dscalar.nii / .dtseries.nii
write_cifti(data(1:end/2), 'dpath', dpath, 'fname', fname_lh, 'surface_name', 'conte69
↪', 'hemi', 'lh')
write_cifti(data(end/2+1:end), 'dpath', dpath, 'fname', fname_rh, 'surface_name',
↪'conte69', 'hemi', 'rh')
```

7.10 Surface data visualization

This page contains descriptions and examples to visualize and manipulate surface data.

How to zoom and rotate surfaces

Users can manipulate cortical and subcortical surfaces within the *Python* and *Matlab* viewers. **To zoom in and out**: Scroll up and down on trackpad or mouse wheel (*Python*) or use the magnifying glass buttons in the figure toolbar and click on the surface (*Matlab*). **To rotate surfaces in 3D**: Click and hold down the left mouse button and move it around (*Python*) or press the 3D rotate figure toolbar button then click and hold down the left mouse button and move it around (*Matlab*).

7.10.1 Cortical surface visualization

ENIGMA TOOLBOX comes equipped with fsaverage5 and Conte69 cortical midsurfaces and numerous parcellations. Following the examples below, we can easily map parcellated data (*e.g.*, Desikan-Killiany) to fsaverage5 surface space (*i.e.*, vertices). In the following example, we will display cortical atrophy in individuals with left TLE.

Prerequisites
Load *summary statistics*, *example data*, **or** *your own data*
Z-score data (*mega only*)

Python | meta

```
>>> from enigmatoolbox.utils.parcellation import parcel_to_surface
>>> from enigmatoolbox.plotting import plot_cortical

>>> # Map parcellated data to the surface
>>> CT_d_fsa5 = parcel_to_surface(CT_d, 'aparc_fsa5')

>>> # Project the results on the surface brain
>>> plot_cortical(array_name=CT_d_fsa5, surface_name="fsa5", size=(800, 400),
...               cmap='RdBu_r', color_bar=True, color_range=(-0.5, 0.5))
```

Matlab | meta

```
% Map parcellated data to the surface
CT_d_fsa5 = parcel_to_surface(CT_d, 'aparc_fsa5');

% Project the results on the surface brain
f = figure,
    plot_cortical(CT_d_fsa5, 'surface_name', 'fsa5', 'color_range', ...
                  [-0.5 0.5], 'cmap', 'RdBu_r')
```

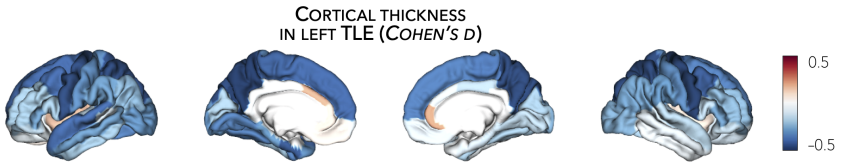
If you have **meta**-analysis data (*e.g.*, summary statistics)
If you have individual site or **mega**-analysis data

Python | mega

```
>>> from enigmatoolbox.utils.parcellation import parcel_to_surface
>>> from enigmatoolbox.plotting import plot_cortical

>>> # Before visualizing the data, we need to map the parcellated data to the surface
>>> CT_z_mean_fsa5 = parcel_to_surface(CT_z_mean, 'aparc_fsa5')

>>> # Project the results on the surface brain
>>> plot_cortical(array_name=CT_z_mean_fsa5, surface_name="fsa5", size=(800, 400),
>>>               cmap='Blues_r', color_bar=True, color_range=(-2, 0))
```

Matlab | mega

```
% Map parcellated data to the surface
CT_d_fsa5 = parcel_to_surface(CT_z_mean{:, :}, 'aparc_fsa5');

% Project the results on the surface brain
f = figure,
    plot_cortical(CT_z_mean_fsa5, 'surface_name', 'fsa5', 'color_range', ...
                  [-2 0], 'cmap', 'Blues_r')
```



7.10.2 Subcortical surface visualization

The **ENIGMA TOOLBOX**'s subcortical viewer includes 16 segmented subcortical structures obtained from the Desikan-Killiany atlas (aparc+aseg.mgz). Subcortical regions include bilateral accumbens, amygdala, caudate, hippocampus (technically not subcortical but considered as such by FreeSurfer), pallidum, putamen, thalamus, and ventricles. In the following example, we will display subcortical atrophy in individuals with left TLE.

We've mentioned this already, but don't forget that...

Subcortical input values are ordered as follows: left-accumbens, left-amygdala, left-caudate, left-hippocampus, left-pallidum, left-putamen, left-thalamus, left-ventricles, right-accumbens, right-amygdala, right-caudate, right-hippocampus, right-pallidum, right-putamen, right-thalamus, right-ventricles! You can re-order your subcortical dataset using our `reorder_sctx()` function. *Ventricles are optional.

Prerequisites

Load *summary statistics* **or** *example data*
Re-order subcortical data (*mega only*)
Z-score data (*mega only*)

Python | meta

```
>>> from enigmatoolbox.plotting import plot_subcortical

>>> # Project the results on the surface brain
```

(continues on next page)

(continued from previous page)

```
>>> plot_subcortical(array_name=SV_d, size=(800, 400),
...                  cmap='RdBu_r', color_bar=True, color_range=(-0.5, 0.5))
```

Matlab | meta

```
% Project the results on the surface brain
f = figure,
    plot_subcortical(SV_d, 'color_range', [-0.5 0.5], 'cmap', 'RdBu_r')
```

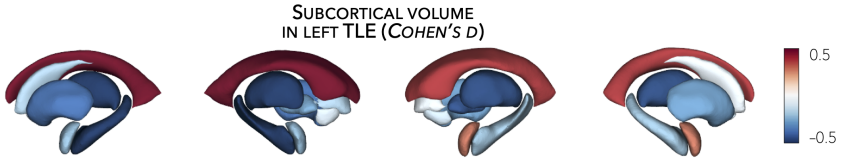
If you have **meta**-analysis data (*e.g.*, summary statistics)
If you have individual site or **mega**-analysis data

Python | mega

```
>>> from enigmatoolbox.plotting import plot_subcortical

>>> # Project the results on the surface brain
>>> plot_subcortical(array_name=SV_z_mean, size=(800, 400),
>>>                  cmap='Blues_r', color_bar=True, color_range=(-3, 0))
```

Matlab | meta

```
% Project the results on the surface brain
f = figure,
    plot_subcortical(SV_z_mean{:, :}, 'color_range', [-2 1], 'cmap', 'Blues_r')
```



7.11 Connectivity data

This page contains descriptions and examples to use Human Connectome Project (HCP) connectivity data.

Neuroimaging, particularly with functional and diffusion MRI, has become a leading tool to characterize human brain network organization *in vivo* and to identify network alterations in brain disorders. The **ENIGMA TOOLBOX** includes high-resolution structural (derived from diffusion-weighted tractography) and functional (derived from resting-state functional MRI) connectivity data from a cohort of unrelated healthy adults from the Human Connectome Project (HCP). Preprocessed cortico-cortical, subcortico-cortical, and subcortico-subcortical functional and structural connectivity matrices can be easily retrieved using the `load_fc()` and `load_sc()` functions.

High-resolution functional and structural data were parcellated according to the Desikan-Killiany, Glasser, as well as Schaefer 100, 200, 300, and 400 parcellations.

Normative functional connectivity matrices were generated by computing pairwise correlations between the time series of all cortical regions and subcortical (nucleus accumbens, amygdala, caudate, hippocampus, pallidum, putamen, thalamus) regions; negative connections were set to zero. Subject-specific connectivity matrices were then z-transformed

and aggregated across participants to construct a group-average functional connectome. Available cortico-cortical, subcortico-cortical, and subcortico-subcortical matrices are unthresholded.

Normative structural connectivity matrices were generated from preprocessed diffusion MRI data using MRtrix3. Anatomical constrained tractography was performed using different tissue types derived from the T1-weighted image, including cortical and subcortical gray matter, white matter, and cerebrospinal fluid. Multi-shell and multi-tissue response functions were estimated104 and constrained spherical deconvolution and intensity normalization were performed. The initial tractogram was generated with 40 million streamlines, with a maximum tract length of 250 and a fractional anisotropy cutoff of 0.06. Spherical-deconvolution informed filtering of tractograms (SIFT2) was applied to reconstruct whole-brain streamlines weighted by the cross-section multipliers. Reconstructed streamlines were mapped onto the 68 cortical and 14 subcortical (including hippocampus) regions to produce subject-specific structural connectivity matrices. The group-average normative structural connectome was defined using a distance-dependent thresholding procedure, which preserved the edge length distribution in individual patients, and was log transformed to reduce connectivity strength variance. As such, structural connectivity was defined by the number of streamlines between two regions (*i.e.*, fiber density).

Additional details on subject inclusion and data preprocessing are provided in the **ENIGMA TOOLBOX** preprint.

One matrix

Are you looking for subcortico-subcortical connectivity? Or do you want cortical and subcortical connectivity in one matrix? If so, check out our functions: `load_fc_as_one()` *here* and `load_sc_as_one()` *here*!

7.11.1 Load cortical connectivity matrices

The **ENIGMA TOOLBOX** provides structural (diffusion MRI) and functional (resting-state functional MRI) connectivity matrices obtained from the HCP.

Python

```
>>> from enigmatoolbox.datasets import load_sc, load_fc
>>> from nilearn import plotting

>>> # Load cortico-cortical functional connectivity data
>>> fc_ctx, fc_ctx_labels, _, _ = load_fc()

>>> # Load cortico-cortical structural connectivity data
>>> sc_ctx, sc_ctx_labels, _, _ = load_sc()

>>> # Plot cortico-cortical connectivity matrices
>>> fc_plot = plotting.plot_matrix(fc_ctx, figure=(9, 9), labels=fc_ctx_labels,␣
↪vmax=0.8, vmin=0, cmap='Reds')
>>> sc_plot = plotting.plot_matrix(sc_ctx, figure=(9, 9), labels=sc_ctx_labels,␣
↪vmax=10, vmin=0, cmap='Blues')
```

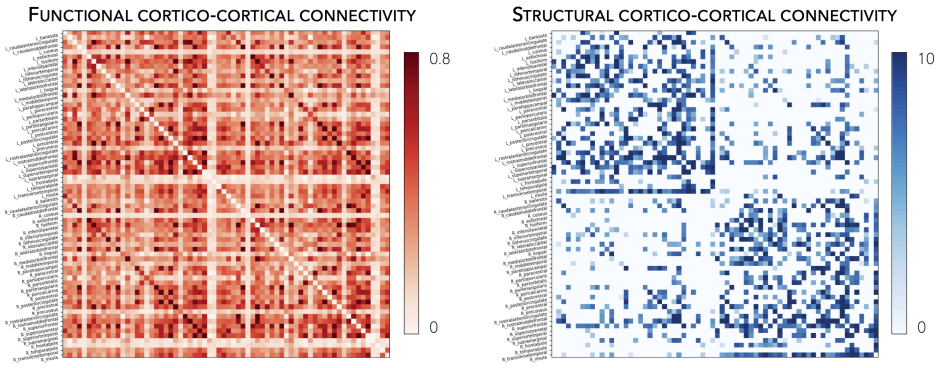
Matlab

```
% Load cortico-cortical functional connectivity data
[fc_ctx, fc_ctx_labels, ~, ~] = load_fc();

% Load cortico-cortical structural connectivity data
[sc_ctx, sc_ctx_labels, ~, ~] = load_sc();

% Plot cortico-cortical connectivity matrices
f = figure,
```

(continues on next page)

(continued from previous page)

```
  imagesc(fc_ctx, [0 0.8]);
  axis square;
  colormap(Reds);
  colorbar;
  set(gca, 'YTick', 1:1:length(fc_ctx_labels), ...
      'YTickLabel', fc_ctx_labels)

f = figure,
  imagesc(sc_ctx, [0 10]);
  axis square;
  colormap(Blues);
  colorbar;
  set(gca, 'YTick', 1:1:length(sc_ctx_labels), ...
      'YTickLabel', sc_ctx_labels)
```



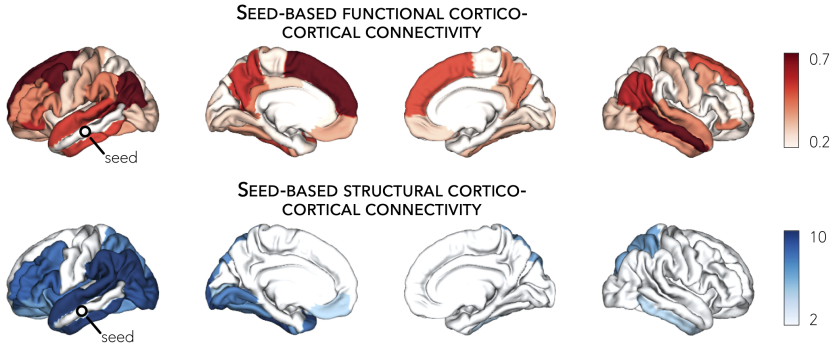
Following the example below, we can also extract seed-based cortico-cortical connectivity, using the left middle temporal gyrus as example seed.

Python

```
>>> from enigmatoolbox.utils.parcellation import parcel_to_surface
>>> from enigmatoolbox.plotting import plot_cortical

>>> # Extract seed-based connectivity
>>> seed = "L_middletemporal"
>>> seed_conn_fc = fc_ctx[[i for i, item in enumerate(fc_ctx_labels) if seed in item],
 ↪ ]
>>> seed_conn_sc = sc_ctx[[i for i, item in enumerate(sc_ctx_labels) if seed in item],
 ↪ ]

>>> # Map parcellated data to the surface
>>> seed_conn_fc_fsa5 = parcel_to_surface(seed_conn_fc, 'aparc_fsa5')
>>> seed_conn_sc_fsa5 = parcel_to_surface(seed_conn_sc, 'aparc_fsa5')

>>> # Project the results on the surface brain
>>> plot_cortical(array_name=seed_conn_fc_fsa5, surface_name="fsa5", size=(800, 400),
...               cmap='Reds', color_bar=True, color_range=(0.2, 0.7))

>>> plot_cortical(array_name=seed_conn_sc_fsa5, surface_name="fsa5", size=(800, 400),
...               cmap='Blues', color_bar=True, color_range=(2, 10))
```

Matlab

```
% Extract seed-based connectivity
seed = 'L_middletemporal'
seed_conn_fc = fc_ctx(find(strcmp(fc_ctx_labels, seed)), :)
seed_conn_sc = sc_ctx(find(strcmp(sc_ctx_labels, seed)), :)

% Map parcellated data to the surface
seed_conn_fc_fsa5 = parcel_to_surface(seed_conn_fc, 'aparc_fsa5');
seed_conn_sc_fsa5 = parcel_to_surface(seed_conn_sc, 'aparc_fsa5');

% Project the results on the surface brain
f = figure,
    plot_cortical(seed_conn_fc_fsa5, 'cmap', 'Reds', 'color_range', [0.2 0.7])

f = figure,
    plot_cortical(seed_conn_sc_fsa5, 'cmap', 'Blues', 'color_range', [2 10])
```



7.11.2 Load subcortical connectivity matrices

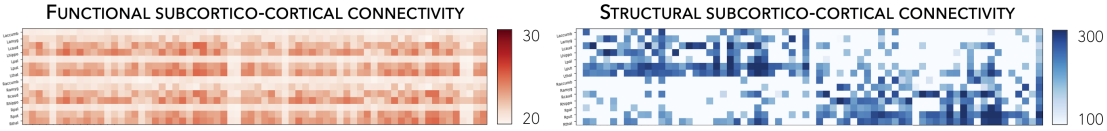
Subcortico-cortical as well as subcortico-subcortical connectivity matrices are also included in the **ENIGMA TOOLBOX**. As above, we can load these structural and functional matrices and extract seed-based connectivity from subcortical seeds.

Python

```
>>> from enigmatoolbox.datasets import load_sc, load_fc
>>> from nilearn import plotting

>>> # Load subcortico-cortical functional connectivity data
>>> _, _, fc_sctx, fc_sctx_labels = load_fc()

>>> # Load subcortico-cortical structural connectivity data
>>> _, _, sc_sctx, sc_sctx_labels = load_sc()

>>> # Plot subcortico-cortical connectivity matrices
>>> fc_plot = plotting.plot_matrix(fc_sctx, figure=(9, 9), labels=fc_sctx_labels,␣
↪vmax=0.5, vmin=0, cmap='Reds')
>>> sc_plot = plotting.plot_matrix(sc_sctx, figure=(9, 9), labels=sc_sctx_labels,␣
↪vmax=10, vmin=0, cmap='Blues')
```

Matlab

```
% Load subcortico-cortical functional connectivity data
[~, ~, fc_sctx, fc_sctx_labels] = load_fc();

% Load subcortico-cortical structural connectivity data
[~, ~, sc_sctx, sc_sctx_labels] = load_sc();

% Plot subcortico-cortical connectivity matrices
f = figure,
  imagesc(fc_sctx, [0 0.5]);
  axis square;
  colormap(Reds);
  colorbar;
  set(gca, 'YTick', 1:1:length(fc_sctx_labels), ...
      'YTickLabel', fc_sctx_labels)

f = figure,
  imagesc(sc_sctx, [0 10]);
  axis square;
  colormap(Blues);
  colorbar;
  set(gca, 'YTick', 1:1:length(sc_sctx_labels), ...
      'YTickLabel', sc_sctx_labels)
```



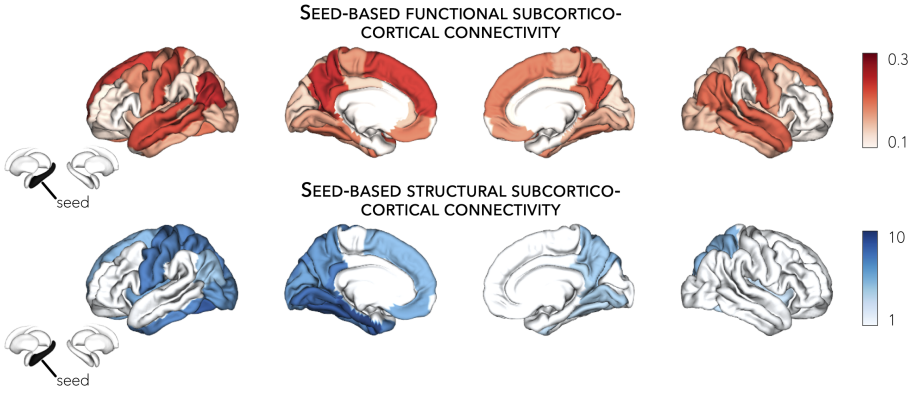
As described above, we can also extract seed-based subcortico-cortical connectivity, using the left hippocampus as example seed.

Python

```
>>> from enigmatoolbox.plotting import plot_cortical

>>> # Extract seed-based connectivity
>>> seed = "Lhippo"
>>> seed_conn_fc = fc_sctx[[i for i, item in enumerate(fc_sctx_labels) if seed in␣
↪item],]
>>> seed_conn_sc = sc_sctx[[i for i, item in enumerate(sc_sctx_labels) if seed in␣
↪item],]

>>> # Map parcellated data to the surface
>>> seed_conn_fc_fsa5 = parcel_to_surface(seed_conn_fc, 'aparc_fsa5')
>>> seed_conn_sc_fsa5 = parcel_to_surface(seed_conn_sc, 'aparc_fsa5')

>>> # Project the results on the surface brain
>>> plot_cortical(array_name=seed_conn_fc_fsa5, surface_name="fsa5", size=(800, 400),
...               cmap='Reds', color_bar=True, color_range=(0.1, 0.3))

>>> plot_cortical(array_name=seed_conn_sc_fsa5, surface_name="fsa5", size=(800, 400),
...               cmap='Blues', color_bar=True, color_range=(1, 10))
```

Matlab

```
% Extract seed-based connectivity
seed = 'Lhippo'
seed_conn_fc = fc_sctx(find(strcmp(fc_sctx_labels, seed)), :)
seed_conn_sc = sc_sctx(find(strcmp(sc_sctx_labels, seed)), :)

% Map parcellated data to the surface
seed_conn_fc_fsa5 = parcel_to_surface(seed_conn_fc, 'aparc_fsa5');
seed_conn_sc_fsa5 = parcel_to_surface(seed_conn_sc, 'aparc_fsa5');

% Project the results on the surface brain
f = figure,
    plot_cortical(seed_conn_fc_fsa5, 'cmap', 'Reds', 'color_range', [0.1 0.3])

f = figure,
    plot_cortical(seed_conn_sc_fsa5, 'cmap', 'Blues', 'color_range', [1 10])
```



7.12 Gene expression data

This page contains descriptions and examples to fetch microarray expression data.

7.12.1 Fetch gene expression data

The **ENGMA TOOLBOX** provides microarray expression data collected from six human donor brains and released by Allen Human Brain Atlas. Microarray expression data were first generated using abagen , a toolbox that provides reproducible workflows for processing and preparing gene co-expression data according to previously established recommendations (Arnatkeviciūtė et al., 2019, NeuroImage); preprocessing steps included intensity-based filtering of microarray probes, selection of a representative probe for each gene across both hemispheres, matching of microarray samples to brain parcels from the Desikan-Killiany, Glasser, and Schaefer parcellations, normalization, and aggregation within parcels and across donors. Moreover, genes whose similarity across donors fell below a threshold ($r < 0.2$) were removed, leaving a total of 12,668 genes for analysis (using the Desikan-Killiany atlas). To accommodate users, we also provide unthresholded gene datasets with varying stability thresholds (r 0.2, r 0.4, r 0.6, r 0.8) for every parcellation (https://github.com/saratheriver/enigma-extra).

Wanna know where we got those genes?

The Allen Human Brain Atlas microarray expression data loaded as part of the **ENIGMA TOOLBOX** was originally fetched from the abagen toolbox using the `abagen.get_expression_data()` command. For more flexibility, check out their toolbox!

Got NaNs?

Please note that two regions (right frontal pole and right temporal pole) in the Desikan-Killiany atlas were not matched to any tissue sample and thus are filled with NaN values in the data matrix.

Slow internet connection?

The command `fetch_ahba()` fetches a large (~24 MB) microarray dataset from the internet and may thus be incredibly slow to load if you lack a good connection. But don't you worry: you can download the relevant file by typing this command in your terminal `wget https://github.com/saratheriver/enigma-extra/raw/master/ahba/allgenes_stable_r0.2.csv` and specifying its path in the `fetch_ahba()` function as follows:`fetch_ahba('/path/to/allgenes_stable_r0.2.csv')`

Python

```
>>> from enigmatoolbox.datasets import fetch_ahba

>>> # Fetch gene expression data
>>> genes = fetch_ahba()

>>> # Obtain region labels
>>> reglabels = genes['label']

