pynets Documentation

Release 1.1.2

The PyNets developers

Sep 05, 2021

Contents

1 PyNets® 1

1.1 About . . . 1
1.2 Documentation . . . 1
1.3 Citing . . . 1
1.4 Contents . . . 2
1.4.1 Installation . . . 2
1.4.2 Hardware Requirements . . . 2
1.4.3 Usage . . . 4
1.4.4 Credits . . . 12
1.4.5 Contributing . . . 13
1.4.6 API . . . 15

Bibliography 73

Index 79

CHAPTER 1

PyNets®

1.1 About

A Reproducible Workflow for Structural and Functional Connectome Ensemble Learning

PyNets leverages the Nipype workflow engine, along with Nilearn and Dipy fMRI and dMRI libraries, to sample individual structural and functional connectomes. Uniquely, PyNets enables the user to specify any of a variety of methodological choices (i.e. that impact node and/or edge definitions) and sampling the resulting connectome estimates in a massively scalable and parallel framework. PyNets is a post-processing workflow, which means that it can be run manually on virtually any preprocessed fMRI or dMRI data. Further, it can be deployed as a BIDS application that takes BIDS derivatives and makes BIDS derivatives. Docker and Singularity containers are further available to facilitate reproducibility of executions. Cloud computing with AWS batch and S3 is also supported.

1.2 Documentation

Official installation, user-guide, and API docs now live here: https://pynets.readthedocs.io/en/latest/

1.3 Citing

A manuscript is in preparation, but for now, please cite all uses with the following entry:

Pisner, D., Hammonds R. (2020) PyNets: A Reproducible Workflow for Structural and Functional Connectome Ensemble Learning. Poster session presented at the 26th Annual Meeting of the Organization for Human Brain Mapping. https://github.com/dPys/PyNets.

1.4 Contents

1.4.1 Installation

There are many ways to use PyNets®: in a *Docker Container* (page 2), in a *Singularity Container* (page 2), using AWS Batch, or in a Manually Prepared Environment (Python 3.6+). Using a local container method is highly recommended. Once you are ready to run pynets, see Usage for details.

1.4.2 Hardware Requirements

PyNets is designed for maximal scalability– it can be run on a supercomputer, but it can also be run on your laptop. Nevertheless, exploring a larger grid-space of the connectome "multiverse" can be accomplished faster and more easily on a supercomputer, even if optimization reveals that only one or a few connectome samples are needed.

With these considerations in mind, the minimal hardware required to run PyNets is 4 vCPUs, at least 8 GB of free RAM, and at least 15-20 GB of free disk space. However, the recommended hardware for ensemble sampling is 8+ vCPU's, 16+ GB of RAM, and 20+ GB of disk space (i.e. high-end desktops and laptops). On AWS and supercomputer clusters, PyNets hypothetically has infinite scalability– because it relies on a forkserver for multiprocessing, it will auto-optimize its concurrency based on the cores/ memory made available to it.

> **Note:** Another important ceiling to consider is I/O. Be sure that when you specify a safe working directory for the heavy metadata disk operations of PyNets. This can be set using the *-work* flag, and unless you have a really good reason, it should almost always be set to some variation of '/tmp'.

Docker Container

In order to run pynets in a Docker container, Docker must be installed. Once Docker is installed, you can simply pull a pre-built image from dockerhub as follows:

```
docker pull dpys/pynets:latest
```

or you can build a container yourself and test it interactively as follows:

```
docker build -t pynets .

docker run -ti --rm --privileged \
    --entrypoint /bin/bash
    -v '/tmp':'/tmp' \
    -v '/var/tmp':'/var/tmp' \
    -v '/input_files_local':'/inputs' \
    -v '/output_files_local':'/outputs' \
    pynets
```

See *External Dependencies* (page 4) for more information (e.g., specific versions) on what is included in the latest Docker images.

Singularity Container

For security reasons, many HPCs (e.g., TACC) do not allow Docker containers, but do allow Singularity containers.

Preparing a Singularity image (Singularity version >= 2.5)

If the version of Singularity on your HPC is modern enough you can create Singularity image directly on the HCP. This is as simple as:

```
singularity build /my_images/pynets-<version>.simg docker://dpys/pynets:<version>
```

Where <version> should be replaced with the desired version of PyNets that you want to download.

Preparing a Singularity image (Singularity version < 2.5)

In this case, start with a machine (e.g., your personal computer) with Docker installed. Use docker2singularity to create a singularity image. You will need an active internet connection and some time.

```
docker run --privileged -t --rm \
    -v '/var/run/docker.sock':'/var/run/docker.sock' \
    -v 'D:\host\path\where\to\output\singularity\image:/output' \
    singularityware/docker2singularity \
    dpys/pynets:<version>
```

Where <version> should be replaced with the desired version of PyNets that you want to download.

Beware of the back slashes, expected for Windows systems. For *nix users the command translates as follows:

```
docker run --privileged -t --rm \
    -V '/var/run/docker.sock':'/var/run/docker.sock' \
    -v '/absolute/path/to/output/folder':'/outputs' \
    singularityware/docker2singularity \
    dpys/pynets:<version>
```

Transfer the resulting Singularity image to the HPC, for example, using scp.

```
scp pynets*.img user@hcpserver.edu:/my_images
```

Manually Prepared Environment (Python 3.6+)

Warning: This method is not recommended! Make sure you would rather do this than use a *Docker Container* (page 2) or a *Singularity Container* (page 2).

Make sure all of pynets's *External Dependencies* (page 4) are installed. These tools must be installed and their binaries available in the system's $PATH. A relatively interpretable description of how your environment can be set-up is found in the Dockerfile.

On a functional Python 3.6 (or above) environment with pip installed, PyNets can be installed using the habitual command:

```
[sudo] pip install pynets [--user]
```

or

```
# Install git-lfs
brew install git-lfs (macOS) or [sudo] apt-get install git-lfs (linux)
git lfs install --skip-repo

# Clone the repository and install
git clone https://github.com/dpys/pynets
cd PyNets
[sudo] python setup.py install [--user]
```

External Dependencies

PyNets is written using Python 3.6 (or above), and is based on nipype.

PyNets requires some other neuroimaging software tools that are not handled by the Python's packaging system (Pypi) used to deploy the `pynets` package:

- FSL (version >=5.0.9). See https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FslInstallation

> **Note:** If you are using a debian/ubuntu OS, installing FSL can be installed using neurodebian:
>
> ```
> [sudo] curl -sSL http://neuro.debian.net/lists/stretch.us-tn.full >> /etc/
> ↪apt/sources.list.d/neurodebian.sources.list
> [sudo] apt-key add {path to PyNets base directory}/docker/files/
> ↪neurodebian.gpg
> [sudo] apt-key adv --refresh-keys --keyserver hkp://ha.pool.sks-
> ↪keyservers.net 0xA5D32F012649A5A9 || true
> [sudo] apt-get update
> [sudo] apt-get install -y fsl-core
> ```

1.4.3 Usage

The exact command to run `PyNets®` depends on several factors:

(1) The Installation method (i.e. pip, docker, singularity, git), along with the environment resources available for computing;

(2) The types and modalities of available data inputs;

(3) The execution objective (e.g. ensemble connectome sampling, unitary connectome sampling, plotting, graph-theory, embedding, optimization/benchmarking).

Required Inputs

Required

(A) An alphanumeric subject identifier must be specified with the *-id* flag. It can be a pre-existing label or an arbitrarily selected one, but it will be used by PyNets for naming of output directories. In the case of BIDS data, this should be *PARTICIPANT*'_'*SESSION*'_'*RUN* from sub-PARTICIPANT, ses-SESSION, run-RUN.

(B) A supported connectivity model specified with the *-mod* flag. If PyNets is executed in multimodal mode (i.e. with both fMRI and dMRI inputs in the same command-line call), multiple modality-applicable connectivity models should be specified (minimally providing at least one for either modality). PyNets will automatically parse which model is appropriate for which data.

(C) If an atlas is not specified with the *-a* flag, then a parcellation file must be specified with the *-a* flag. The following curated list of atlases is currently supported:

Atlas Library

- 'atlas_harvard_oxford'
- 'atlas_aal'
- 'atlas_destrieux_2009'
- 'atlas_talairach_gyrus'
- 'atlas_talairach_ba'
- 'atlas_talairach_lobe'
- 'coords_power_2011' (only valid when using the *-spheres* flag)
- 'coords_dosenbach_2010' (only valid when using the *-spheres* flag)
- 'atlas_msdl'
- 'atlas_pauli_2017'
- 'destrieux2009_rois'
- 'BrainnetomeAtlasFan2016'
- 'VoxelwiseParcellationt0515kLeadDBS'
- 'Juelichgmthr252mmEickhoff2005'
- 'CorticalAreaParcellationfromRestingStateCorrelationsGordon2014'
- 'AICHAreorderedJoliot2015'
- 'HarvardOxfordThr252mmWholeBrainMakris2006'
- 'VoxelwiseParcellationt058kLeadDBS'
- 'MICCAI2012MultiAtlasLabelingWorkshopandChallengeNeuromorphometrics'
- 'Hammers_mithAtlasn30r83Hammers2003Gousias2008'
- 'AALTzourioMazoyer2002'
- 'DesikanKlein2012'
- 'AAL2zourioMazoyer2002'
- 'VoxelwiseParcellationt0435kLeadDBS'
- 'AICHAJoliot2015'
- 'whole_brain_cluster_labels_PCA100'
- 'whole_brain_cluster_labels_PCA200'
- 'RandomParcellationsc05meanalll43Craddock2011'

(D) A set of brain image files. *PyNets* is a post-processing workflow which means that input files should already be preprocessed. Minimally, all DWI, BOLD, and T1W image inputs should be **motion-corrected** (and ideally also susceptibility-corrected + denoised).

anat The T1w can be preprocessed using any method, but should be in its native scanner anatomical space.

func A BOLD/EPI series can be preprocessed using any method, but should in the same scanner anatomical space as the T1w (i.e. coregistered to the T1w anat and not yet normalized to a standard-space template since PyNets must do this in order that it can accurately map parcellations to individual subject anatomy).

dwi A DWI series should ideally be in its native diffusion MRI (dMRI) space (though can also be co-registered to the T1w image) and must contain at least one B0 for reference. If *-dwi* is specified, then *-bvec* and *-bval* must also be. Note that the choice of models specified with *-mod* also depends on the sampling scheme of your dwi data (e.g. CSD will likely overfit your data in the case of too few directional volumes).

Note: Native-space DWI images are preferred for several reasons. Even when rigidly applied, intermodal registration of the diffusion signal to T1-weighted space, for instance, which has considerably different white-matter/grey-matter signal contrast (and lower specificity for the former), will inevitably result in some degree of spatial misalignment and signal loss. Note that this is unlike the case of BOLD EPI – an inherently noisy, temporal (i.e. non-structural) modality – which benefits from being co-registered to T1w images of significantly higher spatial resolution, particularly in grey-matter tissue where BOLD signal is typically observed. To ensure minimal within-subject variance and maximal between-subject variance as a function of numerous hyperparameters used to sample connectome ensembles with PyNets, input DWI data should ideally carry maximal SNR and have undergone the least amount of resampling necessary (e.g. minimally eddy/motion correction).

-g A path to a raw graph can alternatively be specified, in which case the initial stages of the pipeline will be skipped. In this case, the graph should be in .txt, .npy, .csv, .tsv, or .ssv format.

Note: Prior normalization of the *anat*, *func*, or *dwi* inputs to PyNets is not (yet) supported. This is because PyNets relies on the inverse transform from an MNI-template to conform a template-resampled version of the atlas(es) specified (i.e. to define nodes) into native T1w anatomical space. PyNets uses the MNI152 template by default to accomplish this, but you can specify alternative templates in the runconfig.yml advanced settings to override MNI152 (e.g. a Pediatric template), following the naming spec of *templateflow* (See: <https://github.com/templateflow/templateflow>).

Note: If you preprocessed your BOLD data using fMRIprep, then you will need to have specified either *T1w* or *anat* in the list of fmriprep *–output-spaces*.

Note: Input image orientation and voxel resolution are not relevant, as PyNets will create necessary working copies with standardized RAS+ orientations and either 1mm or 2mm voxel resolution reslicing, depending on the runconfig.yml default or resolution override using the *-vox* flag.

Note: All file formats are assumed to be Nifti1Image (i.e. .nii or .nii.gz file suffix), and **absolute** file paths should always be specified to the CLI's.

Note: Tissue segmentations are calculated automatically in PyNets using FAST, but if you are

using the *pynets_bids* CLI on preprocessed BIDS derivatives containing existing segmentations, pynets will alternatively attempt to autodetect and use those.

Custom File Inputs

-m (*fMRI* + *dMRI*) A binarized brain mask of the T1w image in its native anatomical space. Input images need not be skull-stripped. If brain masking has been applied already, *PyNets* will attempt to detect this, else it will attempt to extract automatically using a deep-learning classifier. See [deepbrain]<https://github.com/iitzco/deepbrain> for more information.

-roi (*fMRI* + *dMRI*) A binarized ROI mask used to constrain connectome node-making to restricted brain regions of the parcellation being used. ROI inputs should be in MNI space.

-a (*fMRI* + *dMRI*) A parcellation/atlas image (in MNI space) used to define nodes of a connectome. Labels should be spatially distinct across hemispheres and ordered with consecutive integers with a value of 0 as the background label. This flag can uniquely be listed with multiple, space-separated file inputs.

-ref (*fMRI* + *dMRI*) An atlas reference .txt file that indices intensities corresponding to atlas labels of the parcellation specified with the *-a* flag. This label map is used only to delineate node labels manually. Otherwise, PyNets will attempt to perform automated node labeling via AAL, else sequential numeric labels will be used.

-way (*dMRI*) A binarized white-matter ROI mask (in MNI template space) used to constrain tractography in native diffusion space such that streamlines are retained only if they pass within the vicinity of the mask. Like with ROI inputs, waymasks should be in MNI space.

-cm (*fMRI*) A binarized ROI mask used to spatially-constrained clustering during parcellation-making. Note that if this flag is used, *-k* and *-ct* must also be included. Like with ROI inputs, clustering masks should be in MNI space.

-conf (*fMRI*) An additional noise confound regressor file for extracting a cleaner time-series.

Multimodal Workflow Variations

In the case of running pynets on a single subject, several combinations of input files can be used:

fMRI Connectometry *-func*, *-anat*, (*-conf*), (*-roi*), (*-m*), (*-cm*)

dMRI Connectometry *-dwi*, *-bval*, *-bvec*, *-anat*, (*-roi*), (*-m*), (*-way*)

dMRI + fMRI Multiplex Connectometry All of the above required flags should be included simultaneously. Note that in this case, *-anat* only needs to be specified once.

Raw Graph Connectometry (i.e. for graph analysis/embedding only) *-g*

Command-Line Arguments

Quickstart

Execution on BIDS derivative datasets using the *pynets_bids* CLI

PyNets now includes an API for running single-subject and group workflows on BIDS derivatives (e.g. produced using popular BIDS apps like fmriprep/cpac and dmriprep/qsiprep). In this scenarioo, the input dataset should follow the derivatives specification of the *BIDS (Brain Imaging Data Structure)* format (<https://bids-specification.

readthedocs.io/en/derivatives/05-derivatives/01-introduction.html>), which must include at least one subject's fMRI image or dMRI image (in T1w space), along with a T1w anatomical image.

The *runconfig.yml* file in the base directory includes parameter presets, but all file input options that are included with the *pynets* cli are also exposed to the *pynets_bids* cli.

The common parts of the command follow the BIDS-Apps definition. Example:

```
pynets_bids '/hnu/fMRIprep/fmriprep' '/Users/dPys/outputs/pynets' participant func --
↪participant_label 0025427 0025428 --session_label 1 2 3 -config pynets/config/bids_
↪config.json
```

A similar CLI, *pynets_cloud* has also been made available using AWS Batch and S3, which require a AWS credentials and configuration of job queues and definitions using cloud_config.json:

```
pynets_cloud --bucket 'hnu' --dataset 'HNU' participant func --participant_label␣
↪0025427 --session_label 1 --push_location 's3://hnu/outputs' --jobdir '/Users/
↪derekpisner/.pynets/jobs' -cm 's3://hnu/HNU/masks/MyClusteringROI.nii.gz' -pm '30,
↪110'
```

Manual Execution Using the *pynets* CLI

You have a preprocessed EPI bold dataset from the first session for subject 002, and you wish to analyze a whole-brain network using 'sub-colin27_label-L2018_desc-scale1_atlas', thresholding the connectivity graph proportionally to retain 0.20% of the strongest connections, and you wish to use partial correlation model estimation:

```
pynets -id '002_1' '/Users/dPys/outputs/pynets' \
-func '/Users/dPys/PyNets/tests/examples/sub-002/ses-1/func/BOLD_PREPROCESSED_IN_ANAT_
↪NATIVE.nii.gz' \ # The fMRI BOLD image data.
-anat '/Users/dPys/PyNets/tests/examples/sub-002/ses-1/anat/ANAT_PREPROCESSED_NATIVE.
↪nii.gz' \ # The T1w anatomical image.
-a 'sub-colin27_label-L2018_desc-scale1_atlas' \ # Lausanne parcellation at scale=1.
-mod 'partcorr' \ # The connectivity model.
-thr 0.20 \ # A single proportional threshold to apply post-hoc.
```

Building upon the previous example, let's say you now wish to analyze the Default network for this same subject's data, but based on the 95-node atlas parcellation scheme from Desikan-Klein 2012 called 'DesikanKlein2012' and the Brainnetome Atlas from Fan 2016 called 'BrainnetomeAtlasFan2016', you wish to threshold the graph to achieve a target density of 0.3, and you wish to fit a sparse inverse covariance model in addition to partial correlation, and you wish to plot the results:

```
pynets -id '002_1' '/Users/dPys/outputs/pynets' \
-func '/Users/dPys/PyNets/tests/examples/sub-002/ses-1/func/BOLD_PREPROCESSED_IN_ANAT_
↪NATIVE.nii.gz' \ # The fMRI BOLD image data.
-anat '/Users/dPys/PyNets/tests/examples/sub-002/ses-1/anat/ANAT_PREPROCESSED_NATIVE.
↪nii.gz' \ # The T1w anatomical image.
-a 'DesikanKlein2012' 'BrainnetomeAtlasFan2016' # Multiple spherical atlases.
-mod 'partcorr' 'sps' \ # The connectivity models.
-dt -thr 0.3 \ # The thresholding settings.
-n 'Default' \ # The resting-state network definition to restrict node-making from␣
↪each of the input atlas.
-plt # Activate plotting.
```

Building upon the previous examples, let's say you now wish to analyze the Default and Executive Control Networks for this subject, but this time based on a custom atlas (DesikanKlein2012.nii.gz), this time defining your nodes as parcels (as opposed to spheres), you wish to fit a partial correlation model, you wish to iterate the pipeline over a range of densities (i.e. 0.05-0.10 with 1% step), and you wish to prune disconnected nodes:

```
pynets -id '002_1' '/Users/dPys/outputs/pynets' \
-func '/Users/dPys/PyNets/tests/examples/sub-002/ses-1/func/BOLD_PREPROCESSED_IN_ANAT_
↪NATIVE.nii.gz' \ # The fMRI BOLD image data.
-anat '/Users/dPys/PyNets/tests/examples/sub-002/ses-1/anat/ANAT_PREPROCESSED_NATIVE.
↪nii.gz' \ # The T1w anatomical image.
-a '/Users/dPys/PyNets/pynets/atlases/MyCustomAtlas.nii.gz' \ # A user-supplied atlas␣
↪parcellation.
-mod 'partcorr' \ # The connectivity model.
-dt -min_thr 0.05 -max_thr 0.10 -step_thr 0.01 -p 1 \ # The thresholding settings.
-n 'Default' 'Cont' # The resting-state network definitions to restrict node-making␣
↪from each of the input atlas.
```

Note: In general, parcels are preferable to spheres as nodes because parcels more closely respect cortical topographgy.

