pynets Documentation

Release 1.30

The PyNets developers

Jun 02, 2022
# CONTENTS

## 1 PyNets®

1.1 About ................................................................. 1
1.2 Documentation ....................................................... 1
1.3 Citing ..................................................................... 1
1.4 Contents .................................................................. 1

1.4.1 Installation ......................................................... 1
1.4.2 Hardware Requirements ........................................ 2
1.4.3 Usage .................................................................. 4
1.4.4 Credits .................................................................. 21
1.4.5 Contributing ......................................................... 22
1.4.6 API ..................................................................... 24

## Bibliography

97

## Index

103
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


1.3 Citing

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


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 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’
- ‘coors_power_2011’ (only valid when using the `-spheres` flag)
- ‘coors_dosenbach_2010’ (only valid when using the `-spheres` flag)
- ‘atlas_msd’
- ‘atlas_pauli_2017’
- ‘destrieux2009_rois’
- ‘BrainnetomeAtlasFan2016’
- ‘VoxelwiseParcellation0515kLeadDBS’
- ‘Juelichgmth252mmEickhoff2005’
- ‘CorticalAreaParcellationfromRestingStateCorrelationsGordon2014’
- ‘AICHAreorderedJoliot2015’
- ‘HarvardOxfordThr252mmWholeBrainMakris2006’
- ‘VoxelwiseParcellation058kLeadDBS’
- ‘MICCAI2012MultiAtlasLabelingWorkshopandChallengeNeuromorphometrics’
- ‘Hammers_mithAtlasn30r83Hammers2003Gousias2008’
- ‘AALTzourioMazoyer2002’
- ‘DesikanKlein2012’
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 advanced.yml advanced settings to override MNI152 (e.g. a Pediatric template), following the naming spec of templateflow (See: [https://github.com/templateflow/templateflow](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 advanced.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 [deep-brain](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

PyNets: A Reproducible Workflow for Structural and Functional Connectome Ensemble Learning

usage: pynets [-h] -id A subject id or other unique identifier
  [A subject id or other unique identifier ...]
  [-func Path to input functional file required for functional connectomes]
  [Path to input functional file (required for functional connectomes) ...]]
  [-dwi Path to diffusion-weighted imaging data file (required for dMRI connectomes)]
  [Path to diffusion-weighted imaging data file (required for dMRI connectomes) ...]]
  [-connectomes] [Path to diffusion-weighted imaging data file (required for dMRI connectomes) ...]]
  [-bval Path to b-values file (required for dMRI connectomes) [Path to b-values file (required for dMRI connectomes) ...]]
  [-bvec Path to b-vectors file (required for dMRI connectomes) [Path to b-vectors file (required for dMRI connectomes) ...]]
  [-anat Path to a skull-stripped anatomical Nifti1Image [Path to a skull-stripped anatomical Nifti1Image ...]]
  [-m Path to a T1w brain mask image (if available) in native anatomical space [Path to a T1w brain mask image (if available) in native anatomical space ...]]
  [-conf Confound regressor file (.tsv/.csv format) [Confound regressor file (.tsv/.csv format) ...]]
  [-g Path to graph file input. [Path to graph file input. ...]]
  [-roi Path to binarized Region-of-Interest (ROI) Nifti1Image in template MNI space. [Path to binarized Region-of-Interest (ROI) Nifti1Image in template MNI space. ...]]
  [-ref Atlas reference file path]
  [-way Path to binarized Nifti1Image to constrain tractography [Path to ...]
  [-mod Connectivity estimation/reconstruction method [Connectivity estimation/reconstruction method ...]]
  [-a Atlas [Atlas ...]]
  [-ns Spherical centroid node size [Spherical centroid node size ...]]

(continues on next page)
[-thr Graph threshold]
[-min_thr Multi-thresholding minimum threshold]
[-max_thr Multi-thresholding maximum threshold]
[-step_thr Multi-thresholding step size]
[-hp High-pass filter (Hz) [High-pass filter (Hz) ...]]
[-es Node signal extraction strategy [Node signal extraction strategy ...]]
[-k Number of k clusters [Number of k clusters ...]]
[-ct Clustering type [Clustering type ...]]
[-cm Cluster mask [Cluster mask ...]]
[-sm Smoothing margin (mm FWHM) [Smoothing margin (mm FWHM) ...]]
[-ml Minimum fiber length for tracking [Minimum fiber length for tracking .
...
...]]
[-dg Traversal strategy [Traversal strategy ...]]
[-em Error margin (mm) [Error margin (mm) ...]]
[-norm Normalization strategy for resulting graph(s) [-bin]
[-mplx Perform various levels of multiplex graph analysis (only if both.
...structural and diffusion connectometry is run simultaneously.] [-embed] [-spheres]
[-n Resting-state subnet [Resting-state subnet ...]]
[-vox {1mm,2mm}] [-plt] [-pm Cores,memory]
[-plug Scheduler type] [-v] [-noclean]
[-config Advanced configuration file] [-work Working directory]
[-version]
output_dir

Positional Arguments

**output_dir**  The directory to store pynets derivatives.

Default: “/home/docs”

Named Arguments

**-id**  An subject identifier OR list of subject identifiers, separated by space and of equivalent length to the list of input files indicated with the -func flag. This parameter must be an alphanumeric string and can be arbitrarily chosen. If functional and dmri connectomes are being generated simultaneously, then space-separated id’s need to be repeated to match the total input file count.