>>> # Obtain gene labels
>>> genelabels = list(genes.columns)[1]
```

Matlab

```
% Fetch gene expression data
genes = fetch_ahba();

% Obtain region labels
reglabels = genes.label;

% Obtain gene labels
genelabels = genes.Properties.VariableNames(2:end);
```

MICROARRAY EXPRESSION DATA

	label	A1BG	A1BG-AS1	AACS	AARS	ABCA6	ABCB7	ABCC12	ABCC5	ABCC8	...	ZNF8	ZNF831	ZNF845	ZNF883	ZNRF1	ZSCAN18	ZSCAN26	ZSCAN31	ZSCAN9	ZYX
0	L_banksst	0.601580	0.678272	0.773612	0.618798	0.404300	0.310859	0.676492	0.649539	0.657070	...	0.557786	0.579374	0.489690	0.478107	0.617499	0.717072	0.603006	0.581640	0.427673	0.604982
1	L_caudalanteriorcingulate	0.595611	0.636761	0.833183	0.554223	0.274412	0.272725	0.604715	0.646292	0.620104	...	0.524213	0.656095	0.386915	0.477859	0.531097	0.815701	0.581669	0.529964	0.414542	0.519334
2	L_caudalmiddlefrontal	0.506221	0.658952	0.811766	0.663592	0.437378	0.353123	0.670223	0.559286	0.686908	...	0.580998	0.536016	0.391200	0.462643	0.530420	0.639634	0.558027	0.529937	0.394182	0.655353
3	L_cuneus	0.436322	0.656161	0.787701	0.650927	0.520170	0.419390	0.629259	0.664689	0.739321	...	0.644420	0.472402	0.344207	0.372340	0.593859	0.568385	0.697267	0.604001	0.560236	0.746896
4	L_entorhinal	0.614204	0.614976	0.730651	0.513764	0.459002	0.455035	0.502035	0.605741	0.570617	...	0.579952	0.616730	0.525545	0.620153	0.657757	0.687672	0.499880	0.489746	0.354915	0.456168
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
77	Rcaud	0.394455	0.225060	0.365251	0.474975	0.697754	0.586727	0.023607	0.279043	0.192442	...	0.374146	0.131205	0.650184	0.614419	0.252169	0.371653	0.329511	0.278332	0.645584	0.365695
78	Rhippo	0.738758	0.390843	0.420328	0.275643	0.302411	0.766968	0.157808	0.553915	0.266999	...	0.494237	0.776457	0.593266	0.749790	0.520674	0.831319	0.558408	0.383076	0.374888	0.261280
79	Rpal	0.697285	0.447848	0.367506	0.449150	0.376826	0.722208	0.038749	0.455514	0.427285	...	0.463340	0.541499	0.672838	0.807044	0.434026	0.563669	0.500754	0.371397	0.372596	0.349777
80	Rput	0.619848	0.318972	0.271222	0.283888	0.273621	0.670247	0.048987	0.484805	0.308294	...	0.442804	0.552287	0.608480	0.796927	0.407578	0.505836	0.281885	0.422108	0.281023	0.224110
81	Rthal	0.645202	0.671091	0.750269	0.600841	0.393281	0.352636	0.518519	0.609629	0.714981	...	0.614368	0.624737	0.489976	0.549560	0.624879	0.738588	0.566086	0.532501	0.403335	0.559198

7.13 Disease-related transcriptomics

This page contains descriptions and examples to extract GWAS-implicated gene expression data and project them to cortical and subcortical surfaces. In the following tutorial, we will use epilepsy-related genes (more specifically, genes related to hippocampal sclerosis) as an example, but feel free to replace *epilepsy* with any other disorder listed below.

7.13.1 Extract disease-related genes

As part of the **ENIGMA TOOLBOX**, users can query pre-defined lists of disease-related genes (obtained from several recently published GWAS), including gene sets for:
attention deficit/hyperactivity disorder
autism spectrum disorder
bipolar disorder
depression
common epilepsies
schizophrenia
tourette's syndrome

* *indicates disease-related genes used in the code snippets.*

Caution

Pre-defined gene sets are obtained from individual studies and are liable to be changed. We welcome any suggestions you may have on defining proper disease-related gene sets and are happy to expand this function to include other interesting disorders. Get in touch with us here.

Prerequisites
Fetch *gene expression data*

Python

```
>>> from enigmatoolbox.datasets import risk_genes

>>> # Get the names of epilepsy-related genes (Focal HS phenotype)
>>> epilepsy_genes = risk_genes('epilepsy')['focalhs']

>>> # Extract gene expression data for these Focal HS genes
>>> epilepsy_gene_data = genes[genes.columns.intersection(epilepsy_genes)]
```

Matlab

```
% Get the names of epilepsy-related genes (Focal HS phenotype)
epilepsy_genes = risk_genes('epilepsy');
epilepsy_genes = epilepsy_genes.focalhs;

% Extract gene expression data for these Focal HS genes
epilepsy_gene_data = genes(:, ismember(genes.Properties.VariableNames, ...
                           epilepsy_genes));
```

7.13.2 Visualize disease-related gene expression maps

Following up on the above example, we provide a brief example to project gene expression maps to the surface. Once again, we use genes related to hippocampal sclerosis as an example.

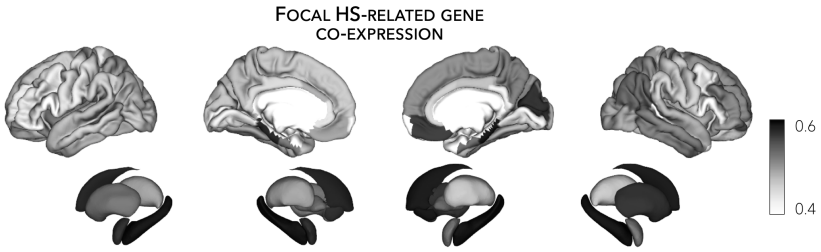
Prerequisites
Fetch *gene expression data*
Extract *disease-related gene data*

Python

```
>>> import numpy as np
>>> from enigmatoolbox.utils.parcellation import parcel_to_surface
>>> from enigmatoolbox.plotting import plot_cortical, plot_subcortical

>>> # Compute the mean co-expression across all Focal HS genes
>>> mean_epilepsy_genes = np.mean(epilepsy_gene_data, axis=1)

>>> # Separate cortical (ctx) from subcortical (sctx) regions
>>> mean_epilepsy_genes_ctx = mean_epilepsy_genes[:68]
>>> mean_epilepsy_genes_sctx = mean_epilepsy_genes[68:]

>>> # Map the parcellated gene expression data to our surface template (cortical␣
↪values only)
>>> mean_epilepsy_genes_ctx_fsa5 = parcel_to_surface(mean_epilepsy_genes_ctx, 'aparc_
↪fsa5')

>>> # Project the results on the surface brain
>>> plot_cortical(array_name=mean_epilepsy_genes_ctx_fsa5, surface_name="fsa5",␣
↪size=(800, 400), nan_color=(1, 1, 1, 1),
...               cmap='Greys', color_bar=True, color_range=(0.4, 0.6))

>>> plot_subcortical(array_name=mean_epilepsy_genes_sctx, ventricles=False, size=(800,
↪ 400),
...                 cmap='Greys', color_bar=True, color_range=(0.4, 0.6))
```

Matlab

```
% Compute the mean co-expression across all Focal HS genes
mean_epilepsy_genes = mean(epilepsy_gene_data{:, :}, 2);

% Separate cortical (ctx) from subcortical (sctx) regions
mean_epilepsy_genes_ctx  = mean_epilepsy_genes(1:68);
mean_epilepsy_genes_sctx = mean_epilepsy_genes(69:end);

% Map the parcellated gene expression data to our surface template (cortical values␣
↪only)
mean_epilepsy_genes_ctx_fsa5 = parcel_to_surface(mean_epilepsy_genes_ctx, 'aparc_fsa5
↪');

% Project the results on the surface brain
f = figure,
    plot_cortical(mean_epilepsy_genes_ctx_fsa5, 'color_range', ...
                  [0.4 0.6], 'cmap', 'Greys')

f = figure,
    plot_subcortical(mean_epilepsy_genes_sctx, 'ventricles', 'False', ...
                     'color_range', [0.4 0.6], 'cmap', 'Greys')
```



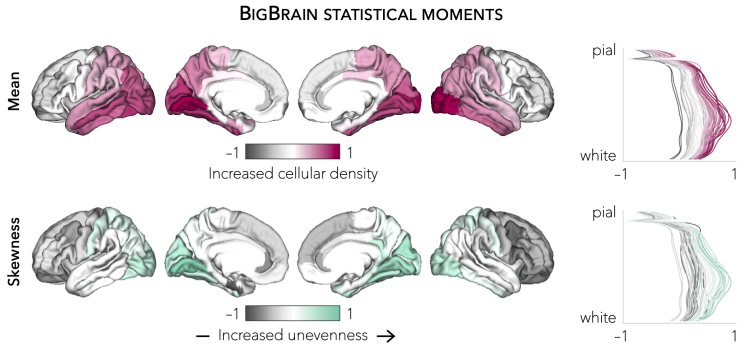
7.14 BigBrain moments & gradient

This page contains descriptions and examples to stratify and visualize surface-based findings according to BigBrain statistical moments and gradient.

BigBrain is a ultra-high resolution, 3D volumetric reconstruction of a *postmortem* Merker-stained and sliced human brain from a 65-year-old male, with specialized pial and white matter surface reconstructions (obtained via the open-access BigBrain repository. The *postmortem* brain was paraffin-embedded, coronally sliced into 7400 20m sections, silver-stained for cell bodies, and digitized. A 3D reconstruction was implemented with a successive coarse-to-fine hierarchical procedure, resulting in a full brain volume. For the **ENIGMA TOOLBOX**, we used the highest resolution full brain volume (100m isotropic voxels), then generated 50 equivolumetric surfaces between the pial and white matter surfaces. The equivolumetric model compensates for cortical folding by varying the Euclidean distance between pairs of intracortical surfaces throughout the cortex, thus preserving the fractional volume between surfaces. Next, staining intensity profiles, representing neuronal density and soma size by cortical depth, were sampled along 327,684 surface points in the direction of cortical columns.

7.14.1 BigBrain statistical moments

As part of the **ENIGMA Toolbox**, we parametrized cytoarchitectural properties of the BigBrain by using two central moments (*i.e.*, mean and skewness) calculated across several cortical depths. In essence, studying the mean of BigBrain intensity profiles across the cortical mantle probes cellular/neuronal density, whereas analysis of skewness contrasts deep and superficial cortical layers, indexing the unevenness of cellular distribution, a critical dimension of laminar differentiation. Finally, the Desikan-Killiany atlas was nonlinearly transformed to the BigBrain histological surfaces Lewis et al., 2019, OHBM and central moments were averaged within each parcels, excluding outlier vertices with values more than three scaled median absolute deviations away from the parcel median.



In the following example, we first threshold and display a cortical thickness map to highlight areas of marked atrophy in patients *vs*. controls (using left TLE *vs*. controls for the example below).

Prerequisites

Load *summary statistics* **or** *example data*
Z-score data (*mega only*)

Python | meta

```
>>> import numpy as np
>>> from enigmatoolbox.plotting import plot_cortical
>>> from enigmatoolbox.utils.parcellation import parcel_to_surface

>>> # Extract FDR-corrected p-values and find regions with p < 0.01
>>> region_idx = np.where(CT['fdr_p'].to_numpy() <= 0.01)

>>> # Visualize thresholded Cohen's d map
>>> CT_d_thr = np.zeros((68,))
>>> CT_d_thr[region_idx] = CT_d.iloc[region_idx]
>>> plot_cortical(array_name=parcel_to_surface(CT_d_thr, 'aparc_fsa5'), surface_name=
↪"fsa5", size=(800, 400),
...               cmap='RdBu_r', color_bar=True, color_range=(-0.5, 0.5))
```

Matlab | meta

```
% Extract FDR-corrected p-values and find regions with p < 0.01
region_idx  = find(CT.fdr_p <= 0.01);

% Visualize thresholded Cohen's d map
CT_d_thr              = zeros(length(CT_d), 1);
CT_d_thr(region_idx)  = CT_d(region_idx);
f = figure,
    plot_cortical(parcel_to_surface(CT_d_thr), 'color_range', [-0.5 0.5])
```

If you have **meta**-analysis data (*e.g.*, summary statistics)
If you have individual site or **mega**-analysis data

Python | mega

```
>>> import numpy as np
>>> from enigmatoolbox.plotting import plot_cortical
>>> from enigmatoolbox.utils.parcellation import parcel_to_surface

>>> # Extract regions with z-score < -1
>>> region_idx = np.where(CT_z_mean.to_numpy() <= -1)

>>> # Visualize z-score thresholded map
>>> CT_z_mean_thr = np.zeros((68,))
>>> CT_z_mean_thr[region_idx] = CT_z_mean.iloc[region_idx]
>>> plot_cortical(array_name=parcel_to_surface(CT_z_mean_thr, 'aparc_fsa5'), surface_
↪name="fsa5",
...               size=(800, 400), cmap='RdBu_r', color_bar=True, color_range=(-1, 1))
```

Matlab | mega

```
% Extract regions with z-score < -1
region_idx  = find(CT_z_mean{:, :} <= -1);

% Visualize z-score thresholded map
CT_z_mean_thr              = zeros(length(CT_z_mean{:, :}), 1);
```

(continues on next page)

(continued from previous page)

```
CT_z_mean_thr(region_idx)  = CT_z_mean{:, :}(region_idx);
f = figure,
    plot_cortical(parcel_to_surface(CT_z_mean_thr), 'color_range', [-1 1])
```

From the following code snippet, we can then contextualize and visualize these marked patterns of atrophy with respect to intensity profiles reflecting microstructural composition (*e.g.*, cellular density, cellular distribution asymmetry) along cortical columns.

Prerequisites

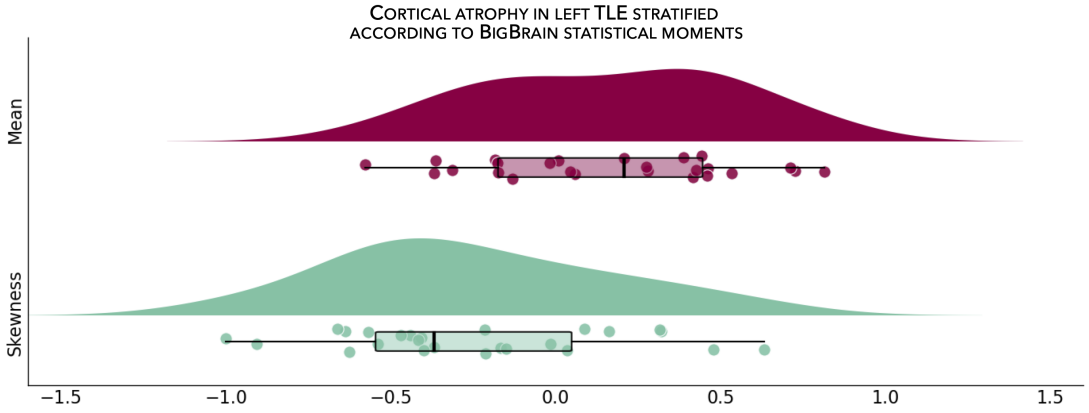
Load *summary statistics* **or** *example data*
Z-score data (*mega only*)
Threshold surface maps

Python

```
>>> from enigmatoolbox.histology import bb_moments_raincloud

>>> # Stratify and plot results according to BigBrain statistical moments
>>> bb_moments_raincloud(region_idx=region_idx)
```

Matlab

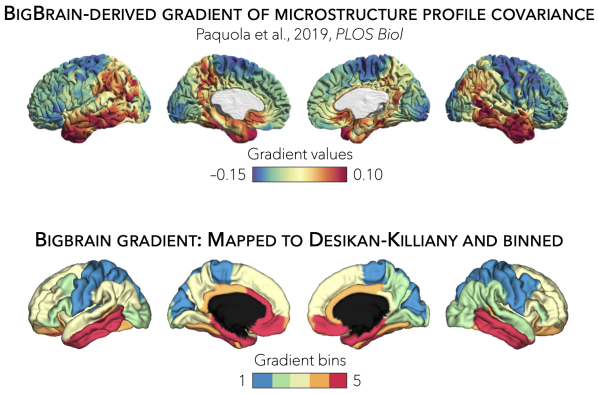
```
% Stratify and plot results according to BigBrain statistical moments
f = figure,
    bb_moments_raincloud(region_idx)
```



7.14.2 BigBrain gradient

In addition to statistical moments, we also incorporated the BigBrain microstructural profile covariance (MPC) gradient from the original publication (Paquola et al., 2019, PLoS Biol). In brief, the authors derived an MPC matrix by correlating BigBrain intensity profiles between every pair of regions in a 1,012 cortical node parcellation, controlling for the average whole-cortex intensity profile. The MPC matrix was thresholded row-wise to retain the top 10% of correlations and converted into a normalized angle matrix. Diffusion map embedding, a nonlinear manifold learning technique, identified the principal axis of variation across cortical areas, *i.e.*, the BigBrain gradient. In this space, cortical nodes that are strongly similar are closer together, whereas nodes with little to no intercovariance are farther apart.

To allow contextualization of ENIGMA-derived surface-based findings, we mapped the BigBrain gradient, which describes a sensory-fugal transition in intracortical microstructure, to the Desikan-Killiany atlas and partitioned it into five equally sized discrete bins. Stratifying cortical findings relative to this gradient can, for example, test whether patterns of changes are conspicuous in cortices with marked laminar differentiation (*e.g.*, 1st bin; sensory and motor cortices) or in those with subtle laminar differentiation (*e.g.*, 5th bin limbic cortices).



In the following example, we can use our thresholded (or unthresholded) a cortical map (*e.g.*, cortical thickness effect sizes) to contextualize and visualize patterns of marked atrophy with respect to each gradient bin.

Prerequisites

Load *summary statistics* **or** *example data*
Z-score data (*mega only*)
Threshold surface maps

Python | meta

```
>>> import numpy as np
>>> from enigmatoolbox.histology import bb_gradient_plot

>>> # Stratify and plot results according to the BigBrain gradient
>>> bb_gradient_plot(data=np.where(CT_d_thr == 0, np.nan, CT_d_thr),
...                  axis_range=(-0.6, 0.25), yaxis_label='Cohen\'s $d$')
```

Matlab | meta

```
% Stratify and plot results according to the BigBrain gradient
CT_d_thr(CT_d_thr == 0) = nan;
f = figure,
    bb_gradient_plot(CT_d_thr, 'axis_range', [-0.6 0.25], ...
                'yaxis_label', 'Cohen'' {\it d}')
```

If you have **meta**-analysis data (*e.g.*, summary statistics)
If you have individual site or **mega**-analysis data

Python | mega

```
>>> import numpy as np
>>> from enigmatoolbox.histology import bb_gradient_plot
```

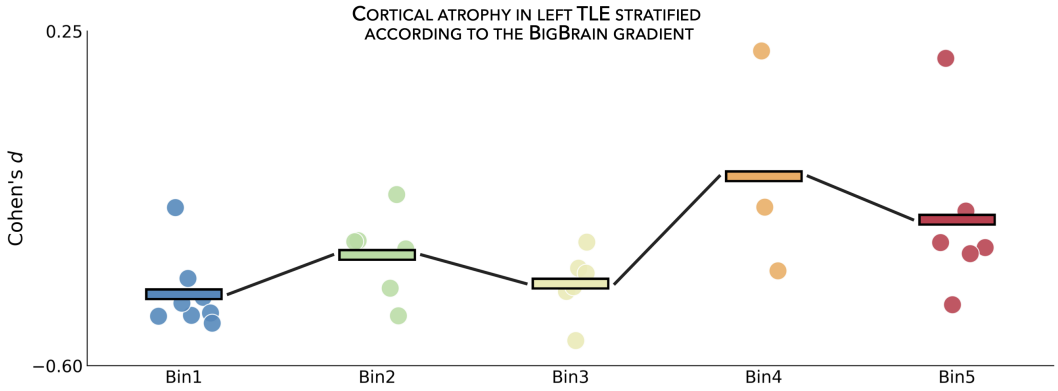
(continues on next page)

(continued from previous page)

```
>>> # Stratify and plot results according to the BigBrain gradient
>>> bb_gradient_plot(data=np.where(CT_z_mean_thr == 0, np.nan, CT_z_mean_thr),
...                  axis_range=(-2, -0.75), yaxis_label='$z$-score')
```

Matlab | mega

```
% Stratify and plot results according to the BigBrain gradient
CT_z_mean_thr(CT_z_mean_thr == 0) = nan;
f = figure,
    bb_gradient_plot(CT_z_mean_thr, 'axis_range', [-2 -0.75], ...
                  'yaxis_label', '{\it z}-score')
```



7.15 Economo-Koskinas cytoarchitectonics

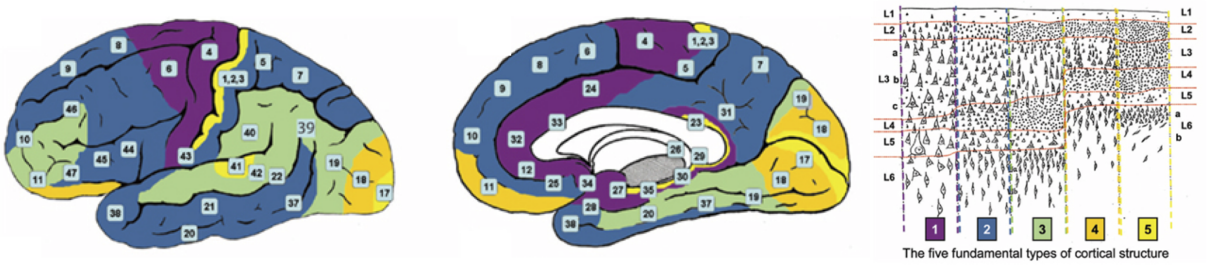
This page contains descriptions and examples to stratify and visualize surface-based findings according to cyatoarchitectural variation.

7.15.1 Economo-Koskinas stratification

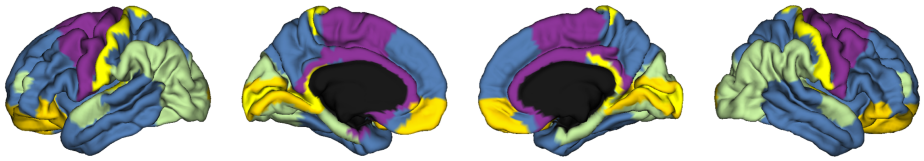
As part of the **ENIGMA TOOLBOX**, we included a digitized parcellation of von Economo and Koskina's cytoarchitectonic mapping of the human cerebral cortex, from which five different structural types of cerebral cortex can be described: *i*) **agranular** (*purple*; thick with large cells but sparse layers II and IV), *ii*) **frontal** (*blue*; thick but not rich in cellular substance, visible layers II and IV), *iii*) **parietal** (*green*; thick and rich in cells with dense layers II and IV but small and slender pyramidal cells), *iv*) **polar** (*orange*; thin but rich in cells, particularly in granular layers), and *v*) **granular** or koniocortex (*yellow*; thin but rich in smalls cells, even in layer IV, and a rarified layer V).

VON ECONOMO AND KOSKINAS CYTOARCHITECTONICS ATLAS

von Economo C & Koskinas GN, 1925; Van Hout Solari S & Stoner R, 2011, *Front Neuroanat*



DIGITIZED VON ECONOMO AND KOSKINAS CYTOARCHITECTONICS ATLAS



In the following example, we contextualized disease-related cortical atrophy patterns with respect to the well-established von Economo and Koskinas cytoarchitectonic atlas by summarizing cortex-wide effects across each of the five structural types of isocortex. To ease interpretation, stratification of findings based on the von Economo and Koskinas atlas are also visualized in a spider plot. Here, negative values (towards the center) represent greater atrophy in disease cases relative to healthy controls.

Prerequisites

Load *summary statistics* **or** *example data*
Z-score data (*mega only*)

Python | meta