Building upon the previous examples, let's say you now wish to create a subject-specific atlas based on the subject's unique spatial-temporal profile. In this case, you can specify the path to a binarized mask within which to performed spatially-constrained spectral clustering, and you want to try this at multiple resolutions of k clusters/nodes (i.e. k=50,100,150). You again also wish to define your nodes spherically with radii at both 2 and 4 mm, fitting a partial correlation and sparse inverse covariance model, you wish to iterate the pipeline over a range of densities (i.e. 0.05-0.10 with 1% step), you wish to prune disconnected nodes, and you wish to plot your results:

```
pynets -id '002_1' '/Users/dPys/outputs/pynets' \
-func '/Users/dPys/PyNets/tests/examples/sub-002/ses-1/func/BOLD_PREPROCESSED_IN_ANAT_
↪NATIVE.nii.gz' \ # The fMRI BOLD image data.
-anat '/Users/dPys/PyNets/tests/examples/sub-002/ses-1/anat/ANAT_PREPROCESSED_NATIVE.
↪nii.gz' \ # The T1w anatomical image.
-mod 'partcorr' 'sps' \ # The connectivity models.
-cm '/Users/dPys/PyNets/tests/examples/MyClusteringROI.nii.gz' -k 50 100 150 -ct 'ward
↪' \ # Node-making specification with spatially-constrained clustering.
-dt -min_thr 0.05 -max_thr 0.10 -step_thr 0.01 -p 1 \ # The thresholding settings.
-plt # Activate plotting.
```

You wish to generate a structural connectome, using deterministic and probabilistic ensemble tractography, based on both constrained-spherical deconvolution (CSD), Constant Solid Angle (CSA), and Sparse Fascicle (SFM) models. You wish to use atlas parcels as defined by both DesikanKlein2012, and AALTzourioMazoyer2002, exploring only those nodes belonging to the Default Mode Network, iterate over a range of graph densities (i.e. 0.05-0.10 with 1% step), and prune disconnected nodes:

```
pynets -id '002_1' '/Users/dPys/outputs/pynets' \
-dwi '/Users/dPys/PyNets/tests/examples/sub-002/ses-1/dwi/DWI_PREPROCESSED_NATIVE.nii.
↪gz' \ # The dMRI diffusion-weighted image data.
-bval '/Users/dPys/PyNets/tests/examples/sub-002/ses-1/dwi/BVAL.bval' \ # The b-
↪values.
-bvec '/Users/dPys/PyNets/tests/examples/sub-002/ses-1/dwi/BVEC.bvec' \ # The b-
↪vectors.
-anat '/Users/dPys/PyNets/tests/examples/sub-002/ses-1/anat/ANAT_PREPROCESSED_NATIVE.
↪nii.gz' \ # The T1w anatomical image.
-a '/Users/dPys/.atlases/DesikanKlein2012.nii.gz' '/Users/dPys/.atlases/
↪AALTzourioMazoyer2002.nii.gz' \ # The atlases.
-mod 'csd' 'csa' 'sfm' \ # The connectivity model.
-dg 'prob' 'det'  \ # The tractography settings.
-dt -min_thr 0.05 -max_thr 0.10 -step_thr 0.01 -p 1 \ # The thresholding settings.
-n 'Default' # The resting-state network definition to restrict node-making from each␣
↪of the input atlases.
```

Note: Spherical nodes can be used by triggering the *-spheres* flag, and for some coordinate-based atlases like coords_power_2011 or coords_dosenbach_2010, only spheres are possible, but in general parcel volumes should be used as the default.

Note: Iterable sampling parameters specified at runtime should always be space-delimited.

There are many other runtime options than these examples demonstrate. To explore all of the possible hyper-parameter combinations that pynets has to offer, see *pynets -h*. A full set of tutorials and python notebooks are coming soon.

Docker and AWS

PyNets includes an API for running *pynets_bids* or *pynets* in a Docker container as well as using AWS Batch. The latter assumes a dataset with BIDS derivatives is stored in an S3 bucket. Docker Example:

```
docker run -ti --rm --privileged -v '/home/dPys/.aws/credentials:/home/neuro/.aws/
↪credentials' dpys/pynets:latest pynets_bids 's3://hnu/HNU' '/outputs' participant␣
↪func --participant_label 0025427 --session_label 1 -plug 'MultiProc' -pm '8,12' -
↪work '/working' -config pynets/config/bids_config.json
```

Running a Singularity Image

If the data to be preprocessed is also on an HPC server, you are ready to run pynets, either manually or as a BIDS application. For example, where PARTICIPANT is a subject identifier and SESSION is a given scan session, we could sample an ensemble of connectomes manually as follows

```
singularity exec -w \
 '/scratch/04171/dPys/pynets_singularity_latest-2020-02-07-eccf145ea766.img' \
 pynets /outputs \
 -p 1 -mod 'partcorr' 'corr' -min_thr 0.20 -max_thr 1.00 -step_thr 0.10 -sm 0 2 4 -hp␣
↪0 0.028 0.080
 -ct 'ward' -k 100 200 -cm '/working/MyClusteringROI.nii.gz' \
 -pm '24,48' \
 -norm 6 \
 -anat '/inputs/sub-PARTICIPANT/ses-SESSION/anat/sub-PARTICIPANT_space-anat_desc-
↪preproc_T1w_brain.nii.gz' \
 -func '/inputs/sub-PARTICIPANT/ses-SESSION/func/sub-PARTICIPANT_ses-SESSION_task-
↪rest_space-anat_desc-smoothAROMAnonaggr_bold_masked.nii.gz' \
 -conf '/inputs/sub-PARTICIPANT/ses-SESSION/func/sub-PARTICIPANT_ses-SESSION_task-
↪rest_desc-confounds_regressors.tsv' \
 -m '/inputs/sub-PARTICIPANT/ses-SESSION/func/sub-PARTICIPANT_ses-SESSION_task-rest_
↪space-anat_desc-brain_mask.nii.gz' \
 -id 'PARTICIPANT_SESSION' -plug 'MultiProc' -work '/tmp'
```

Note: Singularity by default exposes all environment variables from the host inside the container. Because of this your host libraries (such as nipype) could be accidentally used instead of the ones inside the container - if they are included in `PYTHONPATH`. To avoid such situation we sometimes recommend using the `--cleanenv` singularity flag in production use. For example:

```
singularity exec --cleanenv --no-home_clust_est '/scratch/04171/dPys/pynets_latest-
↪2016-12-04-5b74ad9a4c4d.img' \
  pynets /outputs \
  -p 1 -mod 'partcorr' 'corr' -min_thr 0.20 -max_thr 1.00 -step_thr 0.10 -sm 0 2 4 -
↪hp 0 0.028 0.080
  -ct 'ward' -k 100 200 -cm '/working/MyClusteringROI.nii.gz' \
  -norm 6 \
  -anat '/inputs/sub-PARTICIPANT/ses-SESSION/anat/sub-PARTICIPANT_space-anat_desc-
↪preproc_T1w_brain.nii.gz' \
  -func '/inputs/sub-PARTICIPANT/ses-SESSION/func/sub-PARTICIPANT_ses-SESSION_task-
↪rest_space-anat_desc-smoothAROMAnonaggr_bold_masked.nii.gz' \
  -conf '/inputs/sub-PARTICIPANT/ses-SESSION/func/sub-PARTICIPANT_ses-SESSION_task-
↪rest_desc-confounds_regressors.tsv' \
  -m '/inputs/sub-PARTICIPANT/ses-SESSION/func/sub-PARTICIPANT_ses-SESSION_task-rest_
↪space-anat_desc-brain_mask.nii.gz' \
  -id 'PARTICIPANT_SESSION' -plug 'MultiProc' -work '/tmp' -pm '24,48'
```

or, unset the `PYTHONPATH` variable before running:

```
unset PYTHONPATH; singularity exec /scratch/04171/dPys/pynets_latest-2016-12-04-
↪5b74ad9a4c4d.img \
  pynets /outputs \
  -p 1 -mod 'partcorr' 'corr' -min_thr 0.20 -max_thr 1.00 -step_thr 0.10 -sm 0 2 4 -
↪hp 0 0.028 0.080
  -ct 'ward' -cm '/working/MyClusteringROI.nii.gz' -k 100 200 \
  -norm 6 \
  -anat '/inputs/sub-PARTICIPANT/ses-SESSION/anat/sub-PARTICIPANT_space-anat_desc-
↪preproc_T1w_brain.nii.gz' \
  -func '/inputs/sub-PARTICIPANT/ses-SESSION/func/sub-PARTICIPANT_ses-SESSION_task-
↪rest_space-anat_desc-smoothAROMAnonaggr_bold_masked.nii.gz' \
  -conf '/inputs/sub-PARTICIPANT/ses-SESSION/func/sub-PARTICIPANT_ses-SESSION_task-
↪rest_desc-confounds_regressors.tsv' \
  -m '/inputs/sub-PARTICIPANT/ses-SESSION/func/sub-PARTICIPANT_ses-SESSION_task-rest_
↪space-anat_desc-brain_mask.nii.gz' \
  -id 'PARTICIPANT_SESSION' -plug 'MultiProc' -work '/tmp' -pm '24,48'
```

Note: Depending on how Singularity is configured on your cluster it might or might not automatically bind (mount or expose) host folders to the container. If this is not done automatically you will need to bind the necessary folders using the `-B <host_folder>:<container_folder>` Singularity argument. For example:

```
singularity exec_clust_est -B /work:/work /scratch/04171/dPys/pynets_latest-2016-12-
↪04-5b74ad9a4c4d.img \
  -B '/scratch/04171/dPys/pynets_out:/inputs,/scratch/04171/dPys/masks/PARTICIPANT_
↪triple_network_masks_SESSION':'/outputs' \
  pynets /outputs \
  -p 1 -mod 'partcorr' 'corr' -min_thr 0.20 -max_thr 1.00 -step_thr 0.10 -sm 0 2 4 -
↪hp 0 0.028 0.080 \
  -ct 'ward' -k 100 200 -cm '/working/MyClusteringROI.nii.gz' \
  -norm 6 \
  -anat '/inputs/sub-PARTICIPANT/ses-SESSION/anat/sub-PARTICIPANT_space-anat_desc-
↪preproc_T1w_brain.nii.gz' \
  -func '/inputs/sub-PARTICIPANT/ses-SESSION/func/sub-PARTICIPANT_ses-SESSION_task-
↪rest_space-anat_desc-smoothAROMAnonaggr_bold_masked.nii.gz' \
  -conf '/inputs/sub-PARTICIPANT/ses-SESSION/func/sub-PARTICIPANT_ses-SESSION_task-
↪rest_desc-confounds_regressors.tsv' \
```

(continues on next page)

(continued from previous page)

```
  -m '/inputs/sub-PARTICIPANT/ses-SESSION/func/sub-PARTICIPANT_ses-SESSION_task-rest_
↪space-anat_desc-brain_mask.nii.gz' \
  -id 'PARTICIPANT_SESSION' -plug 'MultiProc' -work '/tmp'  -pm '24,48'
```

Debugging

Logs and crashfiles are outputted into the `<working dir>/Wf_single_subject_<id>` directory. To include verbose debugging and resource benchmarking, run pynets with the *-v* flag.

Support and communication

The documentation of this project is found here: http://pynets.readthedocs.org/en/latest/.

All bugs, concerns and enhancement requests for this software can be submitted here: https://github.com/dPys/PyNets/issues.

If you have a problem or would like to ask a question about how to use `pynets`, please submit a question to NeuroStars.org with an `pynets` tag. NeuroStars.org is a platform similar to StackOverflow but dedicated to neuroinformatics.

All previous `pynets` questions are available here: http://neurostars.org/tags/pynets/

To participate in the `pynets` development-related discussions please use the following mailing list: http://mail.python.org/mailman/listinfo/neuroimaging Please add *[pynets]* to the subject line when posting on the mailing list.

Not running on a local machine? - Data transfer

If you intend to run `pynets` on a remote system, you will need to make your data available within that system first.

Alternatively, more comprehensive solutions such as Datalad will handle data transfers with the appropriate settings and commands. Datalad also performs version control over your data.

Interpreting outputs

To collect the graph topological outputs from one or more completed pynets runs, you can use the *pynets_collect* CLI:

```
pynets_collect -basedir '/Users/dPys/outputs/pynets' -modality 'func'
```

which will generate a group summary dataframe in *basedir*, all_subs_neat.csv, where each row is a given subject session and/or run, and each column is a graph topological metric the was calculated, with the prefix indicating correspondence to a given connectome ensemble of interest.

1.4.4 Credits

Development Lead

Derek Pisner <dpisner@utexas.edu>

Contributors

Ryan Hammonds, Aki Nikolaidis, Kamil Bonna, Miriam Kosik, Karolina Finc, James Kunert-Graf, Mathias Goncalves, Andrew Reineberg, Charles Laidi, Josh Faskowitz, Cameron Craddock, Jaewon Chung

1.4.5 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/dpys/pynets/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

PyNets could always use more documentation, whether as part of the official PyNets docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/dpys/pynets/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *pynets* for local development.

1. Fork the *pynets* repo on GitHub.
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/pynets.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
mkvirtualenv pynets
cd pynets/
python setup.py develop
```

4. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
flake8 pynets tests
python setup.py test or py.test
tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6, 3.7, and for PyPy. Check https://travis-ci.org/dpys/pynets/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests

```
py.test tests.test_pynets
```

1.4.6 API

pynets.core package

Subpackages

pynets.core.atlases package

Module contents

pynets.core.labelcharts package

Module contents

Submodules

pynets.core.cloud_utils module

pynets.core.nodemaker module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017 @author: Derek Pisner

pynets.core.nodemaker.**VoxTomm**(*img_affine*, *voxcoords*)
Function to convert a list of voxel coordinates to mm coordinates.

Parameters

img_affine [array] 4 x 4 2D Numpy array that is the affine of the image space that the coordinates inhabit.

voxcoords [list] List of [x, y, z] or (x, y, z) coordinates in voxel-space.

pynets.core.nodemaker.**coords_masker**(*roi*, *coords*, *labels*, *error*, *vox_size='2mm'*)
Evaluate the affinity of any arbitrary list of coordinate nodes for a user-specified ROI mask.

Parameters

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

coords [list] List of (x, y, z) tuples in mm-space corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

labels [list] List of string labels corresponding to ROI nodes.

error [int] Rounded euclidean distance, in units of voxel number, to use as a spatial error cushion in the case of evaluating the spatial affinity of a given list of coordinates to the given ROI mask.

Returns

coords [list] Filtered list of (x, y, z) tuples in mm-space with a spatial affinity for the specified ROI mask.

labels [list] Filtered list of string labels corresponding to ROI nodes with a spatial affinity for the specified ROI mask.

pynets.core.nodemaker.**create_parcel_atlas**(*parcel_list*, *label_intensities=None*)
Create a 3D Nifti1Image atlas parcellation of consecutive integer intensities from an input list of ROI's.

Parameters

parcel_list [list] List of 3D boolean numpy arrays or binarized Nifti1Images corresponding to ROI masks.

Returns

net_parcels_map_nifti [Nifti1Image] A nibabel-based nifti image consisting of a 3D array with integer voxel intensities corresponding to ROI membership.

parcel_list_exp [list] List of 3D boolean numpy arrays or binarized Nifti1Images corresponding to ROI masks, prepended with a background image of zeros.

pynets.core.nodemaker.**create_spherical_roi_volumes**(*node_size*, *coords*, *template_mask*)

Create volume ROI mask of spheres from a given set of coordinates and radius.

Parameters

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's for tracking.

coords [list] List of (x, y, z) tuples in mm-space corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

template_mask [str] Path to binarized version of standard (MNI)-space template Nifti1Image file.

Returns

parcel_list [list]

List of 3D boolean numpy arrays or binarized Nifti1Images corresponding to ROI masks.

par_max [int] The maximum label intensity in the parcellation image.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's for tracking.

parc [bool] Indicates whether to use the raw parcels as ROI nodes instead of coordinates at their center-of-mass.

pynets.core.nodemaker.**drop_badixs_from_parcellation**(*uatlas*, *bad_idxs*, *enf_hemi=True*)

pynets.core.nodemaker.**drop_coords_labels_from_restricted_parcellation**(*parcellation*, *coords*, *labels*)

pynets.core.nodemaker.**enforce_hem_distinct_consecutive_labels**(*uatlas*, *label_names=None*, *background_label=0*)

Check for hemispherically distinct and consecutive labels and rebuild parcellation.

Parameters

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

label_names [list] List of string label names corresponding to ROI nodes.

Returns

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

label_names [list] List of string label names corresponding to ROI nodes.

pynets.core.nodemaker.**fetch_nilearn_atlas_coords**(*atlas*)

Meta-API for nilearn's coordinate atlas fetching API to retrieve any publically-available coordinate atlas by string name.

Parameters

atlas [str] Name of a Nilearn-hosted coordinate atlas supported for fetching. See Nilearn's datasets.atlas module for more detailed reference.

Returns

coords [list] List of (x, y, z) tuples corresponding to a coordinate atlas.

atlas_name [str] Name of atlas parcellation (can differ slightly from fetch API string).

networks_list [list] List of RSN's and their associated cooordinates, if predefined uniquely for a given atlas.

labels [list] List of string labels corresponding to atlas nodes.

pynets.core.nodemaker.**gen_img_list**(*uatlas*)

Return list of boolean nifti masks where each masks corresponds to a unique atlas label for the provided atlas parcellation. Path string to Nifti1Image is input.

Parameters

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

Returns

img_list [Iterator of NiftiImages] List of binarized Nifti1Images corresponding to ROI masks for each unique atlas label.

pynets.core.nodemaker.**gen_network_parcels**(*uatlas*, *network*, *labels*, *dir_path*)

Return a modified verion of an atlas parcellation label, where labels have been filtered based on their spatial affinity for a specified RSN definition.

Parameters

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

network [str]

Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

labels [list] List of string labels corresponding to ROI nodes.

dir_path [str] Path to directory containing subject derivative data for given run.

Returns

out_path [str] File path to a new, RSN-filtered atlas parcellation Nifti1Image.

pynets.core.nodemaker.**get_brainnetome_node_attributes**(*node_files*, *emb_shape*)

pynets.core.nodemaker.**get_names_and_coords_of_parcels**(*uatlas*, *background_label=0*)

Return list of coordinates and max label intensity for a 3D atlas parcellation image.

Parameters

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

Returns

coords [list] List of (x, y, z) tuples corresponding to the center-of-mass of each parcellation node.

atlas [str] An arbitrary identified for the atlas based on the filename.

par_max [int] The maximum label intensity in the parcellation image.

label_intensities [list] A list of integer label intensity values from the parcellation.

pynets.core.nodemaker.**get_node_membership**(*network*, *infile*, *coords*, *labels*, *parc*, *parcel_list*, *perc_overlap=0.75*, *error=4*)

Evaluate the affinity of any arbitrary list of coordinate or parcel nodes for a user-specified RSN based on Yeo-7 or Yeo-17 definitions.

Parameters

network [str]

Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

infile [str] File path to Nifti1Image object whose affine will provide resampling reference for evaluation spatial proximity. Typically, this is an MNI-space template image.

coords [list] List of (x, y, z) tuples in mm-space corresponding to a coordinate atlas used or which represent the center-of-mass of each

parcellation node.

labels [list] List of string labels corresponding to ROI nodes.

parc [bool] Indicates whether to use parcels instead of coordinates as ROI nodes.

parcel_list [list] List of 3D boolean numpy arrays or binarized Nifti1Images corresponding to ROI masks.

perc_overlap [float] Value 0-1 indicating a threshold of spatial overlap to use as a spatial error cushion in the case of evaluating RSN membership from a given list of parcel masks. Default is 0.75.

error [int] Rounded euclidean distance, in units of voxel number, to use as a spatial error cushion in the case of evaluating RSN membership from a given list of coordinates. Default is 4.

Returns

coords_mm [list]

Filtered list of (x, y, z) tuples in mm-space with a spatial affinity for the specified RSN.

RSN_parcels [list]

Filtered list of 3D boolean numpy arrays or binarized Nifti1Images corresponding to ROI masks with a spatial affinity for the specified RSN.

net_labels [list] Filtered list of string labels corresponding to ROI nodes with a spatial affinity for the specified RSN.

network [str]

Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

References

[1], [2]

pynets.core.nodemaker.**get_sphere**(*coords*, *r*, *vox_dims*, *dims*)
Return all points within r mm of coords. Generates a cube and then discards all points outside sphere.

Parameters

coords [list] List of (x, y, z) tuples corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

r [int] Radius for sphere.

vox_dims [array/tuple] 1D vector (x, y, z) of mm voxel resolution for sphere.

dims [array/tuple] 1D vector (x, y, z) of image dimensions for sphere.

Returns

neighbors [list] A list of indices, within the dimensions of the image, that fall within a spherical neighborhood defined by the specified error radius of the list of the input coordinates.

References

[1]

pynets.core.nodemaker.**mask_roi**(*dir_path*, *roi*, *mask*, *img_file*)
Create derivative ROI based on intersection of roi and brain mask.

Parameters

dir_path [str] Path to directory containing subject derivative data for given run.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

mask [str] Path to binarized/boolean brain mask Nifti1Image file.

img_file [str] File path to Nifti1Image to use to generate an epi-mask.

Returns

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file, reduced to the spatial intersection with the input brain mask.

pynets.core.nodemaker.**mmToVox**(*img_affine*, *mmcoords*)
Function to convert a list of mm coordinates to voxel coordinates.

Parameters

img_affine [array] 4 x 4 2D Numpy array that is the affine of the image space that the coordinates inhabit.

mmcoords [list] List of [x, y, z] or (x, y, z) coordinates in mm-space.

pynets.core.nodemaker.**nilearn_atlas_helper**(*atlas*, *parc*)
Meta-API for nilearn's parcellation-based atlas fetching API to retrieve any publically-available parcellation-based atlas by string name.

Parameters

atlas [str] Name of a Nilearn-hosted parcellation/label-based atlas supported for fetching. See Nilearn's datasets.atlas module for more detailed references.

parc [bool] Indicates whether to use the raw parcels as ROI nodes instead of coordinates at their center-of-mass.

Returns

labels [list] List of string labels corresponding to atlas nodes.

networks_list [list] List of RSN's and their associated cooordinates, if predefined uniquely for a given atlas.

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

pynets.core.nodemaker.**node_gen**(*coords*, *parcel_list*, *labels*, *dir_path*, *ID*, *parc*, *atlas*, *uatlas*)

In the case that masking was not applied, this function generate nodes based on atlas definitions established by fetch_nodes_and_labels.

Parameters

coords [list] List of (x, y, z) tuples in mm-space corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

parcel_list [list]

List of 3D boolean numpy arrays or binarized Nifti1Images corresponding to ROI masks.

labels [list] List of string labels corresponding to ROI nodes.

dir_path [str] Path to directory containing subject derivative data for given run.

ID [str] A subject id or other unique identifier.

parc [bool] Indicates whether to use parcels instead of coordinates as ROI nodes.

atlas [str] Name of a Nilearn-hosted coordinate or parcellation/label-based atlas supported for fetching. See Nilearn's datasets.atlas module for more detailed reference.

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

Returns

net_parcels_map_nifti [Nifti1Image]

A nibabel-based nifti image consisting of a 3D array with integer voxel intensities corresponding to ROI membership.

coords [list] List of (x, y, z) tuples in mm-space corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

labels [list] List of string labels corresponding to ROI nodes.

atlas [str] Name of a Nilearn-hosted coordinate or parcellation/label-based atlas supported for fetching. See Nilearn's datasets.atlas module for more detailed reference.

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

dir_path [str] Path to directory containing subject derivative data for given run.

pynets.core.nodemaker.**node_gen_masking**(*roi*, *coords*, *parcel_list*, *labels*, *dir_path*, *ID*, *parc*, *atlas*, *uatlas*, *vox_size*, *perc_overlap=0.1*, *error=2*)

In the case that masking was applied, this function generate nodes based on atlas definitions established by fetch_nodes_and_labels.

Parameters

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

coords [list] List of (x, y, z) tuples in mm-space corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

parcel_list [list]

List of 3D boolean numpy arrays or binarized Nifti1Images corresponding to ROI masks.

labels [list] List of string labels corresponding to ROI nodes.

dir_path [str] Path to directory containing subject derivative data for given run.

ID [str] A subject id or other unique identifier.

parc [bool] Indicates whether to use parcels instead of coordinates as ROI nodes.

atlas [str] Name of a Nilearn-hosted coordinate or parcellation/label-based atlas supported for fetching. See Nilearn's datasets.atlas module for more detailed reference.

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

perc_overlap [float]

Value 0-1 indicating a threshold of spatial overlap to use as a spatial error cushion in the case of evaluating mask/RSN membership from a given list of parcel masks. Default is 0.75.

error [int] Rounded euclidean distance, in units of voxel number, to use as a spatial error cushion in the case of evaluating mask/RSN membership from a given list of coordinates. Default is 4.

Returns

net_parcels_map_nifti [Nifti1Image] A nibabel-based nifti image consisting of a 3D array with integer voxel intensities corresponding to ROI membership.

coords [list] List of (x, y, z) tuples in mm-space corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

labels [list] List of string labels corresponding to ROI nodes.

atlas [str] Name of a Nilearn-hosted coordinate or parcellation/label-based atlas supported for fetching. See Nilearn's datasets.atlas module for more detailed reference.

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

dir_path [str] Path to directory containing subject derivative data for given run.

pynets.core.nodemaker.**parcel_masker**(*roi*, *coords*, *parcel_list*, *labels*, *dir_path*, *ID*, *perc_overlap*, *vox_size*)

Evaluate the affinity of any arbitrary list of parcel nodes for a user-specified ROI mask.

Parameters

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

coords [list] List of (x, y, z) tuples in mm-space corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

parcel_list [list]

List of 3D boolean numpy arrays or binarized Nifti1Images corresponding to ROI masks.

labels [list] List of string labels corresponding to ROI nodes.

dir_path [str] Path to directory containing subject derivative data for given run.

ID [str] A subject id or other unique identifier.

perc_overlap [float] Value 0-1 indicating a threshold of spatial overlap to use as a spatial error cushion in the case of evaluating ROI-mask membership from a given list of parcel masks.

Returns

coords_adj [list] Filtered list of (x, y, z) tuples in mm-space with a spatial affinity for the specified ROI mask.

labels_adj [list] Filtered list of string labels corresponding to ROI nodes with a spatial affinity for the specified ROI mask.

parcel_list_adj [list] Filtered list of 3D boolean numpy arrays or binarized Nifti1Images corresponding to ROI masks with a spatial affinity to the specified ROI mask.

pynets.core.nodemaker.**parcel_naming**(*coords*, *vox_size*)

Perform Automated-Anatomical Labeling of each coordinate from a list of a voxel coordinates. This was adapted from a function of the same name created by Cameron Craddock and included in PyClusterROI (See: <https://github.com/ccraddock/cluster_roi/blob/master/parcel_naming.py>).

Parameters

coords [list] List of (x, y, z) tuples in voxel-space corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

vox_size [str] Voxel resolution (*1mm* or *2mm* stored as strings with units).

Returns

labels [list] List of string labels corresponding to each coordinate's closest anatomical label.

References

[1], [2]

pynets.core.thresholding module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017 @author: Derek Pisner (dPys)

pynets.core.thresholding.**autofix**(*W*, *copy=True*)

Fix a bunch of common problems. More specifically, remove Inf and NaN, ensure exact binariness and symmetry (i.e. remove floating point instability), and zero diagonal.

Parameters

W [np.ndarray] weighted connectivity matrix.

copy [bool] if True, returns a copy of the matrix. Otherwise, modifies the matrix in place. Default value=True.

Returns

——-

W [np.ndarray] connectivity matrix with fixes applied.

References

[1]

pynets.core.thresholding.**binarize**(*W*, *copy=True*)
Binarizes an input weighted connection matrix. If copy is not set, this function will *modify W in place.*

Parameters

W [NxN np.ndarray] weighted connectivity matrix

copy [bool] if True, returns a copy of the matrix. Otherwise, modifies the matrix in place. Default value=True.

Returns

W [NxN np.ndarray] binary connectivity matrix

References

[1]

pynets.core.thresholding.**density_thresholding**(*conn_matrix*, *thr*, *max_iters=10000*, *interval=0.01*)
Iteratively apply an absolute threshold to achieve a target density.

Parameters

conn_matrix [np.ndarray] Weighted connectivity matrix

thr [float] Density value between 0-1.

max_iters [int] Maximum number of iterations for performing absolute thresholding. Default is 1000.

interval [float] Interval for increasing the absolute threshold for each iteration. Default is 0.01.

Returns

conn_matrix [np.ndarray] Thresholded connectivity matrix

References

[1], [2]

pynets.core.thresholding.**disparity_filter**(*G*, *weight='weight'*)
Compute significance scores (alpha) for weighted edges in G as defined in Serrano et al. 2009.

Parameters

G [Object] Weighted NetworkX graph.

weight [str] Key for edge data used as the edge weight w_ij. Default is 'weight'.

Returns

B [Object] Weighted NetworkX graph with a significance score (alpha) assigned to each edge.

References

[1]

pynets.core.thresholding.**disparity_filter_alpha_cut**(*G*, *weight='weight'*, *alpha_t=0.4*, *cut_mode='or'*)

Compute significance scores (alpha) for weighted edges in G as defined in Serrano et al. 2009.

Parameters

G [Object] Weighted NetworkX graph.

weight [str] Key for edge data used as the edge weight w_ij. Default is 'weight'.

alpha_t [float] The threshold, between 0 and 1, for the alpha parameter used to select the surviving edges. Default is 0.4.

cut_mode [str] In the case of directed graphs. It represents the logic operation to filter out edges that do not pass the threshold value, combining the alpha_in and alpha_out attributes resulting from the disparity_filter function. Default is 'or'. Possible strings: 'or', 'and'.

Returns

B [Object] Weighted NetworkX graph with a significance score (alpha) assigned to each edge. The resulting graph contains only edges that survived from the filtering with the alpha_t threshold.

References

[1]

pynets.core.thresholding.**est_density**(*in_mat*)

Calculates the density of a given undirected graph.

Parameters

in_mat [NxN np.ndarray] weighted connectivity matrix.

Returns

density [float] Density of the graph.

pynets.core.thresholding.**invert**(*W*, *copy=False*)

Inverts elementwise the weights in an input connection matrix. In other words, change the from the matrix of internode strengths to the matrix of internode distances. If copy is not set, this function will *modify W in place.*

Parameters

W [np.ndarray] weighted connectivity matrix

copy [bool] if True, returns a copy of the matrix. Otherwise, modifies the matrix in place. Default value=True.

Returns

W [np.ndarray] inverted connectivity matrix

References

[1]

pynets.core.thresholding.**knn**(*conn_matrix*, *k*)

Creates a k-nearest neighbour graph.

Parameters

conn_matrix [array] Weighted NxN matrix.

k [int] Number of nearest neighbours to include in the knn estimation.

Returns

gra [Obj] KNN Weighted NetworkX graph.

pynets.core.thresholding.**local_thresholding_prop**(*conn_matrix*, *thr*)
Threshold the adjacency matrix by building from the minimum spanning tree (MST) and adding successive N-nearest neighbour degree graphs to achieve target proportional threshold.

Parameters

conn_matrix [array] Weighted NxN matrix.

thr [float] A proportional threshold, between 0 and 1, to achieve through local thresholding.

Returns

conn_matrix_thr [array] Weighted local-thresholding using MST, NxN matrix.

References

[1], [2]

pynets.core.thresholding.**normalize**(*W*)
Normalizes an input weighted connection matrix.

Parameters

W [np.ndarray] weighted connectivity matrix

Returns

W [np.ndarray] normalized connectivity matrix

References

[1]

pynets.core.thresholding.**perform_thresholding**(*conn_matrix*, *thr*, *min_span_tree*, *dens_thresh*, *disp_filt*)

References

[1]

pynets.core.thresholding.**standardize**(*W*)
Normalizes an input weighted connection matrix [0, 1]

Parameters

W [np.ndarray] weighted connectivity matrix

Returns

W [np.ndarray] standardized connectivity matrix

References

[1]

pynets.core.thresholding.**thr2prob**(*W*, *copy=True*)

Thresholds the near-zero ranks of a ranked graph.

Parameters

W [NxN np.ndarray] Weighted connectivity matrix of ranks.

Returns

W [NxN np.ndarray] Weighted connectivity matrix of ranks with no near-zero entries.

References

[1]

pynets.core.thresholding.**thresh_func**(*dens_thresh*, *thr*, *conn_matrix*, *conn_model*, *network*, *ID*, *dir_path*, *roi*, *node_size*, *min_span_tree*, *smooth*, *disp_filt*, *parc*, *prune*, *atlas*, *uatlas*, *labels*, *coords*, *norm*, *binary*, *hpass*, *extract_strategy*, *check_consistency=True*)

Threshold a functional connectivity matrix using any of a variety of methods.

Parameters

dens_thresh [bool] Indicates whether a target graph density is to be used as the basis for thresholding.

thr [float] A value, between 0 and 1, to threshold the graph using any variety of methods triggered through other options.

conn_matrix [array] Adjacency matrix stored as an m x n array of nodes and edges.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

ID [str] A subject id or other unique identifier.

dir_path [str] Path to directory containing subject derivative data for given run.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's.

min_span_tree [bool] Indicates whether local thresholding from the Minimum Spanning Tree should be used.

smooth [int] Smoothing width (mm fwhm) to apply to time-series when extracting signal from ROI's.

disp_filt [bool] Indicates whether local thresholding using a disparity filter and 'backbone network' should be used.

parc [bool] Indicates whether to use parcels instead of coordinates as ROI nodes.

prune [bool] Indicates whether to prune final graph of disconnected nodes/isolates.

atlas [str] Name of atlas parcellation used.

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

labels [list] List of string labels corresponding to ROI nodes.

coords [list] List of (x, y, z) tuples corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

norm [int] Indicates method of normalizing resulting graph.

binary [bool] Indicates whether to binarize resulting graph edges to form an unweighted graph.

hpass [float] High-pass filter values (Hz) to apply to node-extracted time-series.

extract_strategy [str] The name of a valid function used to reduce the time-series region extraction.