```
>>> from enigmatoolbox.histology import economo_koskinas_spider

>>> # Stratify cortical atrophy based on Economo-Koskinas classes
>>> class_mean = economo_koskinas_spider(CT_d, axis_range=(-0.4, 0))
```

Matlab | meta

```
% Stratify cortical atrophy based on Economo-Koskinas classes
class_mean = economo_koskinas_spider(CT_d, 'axis_range', [-0.4 0])
```

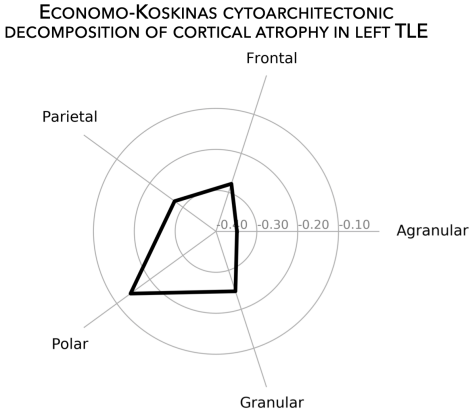
If you have **meta**-analysis data (*e.g.*, summary statistics)
If you have individual site or **mega**-analysis data

Python | mega

```
>>> from enigmatoolbox.histology import economo_koskinas_spider

>>> # Stratify cortical atrophy based on Economo-Koskinas classes
>>> class_mean = economo_koskinas_spider(CT_z_mean, axis_range=(-1, 0))
```

Matlab | meta

```
% Stratify cortical atrophy based on Economo-Koskinas classes
class_mean = economo_koskinas_spider(CT_z_mean{:, :}, 'axis_range', [-1 0])
```



7.16 Hub susceptibility

This page contains descriptions and examples to build hub susceptibility models. For additional details on hub susceptibility models, please see our manuscript entitled Network-based atrophy modeling in the common epilepsies: a worldwide ENIGMA study.

7.16.1 Cortical hubs

Normative structural and functional connectomes hold valuable information for relating macroscopic brain network organization to patterns of disease-related atrophy. Using the *HCP connectivity data*, we can first compute weighted (optimal for unthresholded connectivity matrices) degree centrality to identify structural and functional hub regions (*i.e.*, brain regions with many connections). This is done by simply computing the sum of all weighted cortico-cortical connections for every region. Higher degree centrality denotes increased hubness.

Prerequisites

Load cortico-cortical connectivity matrices

Python

```
>>> import numpy as np
>>> from enigmatoolbox.plotting import plot_cortical
>>> from enigmatoolbox.utils.parcellation import parcel_to_surface

>>> # Compute weighted degree centrality measures from the connectivity data
>>> fc_ctx_dc = np.sum(fc_ctx, axis=0)
>>> sc_ctx_dc = np.sum(sc_ctx, axis=0)

>>> # Map parcellated data to the surface
>>> fc_ctx_dc_fsa5 = parcel_to_surface(fc_ctx_dc, 'aparc_fsa5')
>>> sc_ctx_dc_fsa5 = parcel_to_surface(sc_ctx_dc, 'aparc_fsa5')

>>> # Project the results on the surface brain
>>> plot_cortical(array_name=fc_ctx_dc_fsa5, surface_name="fsa5", size=(800, 400),
...               cmap='Reds', color_bar=True, color_range=(20, 30))
```

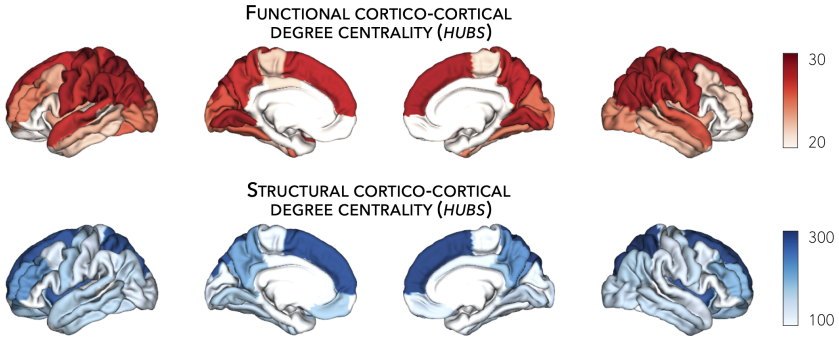
(continues on next page)

(continued from previous page)

```
>>> plot_cortical(array_name=sc_ctx_dc_fsa5, surface_name="fsa5", size=(800, 400),
...                cmap='Blues', color_bar=True, color_range=(100, 300))
```

Matlab

```
% Compute weighted degree centrality measures from the connectivity data
fc_ctx_dc           = sum(fc_ctx);
sc_ctx_dc           = sum(sc_ctx);

% Map parcellated data to the surface
fc_ctx_dc_fsa5      = parcel_to_surface(fc_ctx_dc, 'aparc_fsa5');
sc_ctx_dc_fsa5      = parcel_to_surface(sc_ctx_dc, 'aparc_fsa5');

% Project the results on the surface brain
f = figure,
  plot_cortical(fc_ctx_dc_fsa5, 'surface_name', 'fsa5', 'color_range', [20 30], ...
                'cmap', 'Reds', 'label_text', 'Functional degree centrality')

f = figure,
  plot_cortical(sc_ctx_dc_fsa5, 'surface_name', 'fsa5', 'color_range', [100 300], ...
                'cmap', 'Blues', 'label_text', 'Structural degree centrality')
```



7.16.2 Subcortical hubs

The *HCP connectivity data* can also be used to identify structural and functional subcortico-cortical hub regions. As above, we simply compute the sum of all weighted subcortico-cortical connections for every subcortical area. Once again, higher degree centrality denotes increased hubness.

No ventricles, no problem

Because we do not have connectivity values for the ventricles, do make sure to set the "ventricles" flag to `False` when displaying the findings on the subcortical surfaces.

Prerequisites

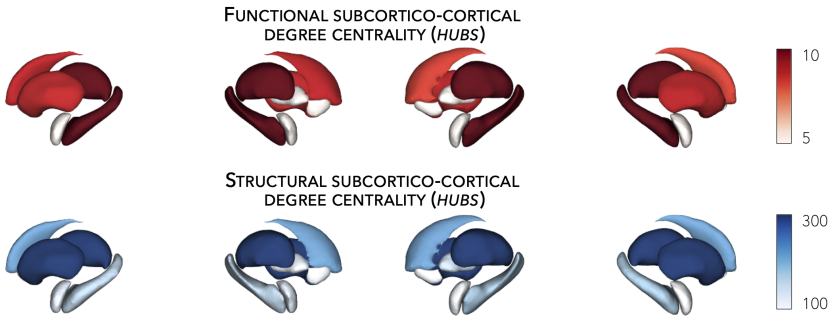
Load subcortico-cortical connectivity matrices

Python

```
>>> import numpy as np
>>> from enigmatoolbox.plotting import plot_subcortical

>>> # Compute weighted degree centrality measures from the connectivity data
>>> fc_sctx_dc = np.sum(fc_sctx, axis=1)
>>> sc_sctx_dc = np.sum(sc_sctx, axis=1)

>>> # Project the results on the surface brain
>>> plot_subcortical(array_name=fc_sctx_dc, ventricles=False, size=(800, 400),
...                  cmap='Reds', color_bar=True, color_range=(5, 10))

>>> plot_subcortical(array_name=sc_sctx_dc, ventricles=False, size=(800, 400),
...                  cmap='Blues', color_bar=True, color_range=(100, 300))
```

Matlab

```
% Compute weighted degree centrality measures from the connectivity data
fc_sctx_dc          = sum(fc_sctx, 2);
sc_sctx_dc          = sum(sc_sctx, 2);

% Project the results on the surface brain
f = figure,
  plot_subcortical(fc_sctx_dc, 'ventricles', 'False', 'color_range', [5 10], ...
                   'cmap', 'Reds', 'label_text', 'Functional degree centrality')

f = figure,
  plot_subcortical(sc_sctx_dc, 'ventricles', 'False', 'color_range', [100 300], ...
                   'cmap', 'Blues', 'label_text', 'Structural degree centrality')
```



7.16.3 Hub-atrophy correlations

Now that we have established the spatial distribution of hubs in the brain, we can then assess whether these hub regions are selectively vulnerable to syndrome-specific atrophy patterns. For simplicity, in the following example, we will spatially correlate degree centrality measures to measures of cortical and subcortical atrophy (where lower values indicate greater atrophy relative to controls).

Prerequisites

Load *summary statistics* **or** *example data*
Re-order subcortical data (*mega only*)
Z-score data (*mega only*)
Load *cortico-cortical* and *subcortico-cortical* connectivity matrices

Compute *cortical-cortical* and *subcortico-cortical* degree centrality

Python | meta

```
>>> import numpy as np

>>> # Remove subcortical values corresponding to the ventricles
>>> # (as we don't have connectivity values for them!)
>>> SV_d_noVent = SV_d.drop([np.where(SV['Structure'] == 'LLatVent')[0][0],
...                          np.where(SV['Structure'] == 'RLatVent')[0][0]]).reset_
↪index(drop=True)

>>> # Perform spatial correlations between functional hubs and Cohen's d
>>> fc_ctx_r = np.corrcoef(fc_ctx_dc, CT_d)[0, 1]
>>> fc_sctx_r = np.corrcoef(fc_sctx_dc, SV_d_noVent)[0, 1]

>>> # Perform spatial correlations between structural hubs and Cohen's d
>>> sc_ctx_r = np.corrcoef(sc_ctx_dc, CT_d)[0, 1]
>>> sc_sctx_r = np.corrcoef(sc_sctx_dc, SV_d_noVent)[0, 1]

>>> # Store correlation coefficients
>>> rvals = {'functional cortical hubs': fc_ctx_r, 'functional subcortical hubs': fc_
↪sctx_r,
...          'structural cortical hubs': sc_ctx_r, 'structural subcortical hubs': sc_
↪sctx_r}
```

Matlab | meta

```
% Remove subcortical values corresponding the ventricles
% (as we don't have connectivity values for them!)
SV_d_noVent = SV_d;
SV_d_noVent([find(strcmp(SV.Structure, 'LLatVent')); ...
             find(strcmp(SV.Structure, 'RLatVent'))], :) = [];

% Perform spatial correlations between cortical hubs and Cohen's d
fc_ctx_r = corrcoef(fc_ctx_dc, CT_d);
sc_ctx_r = corrcoef(sc_ctx_dc, CT_d);

% Perform spatial correlations between structural hubs and Cohen's d
fc_sctx_r = corrcoef(fc_sctx_dc, SV_d_noVent);
sc_sctx_r = corrcoef(sc_sctx_dc, SV_d_noVent);

% Store correlation coefficients
rvals = cell2struct({fc_ctx_r(1, 2), fc_sctx_r(1, 2), sc_ctx_r(1, 2), sc_sctx_r(1, 2)}
↪, ...
                    {'functional_cortical_hubs', 'functional_subcortical_hubs', ...
                     'structural_cortical_hubs', 'structural_subcortical_hubs'}, 2);
```

If you have **meta**-analysis data (*e.g.*, summary statistics)
If you have individual site or **mega**-analysis data

Python | mega

```
>>> import numpy as np

>>> # Remove subcortical values corresponding to the ventricles
>>> # (as we don't have connectivity values for them!)
>>> SV_z_mean_noVent = SV_z_mean.drop(['LLatVent', 'RLatVent']).reset_index(drop=True)

>>> # Perform spatial correlations between functional hubs and z-scores
>>> fc_ctx_r = np.corrcoef(fc_ctx_dc, CT_z_mean)[0, 1]
>>> fc_sctx_r = np.corrcoef(fc_sctx_dc, SV_z_mean_noVent)[0, 1]

>>> # Perform spatial correlations between structural hubs and z-scores
>>> sc_ctx_r = np.corrcoef(sc_ctx_dc, CT_z_mean)[0, 1]
>>> sc_sctx_r = np.corrcoef(sc_sctx_dc, SV_z_mean_noVent)[0, 1]

>>> # Store correlation coefficients
>>> rvals = {'functional cortical hubs': fc_ctx_r, 'functional subcortical hubs': fc_
↪sctx_r,
...          'structural cortical hubs': sc_ctx_r, 'structural subcortical hubs': sc_
↪sctx_r}
```

Matlab | mega

```
% Remove subcortical values corresponding the ventricles
% (as we don't have connectivity values for them!)
SV_z_mean_noVent = SV_z_mean;
SV_z_mean_noVent.LLatVent = [];
SV_z_mean_noVent.RLatVent = [];

% Perform spatial correlations between cortical hubs and Cohen's d
fc_ctx_r = corrcoef(fc_ctx_dc, CT_z_mean{:, :});
sc_ctx_r = corrcoef(sc_ctx_dc, CT_z_mean{:, :});

% Perform spatial correlations between structural hubs and Cohen's d
fc_sctx_r = corrcoef(fc_sctx_dc, SV_z_mean_noVent{:, :});
sc_sctx_r = corrcoef(sc_sctx_dc, SV_z_mean_noVent{:, :});

% Store correlation coefficients
rvals = cell2struct({fc_ctx_r(1, 2), fc_sctx_r(1, 2), sc_ctx_r(1, 2), sc_sctx_r(1, 2)}
↪, ...
                    {'functional_cortical_hubs', 'functional_subcortical_hubs', ...
                     'structural_cortical_hubs', 'structural_subcortical_hubs'}, 2);
```

7.16.4 Plot hub-atrophy correlations

Now that we have done all the necessary analyses, we can finally display our correlations. Here, a negative correlation indicates that greater atrophy correlates with the spatial distribution of hub regions (greater degree centrality).

Prerequisites

The script below can be used to show relationships between any two␣
↪variables, as for example:

degree centrality *vs.* atrophy

Load *summary statistics* **or** *example data*

Re-order subcortical data (*mega only*)

```
        Z-score data (mega only)
        Load cortico-cortical and subcortico-cortical connectivity matrices
        Compute cortical-cortical and subcortico-cortical degree centrality
        Assess statistical significance via spin permutation tests
```

Python | meta

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np

>>> # Store degree centrality and atrophy measures
>>> meas = {('functional cortical hubs', 'cortical thickness'): [fc_ctx_dc, CT_d],
...         ('functional subcortical hubs', 'subcortical volume'): [fc_sctx_dc, SV_d_
↪noVent],
...         ('structural cortical hubs', 'cortical thickness'): [sc_ctx_dc, CT_d],
...         ('structural subcortical hubs', 'subcortical volume'): [sc_sctx_dc, SV_d_
↪noVent]}

>>> fig, axs = plt.subplots(1, 4, figsize=(15, 3))

>>> for k, (fn, dd) in enumerate(meas.items()):
>>>     # Define scatter colors
>>>     if k <= 1:
>>>         col = '#A8221C'
>>>     else:
>>>         col = '#324F7D'

>>>     # Plot relationships between hubs and atrophy
>>>     axs[k].scatter(meas[fn][0], meas[fn][1], color=col,
...                    label='$r$={:.2f}'.format(rvals[fn[0]]) + '\n$p$={:.3f}'.
↪format(p_and_d[fn[0]][0]))
>>>     m, b = np.polyfit(meas[fn][0], meas[fn][1], 1)
>>>     axs[k].plot(meas[fn][0], m * meas[fn][0] + b, color=col)
>>>     axs[k].set_ylim((-1, 0.5))
>>>     axs[k].set_xlabel('{}'.format(fn[0].capitalize()))
>>>     axs[k].set_ylabel('{}'.format(fn[1].capitalize()))
>>>     axs[k].spines['top'].set_visible(False)
>>>     axs[k].spines['right'].set_visible(False)
>>>     axs[k].legend(loc=1, frameon=False, markerscale=0)

>>> fig.tight_layout()
>>> plt.show()
```

Matlab | meta

```
% Store degree centrality measures
meas  =  cell2struct({fc_ctx_dc.', fc_sctx_dc, sc_ctx_dc.', sc_sctx_dc}, ...
                     {'Functional_cortical_hubs', 'Functional_subcortical_hubs', ...
                     'Structural_cortical_hubs', 'Structural_subcortical_hubs'}, 2);
fns   = fieldnames(meas);

% Store atrophy measures
meas2 =  cell2struct({CT_d, SV_d_noVent}, {'Cortical_thickness', 'Subcortical_volume'}
↪, 2);
fns2  = fieldnames(meas2);

f = figure,
    set(gcf,'color','w');
```

(continues on next page)

(continued from previous page)

```
    set(gcf,'units','normalized','position',[0 0 1 0.3])
    k2 = [1 2 1 2];

    for k = 1:numel(fieldnames(meas))
        j = k2(k);

        % Define plot colors
        if k <= 2; col = [0.66 0.13 0.11]; else; col = [0.2 0.33 0.49]; end

        % Plot relationships between hubs and atrophy
        axs = subplot(1, 4, k); hold on
        s   = scatter(meas.(fns{k}), meas2.(fns2{j}), 88, col, 'filled');
        P1      = polyfit(meas.(fns{k}), meas2.(fns2{j}), 1);
        yfit_1  = P1(1) * meas.(fns{k}) + P1(2);
        plot(meas.(fns{k}), yfit_1, 'color',col, 'LineWidth', 3)
        ylim([-1 0.5])
        xlabel(strrep(fns{k}, '_', ' '))
        ylabel(strrep(fns2{j}, '_', ' '))
        legend(s, ['{\it r}=' num2str(round(rvals.(lower(fns{k})), 2)) newline ...
                   '{\it p}=' num2str(round(p_and_d.(lower(fns{k}))(1), 3))])
        legend boxoff
    end
```

If you have **meta**-analysis data (*e.g.*, summary statistics)
If you have individual site or **mega**-analysis data

Python | mega

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np

>>> # Store degree centrality and atrophy measures
>>> meas = {('functional cortical hubs', 'cortical thickness'): [fc_ctx_dc, CT_z_
↪mean],
...          ('functional subcortical hubs', 'subcortical volume'): [fc_sctx_dc, SV_z_
↪mean_noVent],
...          ('structural cortical hubs', 'cortical thickness'): [sc_ctx_dc, CT_z_
↪mean],
...          ('structural subcortical hubs', 'subcortical volume'): [sc_sctx_dc, SV_z_
↪mean_noVent]}

>>> fig, axs = plt.subplots(1, 4, figsize=(15, 3))

>>> for k, (fn, dd) in enumerate(meas.items()):
>>>     # Define scatter colors
>>>     if k <= 1:
>>>         col = '#A8221C'
>>>     else:
>>>         col = '#324F7D'

>>>     # Plot relationships between hubs and atrophy
>>>     axs[k].scatter(meas[fn][0], meas[fn][1], color=col,
```

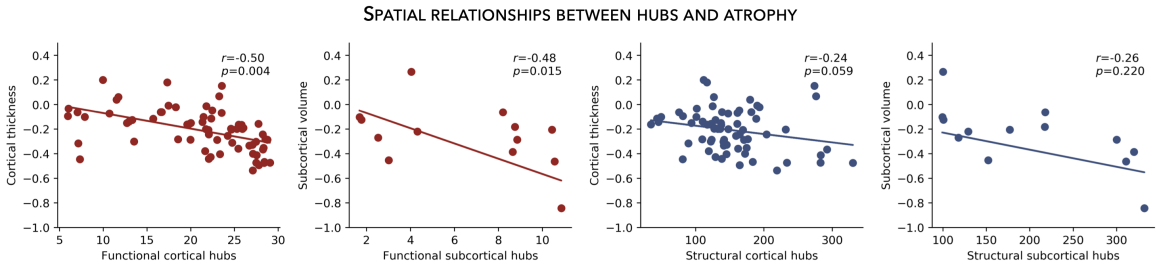
(continues on next page)

(continued from previous page)

```
...                     label='$r$={:.2f}'.format(rvals[fn[0]]) + '\n$p$={:.3f}'.
↪format(p_and_d[fn[0]][0]))
>>>     m, b = np.polyfit(meas[fn][0], meas[fn][1], 1)
>>>     axs[k].plot(meas[fn][0], m * meas[fn][0] + b, color=col)
>>>     axs[k].set_ylim((-3.5, 1.5))
>>>     axs[k].set_xlabel('{}'.format(fn[0].capitalize()))
>>>     axs[k].set_ylabel('{}'.format(fn[1].capitalize()))
>>>     axs[k].spines['top'].set_visible(False)
>>>     axs[k].spines['right'].set_visible(False)
>>>     axs[k].legend(loc=1, frameon=False, markerscale=0)

>>> fig.tight_layout()
>>> plt.show()
```

Matlab | mega

```
% Store degree centrality measures
meas  =  cell2struct({fc_ctx_dc, fc_sctx_dc.', sc_ctx_dc, sc_sctx_dc.'}, ...
                     {'Functional_cortical_hubs', 'Functional_subcortical_hubs', ...
             'Structural_cortical_hubs', 'Structural_subcortical_hubs'}, 2);
fns   = fieldnames(meas);

% Store atrophy measures
meas2 =  cell2struct({CT_z_mean{:, :}, SV_z_mean_noVent{:, :}}, ...
                     {'Cortical_thickness', 'Subcortical_volume'}, 2);
fns2  = fieldnames(meas2);

f = figure,
    set(gcf,'color','w');
    set(gcf,'units','normalized','position',[0 0 1 0.3])
    k2 = [1 2 1 2];

    for k = 1:numel(fieldnames(meas))
        j = k2(k);

        % Define plot colors
        if k <= 2; col = [0.66 0.13 0.11]; else; col = [0.2 0.33 0.49]; end