Returns

conn_matrix_thr [array] Weighted, thresholded, NxN matrix.

edge_threshold [str] The string percentage representation of thr.

est_path [str] File path to the thresholded graph, conn_matrix_thr, saved as a numpy array in .npy format.

thr [float] The value, between 0 and 1, used to threshold the graph using any variety of methods triggered through other options.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

smooth [int] Smoothing width (mm fwhm) to apply to time-series when extracting signal from ROI's.

prune [bool] Indicates whether to prune final graph of disconnected nodes/isolates.

ID [str] A subject id or other unique identifier.

dir_path [str] Path to directory containing subject derivative data for given run.

atlas [str] Name of atlas parcellation used.

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

labels [list] List of string labels corresponding to ROI nodes.

coords [list] List of (x, y, z) tuples corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

norm [int] Indicates method of normalizing resulting graph.

binary [bool] Indicates whether to binarize resulting graph edges to form an unweighted graph.

hpass [float] High-pass filter values (Hz) to apply to node-extracted time-series.

extract_strategy [str] The name of a valid function used to reduce the time-series region extraction.

References

[1]

pynets.core.thresholding.**thresh_raw_graph**(*conn_matrix*, *thr*, *min_span_tree*, *dens_thresh*, *disp_filt*, *est_path*)

pynets.core.thresholding.**thresh_struct**(*dens_thresh*, *thr*, *conn_matrix*, *conn_model*, *network*, *ID*, *dir_path*, *roi*, *node_size*, *min_span_tree*, *disp_filt*, *parc*, *prune*, *atlas*, *uatlas*, *labels*, *coords*, *norm*, *binary*, *target_samples*, *track_type*, *atlas_for_streams*, *streams*, *directget*, *min_length*, *error_margin*, *check_consistency=True*)

Threshold a structural connectivity matrix using any of a variety of methods.

Parameters

dens_thresh [bool] Indicates whether a target graph density is to be used as the basis for thresholding.

thr [float] A value, between 0 and 1, to threshold the graph using any variety of methods triggered through other options.

conn_matrix [array] Adjacency matrix stored as an m x n array of nodes and edges.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

ID [str] A subject id or other unique identifier.

dir_path [str] Path to directory containing subject derivative data for given run.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's.

min_span_tree [bool] Indicates whether local thresholding from the Minimum Spanning Tree should be used.

disp_filt [bool] Indicates whether local thresholding using a disparity filter and 'backbone network' should be used.

parc [bool] Indicates whether to use parcels instead of coordinates as ROI nodes.

prune [bool] Indicates whether to prune final graph of disconnected nodes/isolates.

atlas [str] Name of atlas parcellation used.

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

labels [list] List of string labels corresponding to ROI nodes.

coords [list] List of (x, y, z) tuples corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

norm [int] Indicates method of normalizing resulting graph.

binary [bool] Indicates whether to binarize resulting graph edges to form an unweighted graph.

target_samples [int] Total number of streamline samples specified to generate streams.

track_type [str] Tracking algorithm used (e.g. 'local' or 'particle').

atlas_for_streams [str] File path to atlas parcellation Nifti1Image in the morphological space of the streamlines.

streams [str] File path to save streamline array sequence in .trk format.

directget [str] The statistical approach to tracking. Options are: det (deterministic), closest (clos), boot (bootstrapped), and prob (probabilistic).

min_length [int] Minimum fiber length threshold in mm to restrict tracking.

Returns

conn_matrix_thr [array] Weighted, thresholded, NxN matrix.

edge_threshold [str] The string percentage representation of thr.

est_path [str] File path to the thresholded graph, conn_matrix_thr, saved as a numpy array in .npy format.

thr [float] The value, between 0 and 1, used to threshold the graph using any variety of methods triggered through other options.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

prune [bool] Indicates whether to prune final graph of disconnected nodes/isolates.

ID [str] A subject id or other unique identifier.

dir_path [str] Path to directory containing subject derivative data for given run.

atlas [str] Name of atlas parcellation used.

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

labels [list] List of string labels corresponding to ROI nodes.

coords [list] List of (x, y, z) tuples corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

norm [int] Indicates method of normalizing resulting graph.

binary [bool] Indicates whether to binarize resulting graph edges to form an unweighted graph.

target_samples [int] Total number of streamline samples specified to generate streams.

track_type [str] Tracking algorithm used (e.g. 'local' or 'particle').

atlas_for_streams [str] File path to atlas parcellation Nifti1Image in the morphological space of the streamlines.

streams [str] File path to save streamline array sequence in .trk format.

directget [str] The statistical approach to tracking. Options are: det (deterministic), closest (clos), boot (bootstrapped), and prob (probabilistic).

min_length [int] Minimum fiber length threshold in mm to restrict tracking.

References

[1]

pynets.core.thresholding.**threshold_absolute**(*W*, *thr*, *copy=True*)

This function thresholds the connectivity matrix by absolute weight magnitude. All weights below the given threshold, and all weights on the main diagonal (self-self connections) are set to 0. If copy is not set, this function will *modify W in place.*

Parameters

W [np.ndarray] weighted connectivity matrix

thr [float] absolute weight threshold

copy [bool] if True, returns a copy of the matrix. Otherwise, modifies the matrix in place. Default value=True.

Returns

W [np.ndarray] thresholded connectivity matrix

References

[1]

pynets.core.thresholding.**threshold_proportional**(*W*, *p*, *copy=True*)

This function "thresholds" the connectivity matrix by preserving a proportion p (0<p<1) of the strongest weights. All other weights, and all weights on the main diagonal (self-self connections) are set to 0. If copy is not set, this function will *modify W in place.*

Parameters

W [np.ndarray] weighted connectivity matrix

p [float] proportional weight threshold (0<p<1)

copy [bool] if True, returns a copy of the matrix. Otherwise, modifies the matrix in place. Default value=True.

Returns

W [np.ndarray] thresholded connectivity matrix

Notes

The proportion of elements set to 0 is a fraction of all elements in the matrix, whether or not they are already 0. That is, this function has the following behavior: >> x = np.random.random_sample((10,10)) >> x_25 = threshold_proportional(x, .25) >> np.size(np.where(x_25)) #note this double counts each nonzero element 46 >> x_125 = threshold_proportional(x, .125) >> np.size(np.where(x_125)) 22 >> x_test = threshold_proportional(x_25, .5) >> np.size(np.where(x_test)) 46 That is, the 50% thresholding of x_25 does nothing because >=50% of the elements in x_25 are aleady <=0. This behavior is the same as in BCT. Be careful with matrices that are both signed and sparse.

References

[1]

pynets.core.thresholding.**weight_conversion**(*W*, *wcm*, *copy=True*)

W_bin = weight_conversion(W, 'binarize'); W_nrm = weight_conversion(W, 'normalize'); L = weight_conversion(W, 'lengths'); This function may either binarize an input weighted connection matrix, normalize an input weighted connection matrix or convert an input weighted connection matrix to a weighted connection-length matrix. Binarization converts all present connection weights to 1. Normalization scales all weight magnitudes to the range [0,1] and should be done prior to computing some weighted measures, such as the weighted clustering coefficient. Conversion of connection weights to connection lengths is needed prior to computation of weighted distance-based measures, such as distance and betweenness centrality. In a weighted connection network, higher weights are naturally interpreted as shorter lengths. The connection-lengths matrix here is defined as the inverse of the connection-weights matrix. If copy is not set, this function will *modify W in place.*

Parameters

W [NxN np.ndarray] weighted connectivity matrix

wcm [str] weight conversion command. 'binarize' : binarize weights 'normalize' : normalize weights 'lengths' : convert weights to lengths (invert matrix)

copy [bool] if True, returns a copy of the matrix. Otherwise, modifies the matrix in place. Default value=True.

Returns

W [NxN np.ndarray] connectivity matrix with specified changes

Notes

This function is included for compatibility with BCT. But there are other functions binarize(), normalize() and invert() which are simpler to call directly.

References

[1]

pynets.core.thresholding.**weight_to_distance**(*G*)

Inverts all the edge weights so they become equivalent to distance measure. With a weight, the higher the value the stronger the connection. With a distance, the higher the value the "weaker" the connection. In this case there is no measurement unit for the distance, as it is just a conversion from the weights. The distances can be accessed in each node's property with constants.

Parameters

G [Object] Weighted NetworkX graph.

Returns

G [Object] Inverted NetworkX graph equivalent to the distance measure.

pynets.core.utils module

Created on Fri Nov 10 15:44:46 2017 Copyright (C) 2017 @author: Derek Pisner (dPys)

pynets.core.utils.**as_directory**(*dir_*, *remove=False*, *return_as_path=False*)

Convenience function to make a directory while returning it.

Parameters

dir_ [str, Path] File location to directory.

remove [bool, optional] Whether to remove a previously existing directory, by default False

Returns

str Directory string.

pynets.core.utils.**as_list**(*x*)
A function to convert an item to a list if it is not, or pass it through otherwise.

pynets.core.utils.**build_args_from_config**(*modality*, *arg_dict*)

class pynets.core.utils.**build_sql_db**(*dir_path*, *ID*)
Bases: object

A SQL exporter for AUC metrics.

Methods

add_hp_columns	
add_row_from_df	
create_modality_table	

add_hp_columns(*hyperparams*)

add_row_from_df(*df_summary_auc*, *hyperparam_dict*)

create_modality_table(*modality*)

pynets.core.utils.**checkConsecutive**(*l*)

pynets.core.utils.**check_est_path_existence**(*est_path_list*)
Checks for the existence of each graph estimated and saved to disk.

Parameters

est_path_list [list] List of file paths to .npy file containing graph with thresholding applied.

Returns

est_path_list_ex [list] List of existing file paths to .npy file containing graph with thresholding applied.

bad_ixs [int] List of indices in est_path_list with non-existent and/or corrupt files.

pynets.core.utils.**check_template_loads**(*template*, *template_mask*, *template_name*)

pynets.core.utils.**collect_pandas_df**(*network*, *ID*, *net_mets_csv_list*, *plot_switch*, *multi_nets*, *multimodal*, *embed*)

API for summarizing independent lists of pickled pandas dataframes of graph metrics for each modality, RSN, and roi.

Parameters

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

ID [str] A subject id or other unique identifier.

net_mets_csv_list [list] List of file paths to pickled pandas dataframes as themselves.

plot_switch [bool] Activate summary plotting (histograms, ROC curves, etc.)

multi_nets [list] List of Yeo RSN's specified in workflow(s).

multimodal [bool] Indicates whether multiple modalities of input data have been specified.

Returns

combination_complete [bool] If True, then collect_pandas_df completed successfully.

pynets.core.utils.**collectpandasjoin**(*net_mets_csv*)

Passes csv pandas dataframe as metadata.

Parameters

net_mets_csv [str] File path to csv pandas dataframe.

Returns

net_mets_csv_out [str] File path to csv pandas dataframe as itself.

pynets.core.utils.**create_csv_path**(*dir_path*, *est_path*)

Create a csv path to save graph metrics.

Parameters

dir_path [str] Path to directory containing subject derivative data for given run.

est_path [str] File path to .npy file containing graph with thresholding applied.

Returns

out_path [str] File path to .csv with graph metrics.

pynets.core.utils.**create_est_path_diff**(*ID*, *network*, *conn_model*, *thr*, *roi*, *dir_path*, *node_size*, *target_samples*, *track_type*, *thr_type*, *parc*, *directget*, *min_length*, *error_margin*)

Name the thresholded structural connectivity matrix file based on relevant graph-generating parameters.

Parameters

ID [str] A subject id or other unique identifier.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

thr [float] A value, between 0 and 1, to threshold the graph using any variety of methods triggered through other options.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

dir_path [str] Path to directory containing subject derivative data for given run.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's.

target_samples [int] Total number of streamline samples specified to generate streams.

track_type [str] Tracking algorithm used (e.g. 'local' or 'particle').

thr_type [str] Type of thresholding performed (e.g. prop, abs, dens, mst, disp)

parc [bool] Indicates whether to use parcels instead of coordinates as ROI nodes.

directget [str] The statistical approach to tracking. Options are: det (deterministic), closest (clos), boot (bootstrapped), and prob (probabilistic).

min_length [int] Minimum fiber length threshold in mm to restrict tracking.

Returns

est_path [str] File path to .npy file containing graph with thresholding applied.

pynets.core.utils.**create_est_path_func**(*ID*, *network*, *conn_model*, *thr*, *roi*, *dir_path*, *node_size*, *smooth*, *thr_type*, *hpass*, *parc*, *extract_strategy*)

Name the thresholded functional connectivity matrix file based on relevant graph-generating parameters.

Parameters

ID [str] A subject id or other unique identifier.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

thr [float] A value, between 0 and 1, to threshold the graph using any variety of methods triggered through other options.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

dir_path [str] Path to directory containing subject derivative data for given run.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's.

smooth [int] Smoothing width (mm fwhm) to apply to time-series when extracting signal from ROI's.

thr_type [str] Type of thresholding performed (e.g. prop, abs, dens, mst, disp)

hpass [bool] High-pass filter values (Hz) to apply to node-extracted time-series.

parc [bool] Indicates whether to use parcels instead of coordinates as ROI nodes.

extract_strategy [str] The name of a valid function used to reduce the time-series region extraction.

Returns

est_path [str] File path to .npy file containing graph with all specified combinations of hyperparameter characteristics.

pynets.core.utils.**create_raw_path_diff**(*ID*, *network*, *conn_model*, *roi*, *dir_path*, *node_size*, *target_samples*, *track_type*, *parc*, *directget*, *min_length*, *error_margin*)

Name the raw structural connectivity matrix file based on relevant graph-generating parameters.

Parameters

ID [str] A subject id or other unique identifier.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

dir_path [str] Path to directory containing subject derivative data for given run.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's.

target_samples [int] Total number of streamline samples specified to generate streams.

track_type [str] Tracking algorithm used (e.g. 'local' or 'particle').

parc [bool] Indicates whether to use parcels instead of coordinates as ROI nodes.

directget [str] The statistical approach to tracking. Options are: det (deterministic), closest (clos), boot (bootstrapped), and prob (probabilistic).

min_length [int] Minimum fiber length threshold in mm to restrict tracking.

Returns

est_path [str] File path to .npy file containing graph with thresholding applied.

pynets.core.utils.**create_raw_path_func**(*ID*, *network*, *conn_model*, *roi*, *dir_path*, *node_size*, *smooth*, *hpass*, *parc*, *extract_strategy*)

Name the raw functional connectivity matrix file based on relevant graph-generating parameters.

Parameters

ID [str] A subject id or other unique identifier.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

dir_path [str] Path to directory containing subject derivative data for given run.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's.

smooth [int] Smoothing width (mm fwhm) to apply to time-series when extracting signal from ROI's.

hpass [bool] High-pass filter values (Hz) to apply to node-extracted time-series.

parc [bool] Indicates whether to use parcels instead of coordinates as ROI nodes.

extract_strategy [str] The name of a valid function used to reduce the time-series region extraction.

Returns

est_path [str] File path to .npy file containing graph with all specified combinations of hyperparameter characteristics.

pynets.core.utils.**decompress_nifti**(*infile*)

pynets.core.utils.**do_dir_path**(*atlas*, *outdir*)

Creates an atlas subdirectory from the base directory of the given subject's input file.

Parameters

atlas [str] Name of atlas parcellation used.

outdir [str] Path to base derivatives directory.

Returns

dir_path [str] Path to directory containing subject derivative data for given run.

pynets.core.utils.**dumpstacks**(*signal*, *frame*)

pynets.core.utils.**filter_cols_from_targets**(*df*, *targets*)

pynets.core.utils.**flatten**(*l*)
Flatten list of lists.

pynets.core.utils.**get_file**()
Get a file's base directory path.

pynets.core.utils.**get_template_tf**(*template_name*, *vox_size*)

pynets.core.utils.**kill_process_family**(*parent_pid*)

pynets.core.utils.**load_mat**(*est_path*)
Load an adjacency matrix using any of a variety of methods.

Parameters

est_path [str] File path to .npy file containing graph with thresholding applied.

pynets.core.utils.**load_mat_ext**(*est_path*, *ID*, *network*, *conn_model*, *roi*, *prune*, *norm*, *binary*, *min_span_tree*, *dens_thresh*, *disp_filt*)

pynets.core.utils.**load_runconfig**()

pynets.core.utils.**merge_dicts**(*x*, *y*)
A function to merge two dictionaries, making it easier for us to make modality specific queries for dwi images (since they have variable extensions due to having an nii, bval, and bvec file).

pynets.core.utils.**mergedicts**(*dict1*, *dict2*)

pynets.core.utils.**missing_elements**(*L*)

pynets.core.utils.**pass_meta_ins**(*conn_model*, *est_path*, *network*, *thr*, *prune*, *ID*, *roi*, *norm*, *binary*)
Passes parameters as metadata.

Parameters

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

est_path [str] File path to .npy file containing graph with thresholding applied.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

thr [float] A value, between 0 and 1, to threshold the graph using any variety of methods triggered through other options.

prune [bool] Indicates whether to prune final graph of disconnected nodes/isolates.

ID [str] A subject id or other unique identifier.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

norm [int] Indicates method of normalizing resulting graph.

binary [bool] Indicates whether to binarize resulting graph edges to form an unweighted graph.

Returns

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

est_path [str] File path to .npy file containing graph with thresholding applied.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

thr [float] A value, between 0 and 1, to threshold the graph using any variety of methods triggered through other options.

prune [bool] Indicates whether to prune final graph of disconnected nodes/isolates.

ID [str] A subject id or other unique identifier.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

norm [int] Indicates method of normalizing resulting graph.

binary [bool] Indicates whether to binarize resulting graph edges to form an unweighted graph.

pynets.core.utils.**pass_meta_ins_multi**(*conn_model_func*, *est_path_func*, *network_func*, *thr_func*, *prune_func*, *ID_func*, *roi_func*, *norm_func*, *binary_func*, *conn_model_struct*, *est_path_struct*, *network_struct*, *thr_struct*, *prune_struct*, *ID_struct*, *roi_struct*, *norm_struct*, *binary_struct*)

Passes multimodal iterable parameters as metadata.

Parameters

conn_model_func [str] Functional connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

est_path_func [str] File path to .npy file containing functional graph with thresholding applied.

network_func [str] Functional resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

thr_func [float] A value, between 0 and 1, to threshold the functional graph using any variety of methods triggered through other options.

prune_func [bool] Indicates whether to prune final functional graph of disconnected nodes/isolates.

ID_func [str] A subject id or other unique identifier for the functional workflow.

roi_func [str] File path to binarized/boolean region-of-interest Nifti1Image file applied to the functional data.

norm_func [int] Indicates method of normalizing resulting functional graph.

binary_func [bool] Indicates whether to binarize resulting graph edges to form an unweighted functional graph.

conn_model_struct [str] Diffusion structural connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr

for partial correlation). sps type is used by default.

est_path_struct [str] File path to .npy file containing diffusion structural graph with thresholding applied.

network_struct [str] Diffusion structural resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

thr_struct [float] A value, between 0 and 1, to threshold the diffusion structural graph using any variety of methods triggered through other options.

prune_struct [bool] Indicates whether to prune final diffusion structural graph of disconnected nodes/isolates.

ID_struct [str] A subject id or other unique identifier for the diffusion structural workflow.

roi_struct [str] File path to binarized/boolean region-of-interest Nifti1Image file applied too the dwi data.

norm_struct [int] Indicates method of normalizing resulting diffusion structural graph.

binary_struct [bool] Indicates whether to binarize resulting diffusion structural graph edges to form an unweighted graph.

Returns

conn_model_iterlist [list] List of connectivity estimation model parameters (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

est_path_iterlist [list] List of file paths to .npy file containing graph with thresholding applied.

network_iterlist [list] List of resting-state networks based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

thr_iterlist [list] List of values, between 0 and 1, to threshold the graph using any variety of methods triggered through other options.

prune_iterlist [list] List of booleans indicating whether final graphs were pruned of disconnected nodes/isolates.

ID_iterlist [list] List of repeated subject id strings.

roi_iterlist [list] List of file paths to binarized/boolean region-of-interest Nifti1Image files.

norm_iterlist [list] Indicates method of normalizing resulting graph.

binary_iterlist [list] List of booleans indicating whether resulting graph edges to form an unweighted graph were binarized.

embed_iterlist [list] List of booleans indicating whether omnibus embedding of graph population was performed.

multimodal_iterlist [list] List of booleans indicating whether multiple modalities of input data have been specified.

pynets.core.utils.**pass_meta_outs**(*conn_model_iterlist*, *est_path_iterlist*, *network_iterlist*, *thr_iterlist*, *prune_iterlist*, *ID_iterlist*, *roi_iterlist*, *norm_iterlist*, *binary_iterlist*)

Passes lists of iterable parameters as metadata.

Parameters

conn_model_iterlist [list] List of connectivity estimation model parameters (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

est_path_iterlist [list] List of file paths to .npy file containing graph with thresholding applied.

network_iterlist [list] List of resting-state networks based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

thr_iterlist [list] List of values, between 0 and 1, to threshold the graph using any variety of methods triggered through other options.

prune_iterlist [list] List of booleans indicating whether final graphs were pruned of disconnected nodes/isolates.

ID_iterlist [list] List of repeated subject id strings.

roi_iterlist [list] List of file paths to binarized/boolean region-of-interest Nifti1Image files.

norm_iterlist [list] Indicates method of normalizing resulting graph.

binary_iterlist [list] List of booleans indicating whether resulting graph edges to form an unweighted graph were binarized.

Returns

conn_model_iterlist [list] List of connectivity estimation model parameters (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

est_path_iterlist [list] List of file paths to .npy file containing graph with thresholding applied.

network_iterlist [list] List of resting-state networks based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

thr_iterlist [list] List of values, between 0 and 1, to threshold the graph using any variety of methods triggered through other options.

prune_iterlist [list] List of booleans indicating whether final graphs were pruned of disconnected nodes/isolates.

ID_iterlist [list] List of repeated subject id strings.

roi_iterlist [list] List of file paths to binarized/boolean region-of-interest Nifti1Image files.

norm_iterlist [list] Indicates method of normalizing resulting graph.

binary_iterlist [list] List of booleans indicating whether resulting graph edges to form an unweighted graph were binarized.

embed_iterlist [list] List of booleans indicating whether omnibus embedding of graph population was performed.

multimodal_iterlist [list] List of booleans indicating whether multiple modalities of input data have been specified.

pynets.core.utils.**proportional**(*k*, *voxels_list*)
Hagenbach-Bischoff Quota

pynets.core.utils.**prune_suffices**(*res*)

pynets.core.utils.**save_3d_to_4d**(*in_files*)

pynets.core.utils.**save_4d_to_3d**(*in_file*)

pynets.core.utils.**save_coords_and_labels_to_json**(*coords*, *labels*, *dir_path*, *network='all_nodes'*, *indices=None*)
Save coordinates and labels to json.

Parameters

coords [list] List of (x, y, z) tuples corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

labels [list] List of string labels corresponding to ROI nodes.

dir_path [str] Path to directory containing subject derivative data for given run.

network [str] Restricted sub-network name.

Returns

nodes_path [str] Path to nodes json metadata file.

pynets.core.utils.**save_mat**(*conn_matrix*, *est_path*, *fmt=None*)
Save an adjacency matrix using any of a variety of methods.

Parameters

conn_matrix [array] Adjacency matrix stored as an m x n array of nodes and edges.

est_path [str] File path to .npy file containing graph.

fmt [str]

Format to save connectivity matrix/graph (e.g. .npy, .pkl, .graphml, .txt, .ssv, .csv).

pynets.core.utils.**save_mat_thresholded**(*conn_matrix*, *est_path_orig*, *thr_type*, *ID*, *network*, *thr*, *conn_model*, *roi*, *prune*, *norm*, *binary*)

pynets.core.utils.**save_nifti_parcels_map**(*ID*, *dir_path*, *network*, *net_parcels_map_nifti*, *vox_size*)

This function takes a Nifti1Image parcellation object resulting from some form of masking and saves it to disk.

Parameters

ID [str] A subject id or other unique identifier.

dir_path [str] Path to directory containing subject derivative data for given run.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

net_parcels_map_nifti [Nifti1Image] A nibabel-based nifti image consisting of a 3D array with integer voxel intensities corresponding to ROI membership.

vox_size [str] Voxel size in mm. (e.g. 2mm).

Returns

net_parcels_nii_path [str] File path to Nifti1Image consisting of a 3D array with integer voxel intensities corresponding to ROI membership.

pynets.core.utils.**save_ts_to_file**(*roi*, *network*, *ID*, *dir_path*, *ts_within_nodes*, *smooth*, *hpass*, *node_size*, *extract_strategy*)

This function saves the time-series 4D numpy array to disk as a .npy file.

Parameters

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

ID [str] A subject id or other unique identifier.

dir_path [str] Path to directory containing subject derivative data for given run.

ts_within_nodes [array] 2D m x n array consisting of the time-series signal for each ROI node where m = number of scans and n = number of ROI's, where ROI's are parcel volumes.

smooth [int] Smoothing width (mm fwhm) to apply to time-series when extracting signal from ROI's.

hpass [bool] High-pass filter values (Hz) to apply to node-extracted time-series.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's for time-series extraction.

extract_strategy [str] The name of a valid function used to reduce the time-series region extraction.

Returns

out_path_ts [str] Path to .npy file containing array of fMRI time-series extracted from nodes.

pynets.core.utils.**timeout**(*seconds*)
Timeout function for hung calculations.

class pynets.core.utils.**watchdog**
Bases: object

Methods

run	

run()

pynets.core.workflows module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017 @author: Derek Pisner (dPys)

pynets.core.workflows.**dmri_connectometry**(*ID*, *atlas*, *network*, *node_size*, *roi*, *uatlas*, *plot_switch*, *parc*, *ref_txt*, *procmem*, *dwi_file*, *fbval*, *fbvec*, *anat_file*, *thr*, *dens_thresh*, *conn_model*, *user_atlas_list*, *multi_thr*, *multi_atlas*, *max_thr*, *min_thr*, *step_thr*, *node_size_list*, *conn_model_list*, *min_span_tree*, *use_parcel_naming*, *disp_filt*, *plugin_type*, *multi_nets*, *prune*, *mask*, *norm*, *binary*, *target_samples*, *curv_thr_list*, *step_list*, *track_type*, *min_length*, *maxcrossing*, *error_margin*, *directget*, *tiss_class*, *runtime_dict*, *execution_dict*, *multi_directget*, *template_name*, *vox_size*, *waymask*, *min_length_list*, *error_margin_list*, *outdir*)
A function interface for generating a dMRI connectometry nested workflow

pynets.core.workflows.**fmri_connectometry**(*func_file*, *ID*, *atlas*, *network*, *node_size*, *roi*, *thr*, *uatlas*, *conn_model*, *dens_thresh*, *conf*, *plot_switch*, *parc*, *ref_txt*, *procmem*, *multi_thr*, *multi_atlas*, *max_thr*, *min_thr*, *step_thr*, *k*, *clust_mask*, *k_list*, *k_clustering*, *user_atlas_list*, *clust_mask_list*, *node_size_list*, *conn_model_list*, *min_span_tree*, *use_parcel_naming*, *smooth*, *smooth_list*, *disp_filt*, *prune*, *multi_nets*, *clust_type*, *clust_type_list*, *plugin_type*, *mask*, *norm*, *binary*, *anat_file*, *runtime_dict*, *execution_dict*, *hpass*, *hpass_list*, *template_name*, *vox_size*, *local_corr*, *extract_strategy*, *extract_strategy_list*, *outdir*)
A function interface for generating an fMRI connectometry nested workflow

pynets.core.workflows.**raw_graph_workflow**(*multi_thr*, *thr*, *multi_graph*, *graph*, *ID*, *network*, *conn_model*, *roi*, *prune*, *norm*, *binary*, *min_span_tree*, *dens_thresh*, *disp_filt*, *min_thr*, *max_thr*, *step_thr*, *wf*, *net_mets_node*, *runtime_dict*)

pynets.core.workflows.**workflow_selector**(*func_file*, *ID*, *atlas*, *network*, *node_size*, *roi*, *thr*, *uatlas*, *multi_nets*, *conn_model*, *dens_thresh*, *conf*, *plot_switch*, *dwi_file*, *anat_file*, *parc*, *ref_txt*, *procmem*, *multi_thr*, *multi_atlas*, *max_thr*, *min_thr*, *step_thr*, *k*, *clust_mask*, *k_list*, *k_clustering*, *user_atlas_list*, *clust_mask_list*, *prune*, *node_size_list*, *conn_model_list*, *min_span_tree*, *verbose*, *plugin_type*, *use_parcel_naming*, *smooth*, *smooth_list*, *disp_filt*, *clust_type*, *clust_type_list*, *mask*, *norm*, *binary*, *fbval*, *fbvec*, *target_samples*, *curv_thr_list*, *step_list*, *track_type*, *min_length*, *maxcrossing*, *error_margin*, *directget*, *tiss_class*, *runtime_dict*, *execution_dict*, *embed*, *multi_directget*, *multimodal*, *hpass*, *hpass_list*, *vox_size*, *multiplex*, *waymask*, *local_corr*, *min_length_list*, *error_margin_list*, *extract_strategy*, *extract_strategy_list*, *outdir*, *clean=True*)

A meta-interface for selecting modality-specific workflows to nest into a single-subject workflow

Module contents

pynets.dmri package

Submodules

pynets.dmri.dmri_utils module

pynets.dmri.estimation module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017 @author: Derek Pisner (dPys)

pynets.dmri.estimation.**create_anisopowermap**(*gtab_file*, *dwi_file*, *B0_mask*)

Estimate an anisotropic power map image to use for registrations.

Parameters

gtab_file [str] File path to pickled DiPy gradient table object.

dwi_file [str] File path to diffusion weighted image.

B0_mask [str] File path to B0 brain mask.

Returns

anisopwr_path [str] File path to the anisotropic power Nifti1Image.

B0_mask [str] File path to B0 brain mask Nifti1Image.

gtab_file [str] File path to pickled DiPy gradient table object.

dwi_file [str] File path to diffusion weighted Nifti1Image.

References

[1]

pynets.dmri.estimation.**csa_mod_est**(*gtab*, *data*, *B0_mask*, *sh_order=8*)
Estimate a Constant Solid Angle (CSA) model from dwi data.

Parameters

gtab [Obj] DiPy object storing diffusion gradient information

data [array] 4D numpy array of diffusion image data.

B0_mask [str] File path to B0 brain mask.

sh_order [int] The order of the SH model. Default is 8.

Returns

csa_mod [ndarray] Coefficients of the csa reconstruction.

model [obj] Fitted csa model.

References

[1]

pynets.dmri.estimation.**csd_mod_est**(*gtab*, *data*, *B0_mask*, *sh_order=8*)
Estimate a Constrained Spherical Deconvolution (CSD) model from dwi data.

Parameters

gtab [Obj] DiPy object storing diffusion gradient information.

data [array] 4D numpy array of diffusion image data.

B0_mask [str] File path to B0 brain mask.

sh_order [int] The order of the SH model. Default is 8.

Returns

csd_mod [ndarray] Coefficients of the csd reconstruction.

model [obj] Fitted csd model.

References

[1], [2], [3], [4]

pynets.dmri.estimation.**sfm_mod_est**(*gtab*, *data*, *B0_mask*)
Estimate a Sparse Fascicle Model (SFM) from dwi data.

Parameters

gtab [Obj] DiPy object storing diffusion gradient information.

data [array] 4D numpy array of diffusion image data.

B0_mask [str] File path to B0 brain mask.

Returns

sf_mod [ndarray] Coefficients of the sfm reconstruction.

model [obj] Fitted sf model.

References

[1], [2]

pynets.dmri.estimation.**streams2graph**(*atlas_for_streams*, *streams*, *dir_path*, *track_type*, *target_samples*, *conn_model*, *network*, *node_size*, *dens_thresh*, *ID*, *roi*, *min_span_tree*, *disp_filt*, *parc*, *prune*, *atlas*, *uatlas*, *labels*, *coords*, *norm*, *binary*, *directget*, *warped_fa*, *min_length*, *error_margin*)

Use tracked streamlines as a basis for estimating a structural connectome.

Parameters

atlas_for_streams [str] File path to atlas parcellation Nifti1Image in T1w-conformed space.

streams [str] File path to streamline array sequence in .trk format.

dir_path [str] Path to directory containing subject derivative data for a given pynets run.

track_type [str] Tracking algorithm used (e.g. 'local' or 'particle').

target_samples [int] Total number of streamline samples specified to generate streams.

conn_model [str] Connectivity reconstruction method (e.g. 'csa', 'tensor', 'csd').

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's for tracking.

dens_thresh [bool] Indicates whether a target graph density is to be used as the basis for thresholding.

ID [str] A subject id or other unique identifier.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

min_span_tree [bool] Indicates whether local thresholding from the Minimum Spanning Tree should be used.

disp_filt [bool] Indicates whether local thresholding using a disparity filter and 'backbone network' should be used.

parc [bool] Indicates whether to use parcels instead of coordinates as ROI nodes.

prune [bool] Indicates whether to prune final graph of disconnected nodes/isolates.

atlas [str] Name of atlas parcellation used.

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

labels [list] List of string labels corresponding to graph nodes.

coords [list] List of (x, y, z) tuples corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

norm [int] Indicates method of normalizing resulting graph.

binary [bool] Indicates whether to binarize resulting graph edges to form an unweighted graph.

directget [str] The statistical approach to tracking. Options are: det (deterministic), closest (clos), boot (bootstrapped), and prob (probabilistic).

warped_fa [str] File path to MNI-space warped FA Nifti1Image.

min_length [int] Minimum fiber length threshold in mm to restrict tracking.

error_margin [int]

Euclidean margin of error for classifying a streamline as a connection to an ROI. Default is 2 voxels.

Returns

atlas_for_streams [str] File path to atlas parcellation Nifti1Image in T1w-conformed space.

streams [str] File path to streamline array sequence in .trk format.

conn_matrix [array] Adjacency matrix stored as an m x n array of nodes and edges.

track_type [str] Tracking algorithm used (e.g. 'local' or 'particle').

target_samples [int] Total number of streamline samples specified to generate streams.

dir_path [str] Path to directory containing subject derivative data for given run.

conn_model [str] Connectivity reconstruction method (e.g. 'csa', 'tensor', 'csd').

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's for tracking.

dens_thresh [bool] Indicates whether a target graph density is to be used as the basis for thresholding.

ID [str] A subject id or other unique identifier.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

min_span_tree [bool] Indicates whether local thresholding from the Minimum Spanning Tree should be used.

disp_filt [bool] Indicates whether local thresholding using a disparity filter and 'backbone network' should be used.

parc [bool] Indicates whether to use parcels instead of coordinates as ROI nodes.

prune [bool] Indicates whether to prune final graph of disconnected nodes/isolates.

atlas [str] Name of atlas parcellation used.

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

labels [list] List of string labels corresponding to graph nodes.

coords [list] List of (x, y, z) tuples corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

norm [int] Indicates method of normalizing resulting graph.

binary [bool] Indicates whether to binarize resulting graph edges to form an unweighted graph.

directget [str] The statistical approach to tracking. Options are: det (deterministic), closest (clos), boot (bootstrapped), and prob (probabilistic).

min_length [int] Minimum fiber length threshold in mm to restrict tracking.

error_margin [int]

Euclidean margin of error for classifying a streamline as a connection to an ROI. Default is 2 voxels.

References

[1], [2], [3]

pynets.dmri.estimation.**tens_mod_est**(*gtab*, *data*, *B0_mask*)
Estimate a tensor ODF model from dwi data.

Parameters

gtab [Obj] DiPy object storing diffusion gradient information

data [array] 4D numpy array of diffusion image data.

B0_mask [str] File path to B0 brain mask.

Returns

mod_odf [ndarray] Coefficients of the tensor reconstruction.

model [obj] Fitted tensor model.

References

[1], [2]

pynets.dmri.estimation.**tens_mod_fa_est**(*gtab_file*, *dwi_file*, *B0_mask*)
Estimate a tensor FA image to use for registrations.

Parameters

gtab_file [str] File path to pickled DiPy gradient table object.

dwi_file [str] File path to diffusion weighted image.

B0_mask [str] File path to B0 brain mask.

Returns

fa_path [str] File path to FA Nifti1Image.

B0_mask [str] File path to B0 brain mask Nifti1Image.

gtab_file [str] File path to pickled DiPy gradient table object.

dwi_file [str] File path to diffusion weighted Nifti1Image.

fa_md_path [str] File path to FA/MD mask Nifti1Image.

pynets.dmri.track module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017 @author: Derek Pisner (dPys)

pynets.dmri.track.**create_density_map**(*fa_img*, *dir_path*, *streamlines*, *conn_model*, *target_samples*, *node_size*, *curv_thr_list*, *step_list*, *network*, *roi*, *directget*, *min_length*, *namer_dir*)
Create a density map of the list of streamlines.

Parameters

fa_img [Nifti1Image] Dwi data stored as a Nifti1image object.

dir_path [str] Path to directory containing subject derivative data for a given pynets run.

streamlines [ArraySequence] DiPy list/array-like object of streamline points from tractography.

conn_model [str] Connectivity reconstruction method (e.g. 'csa', 'tensor', 'csd').

target_samples [int] Total number of streamline samples specified to generate streams.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's for tracking.

curv_thr_list [list] List of integer curvature thresholds used to perform ensemble tracking.

step_list [list] List of float step-sizes used to perform ensemble tracking.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

directget [str] The statistical approach to tracking. Options are: det (deterministic), closest (clos), boot (bootstrapped), and prob (probabilistic).

min_length [int] Minimum fiber length threshold in mm to restrict tracking.

Returns

streams [str] File path to saved streamline array sequence in DTK-compatible trackvis (.trk) format.

dir_path [str] Path to directory containing subject derivative data for a given pynets run.

dm_path [str] File path to fiber density map Nifti1Image.

pynets.dmri.track.**prep_tissues**(*t1_mask*, *gm_in_dwi*, *vent_csf_in_dwi*, *wm_in_dwi*, *tiss_class*, *B0_mask*, *cmc_step_size=0.2*)

Estimate a tissue classifier for tractography.

Parameters

t1_mask [Nifti1Image] T1w mask img.

gm_in_dwi [Nifti1Image] Grey-matter tissue segmentation Nifti1Image.

vent_csf_in_dwi [Nifti1Image] Ventricular CSF tissue segmentation Nifti1Image.

wm_in_dwi [Nifti1Image] White-matter tissue segmentation Nifti1Image.

tiss_class [str] Tissue classification method.

cmc_step_size [float] Step size from CMC tissue classification method.

Returns

tiss_classifier [obj] Tissue classifier object.

References

[1], [2]

pynets.dmri.track.**reconstruction**(*conn_model*, *gtab*, *dwi_data*, *B0_mask*)

Estimate a tensor model from dwi data.

Parameters

conn_model [str] Connectivity reconstruction method (e.g. 'csa', 'tensor', 'csd', 'sfm').

gtab [Obj] DiPy object storing diffusion gradient information.

dwi_data [array] 4D array of dwi data.

B0_mask [str] File path to B0 brain mask.

Returns

mod_fit [ndarray] Fitted connectivity reconstruction model.

mod [obj] Connectivity reconstruction model.

References

[1]

pynets.dmri.track.**run_tracking**(*step_curv_combinations*, *recon_path*, *n_seeds_per_iter*, *directget*, *maxcrossing*, *max_length*, *pft_back_tracking_dist*, *pft_front_tracking_dist*, *particle_count*, *roi_neighborhood_tol*, *waymask*, *min_length*, *track_type*, *min_separation_angle*, *sphere*, *tiss_class*, *tissues4d*, *cache_dir*, *min_seeds=100*)

pynets.dmri.track.**track_ensemble**(*target_samples*, *atlas_data_wm_gm_int*, *labels_im_file*, *recon_path*, *sphere*, *directget*, *curv_thr_list*, *step_list*, *track_type*, *maxcrossing*, *roi_neighborhood_tol*, *min_length*, *waymask*, *B0_mask*, *t1w2dwi*, *gm_in_dwi*, *vent_csf_in_dwi*, *wm_in_dwi*, *tiss_class*, *cache_dir*)

Perform native-space ensemble tractography, restricted to a vector of ROI masks.

target_samples [int] Total number of streamline samples specified to generate streams.

atlas_data_wm_gm_int [str] File path to Nifti1Image in T1w-warped native diffusion space, restricted to wm-gm interface.

parcels [list] List of 3D boolean numpy arrays of atlas parcellation ROI masks from a Nifti1Image in T1w-warped native diffusion space.

recon_path [str] File path to diffusion reconstruction model.

tiss_classifier [str] Tissue classification method.

sphere [obj] DiPy object for modeling diffusion directions on a sphere.

directget [str] The statistical approach to tracking. Options are: det (deterministic), closest (clos), and prob (probabilistic).

curv_thr_list [list] List of integer curvature thresholds used to perform ensemble tracking.

step_list [list] List of float step-sizes used to perform ensemble tracking.

track_type [str] Tracking algorithm used (e.g. 'local' or 'particle').

maxcrossing [int] Maximum number if diffusion directions that can be assumed per voxel while tracking.

roi_neighborhood_tol [float] Distance (in the units of the streamlines, usually mm). If any coordinate in the streamline is within this distance from the center of any voxel in the ROI, the filtering criterion is set to True for this streamline, otherwise False. Defaults to the distance between the center of each voxel and the corner of the voxel.

min_length [int] Minimum fiber length threshold in mm.

waymask_data [ndarray] Tractography constraint mask array in native diffusion space.

B0_mask_data [ndarray] B0 brain mask data.

n_seeds_per_iter [int] Number of seeds from which to initiate tracking for each unique ensemble combination. By default this is set to 250.

max_length [int] Maximum number of steps to restrict tracking.

particle_count pft_back_tracking_dist : float Distance in mm to back track before starting the particle filtering tractography. The total particle filtering tractography distance is equal to back_tracking_dist + front_tracking_dist. By default this is set to 2 mm.

pft_front_tracking_dist [float] Distance in mm to run the particle filtering tractography after the the back track distance. The total particle filtering tractography distance is equal to back_tracking_dist + front_tracking_dist. By default this is set to 1 mm.

particle_count [int] Number of particles to use in the particle filter.

min_separation_angle [float] The minimum angle between directions [0, 90].

Returns

streamlines [ArraySequence] DiPy list/array-like object of streamline points from tractography.

References

[1]

Module contents

pynets.fmri package

Submodules

pynets.fmri.clustools module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017 @author: Derek Pisner (dPys)

class pynets.fmri.clustools.**NiParcellate**(*func_file*, *clust_mask*, *k*, *clust_type*, *local_corr*, *outdir*, *conf=None*, *mask=None*)

Bases: object