        % Plot relationships between hubs and atrophy
        axs = subplot(1, 4, k); hold on
        s   = scatter(meas.(fns{k}), meas2.(fns2{j}), 88, col, 'filled');
        P1      = polyfit(meas.(fns{k}), meas2.(fns2{j}), 1);
        yfit_1  = P1(1) * meas.(fns{k}) + P1(2);
        plot(meas.(fns{k}), yfit_1, 'color',col, 'LineWidth', 3)
        ylim([-3 1.5])
        xlabel(strrep(fns{k}, '_', ' '))
        ylabel(strrep(fns2{j}, '_', ' '))
        legend(s, ['{\it r}=' num2str(round(rvals.(lower(fns{k})), 2)) newline ...
                   '{\it p}=' num2str(round(p_and_d.(lower(fns{k}))(1), 3))])
        legend boxoff
    end
```



7.17 Epicenter mapping

This page contains descriptions and examples to identify disease epicenters. For additional details on disease epicenter mapping, please see our manuscript entitled Network-based atrophy modeling in the common epilepsies: a worldwide ENIGMA study.

7.17.1 Cortical epicenters

Using the *HCP connectivity data*, we can also identify epicenters of cortical atrophy. Disease epicenters are regions whose functional and/or structural connectivity profile spatially resembles a given disease-related atrophy map. Hence, disease epicenters can be identified by spatially correlating every region's healthy functional and/or structural connectivity profiles (*i.e.*, cortico-cortical connectivity) to whole-brain atrophy patterns in a given disease. This approach must be repeated systematically across the whole brain, assessing the statistical significance of the spatial similarity of every region's functional and/or structural connectivity profiles to disease-specific abnormality maps with *spatial permutation tests*. Cortical and subcortical (see *tutorial* below) epicenter regions can then be identified if their connectivity profiles are significantly correlated with the disease-specific abnormality map. Regardless of its atrophy level, a cortical (or subcortical) region could potentially be an epicenter if it is (*i*) strongly connected to other high-atrophy regions and (*ii*) weakly connected to low-atrophy regions.

In this tutorial, our *atrophy map* was derived from cortical thickness decreases in individuals with left TLE.

Epicenters?

Cortical and subcortical epicenter regions are identified if their connectivity profiles correlate with a disease-specific *cortical* atrophy map. In the following examples, regions with strong *negative* correlations represent disease epicenters. Moreover, and regardless of its atrophy level, a cortical or subcortical region can be an epicenter if it is (*i*) strongly connected to other high-atrophy cortical regions and (*ii*) weakly connected to low-atrophy cortical regions.

Prerequisites

Load *summary statistics* **or** *example data*
Z-score data (*mega only*)
Load *cortico-cortical connectivity matrices*

Python | meta

```
>>> import numpy as np
>>> from enigmatoolbox.permutation_testing import spin_test

>>> # Identify cortical epicenters (from functional connectivity)
>>> fc_ctx_epi = []
>>> fc_ctx_epi_p = []
>>> for seed in range(fc_ctx.shape[0]):
```

(continues on next page)

(continued from previous page)

```
>>>     seed_con = fc_ctx[:, seed]
>>>     fc_ctx_epi = np.append(fc_ctx_epi, np.corrcoef(seed_con, CT_d)[0, 1])
>>>     fc_ctx_epi_p = np.append(fc_ctx_epi_p,
...                              spin_test(seed_con, CT_d, surface_name='fsa5',␣
↪parcellation_name='aparc',
...                                        type='pearson', n_rot=1000, null_
↪dist=False))

>>> # Identify cortical epicenters (from structural connectivity)
>>> sc_ctx_epi = []
>>> sc_ctx_epi_p = []
>>> for seed in range(sc_ctx.shape[0]):
>>>     seed_con = sc_ctx[:, seed]
>>>     sc_ctx_epi = np.append(sc_ctx_epi, np.corrcoef(seed_con, CT_d)[0, 1])
>>>     sc_ctx_epi_p = np.append(sc_ctx_epi_p,
...                              spin_test(seed_con, CT_d, surface_name='fsa5',␣
↪parcellation_name='aparc',
...                                        type='pearson', n_rot=1000, null_
↪dist=False))
```

Matlab | meta

```
% Identify cortical epicenter values (from functional connectivity)
fc_ctx_epi              = zeros(size(fc_ctx, 1), 1);
fc_ctx_epi_p            = zeros(size(fc_ctx, 1), 1);
for seed = 1:size(fc_ctx, 1)
    seed_conn           = fc_ctx(:, seed);
    r_tmp               = corrcoef(seed_conn, CT_d);
    fc_ctx_epi(seed)    = r_tmp(1, 2);
    fc_ctx_epi_p(seed)  = spin_test(seed_conn, CT_d, 'surface_name', 'fsa5',
↪'parcellation_name', ...
                                    'aparc', 'n_rot', 1000, 'type', 'pearson');
end

% Identify cortical epicenter values (from structural connectivity)
sc_ctx_epi              = zeros(size(sc_ctx, 1), 1);
sc_ctx_epi_p            = zeros(size(sc_ctx, 1), 1);
for seed = 1:size(sc_ctx, 1)
    seed_conn           = sc_ctx(:, seed);
    r_tmp               = corrcoef(seed_conn, CT_d);
    sc_ctx_epi(seed)    = r_tmp(1, 2);
    sc_ctx_epi_p(seed)  = spin_test(seed_conn, CT_d, 'surface_name', 'fsa5',
↪'parcellation_name', ...
                                    'aparc', 'n_rot', 1000, 'type', 'pearson');
end
```

If you have **meta**-analysis data (*e.g.*, summary statistics)
If you have individual site or **mega**-analysis data

Python | mega

```
>>> import numpy as np
>>> from enigmatoolbox.permutation_testing import spin_test

>>> # Identify cortical epicenters (from functional connectivity)
>>> fc_ctx_epi = []
>>> fc_ctx_epi_p = []
>>> for seed in range(fc_ctx.shape[0]):
>>>     seed_con = fc_ctx[:, seed]
>>>     fc_ctx_epi = np.append(fc_ctx_epi, np.corrcoef(seed_con, CT_z_mean)[0, 1])
>>>     fc_ctx_epi_p = np.append(fc_ctx_epi_p,
...                              spin_test(seed_con, CT_z_mean, surface_name='fsa5',␣
↪parcellation_name='aparc',
...                                        type='pearson', n_rot=1000, null_
↪dist=False))

>>> # Identify cortical epicenters (from structural connectivity)
>>> sc_ctx_epi = []
>>> sc_ctx_epi_p = []
>>> for seed in range(sc_ctx.shape[0]):
>>>     seed_con = sc_ctx[:, seed]
>>>     sc_ctx_epi = np.append(sc_ctx_epi, np.corrcoef(seed_con, CT_z_mean)[0, 1])
>>>     sc_ctx_epi_p = np.append(sc_ctx_epi_p,
...                              spin_test(seed_con, CT_z_mean, surface_name='fsa5',␣
↪parcellation_name='aparc',
...                                        type='pearson', n_rot=1000, null_
↪dist=False))
```

Matlab | mega

```
% Identify cortical epicenter values (from functional connectivity)
fc_ctx_epi              = zeros(size(fc_ctx, 1), 1);
fc_ctx_epi_p            = zeros(size(fc_ctx, 1), 1);
for seed = 1:size(fc_ctx, 1)
    seed_conn           = fc_ctx(:, seed);
    r_tmp               = corrcoef(seed_conn, CT_z_mean{:, :});
    fc_ctx_epi(seed)    = r_tmp(1, 2);
    fc_ctx_epi_p(seed)  = spin_test(seed_conn, CT_z_mean{:, :}, 'surface_name', 'fsa5
↪', ...
                                    'parcellation_name', 'aparc', 'n_rot', 1000, 'type
↪', 'pearson');
end

% Identify cortical epicenter values (from structural connectivity)
sc_ctx_epi              = zeros(size(sc_ctx, 1), 1);
sc_ctx_epi_p            = zeros(size(sc_ctx, 1), 1);
for seed = 1:size(sc_ctx, 1)
    seed_conn           = sc_ctx(:, seed);
    r_tmp               = corrcoef(seed_conn, CT_z_mean{:, :});
    sc_ctx_epi(seed)    = r_tmp(1, 2);
    sc_ctx_epi_p(seed)  = spin_test(seed_conn, CT_z_mean{:, :}, 'surface_name', 'fsa5
↪', ...
                                    'parcellation_name', 'aparc', 'n_rot', 1000, 'type
↪', 'pearson');
end
```

As we have assessed the significance of every spatial correlation between seed-based cortico-cortical connectivity and cortical atrophy measures using spin permutation tests, we can set a significance threshold to identify disease epicenters. In the following example, we set a lenient threshold of $p < 0.1$ (*i.e.*, correlation coefficients were set to

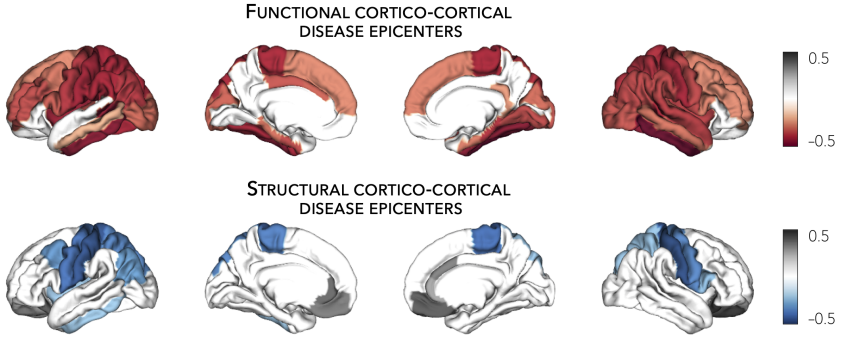
zeros for regions whose p-values were greater than 0.1). We are, thus, displaying only correlation coefficients whose significances passes at least these lenient thresholds.

Python

```
>>> import numpy as np
>>> from enigmatoolbox.utils.parcellation import parcel_to_surface
>>> from enigmatoolbox.plotting import plot_cortical

>>> # Project the results on the surface brain
>>> # Selecting only regions with p < 0.1 (functional epicenters)
>>> fc_ctx_epi_p_sig = np.zeros_like(fc_ctx_epi_p)
>>> fc_ctx_epi_p_sig[np.argwhere(fc_ctx_epi_p < 0.1)] = fc_ctx_epi[np.argwhere(fc_ctx_
↪epi_p < 0.1)]
>>> plot_cortical(array_name=parcel_to_surface(fc_ctx_epi_p_sig, 'aparc_fsa5'),␣
↪surface_name="fsa5", size=(800, 400),
...                cmap='GyRd_r', color_bar=True, color_range=(-0.5, 0.5))

>>> # Selecting only regions with p < 0.1 (structural epicenters)
>>> sc_ctx_epi_p_sig = np.zeros_like(sc_ctx_epi_p)
>>> sc_ctx_epi_p_sig[np.argwhere(sc_ctx_epi_p < 0.1)] = sc_ctx_epi[np.argwhere(sc_ctx_
↪epi_p < 0.1)]
>>> plot_cortical(array_name=parcel_to_surface(sc_ctx_epi_p_sig, 'aparc_fsa5'),␣
↪surface_name="fsa5", size=(800, 400),
...                cmap='GyBu_r', color_bar=True, color_range=(-0.5, 0.5))
```

Matlab

```
% Project the results on the surface brain
% Selecting only regions with p < 0.1 (functional epicenters)
fc_ctx_epi_p_sig = zeros(length(fc_ctx_epi_p), 1);
fc_ctx_epi_p_sig(find(fc_ctx_epi_p < 0.1)) = fc_ctx_epi(fc_ctx_epi_p<0.1);
f = figure,
    plot_cortical(parcel_to_surface(fc_ctx_epi_p_sig, 'aparc_fsa5'), ...
                'color_range', [-0.5 0.5], 'cmap', 'GyRd_r')

% Selecting only regions with p < 0.1 (structural epicenters)
sc_ctx_epi_p_sig = zeros(length(sc_ctx_epi_p), 1);
sc_ctx_epi_p_sig(find(sc_ctx_epi_p < 0.1)) = sc_ctx_epi(sc_ctx_epi_p<0.1);
f = figure,
    plot_cortical(parcel_to_surface(sc_ctx_epi_p_sig, 'aparc_fsa5'), ...
                'color_range', [-0.5 0.5], 'cmap', 'GyBu_r')
```



7.17.2 Subcortical epicenters

Similar to the *cortical epicenter approach*, we can identify subcortical epicenters of cortical atrophy by correlating every subcortical region's seed-based connectivity profile (*e.g.*, subcortico-cortical connectivity) with a whole-brain cortical atrophy map. As above, our *atrophy map* was derived from cortical thickness decreases in individuals with left TLE.

Prerequisites
Load *summary statistics* **or** *example data*
Z-score data (*mega only*)
Load *subcortico-cortical connectivity matrices*

Python | meta

```
>>> import numpy as np
>>> from enigmatoolbox.permutation_testing import spin_test

>>> # Identify subcortical epicenters (from functional connectivity)
>>> fc_sctx_epi = []
>>> fc_sctx_epi_p = []
>>> for seed in range(fc_sctx.shape[0]):
>>>     seed_con = fc_sctx[seed, :]
>>>     fc_sctx_epi = np.append(fc_sctx_epi, np.corrcoef(seed_con, CT_d)[0, 1])
>>>     fc_sctx_epi_p = np.append(fc_sctx_epi_p,
...                               spin_test(seed_con, CT_d, surface_name='fsa5', n_
↪rot=1000))

>>> # Identify subcortical epicenters (from structural connectivity)
>>> sc_sctx_epi = []
>>> sc_sctx_epi_p = []
>>> for seed in range(sc_sctx.shape[0]):
>>>     seed_con = sc_sctx[seed, :]
>>>     sc_sctx_epi = np.append(sc_sctx_epi, np.corrcoef(seed_con, CT_d)[0, 1])
>>>     sc_sctx_epi_p = np.append(sc_sctx_epi_p,
...                               spin_test(seed_con, CT_d, surface_name='fsa5', n_
↪rot=1000))
```

Matlab | meta

```
% Identify subcortical epicenter values (from functional connectivity)
fc_sctx_epi             = zeros(size(fc_sctx, 1), 1);
fc_sctx_epi_p           = zeros(size(fc_sctx, 1), 1);
for seed = 1:size(fc_sctx, 1)
    seed_conn           = fc_sctx(seed, :);
    r_tmp               = corrcoef(seed_conn, CT_d);
    fc_sctx_epi(seed)   = r_tmp(1, 2);
    fc_sctx_epi_p(seed) = spin_test(seed_conn, CT_d, 'surface_name', 'fsa5',
↪'parcellation_name', ...
                                    'aparc', 'n_rot', 1000, 'type', 'pearson');
end

% Identify subcortical epicenter values (from structural connectivity)
sc_sctx_epi             = zeros(size(sc_sctx, 1), 1);
sc_sctx_epi_p           = zeros(size(sc_sctx, 1), 1);
for seed = 1:size(sc_sctx, 1)
    seed_conn           = sc_sctx(seed, :);
    r_tmp               = corrcoef(seed_conn, CT_d);
    sc_sctx_epi(seed)   = r_tmp(1, 2);
```

(continues on next page)

(continued from previous page)

```
    sc_sctx_epi_p(seed) = spin_test(seed_conn, CT_d, 'surface_name', 'fsa5',
↪'parcellation_name', ...
                                    'aparc', 'n_rot', 1000, 'type', 'pearson');
end
```

If you have **meta**-analysis data (*e.g.*, summary statistics)
If you have individual site or **mega**-analysis data

Python | mega

```
>>> import numpy as np
>>> from enigmatoolbox.permutation_testing import spin_test

>>> # Identify subcortical epicenters (from functional connectivity)
>>> fc_sctx_epi = []
>>> fc_sctx_epi_p = []
>>> for seed in range(fc_sctx.shape[0]):
>>>     seed_con = fc_sctx[seed, :]
>>>     fc_sctx_epi = np.append(fc_sctx_epi, np.corrcoef(seed_con, CT_z_mean)[0, 1])
>>>     fc_sctx_epi_p = np.append(fc_sctx_epi_p,
...                               spin_test(seed_con, CT_z_mean, surface_name='fsa5',␣
↪n_rot=1000))

>>> # Identify subcortical epicenters (from structural connectivity)
>>> sc_sctx_epi = []
>>> sc_sctx_epi_p = []
>>> for seed in range(sc_sctx.shape[0]):
>>>     seed_con = sc_sctx[seed, :]
>>>     sc_sctx_epi = np.append(sc_sctx_epi, np.corrcoef(seed_con, CT_z_mean)[0, 1])
>>>     sc_sctx_epi_p = np.append(sc_sctx_epi_p,
...                               spin_test(seed_con, CT_z_mean, surface_name='fsa5',␣
↪n_rot=1000))
```

Matlab | mega

```
% Identify subcortical epicenter values (from functional connectivity)
fc_sctx_epi             = zeros(size(fc_sctx, 1), 1);
fc_sctx_epi_p           = zeros(size(fc_sctx, 1), 1);
for seed = 1:size(fc_sctx, 1)
    seed_conn           = fc_sctx(seed, :);
    r_tmp               = corrcoef(seed_conn, CT_z_mean{:, :});
    fc_sctx_epi(seed)   = r_tmp(1, 2);
    fc_sctx_epi_p(seed) = spin_test(seed_conn, CT_z_mean{:, :}, 'surface_name', 'fsa5
↪', ...
                                    'parcellation_name', 'aparc', 'n_rot', 1000, 'type
↪', 'pearson');
end

% Identify subcortical epicenter values (from structural connectivity)
sc_sctx_epi             = zeros(size(sc_sctx, 1), 1);
sc_sctx_epi_p           = zeros(size(sc_sctx, 1), 1);
for seed = 1:size(sc_sctx, 1)
```

(continues on next page)

(continued from previous page)

```
    seed_conn           = sc_sctx(seed, :);
    r_tmp               = corrcoef(seed_conn, CT_z_mean{:, :});
    sc_sctx_epi(seed)   = r_tmp(1, 2);
    sc_sctx_epi_p(seed) = spin_test(seed_conn, CT_z_mean{:, :}, 'surface_name', 'fsa5
↪', ...
                                    'parcellation_name', 'aparc', 'n_rot', 1000, 'type
↪', 'pearson');
end
```

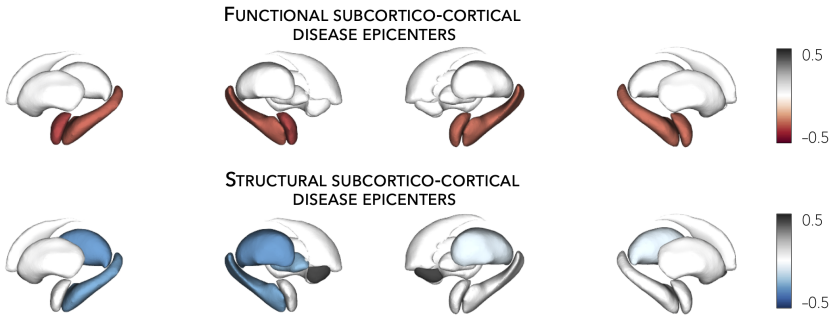
As in the *cortical epicenters* example above, we have assessed the significance of every spatial correlation between seed-based subcortico-cortical connectivity and cortical atrophy measures using spin permutation tests, and set a lenient threshold of $p < 0.1$ (*i.e.*, correlation coefficients were set to zeros for regions whose p-values were greater than 0.1). We are, thus, displaying only correlation coefficients whose significances passes at least these lenient thresholds.

Python

```
>>> import numpy as np
>>> from enigmatoolbox.plotting import plot_subcortical

>>> # Project the results on the surface brain
>>> # Selecting only regions with p < 0.1 (functional epicenters)
>>> fc_sctx_epi_p_sig = np.zeros_like(fc_sctx_epi_p)
>>> fc_sctx_epi_p_sig[np.argwhere(fc_sctx_epi_p < 0.1)] = fc_sctx_epi[np.argwhere(fc_
↪sctx_epi_p < 0.1)]
>>> plot_subcortical(fc_sctx_epi_p_sig, ventricles=False, size=(800, 400),
...                  cmap='GyRd_r', color_bar=True, color_range=(-0.5, 0.5))

>>> # Selecting only regions with p < 0.1 (functional epicenters)
>>> sc_sctx_epi_p_sig = np.zeros_like(sc_sctx_epi_p)
>>> sc_sctx_epi_p_sig[np.argwhere(sc_sctx_epi_p < 0.1)] = sc_sctx_epi[np.argwhere(sc_
↪sctx_epi_p < 0.1)]
>>> plot_subcortical(sc_sctx_epi_p_sig, ventricles=False, size=(800, 400),
...                  cmap='GyBu_r', color_bar=True, color_range=(-0.5, 0.5))
```

Matlab

```
% Project the results on the surface brain
% Selecting only regions with p < 0.1 (functional epicenters)
fc_sctx_epi_p_sig = zeros(length(fc_sctx_epi_p), 1);
fc_sctx_epi_p_sig(find(fc_sctx_epi_p < 0.1)) = fc_sctx_epi(fc_sctx_epi_p<0.1);
f = figure,
    plot_subcortical(fc_sctx_epi_p_sig, 'ventricles', 'False', ...
                     'color_range', [-0.5 0.5], 'cmap', 'GyRd_r')

% Selecting only regions with p < 0.1 (structural epicenters)
sc_sctx_epi_p_sig = zeros(length(sc_sctx_epi_p), 1);
sc_sctx_epi_p_sig(find(sc_sctx_epi_p < 0.1)) = sc_sctx_epi(sc_sctx_epi_p<0.1);
f = figure,
    plot_subcortical(sc_sctx_epi_p_sig, 'ventricles', 'False', ...
                     'color_range', [-0.5 0.5], 'cmap', 'GyBu_r')
```



7.18 Spin permutation tests

This page contains descriptions and examples to assess statistical significance of two spatial maps.

7.18.1 Assess statistical significance

The intrinsic spatial smoothness in two given **cortical maps** may inflate the significance of their spatial correlation. To overcome this challenge, we assess statistical significance using *spin permutation tests*, a framework proposed by Alexander-Bloch and colleagues. To do so, we generate null models of overlap between cortical maps by projecting the spatial coordinates of cortical data onto the surface spheres, apply randomly sampled rotations, and reassign cortical values. We then compare the original correlation coefficients against the empirical distribution determined by the ensemble of spatially permuted correlation coefficients.

To compare spatial overlap between **subcortical maps**, we employed a similar approach with the exception that subcortical labels were randomly shuffled as opposed to being projected onto spheres.

Prerequisites