Class for implementing various clustering routines.

Methods

create_clean_mask (page 49)([num_std_dev])	Create a subject-refined version of the clustering mask.
create_local_clustering (page 49)(overwrite, r_thresh)	API for performing any of a variety of clustering routines available

prep_boot

create_clean_mask(*num_std_dev=1.5*)

Create a subject-refined version of the clustering mask.

create_local_clustering(*overwrite*, *r_thresh*, *min_region_size=80*)

API for performing any of a variety of clustering routines available through NiLearn.

prep_boot(*blocklength=1*)

pynets.fmri.clustools.**discretisation**(*eigen_vec*)

This function performs the second step of normalized cut clustering which assigns features to clusters based on the eigen vectors from the LaPlacian of a similarity matrix. There are a few different ways to perform this task. Shi and Malik (2000) iteratively bisect the features based on the positive and negative loadings of the eigenvectors. Ng, Jordan and Weiss (2001) proposed to perform K-means clustering on the rows of the eigenvectors. The method implemented here was proposed by Yu and Shi (2003) and it finds a discrete solution by iteratively rotating a binarised set of vectors until they are maximally similar to the the eigenvectors. An advantage of this method over K-means is that it is _more_ deterministic, i.e. you should get very similar results every time you run the algorithm on the same data.

The number of clusters that the features are clustered into is determined by the number of eignevectors (number of columns) in the input array eigen_vec. A caveat of this method, is that number of resulting clusters is bound by the number of eignevectors, but it may contain less.

Parameters

eigen_vec [array] Eigenvectors of the normalized LaPlacian calculated from the similarity matrix for the corresponding clustering problem.

Returns

eigen_vec_discrete [array] Discretised eigenvector outputs, i.e. vectors of 0 and 1 which indicate whether or not a feature belongs to the cluster defined by the eigen vector. e.g. a one in the 10th row of the 4th eigenvector (column) means that feature 10 belongs to cluster #4.

References

[1], [2], [3]

pynets.fmri.clustools.**ensemble_parcellate**(*infiles*, *k*)

pynets.fmri.clustools.**indx_1dto3d**(*idx*, *sz*)

Translate 1D vector coordinates to 3D matrix coordinates for a 3D matrix of size sz.

Parameters

idx [array] A 1D numpy coordinate vector.

sz [array] Shape of 3D matrix idx.

Returns

x [int] x-coordinate of 3D matrix coordinates.

y [int] y-coordinate of 3D matrix coordinates.

z [int] z-coordinate of 3D matrix coordinates.

pynets.fmri.clustools.**indx_3dto1d**(*idx*, *sz*)

Translate 3D matrix coordinates to 1D vector coordinates for a 3D matrix of size sz.

Parameters

idx [array] A 3D numpy array of matrix coordinates.

sz [array] Shape of 3D matrix idx.

Returns

idx1 [array] A 1D numpy coordinate vector.

pynets.fmri.clustools.**make_local_connectivity_scorr**(*func_img*, *clust_mask_img*, *thresh*)

Constructs a spatially constrained connectivity matrix from a fMRI dataset. The weights w_ij of the connectivity matrix W correspond to the spatial correlation between the whole brain FC maps generated from the time series from voxel i and voxel j. Connectivity is only calculated between a voxel and the 27 voxels in its 3D neighborhood (face touching and edge touching).

Parameters

func_img [Nifti1Image] 4D Nifti1Image containing fMRI data.

clust_mask_img [Nifti1Image] 3D NIFTI file containing a mask, which restricts the voxels used in the analysis.

thresh [str] Threshold value, correlation coefficients lower than this value will be removed from the matrix (set to zero).

Returns

W [Compressed Sparse Matrix] A Scipy sparse matrix, with weights corresponding to the spatial correlation between the time series from voxel i and voxel j

References

[1]

pynets.fmri.clustools.**make_local_connectivity_tcorr**(*func_img*, *clust_mask_img*, *thresh*)

Constructs a spatially constrained connectivity matrix from a fMRI dataset. The weights w_ij of the connectivity matrix W correspond to the temporal correlation between the time series from voxel i and voxel j. Connectivity is only calculated between a voxel and the 27 voxels in its 3D neighborhood (face touching and edge touching).

Parameters

func_img [Nifti1Image] 4D Nifti1Image containing fMRI data.

clust_mask_img [Nifti1Image] 3D NIFTI file containing a mask, which restricts the voxels used in the analysis.

thresh [str] Threshold value, correlation coefficients lower than this value will be removed from the matrix (set to zero).

Returns

W [Compressed Sparse Matrix] A Scipy sparse matrix, with weights corresponding to the temporal correlation between the time series from voxel i and voxel j

References

[1]

pynets.fmri.clustools.**ncut**(*W*, *nbEigenValues*)

This function performs the first step of normalized cut spectral clustering. The normalized LaPlacian is calculated on the similarity matrix W, and top nbEigenValues eigenvectors are calculated. The number of eigenvectors corresponds to the maximum number of classes (K) that will be produced by the clustering algorithm.

Parameters

W [array] Numpy array containing a symmetric #feature x #feature sparse matrix representing the similarity between voxels, traditionally this matrix should be positive semidefinite, but regularization is employed to allow negative matrix entries (Yu 2001).

nbEigenValues [int] Number of eigenvectors that should be calculated, this determines the maximum number of clusters (K) that can be derived from the result.

Returns

eigen_val [array] Eigenvalues from the eigen decomposition of the LaPlacian of W.

eigen_vec [array] Eigenvectors from the eigen decomposition of the LaPlacian of W.

References

[1], [2], [3]

pynets.fmri.clustools.**parcellate**(*func_boot_img*, *local_corr*, *clust_type*, *_local_conn_mat_path*, *num_conn_comps*, *_clust_mask_corr_img*, *_standardize*, *_detrending*, *k*, *_local_conn*, *conf*, *_dir_path*, *_conn_comps*)

API for performing any of a variety of clustering routines available through NiLearn.

pynets.fmri.clustools.**parcellate_ncut**(*W*, *k*, *mask_img*)

Converts a connectivity matrix into a nifti file where each voxel intensity corresponds to the number of the cluster to which it belongs. Clusters are renumberd to be contiguous.

Parameters

W [Compressed Sparse Matrix] A Scipy sparse matrix, with weights corresponding to the temporal/spatial correlation between the time series from voxel i and voxel j.

k [int] Numbers of clusters that will be generated.

mask_img [Nifti1Image] 3D NIFTI file containing a mask, which restricts the voxels used in the analysis.

References

[1]

pynets.fmri.estimation module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017 @author: Derek Pisner (dPys)

class pynets.fmri.estimation.**TimeseriesExtraction**(*net_parcels_nii_path*, *node_size*, *conf*, *func_file*, *roi*, *dir_path*, *ID*, *network*, *smooth*, *hpass*, *mask*, *extract_strategy*)

Bases: object

Class for implementing various time-series extracting routines.

Methods

extract_ts_parc (page 53)()	API for employing Nilearn's NiftiLabelsMasker to extract fMRI time-series data from spherical ROI's based on a given 3D atlas image of integer-based voxel intensities.
prepare_inputs (page 53)([num_std_dev])	Helper function to creating temporary nii's and prepare inputs from time-series extraction
save_and_cleanup (page 53)()	Save the extracted time-series and clean cache

extract_ts_parc()
API for employing Nilearn's NiftiLabelsMasker to extract fMRI time-series data from spherical ROI's based on a given 3D atlas image of integer-based voxel intensities. The resulting time-series can then optionally be resampled using circular-block bootrapping. The final 2D m x n array is ultimately saved to file in .npy format.

prepare_inputs(*num_std_dev=1.5*)
Helper function to creating temporary nii's and prepare inputs from time-series extraction

save_and_cleanup()
Save the extracted time-series and clean cache

pynets.fmri.estimation.**fill_confound_nans**(*confounds*, *dir_path*, *drop_thr=0.5*)
Fill the NaN values of a confounds dataframe with mean values

pynets.fmri.estimation.**get_conn_matrix**(*time_series*, *conn_model*, *dir_path*, *node_size*, *smooth*, *dens_thresh*, *network*, *ID*, *roi*, *min_span_tree*, *disp_filt*, *parc*, *prune*, *atlas*, *uatlas*, *labels*, *coords*, *norm*, *binary*, *hpass*, *extract_strategy*)
Computes a functional connectivity matrix based on a node-extracted time-series array. Includes a library of routines across Nilearn, scikit-learn, and skggm packages, among others.

Parameters

time_series [array] 2D m x n array consisting of the time-series signal for each ROI node where m = number of scans and n = number of ROI's.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

dir_path [str] Path to directory containing subject derivative data for given run.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's.

smooth [int] Smoothing width (mm fwhm) to apply to time-series when extracting signal from ROI's.

dens_thresh [bool] Indicates whether a target graph density is to be used as the basis for thresholding.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

ID [str] A subject id or other unique identifier.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

min_span_tree [bool] Indicates whether local thresholding from the Minimum Spanning Tree should be used.

disp_filt [bool] Indicates whether local thresholding using a disparity filter and 'backbone network' should be used.

parc [bool] Indicates whether to use parcels instead of coordinates as ROI nodes.

prune [bool] Indicates whether to prune final graph of disconnected nodes/isolates.

atlas [str] Name of atlas parcellation used.

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

labels [list] List of string labels corresponding to ROI nodes.

coords [list] List of (x, y, z) tuples corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

norm [int] Indicates method of normalizing resulting graph.

binary [bool] Indicates whether to binarize resulting graph edges to form an unweighted graph.

hpass [bool] High-pass filter values (Hz) to apply to node-extracted time-series.

extract_strategy [str] The name of a valid function used to reduce the time-series region extraction.

Returns

conn_matrix [array] Adjacency matrix stored as an m x n array of nodes and edges.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

dir_path [str] Path to directory containing subject derivative data for given run.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's for tracking.

smooth [int] Smoothing width (mm fwhm) to apply to time-series when extracting signal from ROI's.

dens_thresh [bool] Indicates whether a target graph density is to be used as the basis for thresholding.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

ID [str] A subject id or other unique identifier.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

min_span_tree [bool] Indicates whether local thresholding from the Minimum Spanning Tree should be used.

disp_filt [bool] Indicates whether local thresholding using a disparity filter and 'backbone network' should be used.

parc [bool] Indicates whether to use parcels instead of coordinates as ROI nodes.

prune [bool] Indicates whether to prune final graph of disconnected nodes/isolates.

atlas [str] Name of atlas parcellation used.

uatlas [str] File path to atlas parcellation Nifti1Image in MNI template space.

labels [list] List of string labels corresponding to graph nodes.

coords [list] List of (x, y, z) tuples corresponding to a coordinate atlas used or which represent the center-of-mass of each parcellation node.

norm [int] Indicates method of normalizing resulting graph.

binary [bool] Indicates whether to binarize resulting graph edges to form an unweighted graph.

hpass [bool] High-pass filter values (Hz) to apply to node-extracted time-series.

extract_strategy [str] The name of a valid function used to reduce the time-series region extraction.

References

[1], [2]

pynets.fmri.estimation.**get_optimal_cov_estimator**(*time_series*)

pynets.fmri.estimation.**timeseries_bootstrap**(*tseries*, *block_size*)

Generates a bootstrap sample derived from the input time-series. Utilizes Circular-block-bootstrap method described in [1].

Parameters

tseries [array_like] A matrix of shapes (*M*, *N*) with *M* timepoints and *N* variables

block_size [integer] Size of the bootstrapped blocks

Returns

bseries [array_like] Bootstrap sample of the input timeseries

References

[1]

Module contents

pynets.plotting package

Submodules

pynets.plotting.plot_gen module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017 @author: Derek Pisner (dPys)

pynets.plotting.plot_gen.**create_gb_palette**(*mat*, *edge_cmap*, *coords*, *labels*, *node_size='auto'*, *node_cmap=None*, *prune=True*, *centrality_type='eig'*, *max_node_size=None*, *node_aff_mat=None*)

Create connectome color palette based on graph topography.

Parameters

mat [array] NxN matrix.

edge_cmap: colormap colormap used for representing the weight of the edges.

coords [list] List of (x, y, z) tuples corresponding to an a-priori defined set (e.g. a coordinate atlas).

labels [list] List of string labels corresponding to ROI nodes.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's.

node_size: scalar or array_like size(s) of the nodes in points^2.

node_cmap: colormap colormap used for representing the community assignment of the nodes.

pynets.plotting.plot_gen.**plot_all_func**(*conn_matrix*, *conn_model*, *atlas*, *dir_path*, *ID*, *network*, *labels*, *roi*, *coords*, *thr*, *node_size*, *edge_threshold*, *smooth*, *prune*, *uatlas*, *norm*, *binary*, *hpass*, *extract_strategy*, *edge_color_override=False*)

Plot adjacency matrix, connectogram, and glass brain for functional connectome.

Parameters

conn_matrix [array] NxN matrix.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

atlas [str] Name of atlas parcellation used.

dir_path [str] Path to directory containing subject derivative data for given run.

ID [str] A subject id or other unique identifier.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

labels [list] List of string labels corresponding to ROI nodes.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

coords [list] List of (x, y, z) tuples corresponding to an a-priori defined set (e.g. a coordinate atlas).

thr [float] A value, between 0 and 1, to threshold the graph using any variety of methods triggered through other options.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's.

edge_threshold [float] The actual value, between 0 and 1, that the graph was thresholded (can differ from thr if target was not successfully obtained.

smooth [int] Smoothing width (mm fwhm) to apply to time-series when extracting signal from ROI's.

prune [bool] Indicates whether to prune final graph of disconnected nodes/isolates.

uatlas [str] File path to atlas parcellation Nifti1Image.

norm [int] Indicates method of normalizing resulting graph.

binary [bool] Indicates whether to binarize resulting graph edges to form an unweighted graph.

hpass [bool] High-pass filter values (Hz) to apply to node-extracted time-series.

extract_strategy [str] The name of a valid function used to reduce the time-series region extraction.

edge_color_override [bool] Switch that enables random sequential colormap selection for edges.

pynets.plotting.plot_gen.**plot_all_struct**(*conn_matrix*, *conn_model*, *atlas*, *dir_path*, *ID*, *network*, *labels*, *roi*, *coords*, *thr*, *node_size*, *edge_threshold*, *prune*, *uatlas*, *target_samples*, *norm*, *binary*, *track_type*, *directget*, *min_length*, *error_margin*)

Plot adjacency matrix, connectogram, and glass brain for functional connectome.

Parameters

conn_matrix [array] NxN matrix.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

atlas [str] Name of atlas parcellation used.

dir_path [str] Path to directory containing subject derivative data for given run.

ID [str] A subject id or other unique identifier.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

labels [list] List of string labels corresponding to ROI nodes.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

coords [list] List of (x, y, z) tuples corresponding to an a-priori defined set (e.g. a coordinate atlas).

thr [float] A value, between 0 and 1, to threshold the graph using any variety of methods triggered through other options.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's.

edge_threshold [float] The actual value, between 0 and 1, that the graph was thresholded (can differ from thr if target was not successfully obtained.

prune [bool] Indicates whether to prune final graph of disconnected nodes/isolates.

uatlas [str] File path to atlas parcellation Nifti1Image.

target_samples [int] Total number of streamline samples specified to generate streams.

norm [int] Indicates method of normalizing resulting graph.

binary [bool] Indicates whether to binarize resulting graph edges to form an unweighted graph.

track_type [str] Tracking algorithm used (e.g. 'local' or 'particle').

directget [str] The statistical approach to tracking. Options are: det (deterministic), closest (clos), boot (bootstrapped), and prob (probabilistic).

min_length [int] Minimum fiber length threshold in mm to restrict tracking.

pynets.plotting.plot_gen.**plot_all_struct_func**(*mG_path*, *namer_dir*, *name*, *modality_paths*, *metadata*)

Plot adjacency matrix and glass brain for structural-functional multiplex connectome.

Parameters

mG_path [str] A gpickle file containing a a MultilayerGraph object (See https://github.com/nkoub/multinetx).

namer_dir [str] Path to output directory for multiplex data.

name [str] Concatenation of multimodal graph filenames.

modality_paths [tuple] A tuple of filepath strings to the raw structural and raw functional connectome graph files (.npy).

metadata [dict] Dictionary coontaining coords and labels shared by each layer of the multilayer graph.

pynets.plotting.plot_gen.**plot_graph_measure_hists**(*csv_all_metrics*)

Plot histograms for each graph theoretical measure for a given subject.

Parameters

csv_all_metrics [str] CSV file of concatenated graph measures across ensemble.

pynets.plotting.plot_gen.**plot_network_clusters**(*graph*, *communities*, *out_path*, *figsize=(8, 8)*, *node_size=50*, *plot_overlaps=False*, *plot_labels=False*)

Plot a graph with node color coding for communities.

Parameters

graph [NetworkX graph]

communities [array] Community affiliation vector

out_path [str] Path to save figure.

figsize [Tuple of integers] The figure size; it is a pair of float, default (8, 8).

node_size: int Default 50.

plot_overlaps [bool] Flag to control if multiple algorithms memberships are plotted. Default is False.

plot_labels [bool] Flag to control if node labels are plotted. Default is False.

pynets.plotting.plot_gen.**plot_timeseries**(*time_series*, *network*, *ID*, *dir_path*, *atlas*, *labels*)

Plot time-series.

Parameters

time-series [array] 2D m x n array consisting of the time-series signal for each ROI node where m = number of scans and n = number of ROI's.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

ID [str] A subject id or other unique identifier.

dir_path [str] Path to directory containing subject derivative data for given run.

atlas [str] Name of atlas parcellation used.

labels [list] List of string labels corresponding to ROI nodes.

pynets.plotting.plot_gen.**show_template_bundles**(*final_streamlines*, *template_path*, *fname*)

pynets.plotting.plot_gen.**view_tractogram**(*streams*, *atlas*)

pynets.plotting.plot_graphs module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017 @author: Derek Pisner (dPys)

pynets.plotting.plot_graphs.**plot_community_conn_mat**(*conn_matrix*, *labels*, *out_path_fig_comm*, *community_aff*, *cmap*, *dpi_resolution=300*)

Plot a community-parcellated connectivity matrix.

Parameters

conn_matrix [array] NxN matrix.

labels [list] List of string labels corresponding to ROI nodes.

out_path_fig_comm [str] File path to save the community-parcellated connectivity matrix image as a .png figure.

community_aff [array] Community-affiliation vector.

pynets.plotting.plot_graphs.**plot_conn_mat**(*conn_matrix*, *labels*, *out_path_fig*, *cmap*, *binarized=False*, *dpi_resolution=300*)

Plot a connectivity matrix.

Parameters

conn_matrix [array] NxN matrix.

labels [list] List of string labels corresponding to ROI nodes.

out_path_fig [str] File path to save the connectivity matrix image as a .png figure.

pynets.plotting.plot_graphs.**plot_conn_mat_func**(*conn_matrix*, *conn_model*, *atlas*, *dir_path*, *ID*, *network*, *labels*, *roi*, *thr*, *node_size*, *smooth*, *hpass*, *extract_strategy*)

API for selecting among various functional connectivity matrix plotting approaches.

Parameters

conn_matrix [array] NxN matrix.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

atlas [str] Name of atlas parcellation used.

dir_path [str] Path to directory containing subject derivative data for given run.

ID [str] A subject id or other unique identifier.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

labels [list] List of string labels corresponding to ROI nodes.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

thr [float] A value, between 0 and 1, to threshold the graph using any variety of methods triggered through other options.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's.

smooth [int] Smoothing width (mm fwhm) to apply to time-series when extracting signal from ROI's.

hpass [bool] High-pass filter values (Hz) to apply to node-extracted time-series.

extract_strategy [str] The name of a valid function used to reduce the time-series region extraction.

pynets.plotting.plot_graphs.**plot_conn_mat_struct**(*conn_matrix*, *conn_model*, *atlas*, *dir_path*, *ID*, *network*, *labels*, *roi*, *thr*, *node_size*, *target_samples*, *track_type*, *directget*, *min_length*, *error_margin*)

API for selecting among various structural connectivity matrix plotting approaches.

Parameters

conn_matrix [array] NxN matrix.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

atlas [str] Name of atlas parcellation used.

dir_path [str] Path to directory containing subject derivative data for given run.

ID [str] A subject id or other unique identifier.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

labels [list] List of string labels corresponding to ROI nodes.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

thr [float] A value, between 0 and 1, to threshold the graph using any variety of methods triggered through other options.

node_size [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI's.

target_samples [int] Total number of streamline samples specified to generate streams.

track_type [str] Tracking algorithm used (e.g. 'local' or 'particle').

directget [str] The statistical approach to tracking. Options are: det (deterministic), closest (clos), boot (bootstrapped), and prob (probabilistic).

min_length [int] Minimum fiber length threshold in mm to restrict tracking.

Module contents

pynets.registration package

Submodules

pynets.registration.reg_utils module

pynets.registration.register module

Module contents

pynets package

Subpackages

pynets.stats package

Submodules

pynets.stats.benchmarking module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017 @authors: Derek Pisner

pynets.stats.benchmarking.**benchmark_reproducibility**(*base_dir*, *comb*, *modality*, *alg*, *par_dict*, *disc*, *final_missingness_summary*, *icc_tmps_dir*, *icc*, *mets*, *ids*, *template*)

pynets.stats.benchmarking.**beta_lin_comb**(*beta*, *GVDAT*, *meta*)

This function calculates linear combinations of graph vectors stored in GVDAT for all subjects and all sessions given the weights vector beta. This was adapted from a function of the same name, written by Kamil Bonna and Miriam Kosik 10.09.2018.

Parameters

beta [list] List of metaparameter weights.

GVDAT [ndarray] 5d data structure storing graph vectors.

Returns

gv_array [ndarray] 2d array of aggregated graph vectors for all sessions, all subjects.

pynets.stats.benchmarking.**discr_stat**(*X*, *Y*, *dissimilarity='euclidean'*, *remove_isolates=True*, *return_rdfs=True*)

Computes the discriminability statistic.

Parameters

X [array, shape (n_samples, n_features) or (n_samples, n_samples)]

Input data. If dissimilarity=='precomputed', the input should be the dissimilarity matrix.

Y [1d-array, shape (n_samples)] Input labels.

dissimilarity [str, {"euclidean" (default), "precomputed"} Dissimilarity] measure can be 'euclidean' (pairwise Euclidean distances between points in the dataset) or 'precomputed' (precomputed dissimilarities).

remove_isolates [bool, optional, default=True] Whether to remove data that have single label.

return_rdfs [bool, optional, default=False] Whether to return rdf for all data points.

Returns

stat [float] Discriminability statistic.

rdfs [array, shape (n_samples, max{len(id)})] Rdfs for each sample. Only returned if `return_rdfs==True`.

pynets.stats.embeddings module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017 @author: Derek Pisner

pynets.stats.embeddings.**build_asetomes**(*est_path_iterlist*, *ID*)
Embeds single graphs using the ASE algorithm.

Parameters

est_path_iterlist [list] List of file paths to .npy files, each containing a graph.

ID [str] A subject id or other unique identifier.

pynets.stats.embeddings.**build_masetome**(*est_path_iterlist*, *ID*)
Embeds structural-functional graph pairs into a common invariant subspace.

Parameters

est_path_iterlist [list] List of list of pairs of file paths (.npy) corresponding to structural and functional connectomes matched at a given node resolution.

ID [str] A subject id or other unique identifier.

References

[1]

pynets.stats.embeddings.**build_omnetome**(*est_path_iterlist*, *ID*)
Embeds ensemble population of graphs into an embedded ensemble feature vector.

Parameters

est_path_iterlist [list] List of file paths to .npy file containing graph.

ID [str] A subject id or other unique identifier.

References

[1], [2]

pynets.stats.netmotifs module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017 @author: Derek Pisner & James Kunert-Graf

pynets.stats.netmotifs.**adaptivethresh**(*in_mat*, *thr*, *mlib*, *N*, *use_gt=False*)
Counts number of motifs with a given absolute threshold.

Parameters

in_mat [ndarray] M x M Connectivity matrix

thr [float] Absolute threshold [0, 1].

mlib [list] List of motif classes.

Returns

mf [ndarray] 1D vector listing the total motifs of size N for each class of mlib.

References

[1]

pynets.stats.netmotifs.**build_multigraphs**(*est_path_iterlist*, *ID*)
Constructs a multimodal multigraph for each available resolution of vertices.

Parameters

est_path_iterlist [list] List of file paths to .npy file containing graph.

ID [str] A subject id or other unique identifier.

Returns

multigraph_list_all [list] List of multiplex graph dictionaries corresponding to each unique node resolution.

graph_path_list_top [list] List of lists consisting of pairs of most similar structural and functional connectomes for each unique node resolution.

References

[1], [2]

pynets.stats.netmotifs.**build_mx_multigraph**(*func_mat*, *struct_mat*, *name*, *namer_dir*)
It creates a symmetric (undirected) MultilayerGraph object from vertex-aligned structural and functional connectivity matrices.

References

[1], [2], [3]

pynets.stats.netmotifs.**compare_motifs**(*struct_mat*, *func_mat*, *name*, *namer_dir*, *bins=20*, *N=4*)
Compare motif structure and population across structural and functional graphs to achieve a homeostatic absolute threshold of each that optimizes multiplex community detection and analysis.

Parameters

in_mat [ndarray] M x M Connectivity matrix

thr [float] Absolute threshold [0, 1].

mlib [list] List of motif classes.

Returns

mf [ndarray] 1D vector listing the total motifs of size N for each class of mlib.

References

[1]

pynets.stats.netmotifs.**countmotifs**(*A*, *N=4*)
Counts number of motifs with size N from A.

Parameters

A [ndarray] M x M Connectivity matrix

N [int] Size of motif type. Default is N=4, only 3 or 4 supported.

Returns

umotifs [int] Total count of size N motifs for graph A.

References

[1]

pynets.stats.netmotifs.**motif_matching**(*paths*, *ID*, *atlas*, *namer_dir*, *name_list*, *metadata_list*, *multigraph_list_all*, *graph_path_list_all*, *rsn=None*)

pynets.stats.netmotifs.**multigraph_matching**(*paths*)

pynets.stats.netstats module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017 @author: Derek Pisner

class pynets.stats.netstats.**CleanGraphs**(*thr*, *conn_model*, *est_path*, *prune*, *norm*, *out_fmt='gpickle'*)

Bases: object

A Class for cleaning graphs in preparation for network analysis.

Parameters

thr [float] The value, between 0 and 1, used to threshold the graph using any variety of methods triggered through other options.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

est_path [str] File path to the thresholded graph, conn_matrix_thr, saved as a numpy array in .npy format.

prune [int] Indicates whether to prune final graph of disconnected nodes/isolates.

norm [int] Indicates method of normalizing resulting graph.

Returns

out_path [str] Path to .csv file where graph analysis results are saved.

References

[1], [2]

Methods

binarize_graph	
create_length_matrix	
normalize_graph	
print_summary	
prune_graph	

binarize_graph()

create_length_matrix()

normalize_graph()

print_summary()

prune_graph(*remove_self_loops=True*)

pynets.stats.netstats.**average_local_efficiency**(*G*, *weight='weight'*, *engine='nx'*)
Return the average local efficiency of all of the nodes in the G

Parameters

G [Obj] NetworkX graph.

Returns

average_local_efficiency [float] Average local efficiency of G.

Notes

Adapted from NetworkX to incorporate weight parameter.

References

[1], [2]

pynets.stats.netstats.**average_shortest_path_length_fast**(*G*, *weight='weight'*)

pynets.stats.netstats.**average_shortest_path_length_for_all**(*G*)
Helper function, in the case of graph disconnectedness, that returns the average shortest path length, calculated iteratively for each distinct subgraph of the G.

Parameters

G [Obj] NetworkX graph.

Returns

average_shortest_path_length [float] The length of the average shortest path for G.

pynets.stats.netstats.**collect_pandas_df_make**(*net_mets_csv_list*, *ID*, *network*, *plot_switch*, *embed=False*, *create_summary=False*, *sql_out=False*)
Summarize list of pickled pandas dataframes of graph metrics unique to each unique combination of metaparameters.

Parameters

net_mets_csv_list [list] List of file paths to pickled pandas dataframes as themselves.

ID [str] A subject id or other unique identifier.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

plot_switch [bool] Activate summary plotting (histograms, central tendency, AUC, etc.)

sql_out [bool] Optionally output data to sql.

Returns

combination_complete [bool] If True, then data integration completed successfully.

References

[1]

pynets.stats.netstats.**community_resolution_selection**(*G*)

pynets.stats.netstats.**create_communities**(*node_comm_aff_mat*, *node_num*)
Create a 1D vector of community assignments from a community affiliation matrix.

Parameters

node_comm_aff_mat [array] Community affiliation matrix produced from modularity estimation (e.g. Louvain).

node_num [int] Number of total connected nodes in the graph used to estimate node_comm_aff_mat.

Returns

com_assign [array] 1D numpy vector of community assignments.

References

[1], [2]

pynets.stats.netstats.**diversity_coef_sign**(*W*, *ci*)
The Shannon-entropy based diversity coefficient measures the diversity of intermodular connections of individual nodes and ranges from 0 to 1.

Parameters

W [NxN np.ndarray] undirected connection matrix with positive and negative weights

ci [Nx1 np.ndarray] community affiliation vector

Returns

Hpos [Nx1 np.ndarray] diversity coefficient based on positive connections

Hneg [Nx1 np.ndarray] diversity coefficient based on negative connections

References

[1]

pynets.stats.netstats.**extractnetstats**(*ID*, *network*, *thr*, *conn_model*, *est_path*, *roi*, *prune*, *norm*, *binary*)
Function interface for performing fully-automated graph analysis.

Parameters

ID [str] A subject id or other unique identifier.

network [str] Resting-state network based on Yeo-7 and Yeo-17 naming (e.g. 'Default') used to filter nodes in the study of brain subgraphs.

thr [float] The value, between 0 and 1, used to threshold the graph using any variety of methods triggered through other options.

conn_model [str] Connectivity estimation model (e.g. corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation). sps type is used by default.

est_path [str] File path to the thresholded graph, conn_matrix_thr, saved as a numpy array in .npy format.

roi [str] File path to binarized/boolean region-of-interest Nifti1Image file.

prune [int] Indicates whether to prune final graph of disconnected nodes/isolates.

norm [int] Indicates method of normalizing resulting graph.

binary [bool] Indicates whether to binarize resulting graph edges to form an unweighted graph.

Returns

out_path [str] Path to .csv file where graph analysis results are saved.

References

[1], [2]

pynets.stats.netstats.**get_betweenness_centrality**(*G_len*, *metric_list_names*, *net_met_val_list_final*, *engine='nx'*)

pynets.stats.netstats.**get_clustering**(*G*, *metric_list_names*, *net_met_val_list_final*, *engine='nx'*)

pynets.stats.netstats.**get_comm_centrality**(*G*, *metric_list_names*, *net_met_val_list_final*)

pynets.stats.netstats.**get_community**(*G*, *net_met_val_list_final*, *metric_list_names*)

pynets.stats.netstats.**get_degree_centrality**(*G*, *metric_list_names*, *net_met_val_list_final*)

pynets.stats.netstats.**get_diversity**(*in_mat*, *ci*, *metric_list_names*, *net_met_val_list_final*)

pynets.stats.netstats.**get_eigen_centrality**(*G*, *metric_list_names*, *net_met_val_list_final*, *engine='nx'*)

pynets.stats.netstats.**get_local_efficiency**(*G*, *metric_list_names*, *net_met_val_list_final*)

pynets.stats.netstats.**get_participation**(*in_mat*, *ci*, *metric_list_names*, *net_met_val_list_final*)

pynets.stats.netstats.**get_prop_type**(*value*, *key=None*)

Performs typing and value conversion for the graph_tool PropertyMap class. If a key is provided, it also ensures the key is in a format that can be used with the PropertyMap. Returns a tuple, (type name, value, key)

pynets.stats.netstats.**get_rich_club_coeff**(*G*, *metric_list_names*, *net_met_val_list_final*, *engine='nx'*)

pynets.stats.netstats.**global_efficiency**(*G*, *weight='weight'*, *engine='nx'*)

Return the global efficiency of the G

Parameters

G [NetworkX graph]

Returns

global_efficiency [float]

Notes

The published definition includes a scale factor based on a completely connected graph. In the case of an unweighted network, the scaling factor is 1 and can be ignored. In the case of a weighted graph, calculating the scaling factor requires somehow knowing the weights of the edges required to make a completely connected

graph. Since that knowlege may not exist, the scaling factor is not included. If that knowlege exists, construct the corresponding weighted graph and calculate its global_efficiency to scale the weighted graph. Distance between nodes is calculated as the sum of weights. If the graph is defined such that a higher weight represents a stronger connection, distance should be represented by 1/weight. In this case, use the invert weights function to generate a graph where the weights are set to 1/weight and then calculate efficiency

References

[1], [2]

pynets.stats.netstats.**iterate_nx_global_measures**(*G*, *metric_list_glob*)

pynets.stats.netstats.**link_communities**(*W*, *type_clustering='single'*)

The optimal community structure is a subdivision of the network into nonoverlapping groups of nodes which maximizes the number of within-group edges and minimizes the number of between-group edges. This algorithm uncovers overlapping community structure via hierarchical clustering of network links. This algorithm is generalized for weighted/directed/fully-connected networks

Parameters

W [NxN np.array] directed weighted/binary adjacency matrix

type_clustering [str] type of hierarchical clustering. 'single' for single-linkage, 'complete' for complete-linkage. Default value='single'

Returns

M [CxN np.ndarray] nodal community affiliation matrix.

References

[1]

pynets.stats.netstats.**local_efficiency**(*G*, *weight='weight'*, *engine='nx'*)

Return the local efficiency of each node in the G

Parameters

G [Obj] NetworkX graph.

Returns

local_efficiency [dict] The keys of the dict are the nodes in the G and the corresponding values are local efficiencies of each node

Notes

The published definition includes a scale factor based on a completely connected graph. In the case of an unweighted network, the scaling factor is 1 and can be ignored. In the case of a weighted graph, calculating the scaling factor requires somehow knowing the weights of the edges required to make a completely connected graph. Since that knowlege may not exist, the scaling factor is not included. If that knowlege exists, construct the corresponding weighted graph and calculate its local_efficiency to scale the weighted graph.

References

[1], [2]

pynets.stats.netstats.**most_important**(*G*, *method='betweenness'*, *sd=1*, *engine='nx'*)
Returns a copy of G with hubs as defined by centrality, core topology, or rich-club topology.

Parameters

G [Obj] NetworkX graph.

method [str] Determines method for defining hubs. Valid inputs are coreness, richclub, and eigenvector centrality. Default is coreness.

sd [int] Number of standard errors as cutoff for low-importance pruning.

Returns

G [Obj] NetworkX graph with isolated and low-importance nodes pruned.

pruned_nodes [list] List of indices of nodes that were pruned from G.

References

[1]

pynets.stats.netstats.**np2gt**(*adj*)

pynets.stats.netstats.**nx2gt**(*nxG*)
Converts a networkx graph to a graph-tool graph.

pynets.stats.netstats.**participation_coef**(*W*, *ci*, *degree='undirected'*)
Participation coefficient is a measure of diversity of intermodular connections of individual nodes.

Parameters

W [NxN np.ndarray] binary/weighted directed/undirected connection matrix

ci [Nx1 np.ndarray] community affiliation vector

degree [str]

Flag to describe nature of graph 'undirected': For undirected graphs 'in': Uses the in-degree 'out': Uses the out-degree

Returns

——-

P [Nx1 np.ndarray] Participation coefficient

References

[1], [2]

pynets.stats.netstats.**participation_coef_sign**(*W*, *ci*)
Participation coefficient is a measure of diversity of intermodular connections of individual nodes.

Parameters

W [NxN np.ndarray] undirected connection matrix with positive and negative weights

ci [Nx1 np.ndarray] community affiliation vector

Returns

Ppos [Nx1 np.ndarray] participation coefficient from positive weights

Pneg [Nx1 np.ndarray] participation coefficient from negative weights

References

[1], [2]

pynets.stats.netstats.**prune_disconnected**(*G*, *min_nodes=10*, *fallback_lcc=True*)

Returns a copy of G with isolates pruned.

Parameters

G [Obj] NetworkX graph with isolated nodes present.

Returns

G [Obj] NetworkX graph with isolated nodes pruned.

pruned_nodes [list] List of indices of nodes that were pruned from G.

References

[1], [2]

pynets.stats.netstats.**raw_mets**(*G*, *i*, *engine='nx'*)

API that iterates across NetworkX algorithms for a G.

Parameters

G [Obj] NetworkX graph.

i [str] Name of the NetworkX algorithm.

Returns

net_met_val [float] Value of the graph metric i that was calculated from G.

pynets.stats.netstats.**rich_club_coefficient**(*G*, *engine='nx'*)

pynets.stats.netstats.**save_netmets**(*dir_path*, *est_path*, *metric_list_names*, *net_met_val_list_final*)

pynets.stats.netstats.**smallworldness**(*G*, *niter=5*, *nrand=10*, *approach='clustering'*, *reference='lattice'*, *engine='nx'*)

Returns the small-world coefficient of a graph

The small-world coefficient of a G is:

omega/sigma = Lr/L - C/Cl

where C and L are respectively the average clustering coefficient/ transitivity and average shortest path length of G. Lr is the average shortest path length of an equivalent random graph and Cl is the average clustering coefficient/transitivity of an equivalent lattice/random graph.

Parameters

G [NetworkX graph] An undirected graph.

niter: integer (optional, default=5) Approximate number of rewiring per edge to compute the equivalent random graph.

nrand: integer (optional, default=10) Number of random graphs generated to compute the average clustering coefficient (Cr) and average shortest path length (Lr).

approach [str] Specifies whether to use clustering coefficient directly *clustering* or *transitivity* method of counting weighted triangles. Default is *clustering*.

reference [str] Specifies whether to use a random *random* or lattice *lattice* reference. Default is *lattice*.

Returns

omega/sigma [float] The smallworld coefficient

References

[1]

pynets.stats.netstats.**subgraph_number_of_cliques_for_all**(*G*)

Helper function, in the case of graph disconnectedness, that returns the number of cliques, calculated iteratively for each distinct subgraph of the G.

Parameters

G [Obj] NetworkX graph.

Returns

number of cliques [int] The average number of cliques for G.

References

[1], [2], [3]

pynets.stats.netstats.**weighted_transitivity**(*G*)

Compute weighted graph transitivity, the fraction of all possible weighted triangles present in G.

Possible triangles are identified by the number of "triads" (two edges with a shared vertex).

The transitivity is

$$T = 3\frac{\#triangles}{\#triads}.$$

Parameters

G [graph]

Returns

out [float] Transitivity

References

[1], [2]

Module contents

Submodules

Module contents

Top-level package for PyNets.

Bibliography

[1] Thomas Yeo, B. T., Krienen, F. M., Sepulcre, J., Sabuncu, M. R., Lashkari, D., Hollinshead, M., ... Buckner, R. L. (2011). The organization of the human cerebral cortex estimated by intrinsic functional connectivity. Journal of Neurophysiology. https://doi.org/10.1152/jn.00338.2011

[2] Schaefer A, Kong R, Gordon EM, Laumann TO, Zuo XN, Holmes AJ, Eickhoff SB, Yeo BTT. Local-Global parcellation of the human cerebral cortex from intrinsic functional connectivity MRI, Cerebral Cortex, 29:3095-3114, 2018.

[1] Tor D., W. (2011). NeuroSynth: a new platform for large-scale automated synthesis of human functional neuroimaging data. Frontiers in Neuroinformatics.

[1] Craddock, R. C., James, G. A., Holtzheimer, P. E., Hu, X. P., & Mayberg, H. S. (2012). A whole brain fMRI atlas generated via spatially constrained spectral clustering. Human Brain Mapping. https://doi.org/10.1002/hbm.21333

[2] N. Tzourio-Mazoyer; B. Landeau; D. Papathanassiou; F. Crivello; O. Etard; N. Delcroix; Bernard Mazoyer & M. Joliot (January 2002). "Automated Anatomical Labeling of activations in SPM using a Macroscopic Anatomical Parcellation of the MNI MRI single-subject brain". NeuroImage. 15 (1): 273–289. doi:10.1006/nimg.2001.0978.

[1] Complex network measures of brain connectivity: Uses and interpretations. Rubinov M, Sporns O (2010) NeuroImage 52:1059-69.

[1] Complex network measures of brain connectivity: Uses and interpretations. Rubinov M, Sporns O (2010) NeuroImage 52:1059-69.

[1] van Wijk, B. C. M., Stam, C. J., & Daffertshofer, A. (2010). Comparing brain networks of different size and connectivity density using graph theory. PLoS ONE. https://doi.org/10.1371/journal.pone.0013701

[2] Complex network measures of brain connectivity: Uses and interpretations. Rubinov M, Sporns O (2010) NeuroImage 52:1059-69.

[1] M. A. Serrano et al. (2009) Extracting the Multiscale backbone of complex weighted networks. PNAS, 106:16, pp. 6483-6488.

[1] M. A. Serrano et al. (2009) Extracting the Multiscale backbone of complex weighted networks. PNAS, 106:16, pp. 6483-6488.

[1] Complex network measures of brain connectivity: Uses and interpretations. Rubinov M, Sporns O (2010) NeuroImage 52:1059-69.

[1] Alexander-Bloch, A. F., Gogtay, N., Meunier, D., Birn, R., Clasen, L., Lalonde, F., ... Bullmore, E. T. (2010). Disrupted modularity and local connectivity of brain functional networks in childhood-onset schizophrenia. Frontiers in Systems Neuroscience. https://doi.org/10.3389/fnsys.2010.00147

[2] Tewarie, P., van Dellen, E., Hillebrand, A., & Stam, C. J. (2015). The minimum spanning tree: An unbiased method for brain network analysis. NeuroImage. https://doi.org/10.1016/j.neuroimage.2014.10.015

[1] Complex network measures of brain connectivity: Uses and interpretations. Rubinov M, Sporns O (2010) NeuroImage 52:1059-69.

[1] Fornito, A., Zalesky, A., & Bullmore, E. T. (2016). Fundamentals of Brain Network Analysis. In Fundamentals of Brain Network Analysis. https://doi.org/10.1016/C2012-0-06036-X

[1] Complex network measures of brain connectivity: Uses and interpretations. Rubinov M, Sporns O (2010) NeuroImage 52:1059-69.

[1] Complex network measures of brain connectivity: Uses and interpretations. Rubinov M, Sporns O (2010) NeuroImage 52:1059-69.

[1] van Wijk, B. C. M., Stam, C. J., & Daffertshofer, A. (2010). Comparing brain networks of different size and connectivity density using graph theory. PLoS ONE. https://doi.org/10.1371/journal.pone.0013701

[1] van Wijk, B. C. M., Stam, C. J., & Daffertshofer, A. (2010). Comparing brain networks of different size and connectivity density using graph theory. PLoS ONE. https://doi.org/10.1371/journal.pone.0013701

[1] Complex network measures of brain connectivity: Uses and interpretations. Rubinov M, Sporns O (2010) NeuroImage 52:1059-69.

[1] Complex network measures of brain connectivity: Uses and interpretations. Rubinov M, Sporns O (2010) NeuroImage 52:1059-69.

[1] Complex network measures of brain connectivity: Uses and interpretations. Rubinov M, Sporns O (2010) NeuroImage 52:1059-69.

[1] Chen, D. Q., Dell'Acqua, F., Rokem, A., Garyfallidis, E., Hayes, D., Zhong, J., & Hodaie, M. (2018). Diffusion Weighted Image Co-registration: Investigation of Best Practices. PLoS ONE.

[1] Aganj, I., et al. 2009. ODF Reconstruction in Q-Ball Imaging with Solid Angle Consideration.

[1] Tournier, J.D., et al. NeuroImage 2007. Robust determination of the fibre orientation distribution in diffusion MRI: Non-negativity constrained super-resolved spherical deconvolution