```
Two brain maps from which you want to assess the signifcance of their␣
↪correlations, as for example:
degree centrality vs. atrophy
      Load summary statistics or example data
      Re-order subcortical data (mega only)
      Z-score data (mega only)
      Load cortico-cortical and subcortico-cortical connectivity matrices
      Compute cortical-cortical and subcortico-cortical degree centrality
```

Python | meta

```
>>> from enigmatoolbox.permutation_testing import spin_test, shuf_test

>>> # Remove subcortical values corresponding to the ventricles
>>> # (as we don't have connectivity values for them!)
>>> SV_d_noVent = SV_d.drop([np.where(SV['Structure'] == 'LLatVent')[0][0],
...                          np.where(SV['Structure'] == 'RLatVent')[0][0]]).reset_
↪index(drop=True)

>>> # Spin permutation testing for two cortical maps
>>> fc_ctx_p, fc_ctx_d = spin_test(fc_ctx_dc, CT_d, surface_name='fsa5', parcellation_
↪name='aparc',
```

(continues on next page)

(continued from previous page)

```
...                                  type='pearson', n_rot=1000, null_dist=True)
>>> sc_ctx_p, sc_ctx_d = spin_test(sc_ctx_dc, CT_d, surface_name='fsa5', parcellation_
↪name='aparc',
...                                  type='pearson', n_rot=1000, null_dist=True)

>>> # Shuf permutation testing for two subcortical maps
>>> fc_sctx_p, fc_sctx_d = shuf_test(fc_sctx_dc, SV_d_noVent, n_rot=1000,
...                                    type='pearson', null_dist=True)
>>> sc_sctx_p, sc_sctx_d = shuf_test(sc_sctx_dc, SV_d_noVent, n_rot=1000,
...                                    type='pearson', null_dist=True)

>>> # Store p-values and null distributions
>>> p_and_d = {'functional cortical hubs': [fc_ctx_p, fc_ctx_d], 'functional␣
↪subcortical hubs': [fc_sctx_p, fc_sctx_d],
...            'structural cortical hubs': [sc_ctx_p, sc_ctx_d], 'structural␣
↪subcortical hubs': [sc_sctx_p, sc_sctx_d]}
```

Matlab | meta

```
% Remove subcortical values corresponding to the ventricles
% (as we don't have connectivity values for them!)
SV_d_noVent = SV_d;
SV_d_noVent([find(strcmp(SV.Structure, 'LLatVent')); ...
             find(strcmp(SV.Structure, 'RLatVent'))], :) = [];

% Spin permutation testing for two cortical maps
[fc_ctx_p, fc_ctx_d]   = spin_test(fc_ctx_dc, CT_d, 'surface_name', 'fsa5', ...
                                   'parcellation_name', 'aparc', 'n_rot', 1000, ...
                                   'type', 'pearson');
[sc_ctx_p, sc_ctx_d]   = spin_test(sc_ctx_dc, CT_d, 'surface_name', 'fsa5', ...
                                   'parcellation_name', 'aparc', 'n_rot', 1000, ...
                                   'type', 'pearson');

% Shuf permutation testing for two subcortical maps
[fc_sctx_p, fc_sctx_d] = shuf_test(fc_sctx_dc, SV_d_noVent, ...
                                   'n_rot', 1000, 'type', 'pearson');
[sc_sctx_p, sc_sctx_d] = shuf_test(sc_sctx_dc, SV_d_noVent, ...
                                   'n_rot', 1000, 'type', 'pearson');

% Store p-values and null distributions
p_and_d =  cell2struct({[fc_ctx_p; fc_ctx_d], [fc_sctx_p; fc_sctx_d], [sc_ctx_p; sc_
↪ctx_d], [sc_sctx_p; sc_sctx_d]}, ...
                       {'functional_cortical_hubs', 'functional_subcortical_hubs', ...
                        'structural_cortical_hubs', 'structural_subcortical_hubs'},␣
↪2);
```

If you have **meta**-analysis data (*e.g.*, summary statistics)
If you have individual site or **mega**-analysis data

Python | mega

```
>>> from enigmatoolbox.permutation_testing import spin_test, shuf_test

>>> # Remove subcortical values corresponding to the ventricles
>>> # (as we don't have connectivity values for them!)
>>> SV_z_mean_noVent = SV_z_mean.drop(['LLatVent', 'RLatVent']).reset_index(drop=True)

>>> # Spin permutation testing for two cortical maps
>>> fc_ctx_p, fc_ctx_d = spin_test(fc_ctx_dc, CT_z_mean, surface_name='fsa5',␣
↪parcellation_name='aparc',
...                                type='pearson', n_rot=1000, null_dist=True)
>>> sc_ctx_p, sc_ctx_d = spin_test(sc_ctx_dc, CT_z_mean, surface_name='fsa5',␣
↪parcellation_name='aparc',
...                                type='pearson', n_rot=1000, null_dist=True)

>>> # Shuf permutation testing for two subcortical maps
>>> fc_sctx_p, fc_sctx_d = shuf_test(fc_sctx_dc, SV_z_mean_noVent, n_rot=1000,
...                                  type='pearson', null_dist=True)
>>> sc_sctx_p, sc_sctx_d = shuf_test(sc_sctx_dc, SV_z_mean_noVent, n_rot=1000,
...                                  type='pearson', null_dist=True)

>>> # Store p-values and null distributions
>>> p_and_d = {'functional cortical hubs': [fc_ctx_p, fc_ctx_d], 'functional␣
↪subcortical hubs': [fc_sctx_p, fc_sctx_d],
...            'structural cortical hubs': [sc_ctx_p, sc_ctx_d], 'structural␣
↪subcortical hubs': [sc_sctx_p, sc_sctx_d]}
```

Matlab | mega

```
% Remove subcortical values corresponding to the ventricles
% (as we don't have connectivity values for them!)
SV_z_mean_noVent = SV_z_mean;
SV_z_mean_noVent.LLatVent = [];
SV_z_mean_noVent.RLatVent = [];

% Spin permutation testing for two cortical maps
[fc_ctx_p, fc_ctx_d]   = spin_test(fc_ctx_dc, CT_z_mean{:, :}, 'surface_name', ...
                                   'fsa5', 'parcellation_name', 'aparc', 'n_rot', ...
                                   1000, 'type', 'pearson');
[sc_ctx_p, sc_ctx_d]   = spin_test(sc_ctx_dc, CT_z_mean{:, :}, 'surface_name', ...
                                   'fsa5', 'parcellation_name', 'aparc', 'n_rot', ...
                                   1000, 'type', 'pearson');

% Shuf permutation testing for two subcortical maps
[fc_sctx_p, fc_sctx_d] = shuf_test(fc_sctx_dc, SV_z_mean_noVent{:, :}, ...
                                   'n_rot', 1000, 'type', 'pearson');
[sc_sctx_p, sc_sctx_d] = shuf_test(sc_sctx_dc, SV_z_mean_noVent{:, :}, ...
                                   'n_rot', 1000, 'type', 'pearson');

% Store p-values and null distributions
p_and_d =  cell2struct({[fc_ctx_p; fc_ctx_d], [fc_sctx_p; fc_sctx_d], [sc_ctx_p; sc_
↪ctx_d], [sc_sctx_p; sc_sctx_d]}, ...
                       {'functional_cortical_hubs', 'functional_subcortical_hubs', ...
                        'structural_cortical_hubs', 'structural_subcortical_hubs'},␣
↪2);
```

7.18.2 Plot null distributions

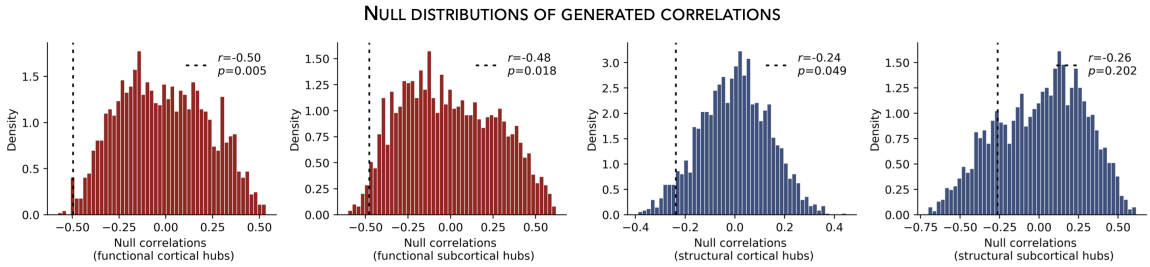
To better interpret statistical significance, we can plot the null distribution of generated correlations (*i.e.*, "spun" or "shuffled" correlations) and overlay the correlation coefficient obtained from the empirical (*i.e.*, real) brain maps.

Python

```
>>> import matplotlib.pyplot as plt

>>> fig, axs = plt.subplots(1, 4, figsize=(15, 3))

>>> for k, (fn, dd) in enumerate(p_and_d.items()):
>>>     # Define plot colors
>>>     if k <= 1:
>>>         col = '#A8221C'      # red for functional hubs
>>>     else:
>>>         col = '#324F7D'      # blue for structural hubs

>>>     # Plot null distributions
>>>     axs[k].hist(dd[1], bins=50, density=True, color=col, edgecolor='white', lw=0.
↪5)
>>>     axs[k].axvline(rvals[fn], lw=1.5, ls='--', color='k', dashes=(2, 3),
...                    label='$r$={:.2f}'.format(rvals[fn]) + '\n$p$={:.3f}'.
↪format(dd[0]))
>>>     axs[k].set_xlabel('Null correlations \n ({})'.format(fn))
>>>     axs[k].set_ylabel('Density')
>>>     axs[k].spines['top'].set_visible(False)
>>>     axs[k].spines['right'].set_visible(False)
>>>     axs[k].legend(loc=1, frameon=False)

>>> fig.tight_layout()
>>> plt.show()
```

Matlab

```
f = figure,
    set(gcf,'color','w');
    set(gcf,'units','normalized','position',[0 0 1 0.3])
    fns = fieldnames(p_and_d);

    for k = 1:numel(fieldnames(rvals))
        % Define plot colors
        if k <= 2; col = [0.66 0.13 0.11]; else; col = [0.2 0.33 0.49]; end

        % Plot null distributions
        axs = subplot(1, 4, k); hold on
        h = histogram(p_and_d.(fns{k})(2:end), 50, 'Normalization', 'pdf', 'edgecolor
↪', 'w', ...
                      'facecolor', col, 'facealpha', 1, 'linewidth', 0.5);
        l = line([rvals.(fns{k}) rvals.(fns{k})], get(gca, 'ylim'), 'linestyle', '--',
↪ ...
                 'color', 'k', 'linewidth', 1.5);
        xlabel(['Null correlations' newline '(' strrep(fns{k}, '_', ' ') ')'])
        ylabel('Density')
        legend(l,['{\it r}=' num2str(round(rvals.(fns{k}), 2)) newline ...
                  '{\it p}=' num2str(round(p_and_d.(fns{k})(1), 3))])
        legend boxoff
    end
```



7.19 Citing the ENIGMA TOOLBOX

Please cite the following when referring to your use of our Toolbox: Larivière, S., Paquola, C., Park, By. et al. The ENIGMA Toolbox: multiscale neural contextualization of multisite neuroimaging datasets. *Nat Methods* **18**, 698–700 (2021). https://doi.org/10.1038/s41592-021-01186-4

7.20 Python API

This is the function reference of the **ENIGMA TOOLBOX**

7.20.1 `enigmatoolbox.datasets`

ENIGMA datasets

enigmatoolbox.datasets.load_example_data()	Loads the ENIGMA example dataset (from one site - MICA-MNI Montreal; author: @saratheriver)
enigmatoolbox.datasets.load_summary_stats([...])	Outputs summary statistics for a given disorder (author: @saratheriver)

enigmatoolbox.datasets.load_example_data

enigmatoolbox.datasets.**load_example_data**()

Loads the ENIGMA example dataset (from one site - MICA-MNI Montreal; author: @saratheriver)

Returns

- **cov** (*pandas.DataFrame*) – Contains information on covariates
- **metr1** (*pandas.DataFrame*) – Contains information on subcortical volume
- **metr2** (*pandas.DataFrame*) – Contains information on cortical thickness
- **metr3** (*pandas.DataFrame*) – Contains information on surface area

enigmatoolbox.datasets.load_summary_stats

enigmatoolbox.datasets.**load_summary_stats**(*disorder=None*)
Outputs summary statistics for a given disorder (author: @saratheriver)

Parameters **disorder** (*{'22q', 'adhd', 'asd', 'bipolar', 'depression', 'epilepsy', 'ocd', 'schizophrenia'}*) – Disorder name, default is None

Returns **summary_stats** – Available summary statistics

Return type pandas.DataFrame

Export functions

enigmatoolbox.datasets.nfaces(surface_name, ...)	Returns number of faces/triangles for a surface (author: @saratheriver)
enigmatoolbox.datasets.getaffine(...)	Returns vox2ras transform for a surface (author: @saratheriver)
enigmatoolbox.datasets.write_cifti(data[, ...])	Writes cifti file (authors: @NicoleEic, @saratheriver)

enigmatoolbox.datasets.nfaces

enigmatoolbox.datasets.**nfaces**(*surface_name*, *hemisphere*)
Returns number of faces/triangles for a surface (author: @saratheriver)

Parameters

- **surface_name** (*string*) – Name of surface {'fsa5', 'conte69'}
- **hemisphere** (*string*) – Name of hemisphere {'lh', 'rh', 'both'}

Returns **numfaces** – number of faces/triangles

Return type int

enigmatoolbox.datasets.getaffine

enigmatoolbox.datasets.**getaffine**(*surface_name*, *hemisphere*)
Returns vox2ras transform for a surface (author: @saratheriver)

Parameters

- **surface_name** (*string*) – Name of surface {'fsa5', 'conte69'}
- **hemisphere** (*string*) – Name of hemisphere {'lh', 'rh', 'both'}

Returns **numfaces** – vox2ras transform, shape = (4, 4)

Return type 2D ndarray

enigmatoolbox.datasets.write_cifti

enigmatoolbox.datasets.**write_cifti**(*data*, *dpath=None*, *fname=None*, *labels=None*, *surface_name='conte69'*, *hemi='lh'*)

Writes cifti file (authors: @NicoleEic, @saratheriver)

Parameters

- **dpath** (*string*) – Path to location for saving file (e.g., '/Users/bonjour/')
- **fname** (*string*) – Name of file (e.g., 'ello.dscalar.nii') Default is None
- **labels** (*list*) – List of region labels Default is None
- **surface_name** (*string*) – Name of surface {'fsa5', 'conte69'} Default is 'conte69'
- **hemi** (*string*) – Name of hemisphere {'lh', 'rh'} Default is 'lh'

Connectivity matrices

enigmatoolbox.datasets.load_fc([parcellation])	Load functional connectivity data (author: @saratheriver)
enigmatoolbox.datasets.load_fc_as_one([...])	Load functional connectivity data (cortical + subcortical in one matrix; author: @saratheriver)
enigmatoolbox.datasets.load_sc([parcellation])	Load structural connectivity data (author: @saratheriver)
enigmatoolbox.datasets.load_sc_as_one([...])	Load structural connectivity data (cortical + subcortical in one matrix; author: @saratheriver)

enigmatoolbox.datasets.load_fc

enigmatoolbox.datasets.**load_fc**(*parcellation='aparc'*)

Load functional connectivity data (author: @saratheriver)

Parameters **parcellation** (*str, optional*) – Parcellation name {'aparc', 'schaefer_100', 'schaefer_200', 'schaefer_300', 'schaefer_400', 'glasser_360'}. Default is 'aparc' with n cortical regions.

Returns

- **funcMatrix_ctx** (*2D ndarray*) – Cortico-cortical connectivity, shape = (n, n)
- **funcLabels_ctx** (*1D ndarray*) – Cortical labels, shape = (n,)
- **funcMatrix_sctx** (*2D ndarray*) – Subcortico-cortical connectivity, shape = (14, n)
- **funcLabels_sctx** (*1D ndarray*) – Subcortical labels, shape = (14,)

enigmatoolbox.datasets.load_fc_as_one

enigmatoolbox.datasets.**load_fc_as_one**(*parcellation='aparc'*)
Load functional connectivity data (cortical + subcortical in one matrix; author: @saratheriver)

Parameters **parcellation** (*str, optional*) – Parcellation name {'aparc', 'schaefer_100', 'schaefer_200', 'schaefer_300', 'schaefer_400', 'glasser'}. Default is 'aparc' with n cortical regions.

Returns

- **funcMatrix_ctx** (*2D ndarray*) – Functional connectivity, shape = (n+14, n+14)
- **funcLabels_ctx** (*1D ndarray*) – Region labels, shape = (n+14,)

enigmatoolbox.datasets.load_sc

enigmatoolbox.datasets.**load_sc**(*parcellation='aparc'*)
Load structural connectivity data (author: @saratheriver)

Parameters **parcellation** (*str, optional*) – Parcellation name {'aparc', 'schaefer_100', 'schaefer_200', 'schaefer_300', 'schaefer_400', 'glasser_360'}. Default is 'aparc' with n cortical regions.

Returns

- **strucMatrix_ctx** (*2D ndarray*) – Cortico-cortical connectivity, shape = (n, n)
- **strucLabels_ctx** (*1D ndarray*) – Cortical labels, shape = (n,)
- **strucMatrix_sctx** (*2D ndarray*) – Subcortico-cortical connectivity, shape = (14, n)
- **strucLabels_sctx** (*1D ndarray*) – Subcortical labels, shape = (14,)

enigmatoolbox.datasets.load_sc_as_one

enigmatoolbox.datasets.**load_sc_as_one**(*parcellation='aparc'*)
Load structural connectivity data (cortical + subcortical in one matrix; author: @saratheriver)

Parameters **parcellation** (*str, optional*) – Parcellation name {'aparc', 'schaefer_100', 'schaefer_200', 'schaefer_300', 'schaefer_400', 'glasser'}. Default is 'aparc' with n cortical regions.

Returns

- **strucMatrix_ctx** (*2D ndarray*) – Structural connectivity, shape = (n+14, n+14)
- **strucLabels_ctx** (*1D ndarray*) – Region labels, shape = (n+14,)

Gene co-expression data

enigmatoolbox.datasets.fetch_ahba([csvfile])	Fetch Allen Human Brain Atlas microarray expression data from all donors and all genes (author: @saratheriver)
enigmatoolbox.datasets.risk_genes([disorder])	Outputs names of GWAS-derived risk genes for a given disorder (author: @saratheriver)

enigmatoolbox.datasets.fetch_ahba

enigmatoolbox.datasets.**fetch_ahba**(*csvfile=None*)

Fetch Allen Human Brain Atlas microarray expression data from all donors and all genes (author: @saratheriver)

Parameters **csvfile** (*None or string, optional*) – Path to downloaded csvfile. For more threshold and parcellation options, download csvfile from here: https://github.com/saratheriver/enigma-extra/tree/master/ahba If None (default), fetches microarray expression data from the internet (aparc and stable r > 0.2).

Returns **genes** – Table of gene co-expression data, shape = (82, 15633)

Return type pandas.DataFrame

enigmatoolbox.datasets.risk_genes

enigmatoolbox.datasets.**risk_genes**(*disorder=None*)

Outputs names of GWAS-derived risk genes for a given disorder (author: @saratheriver)

Parameters **disorder** (*{'adhd', 'asd', 'bipolar', 'depression', 'epilepsy', 'hippocampal_volume', 'ocd', 'schizophrenia', 'tourette'}*) – Name of disorder, default is None

Returns **risk_genes** – Names of genes for a given disorder

Return type set

Surface templates

enigmatoolbox.datasets.load_fsa5([...])	Load fsaverage5 surfaces (author: @saratheriver)
enigmatoolbox.datasets.load_conte69([...])	Load conte69 surfaces (author: @OualidBenkarim)
enigmatoolbox.datasets.load_subcortical([...])	Load subcortical surfaces (author: @saratheriver)

enigmatoolbox.datasets.load_fsa5

enigmatoolbox.datasets.**load_fsa5**(*as_sphere=False*, *with_normals=True*, *join=False*, *with_sctx=False*)

Load fsaverage5 surfaces (author: @saratheriver)

Parameters

- **as_sphere** (*bool, optional*) – Return spheres instead of cortical surfaces. Default is False.
- **with_normals** (*bool, optional*) – Whether to compute surface normals. Default is True.
- **join** (*bool, optional*) – If False, return one surface for left and right hemispheres. Otherwise, return a single surface as a combination of both left and right surfaces. Default is False.
- **with_sctx** (*bool, optional*) – If False, returns cortical surfaces/spheres only, if True, returns combined cortical+subcortical surfaces. Default is False

Returns **surf** – Surfaces for left and right hemispheres. If `join == True`, one surface with both hemispheres.

Return type tuple of BSPolyData or BSPolyData

enigmatoolbox.datasets.load_conte69

enigmatoolbox.datasets.**load_conte69**(*as_sphere=False*, *with_normals=True*, *join=False*)

Load conte69 surfaces (author: @OualidBenkarim)

Parameters

- **as_sphere** (*bool, optional*) – Return spheres instead of cortical surfaces. Default is False.
- **with_normals** (*bool, optional*) – Whether to compute surface normals. Default is True.
- **join** (*bool, optional*) – If False, return one surface for left and right hemispheres. Otherwise, return a single surface as a combination of both left and right surfaces. Default is False.

Returns **surf** – Surfaces for left and right hemispheres. If `join == True`, one surface with both hemispheres.

Return type tuple of BSPolyData or BSPolyData

enigmatoolbox.datasets.load_subcortical

enigmatoolbox.datasets.**load_subcortical**(*with_normals=False*, *join=False*)

Load subcortical surfaces (author: @saratheriver)

Parameters

- **with_normals** (*bool, optional*) – Whether to compute surface normals. Default is False.

- **join** (*bool, optional*) – If False, return one surface for left and right hemispheres. Otherwise, return a single surface as a combination of both left and right surfaces. Default is False

Returns **surf** – Surfaces for left and right hemispheres. If `join == True`, one surface with both hemispheres.

Return type tuple of BSPolyData or BSPolyData

7.20.2 enigmatoolbox.cross_disorder

Cross-disorder effect

enigmatoolbox.cross_disorder.cross_disorder_effect([...])	Cross-disorder effect (authors: @boyongpark, @saratheriver)

enigmatoolbox.cross_disorder.cross_disorder_effect

enigmatoolbox.cross_disorder.**cross_disorder_effect**(*disorder='all_disorder'*, *measure=None*, *additional_data_cortex=None*, *additional_name_cortex=None*, *additional_data_subcortex=None*, *additional_name_subcortex=None*, *ignore=None*, *include=None*, *method='pca'*)

Cross-disorder effect (authors: @boyongpark, @saratheriver)

Parameters

- **disorder** (*list, optional*) – Any combination of disorder name. Default is all available disorders, except 'adhd'. Options are: {'22q', 'adhd', 'asd', 'bipolar', 'depression', 'epilepsy', 'ocd', 'schizophrenia'}.
- **measure** (*list, optional*) – Any combination of measure names. Default is {'CortThick', 'CortSurf', 'SubVol'}.
- **additional_data_cortex** (*ndarray, optional*) – Name for additional cortical ENIGMA-type data. Must also provide 'additional_name_cortex'.
- **additional_name_cortex** (*list, optional*) – Additional cortical ENIGMA-type data (n, 68). Must also provide 'additional_name_cortex'.
- **additional_data_subcortex** (*ndarray, optional*) – Name for additional subcortical ENIGMA-type data. Must also provide 'additional_name_subcortex'.
- **additional_name_subcortex** (*list, optional*) – Additional subcortical ENIGMA-type data (n, 16). Must also provide 'additional_name_subcortex'.
- **ignore** (*list, optional*) – Ignore summary statistics with these expressions. Default is ('mega') as it contains NaNs.
- **include** (*list, optional*) – Include only summary statistics with these expressions. Default is empty, i.e., include everything.
- **method** (*string, optional*) – Analysis method {'pca', 'correlation'}. Default is 'pca'.

Returns

- **components** (*dict*) – Principal components of shared effects in descending order in terms of component variance. Only is method is 'pca'.
- **variance** (*dict*) – Variance of components. Only is method is 'pca'.
- **correlation_matrix** (*dict*) – Correlation matrices of for every pair of shared effect maps. Only is method is 'correlation'.
- **names** (*dict*) – Names of disorder and case-control effect maps included in analysis.

7.20.3 enigmatoolbox.mesh

Read/write functionality

enigmatoolbox.mesh.mesh_io.read_surface(ipth)	Read surface data (author: @OualidBenkarim)
enigmatoolbox.mesh.mesh_io.write_surface(...)	Write surface data (author: @OualidBenkarim)

enigmatoolbox.mesh.mesh_io.read_surface

enigmatoolbox.mesh.mesh_io.**read_surface**(*ipth*, *itype=None*, *return_data=True*, *update=True*)

Read surface data (author: @OualidBenkarim)

See *itype* for supported file types.

Parameters

- **ipth** (*str*) – Input filename.
- **itype** (*{'ply', 'vtp', 'vtk', 'fs', 'asc', 'gii'}, optional*) – Input file type. If None, it is deduced from *ipth*. Default is None.
- **return_data** (*bool, optional*) – Whether to return data instead of filter. Default is False
- **update** (*bool, optional*) – Whether to update filter When return_data=True, filter is automatically updated. Default is True.

Returns **output** – Surface as a filter or BSPolyData.

Return type BSAlgorithm or BSPolyData

Notes

Function can read FreeSurfer geometry data in binary ('fs') and ascii ('asc') format. Gifti surfaces can also be loaded if nibabel is installed.

See also:

write_surface()

enigmatoolbox.mesh.mesh_io.write_surface

enigmatoolbox.mesh.mesh_io.**write_surface**(*ifilter*, *opth*, *oformat=None*, *otype=None*)

Write surface data (author: @OualidBenkarim)

See *otype* for supported file types.

Parameters

- **ifilter** (*BSAlgorithm or BSDataObject*) – Input filter or data.
- **opth** (*str*) – Output filename.
- **oformat** (*{'ascii', 'binary'}, optional*) – File format. Defaults to writer's default format. Only used when writer accepts format. Default is None.
- **otype** (*{'ply', 'vtp', 'vtk', 'fs', 'asc', 'gii'}, optional*) – File type. If None, type is deduced from *opth*. Default is None.

Notes

Function can save data in FreeSurfer binary ('fs') and ascii ('asc') format. Gifti surfaces can also be saved if nibabel is installed.

See also:

read_surface()

7.20.4 enigmatoolbox.permutation_testing

Spin permutations

enigmatoolbox.permutation_testing.spin_test(...)	Spin permutation (author: @saratheriver)
enigmatoolbox.permutation_testing.centroid_extraction_sphere(...)	Extract centroids of a cortical parcellation on a surface sphere (author: @saratheriver)
enigmatoolbox.permutation_testing.rotate_parcellation(...)	Rotate parcellation (author: @saratheriver)
enigmatoolbox.permutation_testing.perm_sphere_p(x, ...)	Generate a p-value for the spatial correlation between two parcellated cortical surface maps (author: @saratheriver)

enigmatoolbox.permutation_testing.spin_test

enigmatoolbox.permutation_testing.**spin_test**(*map1*, *map2*, *surface_name='fsa5'*, *parcellation_name='aparc'*, *n_rot=1000*, *type='pearson'*, *null_dist=False*, *ventricles=False*)

Spin permutation (author: @saratheriver)

Parameters

- **map1** (*narray, ndarray, or pandas.Series*) – One of two map to be correlated

- **map2** (*narray, ndarray, or pandas.Series*) – The other map to be correlated
- **surface_name** (*string, optional*) – Surface name {'fsa5', 'fsa5_with_sctx'}. Use 'fsa5' for Conte69. Default is 'fsa5'.
- **parcellation_name** (*string, optional*) – Parcellation name {'aparc', 'aparc_aseg'}. Default is 'aparc'.
- **n_rot** (*int, optional*) – Number of spin rotations. Default is 1000.
- **type** (*string, optional*) – Correlation type {'pearson', 'spearman'}. Default is 'pearson'.
- **null_dist** (*bool, optional*) – Output null correlations. Default is False.
- **ventricles** (*bool, optional*) – Whether ventricles are present in map1, map2. Only used when `parcellation_name is 'aparc_aseg'`. Default is False.

Returns

- **p_spin** (*float*) – Permutation p-value
- **r_dist** (*1D ndarray*) – Null correlations, shape = (n_rot*2,). Only if `null_dist is True`.

See also:

centroid_extraction_sphere(), *rotate_parcellation()*, *perm_sphere_p()*

References

- Alexander-Bloch A, Shou H, Liu S, Satterthwaite TD, Glahn DC, Shinohara RT, Vandekar SN and Raznahan A (2018). On testing for spatial correspondence between maps of human brain structure and function. NeuroImage, 178:540-51.
- Vasa F, Seidlitz J, Romero-Garcia R, Whitaker KJ, Rosenthal G, Vertes PE, Shinn M, Alexander-Bloch A, Fonagy P, Dolan RJ, Goodyer IM, the NSPN consortium, Sporns O, Bullmore ET (2017). Adolescent tuning of association cortex in human structural brain networks. Cerebral Cortex, 28(1):281–294.

enigmatoolbox.permutation_testing.centroid_extraction_sphere

enigmatoolbox.permutation_testing.**centroid_extraction_sphere**(*sphere_coords*, *annotfile*, *ventricles=False*)

Extract centroids of a cortical parcellation on a surface sphere (author: @saratheriver)

Parameters

- **sphere_coords** (*ndarray*) – Sphere coordinates, shape = (n, 3)
- **annotfile** (*string*) – Name of annotation file {'fsa5_lh_aparc.annot', 'fsa5_rh_aparc.annot, 'fsa5_with_sctx_lh_aparc_aseg.csv', etc.}
- **ventricles** (*bool, optional*) – Whether ventricle data are present. Only used when 'annotfile' is fsa5_with_sctx_lh_aparc_aseg or fsa5_with_sctx_lh_aparc_aseg``. Default is False.

Returns **coord** – Coordinates of the centroid of each region on the sphere, shape = (m, 3).

Return type ndarray

See also:

spin_test()

References

- Alexander-Bloch A, Shou H, Liu S, Satterthwaite TD, Glahn DC, Shinohara RT, Vandekar SN and Raznahan A (2018). On testing for spatial correspondence between maps of human brain structure and function. NeuroImage, 178:540-51.
- Vasa F, Seidlitz J, Romero-Garcia R, Whitaker KJ, Rosenthal G, Vertes PE, Shinn M, Alexander-Bloch A, Fonagy P, Dolan RJ, Goodyer IM, the NSPN consortium, Sporns O, Bullmore ET (2017). Adolescent tuning of association cortex in human structural brain networks. Cerebral Cortex, 28(1):281–294.

enigmatoolbox.permutation_testing.rotate_parcellation

enigmatoolbox.permutation_testing.**rotate_parcellation**(*coord_l*, *coord_r*, *nrot=1000*)

Rotate parcellation (author: @saratheriver)

Parameters

- **coord_l** (*ndarray*) – Coordinates of left hemisphere regions on the sphere, shape = (m, 3)
- **coord_r** (*ndarray*) – Coordinates of right hemisphere regions on the sphere, shape = (m, 3)
- **nrot** (*int, optional*) – Number of rotations. Default is 1000.

Returns **perm_id** – Array of permutations, shape = (m, nrot)

Return type ndarray

See also:

spin_test()

References

- Alexander-Bloch A, Shou H, Liu S, Satterthwaite TD, Glahn DC, Shinohara RT, Vandekar SN and Raznahan A (2018). On testing for spatial correspondence between maps of human brain structure and function. NeuroImage, 178:540-51.
- Vasa F, Seidlitz J, Romero-Garcia R, Whitaker KJ, Rosenthal G, Vertes PE, Shinn M, Alexander-Bloch A, Fonagy P, Dolan RJ, Goodyer IM, the NSPN consortium, Sporns O, Bullmore ET (2017). Adolescent tuning of association cortex in human structural brain networks. Cerebral Cortex, 28(1):281–294.

enigmatoolbox.permutation_testing.perm_sphere_p

enigmatoolbox.permutation_testing.**perm_sphere_p**(*x*, *y*, *perm_id*, *corr_type='pearson'*, *null_dist=False*)

Generate a p-value for the spatial correlation between two parcellated cortical surface maps (author: @saratheriver)

Parameters

- **x** (*narray, ndarray, or pandas.Series*) – One of two map to be correlated
- **y** (*narray, ndarray, or pandas.Series*) – The other map to be correlated
- **perm_id** (*ndarray*) – Array of permutations, shape = (m, nrot)
- **corr_type** (*string, optional*) – Correlation type {'pearson', 'spearman'}. Default is 'pearson'.
- **null_dist** (*bool, optional*) – Output null correlations. Default is False.

Returns

- **p_perm** (*float*) – Permutation p-value
- **r_dist** (*1D ndarray*) – Null correlations, shape = (n_rot*2,). Only if `null_dist is True`.

See also:

spin_test()

References

- Alexander-Bloch A, Shou H, Liu S, Satterthwaite TD, Glahn DC, Shinohara RT, Vandekar SN and Raznahan A (2018). On testing for spatial correspondence between maps of human brain structure and function. NeuroImage, 178:540-51.
- Vasa F, Seidlitz J, Romero-Garcia R, Whitaker KJ, Rosenthal G, Vertes PE, Shinn M, Alexander-Bloch A, Fonagy P, Dolan RJ, Goodyer IM, the NSPN consortium, Sporns O, Bullmore ET (2017). Adolescent tuning of association cortex in human structural brain networks. Cerebral Cortex, 28(1):281–294.

Shuf permutations

enigmatoolbox.permutation_testing.shuf_test(...)	Shuf permuation (author: @saratheriver)

enigmatoolbox.permutation_testing.shuf_test

enigmatoolbox.permutation_testing.**shuf_test**(*map1*, *map2*, *n_rot=1000*, *type='pearson'*, *null_dist=False*)

Shuf permuation (author: @saratheriver)

Parameters

- **map1** (*narray, ndarray, or pandas.Series*) – One of two map to be correlated
- **map2** (*narray, ndarray, or pandas.Series*) – The other map to be correlated

- **n_rot** (*int, optional*) – Number of shuffles. Default is 1000.
- **type** (*string, optional*) – Correlation type {'pearson', 'spearman'}. Default is 'pearson'.
- **null_dist** (*bool, optional*) – Output null correlations. Default is False.

Returns

- **p_shuf** (*float*) – Permutation p-value
- **r_dist** (*1D ndarray*) – Null correlations, shape = (n_rot*2,). Only if `null_dist is True`.

7.20.5 enigmatoolbox.plotting.surface_plotting

Surface plotting

enigmatoolbox.plotting.surface_plotting.plot_cortical([...])	Plot cortical surface with lateral and medial views (authors: @OualidBenkarim, @saratheriver)
enigmatoolbox.plotting.surface_plotting.plot_subcortical([...])	Plot subcortical surface with lateral and medial views (author: @saratheriver)
enigmatoolbox.plotting.surface_plotting.build_plotter(...)	Build plotter arranged according to the *layout* (author: @OualidBenkarim)
enigmatoolbox.plotting.surface_plotting.plot_surf(...)	Plot surfaces arranged according to the *layout* (author: @OualidBenkarim)

enigmatoolbox.plotting.surface_plotting.plot_cortical

enigmatoolbox.plotting.surface_plotting.**plot_cortical**(*array_name=None*, *surface_name='fsa5'*, *color_bar=False*, *color_range=None*, *label_text=None*, *cmap='RdBu_r'*, *nan_color=(1, 1, 1, 0)*, *zoom=1*, *background=(1, 1, 1)*, *size=(400, 400)*, *interactive=True*, *embed_nb=False*, *screenshot=False*, *filename=None*, *scale=(1, 1)*, *transparent_bg=True*, ***kwargs*)

Plot cortical surface with lateral and medial views (authors: @OualidBenkarim, @saratheriver)

Parameters

- **array_name** (*str, list of str, ndarray or list of ndarray, optional*) – Name of point data array to plot. If ndarray, the array is split for the left and right hemispheres. If list, plot one row per array. Default is None.
- **surface_name** (*str, optional*) – Name of surface {'fsa', 'fsa5', 'conte69}. Default is 'fsa5'.

- **color_bar** (*bool, optional*) – Plot color bar for each array (row). Default is False.
- **color_range** (*{'sym'}, tuple or sequence.*) – Range for each array name. If 'sym', uses a symmetric range. Only used if array has positive and negative values. Default is None.
- **label_text** (*dict[str, array-like], optional*) – Label text for column/row. Possible keys are {'left', 'right', 'top', 'bottom'}, which indicate the location. Default is None.
- **nan_color** (*tuple*) – Color for nan values. Default is (1, 1, 1, 0).
- **zoom** (*float or sequence of float, optional*) – Zoom applied to the surfaces in each layout entry.
- **background** (*tuple*) – Background color. Default is (1, 1, 1).
- **cmap** (*str, optional*) – Colormap name (from matplotlib). Default is 'RdBu_r'.
- **size** (*tuple, optional*) – Window size. Default is (400, 400).
- **interactive** (*bool, optional*) – Whether to enable interaction. Default is True.
- **embed_nb** (*bool, optional*) – Whether to embed figure in notebook. Only used if running in a notebook. Default is False.
- **screenshot** (*bool, optional*) – Take a screenshot instead of rendering. Default is False.
- **filename** (*str, optional*) – Filename to save the screenshot. Default is None.
- **transparent_bg** (*bool, optional*) – Whether to us a transparent background. Only used if `screenshot==True`. Default is False.
- **scale** (*tuple, optional*) – Scale (magnification). Only used if `screenshot==True`. Default is None.
- **kwargs** (*keyword-valued args*) – Additional arguments passed to the plotter.

Returns **figure** – Figure to plot. None if using vtk for rendering (i.e., `embed_nb == False`).

Return type Ipython Image or None

See also:

build_plotter(), *plot_surf()*

enigmatoolbox.plotting.surface_plotting.plot_subcortical

enigmatoolbox.plotting.surface_plotting.**plot_subcortical**(*array_name=None*, *ventricles=True*, *color_bar=False*, *color_range=None*, *label_text=None*, *cmap='RdBu_r'*, *nan_color=(1, 1, 1, 0)*, *zoom=1*, *background=(1, 1, 1)*, *size=(400, 400)*, *interactive=True*, *embed_nb=False*, *screenshot=False*, *filename=None*, *scale=(1, 1)*, *transparent_bg=True*, ***kwargs*)

Plot subcortical surface with lateral and medial views (author: @saratheriver)

Parameters

- **array_name** (*str, list of str, ndarray or list of ndarray, optional*) – Name of point data array to plot. If ndarray, the array is split for the left and right hemispheres. If list, plot one row per array. Default is None.
- **ventricles** (*bool, optional*) – Whether to include ventricles (i.e., array_name must have 16 values). False does not include ventricles (e.g., array_name must have 14 values). Default is True.
- **color_bar** (*bool, optional*) – Plot color bar for each array (row). Default is False.
- **color_range** (*{'sym'}, tuple or sequence.*) – Range for each array name. If 'sym', uses a symmetric range. Only used if array has positive and negative values. Default is None.
- **label_text** (*dict[str, array-like], optional*) – Label text for column/row. Possible keys are {'left', 'right', 'top', 'bottom'}, which indicate the location. Default is None.
- **nan_color** (*tuple*) – Color for nan values. Default is (1, 1, 1, 0).
- **zoom** (*float or sequence of float, optional*) – Zoom applied to the surfaces in each layout entry.
- **background** (*tuple*) – Background color. Default is (1, 1, 1).
- **cmap** (*str, optional*) – Color map name (from matplotlib). Default is 'RdBu_r'.
- **size** (*tuple, optional*) – Window size. Default is (400, 400).
- **interactive** (*bool, optional*) – Whether to enable interaction. Default is True.
- **embed_nb** (*bool, optional*) – Whether to embed figure in notebook. Only used if running in a notebook. Default is False.
- **screenshot** (*bool, optional*) – Take a screenshot instead of rendering. Default is False.
- **filename** (*str, optional*) – Filename to save the screenshot. Default is None.
- **transparent_bg** (*bool, optional*) – Whether to us a transparent background. Only used if `screenshot==True`. Default is False.

- **scale** (*tuple, optional*) – Scale (magnification). Only used if `screenshot==True`. Default is None.
- **kwargs** (*keyword-valued args*) – Additional arguments passed to the plotter.

Returns **figure** – Figure to plot. None if using vtk for rendering (i.e., `embed_nb == False`).

Return type Ipython Image or None

See also:

build_plotter(), *plot_surf()*

enigmatoolbox.plotting.surface_plotting.build_plotter

enigmatoolbox.plotting.surface_plotting.**build_plotter**(*surfs*, *layout*, *array_name=None*, *view=None*, *color_bar=None*, *color_range=None*, *share=False*, *label_text=None*, *cmap='viridis'*, *nan_color=(0, 0, 0, 1)*, *zoom=1*, *background=(1, 1, 1)*, *size=(400, 400)*, ***kwargs*)

Build plotter arranged according to the *layout* (author: @OualidBenkarim)

Parameters

- **surfs** (*dict[str, BSPolyData]*) – Dictionary of surfaces.
- **layout** (*array-like, shape = (n_rows, n_cols)*) – Array of surface keys in *surfs*. Specifies how window is arranged.
- **array_name** (*array-like, optional*) – Names of point data array to plot for each layout entry. Use a tuple with multiple array names to plot multiple arrays (overlays) per layout entry. Default is None.
- **view** (*array-like, optional*) – View for each each layout entry. Possible views are {'lateral', 'medial', 'ventral', 'dorsal'}. If None, use default view. Default is None.
- **color_bar** (*{'left', 'right', 'top', 'bottom'} or None, optional*) – Location where color bars are rendered. If None, color bars are not included. Default is None.
- **color_range** (*{'sym'}, tuple or sequence.*) – Range for each array name. If 'sym', uses a symmetric range. Only used if array has positive and negative values. Default is None.
- **share** (*{'row', 'col', 'both'} or bool, optional*) – If `share == 'row'`, point data for surfaces in the same row share same data range. If `share == 'col'`, the same but for columns. If `share == 'both'`, all data shares same range. If True, similar to `share == 'both'`. Default is False.
- **label_text** (*dict[str, array-like], optional*) – Label text for column/row. Possible keys are {'left', 'right', 'top', 'bottom'}, which indicate the location. Default is None.

- **cmap** (*str or sequence of str, optional*) – Color map name (from matplotlib) for each array name. Default is 'viridis'.
- **nan_color** (*tuple*) – Color for nan values. Default is (0, 0, 0, 1).
- **zoom** (*float or sequence of float, optional*) – Zoom applied to the surfaces in each layout entry.
- **background** (*tuple*) – Background color. Default is (1, 1, 1).
- **size** (*tuple, optional*) – Window size. Default is (400, 400).
- **kwargs** (*keyword-valued args*) – Additional arguments passed to the renderers, actors, mapper, color_bar or plotter.

Returns **plotter** – An instance of Plotter.

Return type Plotter

See also:

plot_surf(), *plot_cortical()*, *plot_subcortical()*

Notes

If sequences, shapes of *array_name*, *view* and *zoom* must be equal or broadcastable to the shape of *layout*. Renderer keywords must also be broadcastable to the shape of *layout*.

If sequences, shapes of *cmap* and *cbar_range* must be equal or broadcastable to the shape of *array_name*, including the number of array names per entry. Actor and mapper keywords must also be broadcastable to the shape of *array_name*.

enigmatoolbox.plotting.surface_plotting.plot_surf

enigmatoolbox.plotting.surface_plotting.**plot_surf**(*surfs*, *layout*, *array_name=None*, *view=None*, *color_bar=None*, *color_range=None*, *share=False*, *label_text=None*, *cmap='viridis'*, *nan_color=(0, 0, 0, 1)*, *zoom=1*, *background=(1, 1, 1)*, *size=(400, 400)*, *embed_nb=False*, *interactive=True*, *scale=(1, 1)*, *transparent_bg=True*, *screenshot=False*, *filename=None*, *return_plotter=False*, ***kwargs*)

Plot surfaces arranged according to the *layout* (author: @OualidBenkarim)

Parameters

- **surfs** (*dict[str, BSPolyData]*) – Dictionary of surfaces.
- **layout** (*array-like, shape = (n_rows, n_cols)*) – Array of surface keys in *surfs*. Specifies how window is arranged.
- **array_name** (*array-like, optional*) – Names of point data array to plot for each layout entry. Use a tuple with multiple array names to plot multiple arrays (overlays) per layout entry. Default is None.
- **view** (*array-like, optional*) – View for each each layout entry. Possible views are {'lateral', 'medial', 'ventral', 'dorsal'}. If None, use default view. Default is None.

- **color_bar** (*{'left', 'right', 'top', 'bottom'} or None, optional*) – Location where color bars are rendered. If None, color bars are not included. Default is None.
- **color_range** (*{'sym'}, tuple or sequence.*) – Range for each array name. If 'sym', uses a symmetric range. Only used if array has positive and negative values. Default is None.
- **share** (*{'row', 'col', 'both'} or bool, optional*) – If `share == 'row'`, point data for surfaces in the same row share same data range. If `share == 'col'`, the same but for columns. If `share == 'both'`, all data shares same range. If True, similar to `share == 'both'`. Default is False.
- **label_text** (*dict[str, array-like], optional*) – Label text for column/row. Possible keys are {'left', 'right', 'top', 'bottom'}, which indicate the location. Default is None.
- **cmap** (*str or sequence of str, optional*) – Color map name (from matplotlib) for each array name. Default is 'viridis'.
- **nan_color** (*tuple*) – Color for nan values. Default is (0, 0, 0, 1).
- **zoom** (*float or sequence of float, optional*) – Zoom applied to the surfaces in each layout entry.
- **background** (*tuple*) – Background color. Default is (1, 1, 1).
- **size** (*tuple, optional*) – Window size. Default is (400, 400).
- **interactive** (*bool, optional*) – Whether to enable interaction. Default is True.
- **embed_nb** (*bool, optional*) – Whether to embed figure in notebook. Only used if running in a notebook. Default is False.
- **screenshot** (*bool, optional*) – Take a screenshot instead of rendering. Default is False.
- **filename** (*str, optional*) – Filename to save the screenshot. Default is None.
- **transparent_bg** (*bool, optional*) – Whether to us a transparent background. Only used if `screenshot==True`. Default is False.
- **scale** (*tuple, optional*) – Scale (magnification). Only used if `screenshot==True`. Default is None.
- **kwargs** (*keyword-valued args*) – Additional arguments passed to the renderers, actors, mapper or plotter.

Returns **figure** – Figure to plot. None if using vtk for rendering (i.e., `embed_nb == False`).

Return type Ipython Image or panel or None

See also:

build_plotter(), *plot_cortical()*, *plot_subcortical()*

Notes

If sequences, shapes of *array_name*, *view* and *zoom* must be equal or broadcastable to the shape of *layout*. Renderer keywords must also be broadcastable to the shape of *layout*.

If sequences, shapes of *cmap* and *cbar_range* must be equal or broadcastable to the shape of *array_name*, including the number of array names per entry. Actor and mapper keywords must also be broadcastable to the shape of *array_name*.

7.20.6 enigmatoolbox.histology

BigBrain stratification

—

von Economo-Koskinas stratification

—

7.20.7 enigmatoolbox.utils

Re-order subcortical data matrix

enigmatoolbox.utils.useful.reorder_sctx(data)	Re-order subcortical volume data alphabetically and by hemisphere (left then right; author: @saratheriver)

enigmatoolbox.utils.useful.reorder_sctx

enigmatoolbox.utils.useful.**reorder_sctx**(*data*)

Re-order subcortical volume data alphabetically and by hemisphere (left then right; author: @saratheriver)

Parameters **data** (*pandas.DataFrame*) – Data matrix

Returns **data_r** – Re-ordered data

Return type pandas.DataFrame

Z-score data matrix

enigmatoolbox.utils.useful.zscore_matrix(...)	Z-score data relative to a given group (author: @saratheriver)

enigmatoolbox.utils.useful.zscore_matrix

enigmatoolbox.utils.useful.**zscore_matrix**(*data*, *group*, *controlCode*)

Z-score data relative to a given group (author: @saratheriver)

Parameters

- **data** (*pandas.DataFrame*) – Data matrix (e.g. thickness data), shape = (n_subject, n_region)
- **group** (*list*) – Group assignment (e.g, [0, 0, 0, 1, 1, 1], same length as n_subject.
- **controlCode** (*int*) – Value that corresponds to "baseline" group

Returns **Z** – Z-scored data relative to control code

Return type pandas.DataFrame

Parcellation

enigmatoolbox.utils.parcellation.parcel_to_surface(...)	Map data in source to target according to their labels (authors: @OualidBenkarim, @saratheriver)
enigmatoolbox.utils.parcellation.surface_to_parcel(...)	Summarize data in *values* according to *labels* (author: @OualidBenkarim)
enigmatoolbox.utils.parcellation.subcorticalvertices([...])	Map one value per subcortical area to surface vertices (author: @saratheriver)

enigmatoolbox.utils.parcellation.parcel_to_surface

enigmatoolbox.utils.parcellation.**parcel_to_surface**(*source_val*, *target_lab*, *mask=None*, *fill=0*, *source_lab=None*)

Map data in source to target according to their labels (authors: @OualidBenkarim, @saratheriver)

Target labels are sorted in ascending order, such that the smallest label indexes the value at position 0 in *source_val*. If *source_lab* is specified, any label in *target_lab* must be in *source_lab*.

Parameters

- **source_val** (*ndarray, shape = (n_val,)*) – Source array of values.
- **target_lab** (*can be a string (e.g., aparc_fsa5) or an ndarray, shape = (n_lab,)*) – Target labels.
- **mask** (*ndarray, shape = (n_lab,), optional*) – If mask is not None, only consider target labels in mask. Default is None.
- **fill** (*float, optional*) – Value used to fill elements outside the mask. Default is 0.
- **source_lab** (*ndarray, shape = (n_val,), optional*) – Source labels for source values. If None, use unique labels in *target_lab* in ascending order. Default is None.

Returns **target_val** – Target array with corresponding source values.

Return type ndarray, shape = (n_lab,)

enigmatoolbox.utils.parcellation.surface_to_parcel

enigmatoolbox.utils.parcellation.**surface_to_parcel**(*values*, *labels*, *weights=None*, *target_labels=None*, *red_op='mean'*, *axis=0*, *dtype=<class 'float'>*)

Summarize data in *values* according to *labels* (author: @OualidBenkarim)

Parameters

- **values** (*1D or 2D ndarray*) – Array of values.
- **labels** (*name of parcellation or 1D ndarray, shape = (n_lab,)*) – Labels used summarize values.
- **weights** (*1D ndarray, shape = (n_lab,), optional*) – Weights associated with labels. Only used when *red_op* is 'average', 'mean', 'sum' or 'mode'. Weights are not normalized. Default is None.
- **target_labels** (*1D ndarray, optional*) – Target labels. Arrange new array following the ordering of labels in the *target_labels*. When None, new array is arranged in ascending order of *labels*. Default is None.
- **red_op** (*str or callable, optional*) – How to summarize data. If str, options are: {'min', 'max', 'sum', 'mean', 'median', 'mode', 'average'}. If callable, it should receive a 1D array of values, array of weights (or None) and return a scalar value. Default is 'mean'.
- **dtype** (*dtype, optional*) – Data type of output array. Default is float.
- **axis** (*{0, 1}, optional*) – If `axis == 0`, apply to each row (reduce number of columns per row). Otherwise, apply to each column (reduce number of rows per column). Default is 0.

Returns **target_values** – Summarized target values.

Return type ndarray

enigmatoolbox.utils.parcellation.subcorticalvertices

enigmatoolbox.utils.parcellation.**subcorticalvertices**(*subcortical_values=None*)

Map one value per subcortical area to surface vertices (author: @saratheriver)

Parameters **subcortical_values** (*1D ndarray*) – Shape = (16,), order of subcortical structure must be = L_accumbens, L_amygdala, L_caudate, L_hippocampus, L_pallidun, L_putamen, L_thalamus, L_ventricles, R_accumbens, R_amygdala, R_caudate, R_hippocampus, R_pallidun, R_putamen, R_thalamus, R_ventricles

Returns **data** – Transformed data, shape = (51278,)

Return type 1D ndarray

7.21 Matlab API

7.21.1 enigma datasets

load_example_data()

Usage [source]:

```
[cov, metr1_SubVol, metr2_CortThick, metr3_CortSurf] = load_example_data()
```

Description: Loads the ENIGMA example dataset (from one site - MICA-MNI Montreal; author: @saratheriver)

Outputs:

- **cov** (*table*) – Contains information on covariates
- **metr1** (*table*) – Contains information on subcortical volume
- **metr2** (*table*) – Contains information on cortical thickness
- **metr3** (*table*) – Contains information on surface area

load_summary_stats(disorder)

Usage [source]:

```
summary_stats = load_summary_stats(disorder)
```

Description: Outputs summary statistics for a given disorder (author: @saratheriver)

Inputs:

- **disorder** ({'22q', 'adhd', 'asd', 'bipolar', 'depression', 'epilepsy', 'ocd', 'schizophrenia'}) – Disorder name, must pick one.

Outputs:

- **summary_stats** (*table*) – Available summary statistics

load_example_data()	Loads the ENIGMA example dataset (from one site - MICA-MNI Montreal; author: @saratheriver)
load_summary_stats(disorder)	Outputs summary statistics for a given disorder (author: @saratheriver)

7.21.2 import / export

nfaces()

Usage [source]:

```
M = nfaces(surface_name, hemisphere)
```

Description: Returns number of faces/triangles for a surface (author: @saratheriver)

Inputs:

- **surface_name** (*string*) - Name of surface {'fsa5', 'conte69'}
- **hemisphere** (*string*) - Name of hemisphere {'lh', 'rh', 'both'}

Outputs:

- **numfaces** (*double*) – Number of faces/triangles

getaffine()

Usage [source]:

```
M = getaffine(surface_name, hemisphere)
```

Description: Returns vox2ras transform for a surface (author: @saratheriver)

Inputs:

- **surface_name** (*string*) - Name of surface {'fsa5', 'conte69'}
- **hemisphere** (*string*) - Name of hemisphere {'lh', 'rh', 'both'}

Outputs:

- **M** (*double*) – Vox2ras transform, size = [4 x 4]

write_cifti()

Usage [source]:

write_cifti (*data*, *varargin*)

Description: Writes cifti file (authors: @NicoleEic, @saratheriver)

Inputs:

- **data** (*double* or *single*) – Data to be saved

Name/value pairs:

- **dpath** (*string, optional*) – Path to location for saving file (e.g., '/Users/bonjour/'). Default is ''.
- **fname** (*string, optional*) – Name of file (e.g., 'ello.dscalar.nii'). Default is ''.
- **surface_name** (*string, optional*) – Surface name {'fsa5', 'conte69'}. Default is 'conte69'.
- **hemi** (*string, optional*) – Name of hemisphere {'lh', 'rh'}. Default is 'lh'.

nfaces(surface_name, hemisphere)	Returns number of faces/triangles for a surface (author: @saratheriver)
getaffine(surface_name, hemisphere)	Returns vox2ras transform for a surface (author: @saratheriver)
write_cifti(data, varargin)	Writes cifti file (authors: @NicoleEic, @saratheriver)

7.21.3 connectivity matrices

load_fc()

Usage [source]:

```
[funcMatrix_ctx, funcLabels_ctx, funcMatrix_sctx, funcLabels_sctx] = load_fc(parcellat
```

Description: Load functional connectivity data (author: @saratheriver)

Inputs:

- **parcellation** (*string, optional*) - Name of parcellation (with n cortical parcels). Default is 'aparc'. Other options are 'schaefer_100', 'schaefer_200', 'schaefer_300', 'schaefer_400'.

Outputs:

- **funcMatrix_ctx** (*double array*) – Cortico-cortical connectivity, size = [n x n]
- **funcLabels_ctx** (*cell array*) – Cortical labels, size = [1 x n]
- **funcMatrix_sctx** (*double array*) – Subcortico-cortical connectivity, size = [14 x n]
- **funcLabels_sctx** (*cell array*) – Subcortical labels, size = [1 x 14]

load_fc_as_one()

Usage [source]:

```
[funcMatrix, funcLabels] = load_fc_as_one(parcellation)
```

Description: Load functional connectivity data (cortical + subcortical in one matrix; author: @saratheriver)

Inputs:

- **parcellation** (*string, optional*) - Name of parcellation (with n cortical parcels). Default is 'aparc'. Other options are 'schaefer_100', 'schaefer_200', 'schaefer_300', 'schaefer_400'.

Outputs:

- **funcMatrix** (*double array*) – Functional connectivity, size = [n+14 x n+14]
- **funcLabels** (*cell array*) – Region labels, size = [1 x n+14]

load_sc()

Usage [source]:

```
[strucMatrix_ctx, strucLabels_ctx, strucMatrix_sctx, strucLabels_sctx] = load_sc(parce
```

Description: Load structural connectivity data (author: @saratheriver)

Inputs:

- **parcellation** (*string, optional*) - Name of parcellation (with n cortical parcels). Default is 'aparc'. Other options are 'schaefer_100', 'schaefer_200', 'schaefer_300', 'schaefer_400'.

Outputs:

- **strucMatrix_ctx** (*double array*) – Cortico-cortical connectivity, size = [n x n]
- **strucLabels_ctx** (*cell array*) – Cortical labels, size = [1 x n]

- **strucMatrix_sctx** (*double array*) – Subcortico-cortical connectivity, size = [14 x n]
- **strucLabels_sctx** (*cell array*) – Subcortical labels, size = [1 x 14]

load_sc_as_one()

Usage [source]:

```
[strucMatrix, strucLabels] = load_sc_as_one(parcellation)
```

Description: Load structural connectivity data (cortical + subcortical in one matrix; author: @saratheriver)

Inputs:

- **parcellation** (*string, optional*) - Name of parcellation (with n cortical parcels). Default is 'aparc'. Other options are 'schaefer_100', 'schaefer_200', 'schaefer_300', 'schaefer_400'.

Outputs:

- **strucMatrix** (*double array*) – Structural connectivity, size = [n+14 x n+14]
- **strucLabels** (*cell array*) – Region labels, size = [1 x n+14]

load_fc()	Load functional connectivity data (author: @saratheriver)
load_fc_sc_one()	Load functional connectivity data (cortical + subcortical in one matrix; author: @saratheriver)
load_sc()	Load structural connectivity data (author: @saratheriver)
load_sc_as_one()	Load structural connectivity data (cortical + subcortical in one matrix; author: @saratheriver)

7.21.4 gene co-expression data

fetch_ahba()

Usage [source]:

```
genes = fetch_ahba();
```

Description: Fetch Allen Human Brain Atlas microarray expression data from all donors and all genes (author: @saratheriver)

Inputs:

- **csvfile** (empty or string) – Path to downloaded csvfile. If empty (default), fetches microarray expression data from the internet. For more threshold and parcellation options, download csvfile from here: https://github.com/saratheriver/enigma-extra/tree/master/ahba. If empty, fetches microarray expression data from the internet (aparc and stable r > 0.2).

Outputs:

- **genes** (*table*) - Gene co-expression data, size = [82 x 15633]

risk_genes(disorder)

Usage [source]:

```
risk_genes = risk_genes(disorder)
```

Description: Outputs names of GWAS-derived risk genes for a given disorder (author: @saratheriver)

Inputs:

- **disorder** ({'adhd', 'asd', 'bipolar', 'depression', 'epilepsy', 'hippocampal_volume', 'ocd', 'schizophrenia', 'tourette'}) –

Disorder name, must pick one.

Outputs:

- **risk_genes** (*cell array*) – Names of genes for a given disorder

fetch_donors()	Fetch Allen Human Brain Atlas microarray expression data from all donors and all genes (author: @saratheriver)
risk_genes(disorder)	Outputs names of GWAS-derived risk genes for a given disorder

7.21.5 cross-disorder effect

cross_disorder_effect()

Usage [source]:

```
[components, variance, ~, names] = cross_disorder_effect(varargin)
[~, ~, correlation_matrix, names] = cross_disorder_effect(varargin)
```

Description: Cross-disorder effect (authors: @boyongpark, @saratheriver)

Name/value pairs:

- **disorder** (**cell array, optional**) - Any combination of disorder name. Default is all available disorders, except 'adhd' due to NaNs. Options are any combination of {'22q', 'adhd', 'asd', 'bipolar', 'depression', 'epilepsy', 'ocd', 'schizophrenia'}.
- **measure** (*cell array, optional*) - Any combination of measure names. Default is {'CortThick', 'CortSurf'}.
- **additional_data_cortex** (*double array, optional*) - Name for additional cortical ENIGMA-type data. Must also provide 'additional_name_cortex'.
- **additional_name_cortex** (*cell array, optional*) - Additional cortical ENIGMA-type data (n, 68). Must also provide 'additional_data_cortex'.
- **additional_data_subcortex** (*double array, optional*) - Name for additional subcortical ENIGMA-type data. Must also provide 'additional_name_subcortex'.
- **additional_name_subcortex** (*cell array, optional*) - Additional subcortical ENIGMA-type data (n, 16). Must also provide 'additional_data_subcortex'.
- **ignore** (*cell array, optional*) - Ignore summary statistics with these expressions. Default is ('mega') as it contains NaNs.
- **include** (c*ell array, optional*) - Include only summary statistics with these expressions. Default is empty.
- **method** (*string, optional*) - Analysis method {'pca', 'correlation'}. Default is 'pca'.

Outputs:

- **components** (*structure*) - Principal components of shared effects in descending order in terms of component variance. Only is method is 'pca'.
- **variance** (*structure*) - Variance of components. Only is method is 'pca'.
- **correlation_matrix** (*structure*) - Correlation matrix of for every pair of shared effect maps. Only is method is 'correlation'.
- **names** (*structure*) - Name of disorder and case-control effect maps included in analysis.

cross_disorder()	Cross-disorder effect (authors: @boyongpark, @saratheriver)

7.21.6 spin permutations

spin_test()

Usage [source]:

```
[p_spin, r_dist] = spin_test(map1, map2, varargin)
```

Description: Spin permutation (author: @saratheriver)

Inputs:

- **map1** (*double array*) – One of two map to be correlated
- **map2** (*double array*) – The other map to be correlated

Name/value pairs:

- **surface_name** (*string, optional*) – Surface name {'fsa5', 'fsa5_with_sctx'}. Use 'fsa5' for Conte69. Default is 'fsa5'.
- **parcellation_name** (*string, optional*) – Parcellation name {'aparc', 'aparc_aseg'}. Default is 'aparc'.
- **n_rot** (*int, optional*) – Number of spin rotations. Default is 100.
- **type** (*string, optional*) – Correlation type {'pearson', 'spearman'}. Default is 'pearson'.
- **ventricles** (*string, optional*) – Whether ventricles are present in map1, map2. Only used when `parcellation_name is 'aparc_aseg'`. Default is 'False' (other option is 'True')

Outputs:

- **p_spin** (*double*) – Permutation p-value
- **r_dist** (*double array*) - Null correlations, size = [n_rot*2 x 1].

References:

- Alexander-Bloch A, Shou H, Liu S, Satterthwaite TD, Glahn DC, Shinohara RT, Vandekar SN and Raznahan A (2018). On testing for spatial correspondence between maps of human brain structure and function. NeuroImage, 178:540-51.
- Vasa F, Seidlitz J, Romero-Garcia R, Whitaker KJ, Rosenthal G, Vertes PE, Shinn M, Alexander-Bloch A, Fonagy P, Dolan RJ, Goodyer IM, the NSPN consortium, Sporns O, Bullmore ET (2017). Adolescent tuning of association cortex in human structural brain networks. Cerebral Cortex, 28(1):281–294.

centroid_extraction_sphere()

Usage [source]:

```
centroid = centroid_extraction_sphere(sphere_coords, annotfile)
```

Description: Extract centroids of a cortical parcellation on a surface sphere (authors: @frantisekvasa, @saratheriver)

Inputs:

- **sphere_coords** (*double array*) – Sphere coordinates, size = [n x 3]
- **annotfile** (*string*) – Name of annotation file {'fsa5_lh_aparc.annot', 'fsa5_rh_aparc.annot, 'fsa5_with_sctx_lh_aparc_aseg.csv', etc.}
- **ventricles** (*string, optional*) – Whether ventricle data are present. Only used when 'annotfile' is fsa5_with_sctx_lh_aparc_aseg or fsa5_with_sctx_lh_aparc_aseg. Default is 'False'.

Outputs:

- **coord** (*double array*) – Coordinates of the centroid of each region on the sphere, size = [m x 3].

References:

- Alexander-Bloch A, Shou H, Liu S, Satterthwaite TD, Glahn DC, Shinohara RT, Vandekar SN and Raznahan A (2018). On testing for spatial correspondence between maps of human brain structure and function. NeuroImage, 178:540-51.
- Vasa F, Seidlitz J, Romero-Garcia R, Whitaker KJ, Rosenthal G, Vertes PE, Shinn M, Alexander-Bloch A, Fonagy P, Dolan RJ, Goodyer IM, the NSPN consortium, Sporns O, Bullmore ET (2017). Adolescent tuning of association cortex in human structural brain networks. Cerebral Cortex, 28(1):281–294.

rotate_parcellation()

Usage [source]:

```
perm_id = rotate_parcellation(coord_l, coord_r, nrot)
```

Description: Rotate parcellation (authors: @frantisekvasa, @saratheriver)

Inputs:

- **coord_l** (*double array*) – Coordinates of left hemisphere regions on the sphere, size = [m x 3]
- **coord_r** (*double array*) – Coordinates of right hemisphere regions on the sphere, size = [m x 3]
- **nrot** (*int, optional*) – Number of rotations. Default is 100.

Outputs:

- **perm_id** (*double array*) – Array of permutations, size = [m x nrot]

References:

- Alexander-Bloch A, Shou H, Liu S, Satterthwaite TD, Glahn DC, Shinohara RT, Vandekar SN and Raznahan A (2018). On testing for spatial correspondence between maps of human brain structure and function. NeuroImage, 178:540-51.
- Vasa F, Seidlitz J, Romero-Garcia R, Whitaker KJ, Rosenthal G, Vertes PE, Shinn M, Alexander-Bloch A, Fonagy P, Dolan RJ, Goodyer IM, the NSPN consortium, Sporns O, Bullmore ET (2017). Adolescent tuning of association cortex in human structural brain networks. Cerebral Cortex, 28(1):281–294.

perm_sphere_p()

Usage [source]:

```
[p_perm, null_dist] = perm_sphere_p(x, y, perm_id, corr_type)
```

Description: Generate a p-value for the spatial correlation between two parcellated cortical surface maps (authors: @frantisekvasa, @saratheriver)

Inputs:

- **x** (*double array*) – One of two map to be correlated
- **y** (*double array*) – The other map to be correlated
- **perm_id** (*double array*) – Array of permutations, size = [m x nrot]
- **corr_type** (*string, optional*) - Correlation type {'pearson', 'spearman'}. Default is 'pearson'.

Outputs:

- **p_perm** (*double*) – Permutation p-value
- **null_dist** (*double array*) - Null correlations, size = [n_rot*2 x 1].

References:

- Alexander-Bloch A, Shou H, Liu S, Satterthwaite TD, Glahn DC, Shinohara RT, Vandekar SN and Raznahan A (2018). On testing for spatial correspondence between maps of human brain structure and function. NeuroImage, 178:540-51.
- Vasa F, Seidlitz J, Romero-Garcia R, Whitaker KJ, Rosenthal G, Vertes PE, Shinn M, Alexander-Bloch A, Fonagy P, Dolan RJ, Goodyer IM, the NSPN consortium, Sporns O, Bullmore ET (2017). Adolescent tuning of association cortex in human structural brain networks. Cerebral Cortex, 28(1):281–294.

spin_test(map1, map2, varargin)	Spin permutation (author: @saratheriver)
centroid_extraction_sphere(sphere_coords, annotfile)	Extract centroids of a cortical parcellation on a surface sphere (authors: @frantisekvasa, @saratheriver)
rotate_parcellation(coord_l, coord_r, nrot)	Rotate parcellation (authors: @frantisekvasa, @saratheriver)
perm_sphere_p(x, y, perm_id, corr_type)	Generate a p-value for the spatial correlation between two parcellated cortical surface maps (authors: @frantisekvasa, @saratheriver)

7.21.7 shuf permutations

shuf_test()

Usage [source]:

```
[p_shuf, r_dist] = shuf_test(map1, map2, varargin)
```

Description: Shuf permuation (author: @saratheriver)

Inputs:

- **map1** (*double array*) – One of two map to be correlated
- **map2** (*double array*) – The other map to be correlated

Name/value pairs:

- **n_rot** (*int, optional*) – Number of shuffles. Default is 100.
- **type** (*string, optional*) – Correlation type {'pearson', 'spearman'}. Default is 'pearson'.

Outputs:

- **p_shuf** (*double*) – Permutation p-value
- **r_dist** (*double array*) - Null correlations, size = [n_rot*2 x 1].

shuf_test(map1, map2, varargin)	Shuf permuation (author: @saratheriver)

7.21.8 surface plotting

plot_cortical()

Usage [source]:

```
[a, cb] = plot_cortical(data, varargin);
```

Description: Plot cortical surface with lateral and medial views (authors: @MICA-MNI, @saratheriver)

Inputs:

- **data** (*double array*) – vector of data, size = [1 x v]

Name/value pairs:

- **surface_name** (*string, optional*) – Name of surface {'fsa5', 'conte69}. Default is 'fsa5'.
- **label_text** (*string, optional*) – Label text for colorbar. Default is empty.
- **background** (*string, double array, optional*) – Background color. Default is 'white'.
- **color_range** (*double array, optional*) – Range of colorbar. Default is [min(data) max(data)].
- **cmap** (*string, double array, optional*) – Colormap name. Default is 'RdBu_r'.

Outputs:

- **a** (*axes*) – vector of handles to the axes, left to right, top to bottom
- **cb** (*colorbar*) - colorbar handle

plot_subcortical()

Usage [source]:

```
[a, cb] = plot_subcortical(data, varargin);
```

Description: Plot subcortical surface with lateral and medial views (author: @saratheriver)

Inputs:

- **data** (*double array*) – vector of data, size = [1 x v]. One value per subcortical structure, in this order: L-accumbens, L-amygdala, L-caudate, L-hippocampus, L-pallidum L-putamen, L-thalamus, L-ventricle, R-accumbens, R-amygdala, R-caudate, R-hippocampus, R-pallidum, R-putamen, R-thalamus, R-ventricle

Name/value pairs:

- **ventricles** (*string, optional*) – If 'True' (default) shows the ventricles (`data must be size = [1 x 16]`). If 'False', then ventricles are not shown and `data must be size = [1 x 14]`.
- **label_text** (*string, optional*) – Label text for colorbar. Default is empty.
- **background** (*string, double array, optional*) – Background color. Default is 'white'.
- **color_range** (*double array, optional*) – Range of colorbar. Default is [min(data) max(data)].
- **cmap** (*string, double array, optional*) – Colormap name. Default is 'RdBu_r'.

Outputs:

- **a** (*axes*) – vector of handles to the axes, left to right, top to bottom
- **cb** (*colorbar*) - colorbar handle

plot_cortical(data, varargin)	Plot cortical surface with lateral and medial views (authors: @MICA-MNI, @saratheriver)
plot_subcortical(data, varargin)	Plot subcortical surface with lateral and medial views (author: @saratheriver)

7.21.9 histology

bb_moments_raincloud()

Usage [source]:

bb_moments_raincloud(*region_idx*, *title*)

Description: Stratify regional data according to BigBrain statistical moments (authors: @caseypaquola, @saratheriver)

Inputs:

- **region_idx** (*double array*) - Vector of data. Indices of regions to be included in analysis
- **parcellation** (*string, optional*) - Name of parcellation. Options are 'aparc', 'schaefer_100', 'schaefer_200', 'schaefer_300', 'schaefer_400', 'glasser_360'. Default is 'aparc'.
- **title** (*string, optional*) - Title of raincloud plot. Default is empty.

Outputs:

- **figure** (*figure*) – Raincloud plot.

bb_gradient_plot()

Usage [source]:

`bb_gradient_plot` (*data*, *varargin*)

Description: Stratify parcellated data according to the BigBrain gradient (authors: @caseypaquola, @saratheriver)

Inputs:

- **data** (*double array*) – vector of data. Parcellated data.

Name/value pairs:

- **parcellation** (*string, optional*) - Name of parcellation. Options are: 'aparc', 'schaefer_100', 'schaefer_200', 'schaefer_300', 'schaefer_400', 'glasser_360'. Default is 'aparc'.
- **title** (*string, optional*) – Title of spider plot. Default is empty.
- **axis_range** (*double array, optional*) - Range of spider plot axes. Default is (min, max).
- **yaxis_label** (*string, optional*) - Label for y-axis. Default is empty.
- **xaxis_label** (*string, optional*) - Label for x-axis. Default is empty.

Outputs:

- **figure** (*figure*) – Gradient plot.

economo_koskinas_spider()

Usage [source]:

```
class_mean = economo_koskinas_spider(parcel_data, varargin)
```

Description: Stratify parcellated data according to von Economo-Koskinas cytoarchitectonic classes (authors: @caseypaquola, @saratheriver)

Inputs:

- **parcel_data** (*double array*) – vector of data. Parcellated data.

Name/value pairs:

- **parcellation** (*string, optional*) - Parcellation to go from parcel_data to surface. Default is 'aparc_fsa5'.
- **fill** (*double, optional*) - Value for mask. Default is 0.
- **title** (*string, optional*) – Title of spider plot. Default is empty.
- **axis_range** (*double array, optional*) - Range of spider plot axes. Default is (min, max).
- **label** (*cell array, optional*) - List of axis labels. Length = 5. Default is names of von Economo-Koskinas cytoarchitectonic classes.
- **color** (*double array, optional*) - Color of line. Default is [0 0 0].

Outputs:

- **figure** (*figure*) – Spider plot.

bb_moments_stratify_regional_data(regional_data, title)	Stratify regional data according to BigBrain statistical moments (authors: @caseypaquola, @saratheriver)
bb_gradient_stratify_parcellated_data(parcellated_data, varargin)	Stratify parcellated data according to the BigBrain gradient (authors: @caseypaquola, @saratheriver)
economo_koskinas_stratify_parcellated_data(parcellated_data, varargin)	Stratify parcellated data according to von Economo-Koskinas cytoarchitectonic classes (authors: @caseypaquola, @saratheriver)

7.21.10 re-order subcortical data matrix

reorder_sctx()

Usage [source]:

```
data_r = reorder_sctx(data)
```

Description: Re-order subcortical volume data alphabetically and by hemisphere (left then right; author: @saratheriver)

Inputs:

- **data** (*table*) - Data matrix

Outputs:

- **data_r** (*table*) – Re-ordered data

reorder_sctx(data)	Re-order subcortical volume data alphabetically and by hemisphere (left then rightl; author: @saratheriver)

7.21.11 z-score data matrix

zscore_matrix()

Usage [source]:

```
Z = zscore_matrix(data, group, controlCode)
```

Description: Z-score data relative to a given group (author: @saratheriver)

Inputs:

- **data** (*double array*) - Data matrix (e.g., thickness data), size = [n_subject x n_region]
- **group** (*double array*) - Vector of values for group assignment (e.g, [0 0 0 1 1 1], same length as n_subject.
- **controlCode** (*int*) - Value that corresponds to "baseline" group.

Outputs:

- **Z** (*doule array*) – Z-scored data relative to control code

zscore_matrix(data, group, controlCode)	Z-score data relative to a given group (author: @saratheriver)

7.21.12 parcellation

parcel_to_surface()

Usage [source]:

```
parcel2surf = parcel_to_surface(parcel_data, parcellation, fill)
```

Description: Map parcellated data to the surface (authors : @MICA-MNI, @saratheriver)

Inputs:

- **parcel_data** (*double array*) - Parcel vector, size = [p x 1]. For example, if Desikan Killiany from ENIGMA data, then parcel_data is size = [68 x 1].
- **parcellation** (*string, optional*) - Default is 'aparc_fsa5'
- **fill** (*double, optional*) - Value for mask. Default is 0.

Outputs:

- **parcel2surf** (*double array*) – Vector of values mapped from a parcellation to the surface

surface_to_parcel()

Usage [source]:

```
surf2parcel = surface_to_parcel(surf_data, parcellation)
```

Description: Map surface data to a parcellation (authors : @MICA-MNI, @saratheriver)

Inputs:

- **surf_data** (*double array*) - Surface vector, size = [v x 1].
- **parcellation** (*string, optional*) - Default is 'aparc_fsa5'

Outputs:

- **surf2parcel** (*double array*) – Vector of values mapped from a surface to a parcellation

parcel_to_surface(parcel_data, parcellation, fill)	Map parcellated data to the surface (authors : @MICA-MNI, @saratheriver)
surface_to_parcel(surf_data, parcellation)	Map surface data to a parcellation (authors : @MICA-MNI, @saratheriver)

7.22 References

7.22.1 ENIGMA datasets

- Sun, D., Ching, C. R., Lin, A., Forsyth, J. K., Kushan, L., Vajdi, A., ... & Jonas, R. K. (2018). Large-scale mapping of cortical alterations in 22q11. 2 deletion syndrome: convergence with idiopathic psychosis and effects of deletion size. Molecular psychiatry, 1-13.
- Boedhoe, P. S., Van Rooij, D., Hoogman, M., Twisk, J. W., Schmaal, L., Abe, Y., ... & Arango, C. (2020). Subcortical brain volume, regional cortical thickness, and cortical surface area across disorders: findings from the ENIGMA ADHD, ASD, and OCD working groups. American Journal of Psychiatry, 177(9), 834-843.
- Van Rooij, D., Anagnostou, E., Arango, C., Auzias, G., Behrmann, M., Busatto, G. F., ... & Dinstein, I. (2018). Cortical and subcortical brain morphometry differences between patients with autism spectrum disorder and healthy individuals across the lifespan: results from the ENIGMA ASD Working Group. American Journal of Psychiatry, 175(4), 359-369.
- Hibar, D. P., Westlye, L. T., Doan, N. T., Jahanshad, N., Cheung, J. W., Ching, C. R., ... & Krämer, B. (2018). Cortical abnormalities in bipolar disorder: an MRI analysis of 6503 individuals from the ENIGMA Bipolar Disorder Working Group. Molecular psychiatry, 23(4), 932-942.
- Whelan, C. D., Altmann, A., Botía, J. A., Jahanshad, N., Hibar, D. P., Absil, J., ... & Bergo, F. P. (2018). Structural brain abnormalities in the common epilepsies assessed in a worldwide ENIGMA study. Brain, 141(2), 391-408.
- Schmaal, L., Hibar, D. P., Sämann, P. G., Hall, G. B., Baune, B. T., Jahanshad, N., ... & Vernooij, M. W. (2017). Cortical abnormalities in adults and adolescents with major depression based on brain scans from 20 cohorts worldwide in the ENIGMA Major Depressive Disorder Working Group. Molecular psychiatry, 22(6), 900-909.
- Schmaal, L., Veltman, D. J., van Erp, T. G., Sämann, P. G., Frodl, T., Jahanshad, N., ... & Vernooij, M. W. (2016). Subcortical brain alterations in major depressive disorder: findings from the ENIGMA Major Depressive Disorder working group. Molecular psychiatry, 21(6), 806-812.
- Boedhoe, P. S., Schmaal, L., Abe, Y., Alonso, P., Ameis, S. H., Anticevic, A., ... & Bollettini, I. (2018). Cortical abnormalities associated with pediatric and adult obsessive-compulsive disorder: findings from the ENIGMA Obsessive-Compulsive Disorder Working Group. American Journal of Psychiatry, 175(5), 453-462.
- Van Erp, T. G., Walton, E., Hibar, D. P., Schmaal, L., Jiang, W., Glahn, D. C., ... & Okada, N. (2018). Cortical brain abnormalities in 4474 individuals with schizophrenia and 5098 control subjects via the Enhancing Neuro Imaging Genetics Through Meta Analysis (ENIGMA) Consortium. Biological psychiatry, 84(9), 644-654.
- van Erp, T. G., Hibar, D. P., Rasmussen, J. M., Glahn, D. C., Pearlson, G. D., Andreassen, O. A., ... & Melle, I. (2016). Subcortical brain volume abnormalities in 2028 individuals with schizophrenia and 2540 healthy controls via the ENIGMA consortium. Molecular psychiatry, 21(4), 547-553.

7.22.2 Connectivity data

- Van Essen, D. C., Smith, S. M., Barch, D. M., Behrens, T. E., Yacoub, E., Ugurbil, K., & Wu-Minn HCP Consortium. (2013). The WU-Minn human connectome project: an overview. Neuroimage, 80, 62-79.
- Glasser, M. F., Sotiropoulos, S. N., Wilson, J. A., Coalson, T. S., Fischl, B., Andersson, J. L., ... & Van Essen, D. C. (2013). The minimal preprocessing pipelines for the Human Connectome Project. Neuroimage, 80, 105-124.

7.22.3 Gene co-expression data

- Arnatkeviciūtė, A., Fulcher, B. D., & Fornito, A. (2019). A practical guide to linking brain-wide gene expression and neuroimaging data. Neuroimage, 189, 353-367.
- Hawrylycz, M. J., Lein, E. S., Guillozet-Bongaarts, A. L., Shen, E. H., Ng, L., Miller, J. A., … & Abajian, C. (2012). An anatomically comprehensive atlas of the adult human brain transcriptome. Nature, 489(7416), 391-399.
- Markello, Ross, Shafiei, Golia, Zheng, Ying-Qiu, Mišić, Bratislav. abagen: A toolbox for the Allen Brain Atlas genetics data. Zenodo; 2020. Available from: https://doi.org/10.5281/zenodo.3726257.

7.22.4 GWAS

- Demontis, D., Walters, R. K., Martin, J., Mattheisen, M., Als, T. D., Agerbo, E., … & Cerrato, F. (2019). Discovery of the first genome-wide significant risk loci for attention deficit/hyperactivity disorder. Nature genetics, 51(1), 63-75.
- Grove, J., Ripke, S., Als, T. D., Mattheisen, M., Walters, R. K., Won, H., … & Awashti, S. (2019). Identification of common genetic risk variants for autism spectrum disorder. Nature genetics, 51(3), 431-444.
- Stahl, E. A., Breen, G., Forstner, A. J., McQuillin, A., Ripke, S., Trubetskoy, V., … & de Leeuw, C. A. (2019). Genome-wide association study identifies 30 loci associated with bipolar disorder. Nature genetics, 51(5), 793-803.
- Howard, D. M., Adams, M. J., Clarke, T. K., Hafferty, J. D., Gibson, J., Shirali, M., … & Alloza, C. (2019). Genome-wide meta-analysis of depression identifies 102 independent variants and highlights the importance of the prefrontal brain regions. Nature neuroscience, 22(3), 343-352.
- Consortium, T. I. L. A. E. (2018). Genome-wide mega-analysis identifies 16 loci and highlights diverse biological mechanisms in the common epilepsies. Nature communications, 9.
- Pardiñas, A. F., Holmans, P., Pocklington, A. J., Escott-Price, V., Ripke, S., Carrera, N., … & Han, J. (2018). Common schizophrenia alleles are enriched in mutation-intolerant genes and in regions under strong background selection. Nature genetics, 50(3), 381-389.
- Yu, D., Sul, J. H., Tsetsos, F., Nawaz, M. S., Huang, A. Y., Zelaya, I., … & Greenberg, E. (2019). Interrogating the genetic determinants of Tourette's syndrome and other tic disorders through genome-wide association studies. American Journal of Psychiatry, 176(3), 217-227.

7.22.5 BigBrain

- Amunts, K., Lepage, C., Borgeat, L., Mohlberg, H., Dickscheid, T., Rousseau, M. É., … & Shah, N. J. (2013). BigBrain: an ultrahigh-resolution 3D human brain model. Science, 340(6139), 1472-1475.
- vPapoulis, A., & Pillai, S. U. (2002). Probability, random variables, and stochastic processes. Tata McGraw-Hill Education.
- Paquola, C., De Wael, R. V., Wagstyl, K., Bethlehem, R. A., Hong, S. J., Seidlitz, J., … & Smallwood, J. (2019). Microstructural and functional gradients are increasingly dissociated in transmodal cortices. PLoS biology, 17(5), e3000284.

7.22.6 von Economo and Koskinas atlas

- von Economo, C. F., & Koskinas, G. N. (1925). Die cytoarchitektonik der hirnrinde des erwachsenen menschen. J. Springer.
- Scholtens, L. H., de Reus, M. A., de Lange, S. C., Schmidt, R., & van den Heuvel, M. P. (2018). An mri von economo–koskinas atlas. NeuroImage, 170, 249-256.
- Triarhou, L. C. (2007). The Economo-Koskinas atlas revisited: cytoarchitectonics and functional context. Stereotactic and functional neurosurgery, 85(5), 195-203.

7.22.7 Network-based atrophy models (*hubs*)

- Fornito, A., Zalesky, A., & Breakspear, M. (2015). The connectomics of brain disorders. Nature Reviews Neuroscience, 16(3), 159-172.
- van den Heuvel, M. P., & Sporns, O. (2013). Network hubs in the human brain. Trends in cognitive sciences, 17(12), 683-696.
- Crossley, N. A., Mechelli, A., Scott, J., Carletti, F., Fox, P. T., McGuire, P., & Bullmore, E. T. (2014). The hubs of the human connectome are generally implicated in the anatomy of brain disorders. Brain, 137(8), 2382-2395.

7.22.8 Network-based atrophy models (*disease epicenters*)

- Larivière, S., Rodríguez-Cruces, R., Royer, J., Caligiuri, M. E., Gambardella, A., Concha, L., … & Gleichgerrcht, E. (2020). Network-based atrophy modeling in the common epilepsies: a worldwide ENIGMA study. Science Advances, 6.
- Shafiei, G., Markello, R. D., Makowski, C., Talpalaru, A., Kirschner, M., Devenyi, G. A., … & Chakravarty, M. M. (2020). Spatial patterning of tissue volume loss in schizophrenia reflects brain network architecture. Biological psychiatry, 87(8), 727-735.
- Zeighami, Y., Ulla, M., Iturria-Medina, Y., Dadar, M., Zhang, Y., Larcher, K. M. H., … & Dagher, A. (2015). Network structure of brain atrophy in de novo Parkinson's disease. Elife, 4, e08440.
- Brown, J. A., Deng, J., Neuhaus, J., Sible, I. J., Sias, A. C., Lee, S. E., … & Grinberg, L. T. (2019). Patient-tailored, connectivity-based forecasts of spreading brain atrophy. Neuron, 104(5), 856-868.

7.22.9 Spin permutations

- Alexander-Bloch, A. F., Shou, H., Liu, S., Satterthwaite, T. D., Glahn, D. C., Shinohara, R. T., … & Raznahan, A. (2018). On testing for spatial correspondence between maps of human brain structure and function. Neuroimage, 178, 540-551.
- Váša, F., Seidlitz, J., Romero-Garcia, R., Whitaker, K. J., Rosenthal, G., Vértes, P. E., … & Jones, P. B. (2018). Adolescent tuning of association cortex in human structural brain networks. Cerebral Cortex, 28(1), 281-294.

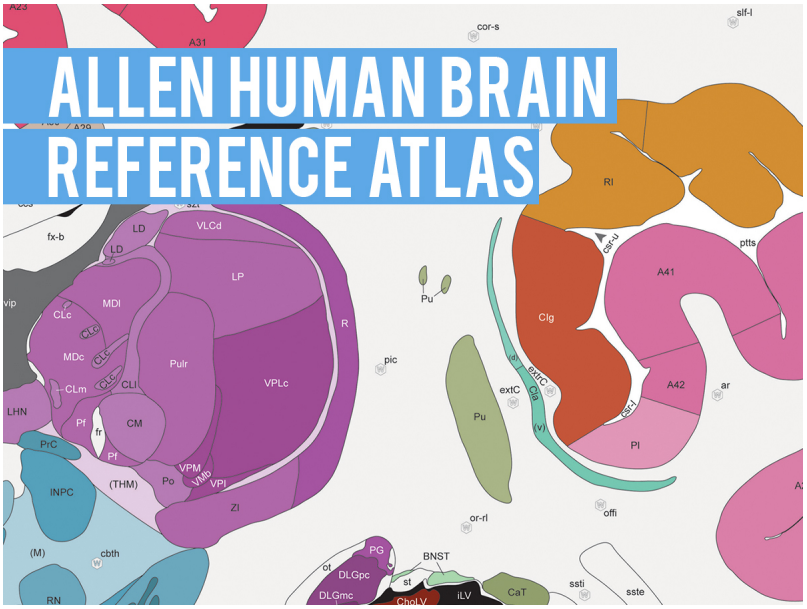
7.23 Acknowledgements

The authors would like to express their gratitude to the open science initiatives that made this work possible:

The ENIGMA Consortium

The Human Connectome Project

Allen Human Brain Atlas



We would also like to recognise funding support from the Canadian Institutes of Health Research (CIHR) and Fonds de la recherche en santé du Québec (FRQS). Core funding for ENIGMA was provided by the NIH Big Data to Knowledge (BD2K) program under consortium grant U54 EB020403, by the ENIGMA World Aging Center (R56 AG058854), and by the ENIGMA Sex Differences Initiative (R01 MH116147).

And finally, a many thanks to our wonderful enigmator collaborators
- **Casey Paquola**, *MICA Lab - Montreal Neurological Institute*
- **Jessica Royer**, *MICA Lab - Montreal Neurological Institute*
- **Bo-Yong Park**, *MICA Lab - Montreal Neurological Institute*
- **Oualid Benkarim**, *MICA Lab - Montreal Neurological Institute*
- **Raúl Rodríguez-Cruces**, *MICA Lab - Montreal Neurological Institute*
- **Nicole Eichert**, *MICA Lab - Montreal Neurological Institute*

CHAPTER

EIGHT

CORE DEVELOPERS

- **Sara Larivière**, *MICA Lab - Montreal Neurological Institute*
- **Boris Bernhardt**, *MICA Lab - Montreal Neurological Institute*

PYTHON MODULE INDEX

e

INDEX

S

W

Z