[2] Descoteaux, M., et al. IEEE TMI 2009. Deterministic and Probabilistic Tractography Based on Complex Fibre Orientation Distributions

[3] Côté, M-A., et al. Medical Image Analysis 2013. Tractometer: Towards validation of tractography pipelines

[4] Tournier, J.D, et al. Imaging Systems and Technology 2012. MRtrix: Diffusion Tractography in Crossing Fiber Regions

[1] Ariel Rokem, Jason D. Yeatman, Franco Pestilli, Kendrick N. Kay, Aviv Mezer, Stefan van der Walt, Brian A. Wandell (2015). Evaluating the accuracy of diffusion MRI models in white matter. PLoS ONE 10(4): e0123272. doi:10.1371/journal.pone.0123272

[2] Ariel Rokem, Kimberly L. Chan, Jason D. Yeatman, Franco Pestilli, Brian A. Wandell (2014). Evaluating the accuracy of diffusion models at multiple b-values with cross-validation. ISMRM 2014.

[1] Sporns, O., Tononi, G., & Kötter, R. (2005). The human connectome: A structural description of the human brain. PLoS Computational Biology. https://doi.org/10.1371/journal.pcbi.0010042

[2] Sotiropoulos, S. N., & Zalesky, A. (2019). Building connectomes using diffusion MRI: why, how and but. NMR in Biomedicine. https://doi.org/10.1002/nbm.3752

[3] Chung, M. K., Hanson, J. L., Adluru, N., Alexander, A. L., Davidson, R. J., & Pollak, S. D. (2017). Integrative Structural Brain Network Analysis in Diffusion Tensor Imaging. Brain Connectivity. https://doi.org/10.1089/brain.2016.0481

[1] Basser PJ, Mattielo J, LeBihan (1994). MR diffusion tensor spectroscopy and imaging.

[2] Pajevic S, Pierpaoli (1999). Color schemes to represent the orientation of anisotropic tissues from diffusion tensor data: application to white matter fiber tract mapping in the human brain.

[1] Zhang, Y., Brady, M. and Smith, S. Segmentation of Brain MR Images Through a Hidden Markov Random Field Model and the Expectation-Maximization Algorithm IEEE Transactions on Medical Imaging, 20(1): 45-56, 2001

[2] Avants, B. B., Tustison, N. J., Wu, J., Cook, P. A. and Gee, J. C. An open source multivariate framework for n-tissue segmentation with evaluation on public data. Neuroinformatics, 9(4): 381-400, 2011.

[1] Soares, J. M., Marques, P., Alves, V., & Sousa, N. (2013). A hitchhiker's guide to diffusion tensor imaging. Frontiers in Neuroscience. https://doi.org/10.3389/fnins.2013.00031

[1] Takemura, H., Caiafa, C. F., Wandell, B. A., & Pestilli, F. (2016). Ensemble Tractography. PLoS Computational Biology. https://doi.org/10.1371/journal.pcbi.1004692

[1] Stella Yu and Jianbo Shi, "Understanding Popout through Repulsion," Computer Vision and Pattern Recognition, December, 2001.

[2] Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), 888-905. doi: 10.1109/34.868688.

[3] Yu, S. X., & Shi, J. (2003). Multiclass spectral clustering. Proceedings Ninth IEEE International Conference on Computer Vision, (1), 313-319 vol.1. Ieee. doi: 10.1109/ICCV.2003.1238361

[1] Craddock, R. C., James, G. A., Holtzheimer, P. E., Hu, X. P., & Mayberg, H. S. (2012). A whole brain fMRI atlas generated via spatially constrained spectral clustering. Human Brain Mapping. https://doi.org/10.1002/hbm.21333

[1] Craddock, R. C., James, G. A., Holtzheimer, P. E., Hu, X. P., & Mayberg, H. S. (2012). A whole brain fMRI atlas generated via spatially constrained spectral clustering. Human Brain Mapping. https://doi.org/10.1002/hbm.21333

[1] Stella Yu and Jianbo Shi, "Understanding Popout through Repulsion," Computer Vision and Pattern Recognition, December, 2001.

[2] Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), 888-905. doi: 10.1109/34.868688.

[3] Yu, S. X., & Shi, J. (2003). Multiclass spectral clustering. Proceedings Ninth IEEE International Conference on Computer Vision, (1), 313-319 vol.1. Ieee. doi: 10.1109/ICCV.2003.1238361

[1] Craddock, R. C., James, G. A., Holtzheimer, P. E., Hu, X. P., & Mayberg, H. S. (2012). A whole brain fMRI atlas generated via spatially constrained spectral clustering. Human Brain Mapping. https://doi.org/10.1002/hbm.21333

[1] Varoquaux, G., & Craddock, R. C. (2013). Learning and comparing functional connectomes across subjects. NeuroImage. https://doi.org/10.1016/j.neuroimage.2013.04.007

[2] Jason Laska, Manjari Narayan, 2017. skggm 0.2.7: A scikit-learn compatible package for Gaussian and related Graphical Models. doi:10.5281/zenodo.830033

[1] P. Bellec; G. Marrelec; H. Benali, A bootstrap test to investigate changes in brain connectivity for functional MRI. Statistica Sinica, special issue on Statistical Challenges and Advances in Brain Science, 2008, 18: 1253-1268.

[1] Rosenthal, G., Váša, F., Griffa, A., Hagmann, P., Amico, E., Goñi, J., Sporns, O. (2018). Mapping higher-order relations between brain structure and function with embedded vector representations of connectomes. Nature Communications. https://doi.org/10.1038/s41467-018-04614-w

[1] Liu, Y., He, L., Cao, B., Yu, P. S., Ragin, A. B., & Leow, A. D. (2018). Multi-view multi-graph embedding for brain network clustering analysis. 32nd AAAI Conference on Artificial Intelligence, AAAI 2018.

[2] Levin, K., Athreya, A., Tang, M., Lyzinski, V., & Priebe, C. E. (2017, November). A central limit theorem for an omnibus embedding of multiple random dot product graphs. In Data Mining Workshops (ICDMW), 2017 IEEE International Conference on (pp. 964-967). IEEE.

[1] Battiston, F., Nicosia, V., Chavez, M., & Latora, V. (2017). Multilayer motif analysis of brain networks. Chaos. https://doi.org/10.1063/1.4979282

[1] Bullmore, E., & Sporns, O. (2009). Complex brain networks: Graph theoretical analysis of structural and functional systems. Nature Reviews Neuroscience. https://doi.org/10.1038/nrn2575

[2] Vaiana, M., & Muldoon, S. F. (2018). Multilayer Brain Networks. Journal of Nonlinear Science. https://doi.org/10.1007/s00332-017-9436-8

[1] R. Amato, N. E Kouvaris, M. San Miguel and A. Diaz-Guilera, Opinion competition dynamics on multiplex networks, New J. Phys. DOI: https://doi.org/10.1088/1367-2630/aa936a

[2] N. E. Kouvaris, S. Hata and A. Diaz-Guilera, Pattern formation in multiplex networks, Scientific Reports 5, 10840 (2015). http://www.nature.com/srep/2015/150604/srep10840/full/srep10840.html

[3] A. Sole-Ribata, M. De Domenico, N. E. Kouvaris, A. Diaz-Guilera, S. Gomez and A. Arenas, Spectral properties of the Laplacian of a multiplex network, Phys. Rev. E 88, 032807 (2013). http://journals.aps.org/pre/abstract/10.1103/PhysRevE.88.032807

[1] Battiston, F., Nicosia, V., Chavez, M., & Latora, V. (2017). Multilayer motif analysis of brain networks. Chaos. https://doi.org/10.1063/1.4979282

[1] Sporns, O., & Kötter, R. (2004). Motifs in Brain Networks. PLoS Biology. https://doi.org/10.1371/journal.pbio.0020369

[1] Qin, Tai, and Karl Rohe. "Regularized spectral clustering under the degree-corrected stochastic blockmodel." In Advances in Neural Information Processing Systems, pp. 3120-3128. 2013

[2] Rohe, Karl, Tai Qin, and Bin Yu. "Co-clustering directed graphs to discover asymmetries and directional communities." Proceedings of the National Academy of Sciences 113.45 (2016): 12679-12684.

[1] Latora, V., and Marchiori, M. (2001). Efficient behavior of small-world networks. Physical Review Letters 87.

[2] Latora, V., and Marchiori, M. (2003). Economic small-world behavior in weighted networks. Eur Phys J B 32, 249-263.

[1] Drakesmith, M., Caeyenberghs, K., Dutt, A., Lewis, G., David, A. S., & Jones, D. K. (2015). Overcoming the effects of false positives and threshold bias in graph theoretical analyses of neuroimaging data. NeuroImage. https://doi.org/10.1016/j.neuroimage.2015.05.011

[1] Newman, M.E.J. & Girvan, M. Finding and evaluating community structure in networks. Physical Review E 69, 26113(2004).

[2] Blondel, V.D. et al. Fast unfolding of communities in large networks. J. Stat. Mech 10008, 1-12(2008).

[1] Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. NeuroImage, 52, 1059-1069.

[1] Fornito, A., Zalesky, A., & Bullmore, E. T. (2016). Fundamentals of Brain Network Analysis. In Fundamentals of Brain Network Analysis. https://doi.org/10.1016/C2012-0-06036-X

[2] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, "Exploring network structure, dynamics, and function using NetworkX", in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008

[1] Latora, V., and Marchiori, M. (2001). Efficient behavior of small-world networks. Physical Review Letters 87.

[2] Latora, V., and Marchiori, M. (2003). Economic small-world behavior in weighted networks. Eur Phys J B 32, 249-263.

[1] de Reus, M. A., Saenger, V. M., Kahn, R. S., & van den Heuvel, M. P. (2014). An edge-centric perspective on the human connectome: Link communities in the brain. Philosophical Transactions of the Royal Society B: Biological Sciences. https://doi.org/10.1098/rstb.2013.0527

[1] Latora, V., and Marchiori, M. (2001). Efficient behavior of small-world networks. Physical Review Letters 87.

[2] Latora, V., and Marchiori, M. (2003). Economic small-world behavior in weighted networks. Eur Phys J B 32, 249-263.

[1] Power, J. D., Schlaggar, B. L., Lessov-Schlaggar, C. N., & Petersen, S. E. (2013). Evidence for hubs in human functional brain networks. Neuron. https://doi.org/10.1016/j.neuron.2013.07.035

[1] Guimera, R., & Amaral, L. A. N. (2005). Functional cartography of complex metabolic networks. Nature, 433, 895-900.

[2] Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. NeuroImage, 52, 1059-1069.

[1] Guimera, R., & Amaral, L. A. N. (2005). Functional cartography of complex metabolic networks. Nature, 433, 895-900.

[2] Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. NeuroImage, 52, 1059-1069.

[1] Hayasaka, S. (2017). Anti-Fragmentation of Resting-State Functional Magnetic Resonance Imaging Connectivity Networks with Node-Wise Thresholding. Brain Connectivity. https://doi.org/10.1089/brain.2017.0523

[2] Fornito, A., Zalesky, A., & Bullmore, E. T. (2016). Fundamentals of Brain Network Analysis. In Fundamentals of Brain Network Analysis. https://doi.org/10.1016/C2012-0-06036-X

[1] Telesford, Joyce, Hayasaka, Burdette, and Laurienti (2011). "The Ubiquity of Small-World Networks". Brain Connectivity. 1 (0038): 367-75. PMC 3604768. PMID 22432451. doi:10.1089/brain.2011.0038.

[1] Bron, C. and Kerbosch, J. "Algorithm 457: finding all cliques of an undirected graph". *Communications of the ACM* 16, 9 (Sep. 1973), 575–577. <http://portal.acm.org/citation.cfm?doid=362342.362367>

[2] Etsuji Tomita, Akira Tanaka, Haruhisa Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments", *Theoretical Computer Science*, Volume 363, Issue 1, Computing and Combinatorics, 10th Annual International Conference on Computing and Combinatorics (COCOON 2004), 25 October 2006, Pages 28-42 <https://doi.org/10.1016/j.tcs.2006.06.015>

[3] F. Cazals, C. Karande, "A note on the problem of reporting maximal cliques", *Theoretical Computer Science*, Volume 407, Issues 1–3, 6 November 2008, Pages 564–568, <https://doi.org/10.1016/j.tcs.2008.05.010>

[1] Wasserman, S., and Faust, K. (1994). Social Network Analysis: Methods and Applications. Cambridge: Cambridge University Press.

[2] Alain Barrat, Marc Barthelemy, Romualdo Pastor-Satorras, Alessandro Vespignani: The architecture of complex weighted networks, Proc. Natl. Acad. Sci. USA 101, 3747 (2004)

Index

A

adaptivethresh() (*in module pynets.stats.netmotifs*), 62
add_hp_columns() (*pynets.core.utils.build_sql_db method*), 32
add_row_from_df() (*pynets.core.utils.build_sql_db method*), 32
as_directory() (*in module pynets.core.utils*), 31
as_list() (*in module pynets.core.utils*), 32
autofix() (*in module pynets.core.thresholding*), 22
average_local_efficiency() (*in module pynets.stats.netstats*), 65
average_shortest_path_length_fast() (*in module pynets.stats.netstats*), 65
average_shortest_path_length_for_all() (*in module pynets.stats.netstats*), 65

B

benchmark_reproducibility() (*in module pynets.stats.benchmarking*), 61
beta_lin_comb() (*in module pynets.stats.benchmarking*), 61
binarize() (*in module pynets.core.thresholding*), 23
binarize_graph() (*pynets.stats.netstats.CleanGraphs method*), 64
build_args_from_config() (*in module pynets.core.utils*), 32
build_asetomes() (*in module pynets.stats.embeddings*), 62
build_masetome() (*in module pynets.stats.embeddings*), 62
build_multigraphs() (*in module pynets.stats.netmotifs*), 63
build_mx_multigraph() (*in module pynets.stats.netmotifs*), 63
build_omnetome() (*in module pynets.stats.embeddings*), 62
build_sql_db (*class in pynets.core.utils*), 32

C

check_est_path_existence() (*in module pynets.core.utils*), 32
check_template_loads() (*in module pynets.core.utils*), 32
checkConsecutive() (*in module pynets.core.utils*), 32
CleanGraphs (*class in pynets.stats.netstats*), 64
collect_pandas_df() (*in module pynets.core.utils*), 32
collect_pandas_df_make() (*in module pynets.stats.netstats*), 65
collectpandasjoin() (*in module pynets.core.utils*), 33
community_resolution_selection() (*in module pynets.stats.netstats*), 66
compare_motifs() (*in module pynets.stats.netmotifs*), 63
coords_masker() (*in module pynets.core.nodemaker*), 15
countmotifs() (*in module pynets.stats.netmotifs*), 63
create_anisopowermap() (*in module pynets.dmri.estimation*), 42
create_clean_mask() (*pynets.fmri.clustools.NiParcellate method*), 49
create_communities() (*in module pynets.stats.netstats*), 66
create_csv_path() (*in module pynets.core.utils*), 33
create_density_map() (*in module pynets.dmri.track*), 46
create_est_path_diff() (*in module pynets.core.utils*), 33
create_est_path_func() (*in module pynets.core.utils*), 34
create_gb_palette() (*in module pynets.plotting.plot_gen*), 55
create_length_matrix() (*pynets.stats.netstats.CleanGraphs method*), 64

create_local_clustering() (pynets.fmri.clustools.NiParcellate method), 49
create_modality_table() (pynets.core.utils.build_sql_db method), 32
create_parcel_atlas() (in module pynets.core.nodemaker), 15
create_raw_path_diff() (in module pynets.core.utils), 34
create_raw_path_func() (in module pynets.core.utils), 35
create_spherical_roi_volumes() (in module pynets.core.nodemaker), 16
csa_mod_est() (in module pynets.dmri.estimation), 42
csd_mod_est() (in module pynets.dmri.estimation), 43

D

decompress_nifti() (in module pynets.core.utils), 35
density_thresholding() (in module pynets.core.thresholding), 23
discr_stat() (in module pynets.stats.benchmarking), 61
discretisation() (in module pynets.fmri.clustools), 50
disparity_filter() (in module pynets.core.thresholding), 23
disparity_filter_alpha_cut() (in module pynets.core.thresholding), 24
diversity_coef_sign() (in module pynets.stats.netstats), 66
dmri_connectometry() (in module pynets.core.workflows), 41
do_dir_path() (in module pynets.core.utils), 35
drop_badixs_from_parcellation() (in module pynets.core.nodemaker), 16
drop_coords_labels_from_restricted_parcellation() (in module pynets.core.nodemaker), 16
dumpstacks() (in module pynets.core.utils), 35

E

enforce_hem_distinct_consecutive_labels() (in module pynets.core.nodemaker), 16
ensemble_parcellate() (in module pynets.fmri.clustools), 50
est_density() (in module pynets.core.thresholding), 24
extract_ts_parc() (pynets.fmri.estimation.TimeseriesExtraction method), 53
extractnetstats() (in module pynets.stats.netstats), 66

F

fetch_nilearn_atlas_coords() (in module pynets.core.nodemaker), 17
fill_confound_nans() (in module pynets.fmri.estimation), 53
filter_cols_from_targets() (in module pynets.core.utils), 35
flatten() (in module pynets.core.utils), 36
fmri_connectometry() (in module pynets.core.workflows), 41

G

gen_img_list() (in module pynets.core.nodemaker), 17
gen_network_parcels() (in module pynets.core.nodemaker), 17
get_betweenness_centrality() (in module pynets.stats.netstats), 67
get_brainnetome_node_attributes() (in module pynets.core.nodemaker), 17
get_clustering() (in module pynets.stats.netstats), 67
get_comm_centrality() (in module pynets.stats.netstats), 67
get_community() (in module pynets.stats.netstats), 67
get_conn_matrix() (in module pynets.fmri.estimation), 53
get_degree_centrality() (in module pynets.stats.netstats), 67
get_diversity() (in module pynets.stats.netstats), 67
get_eigen_centrality() (in module pynets.stats.netstats), 67
get_file() (in module pynets.core.utils), 36
get_local_efficiency() (in module pynets.stats.netstats), 67
get_names_and_coords_of_parcels() (in module pynets.core.nodemaker), 17
get_node_membership() (in module pynets.core.nodemaker), 18
get_optimal_cov_estimator() (in module pynets.fmri.estimation), 55
get_participation() (in module pynets.stats.netstats), 67
get_prop_type() (in module pynets.stats.netstats), 67
get_rich_club_coeff() (in module pynets.stats.netstats), 67
get_sphere() (in module pynets.core.nodemaker), 19
get_template_tf() (in module pynets.core.utils), 36
global_efficiency() (in module pynets.stats.netstats), 67

I

indx_1dto3d() (*in module pynets.fmri.clustools*), 50
indx_3dto1d() (*in module pynets.fmri.clustools*), 50
invert() (*in module pynets.core.thresholding*), 24
iterate_nx_global_measures() (*in module pynets.stats.netstats*), 68

K

kill_process_family() (*in module pynets.core.utils*), 36
knn() (*in module pynets.core.thresholding*), 24

L

link_communities() (*in module pynets.stats.netstats*), 68
load_mat() (*in module pynets.core.utils*), 36
load_mat_ext() (*in module pynets.core.utils*), 36
load_runconfig() (*in module pynets.core.utils*), 36
local_efficiency() (*in module pynets.stats.netstats*), 68
local_thresholding_prop() (*in module pynets.core.thresholding*), 25

M

make_local_connectivity_scorr() (*in module pynets.fmri.clustools*), 51
make_local_connectivity_tcorr() (*in module pynets.fmri.clustools*), 51
mask_roi() (*in module pynets.core.nodemaker*), 19
merge_dicts() (*in module pynets.core.utils*), 36
mergedicts() (*in module pynets.core.utils*), 36
missing_elements() (*in module pynets.core.utils*), 36
mmToVox() (*in module pynets.core.nodemaker*), 19
most_important() (*in module pynets.stats.netstats*), 68
motif_matching() (*in module pynets.stats.netmotifs*), 64
multigraph_matching() (*in module pynets.stats.netmotifs*), 64

N

ncut() (*in module pynets.fmri.clustools*), 51
nilearn_atlas_helper() (*in module pynets.core.nodemaker*), 19
NiParcellate (*class in pynets.fmri.clustools*), 49
node_gen() (*in module pynets.core.nodemaker*), 20
node_gen_masking() (*in module pynets.core.nodemaker*), 20
normalize() (*in module pynets.core.thresholding*), 25
normalize_graph() (*pynets.stats.netstats.CleanGraphs method*), 64
np2gt() (*in module pynets.stats.netstats*), 69
nx2gt() (*in module pynets.stats.netstats*), 69

P

parcel_masker() (*in module pynets.core.nodemaker*), 21
parcel_naming() (*in module pynets.core.nodemaker*), 22
parcellate() (*in module pynets.fmri.clustools*), 52
parcellate_ncut() (*in module pynets.fmri.clustools*), 52
participation_coef() (*in module pynets.stats.netstats*), 69
participation_coef_sign() (*in module pynets.stats.netstats*), 69
pass_meta_ins() (*in module pynets.core.utils*), 36
pass_meta_ins_multi() (*in module pynets.core.utils*), 37
pass_meta_outs() (*in module pynets.core.utils*), 38
perform_thresholding() (*in module pynets.core.thresholding*), 25
plot_all_func() (*in module pynets.plotting.plot_gen*), 56
plot_all_struct() (*in module pynets.plotting.plot_gen*), 56
plot_all_struct_func() (*in module pynets.plotting.plot_gen*), 57
plot_community_conn_mat() (*in module pynets.plotting.plot_graphs*), 58
plot_conn_mat() (*in module pynets.plotting.plot_graphs*), 59
plot_conn_mat_func() (*in module pynets.plotting.plot_graphs*), 59
plot_conn_mat_struct() (*in module pynets.plotting.plot_graphs*), 60
plot_graph_measure_hists() (*in module pynets.plotting.plot_gen*), 58
plot_network_clusters() (*in module pynets.plotting.plot_gen*), 58
plot_timeseries() (*in module pynets.plotting.plot_gen*), 58
prep_boot() (*pynets.fmri.clustools.NiParcellate method*), 50
prep_tissues() (*in module pynets.dmri.track*), 47
prepare_inputs() (*pynets.fmri.estimation.TimeseriesExtraction method*), 53
print_summary() (*pynets.stats.netstats.CleanGraphs method*), 65
proportional() (*in module pynets.core.utils*), 39
prune_disconnected() (*in module pynets.stats.netstats*), 70
prune_graph() (*pynets.stats.netstats.CleanGraphs method*), 65
prune_suffices() (*in module pynets.core.utils*), 39
pynets (*module*), 71

pynets.core (*module*), 42
pynets.core.nodemaker (*module*), 15
pynets.core.thresholding (*module*), 22
pynets.core.utils (*module*), 31
pynets.core.workflows (*module*), 41
pynets.dmri (*module*), 49
pynets.dmri.estimation (*module*), 42
pynets.dmri.track (*module*), 46
pynets.fmri (*module*), 55
pynets.fmri.clustools (*module*), 49
pynets.fmri.estimation (*module*), 52
pynets.plotting (*module*), 60
pynets.plotting.plot_gen (*module*), 55
pynets.plotting.plot_graphs (*module*), 58
pynets.registration (*module*), 60
pynets.stats (*module*), 71
pynets.stats.benchmarking (*module*), 61
pynets.stats.embeddings (*module*), 62
pynets.stats.netmotifs (*module*), 62
pynets.stats.netstats (*module*), 64

R

raw_graph_workflow() (*in module pynets.core.workflows*), 41
raw_mets() (*in module pynets.stats.netstats*), 70
reconstruction() (*in module pynets.dmri.track*), 47
rich_club_coefficient() (*in module pynets.stats.netstats*), 70
run() (*pynets.core.utils.watchdog method*), 41
run_tracking() (*in module pynets.dmri.track*), 48

S

save_3d_to_4d() (*in module pynets.core.utils*), 39
save_4d_to_3d() (*in module pynets.core.utils*), 39
save_and_cleanup() (*pynets.fmri.estimation.TimeseriesExtraction method*), 53
save_coords_and_labels_to_json() (*in module pynets.core.utils*), 39
save_mat() (*in module pynets.core.utils*), 39
save_mat_thresholded() (*in module pynets.core.utils*), 40
save_netmets() (*in module pynets.stats.netstats*), 70
save_nifti_parcels_map() (*in module pynets.core.utils*), 40
save_ts_to_file() (*in module pynets.core.utils*), 40
sfm_mod_est() (*in module pynets.dmri.estimation*), 43
show_template_bundles() (*in module pynets.plotting.plot_gen*), 58
smallworldness() (*in module pynets.stats.netstats*), 70
standardize() (*in module pynets.core.thresholding*), 25
streams2graph() (*in module pynets.dmri.estimation*), 44
subgraph_number_of_cliques_for_all() (*in module pynets.stats.netstats*), 71

T

tens_mod_est() (*in module pynets.dmri.estimation*), 46
tens_mod_fa_est() (*in module pynets.dmri.estimation*), 46
thr2prob() (*in module pynets.core.thresholding*), 26
thresh_func() (*in module pynets.core.thresholding*), 26
thresh_raw_graph() (*in module pynets.core.thresholding*), 28
thresh_struct() (*in module pynets.core.thresholding*), 28
threshold_absolute() (*in module pynets.core.thresholding*), 30
threshold_proportional() (*in module pynets.core.thresholding*), 30
timeout() (*in module pynets.core.utils*), 40
timeseries_bootstrap() (*in module pynets.fmri.estimation*), 55
TimeseriesExtraction (*class in pynets.fmri.estimation*), 52
track_ensemble() (*in module pynets.dmri.track*), 48

V

view_tractogram() (*in module pynets.plotting.plot_gen*), 58
VoxTomm() (*in module pynets.core.nodemaker*), 15

W

watchdog (*class in pynets.core.utils*), 41
weight_conversion() (*in module pynets.core.thresholding*), 30
weight_to_distance() (*in module pynets.core.thresholding*), 31
weighted_transitivity() (*in module pynets.stats.netstats*), 71
workflow_selector() (*in module pynets.core.workflows*), 41