**-func**  Specify either a path to a preprocessed functional Nifti1Image in MNI152 space OR multiple space-separated paths to multiple preprocessed functional Nifti1Image files in MNI152 space and in .nii or .nii.gz format, OR the path to a text file containing a list of paths to subject files.

**-dwi**  Specify either a path to a preprocessed dmri diffusion Nifti1Image in native diffusion space and in .nii or .nii.gz format OR multiple space-separated paths to multiple preprocessed dmri diffusion Nifti1Image files in native diffusion space and in .nii or .nii.gz format.

**-bval**  Specify either a path to a b-values text file containing gradient shell values per diffusion direction OR multiple space-separated paths to multiple b-values text
files in the order of accompanying b-vectors and dwi files.

-bvec
Specify either a path to a b-vectors text file containing gradient directions (x,y,z) per diffusion direction OR multiple space-separated paths to multiple b-vectors text files in the order of accompanying b-values and dwi files.

-anat
Required for dmri and/or functional connectomes. Multiple paths to multiple anatomical files should be specified by space in the order of accompanying functional and/or dmri files. If functional and dmri connectomes are both being generated simultaneously, then anatomical Nifti1Image file paths need to be repeated, but separated by comma.

-m
File path to a T1w brain mask Nifti image (if available) in native anatomical space OR multiple file paths to multiple T1w brain mask Nifti images in the case of running multiple participants, in which case paths should be separated by a space. If no brain mask is supplied, the template mask will be used (see advanced.yaml).

-conf
Optionally specify a path to a confound regressor file to reduce noise in the time-series estimation for the graph. This can also be a list of paths in the case of running multiplicesubjects, which requires separation by space and of equivalent length to the list of input files indicated with the -func flag.

-g
In either .txt, .npy, .graphml, .csv, .ssv, .tsv, or .gpickle format. This skips fMRI and dMRI graph estimation workflows and begins at the thresholding and graph analysis stage. Multiple graph files corresponding to multiple subject ID’s should be separated by space, and multiple graph files corresponding to the same subject ID should be separated by comma. If the -g flag is used, then the -id flag must also be used. Consider also including -thr flag to activate thresholding only or the -p and -norm flags if graph defragmentation or normalization is desired. The -mod flag can be used for additional provenance/file-naming.

-roi
Optionally specify a binarized ROI mask and retain only those nodes of a parcellation contained within that mask for connectome estimation.

-ref
Specify the path to the atlas reference .txt file that maps labels to intensities corresponding to the atlas parcellation file specified with the -a flag.

-way
Optionally specify a binarized ROI mask in MNI-space toconstrain tractography in the case of dmri connectome estimation.

-mod
Possible choices: corr, sps, cov, partcorr, QuicGraphicalLasso, QuicGraphicalLassoCV, QuicGraphicalLassoEBIC, AdaptiveQuicGraphicalLasso, csa, csd, sfm, mcsd
(hyperparameter): Specify connectivity estimation model. For fMRI, possible models include: corr for correlation, cov for covariance, sps for precision covariance, partcorr for partial correlation. If skgmm is installed (https://github.com/skggm/skggm), then QuicGraphicalLasso, QuicGraphicalLassoCV, QuicGraphicalLassoEBIC, and AdaptiveQuicGraphicalLasso. For dMRI, current models include csa, csd, sfm, and mcsd (for multishell data).
Default: “?”

-a
(hyperparameter): Specify an atlas name from nilearn or local (pynets) library, and/or specify a path to a custom parcellation/atlas Nifti1Image file in MNI space. Labels should be spatially distinct across hemispheres and ordered with consecutive integers with a value of 0 as the background label. If specifying a list of paths to multiple parcellations, separate them by space. If you wish to iterate your pynets run over multiple atlases, separate them by space. Available nilearn atlases are:
Available local atlases are:

- destrieux2009_rois
- BrainnetomeAtlasFan2016
- VoxelwiseParcellation0515kLeadDBS
- JuellichgmthThr252mmEickhoff2005
- CorticalAreaParcellationfromRestingStateCorrelationsGordon2014
- whole_brain_cluster_labels_PCA100
- AICHAreorderedJoliot2015
- HarvardOxfordThr252mmWholeBrainMakris2006
- VoxelwiseParcellation058kLeadDBS
- MICCAI2012MultiAtlasLabelingWorkshopandChallengeNeuromorphometrics
- Hammers_mithAtlasn30r83Hammers2003
- Gousias2008
- AALTzourioMazoyer2002
- DesikanKlein2012
- AAL2zourioMazoyer2002
- VoxelwiseParcellation0435kLeadDBS
- AICHAJoliot2015
- whole_brain_cluster_labels_PCA200
- RandomParcellationsc05meanalll43Craddock2011

-ns (hyperparameter): Optionally specify coordinate-based node radius size(s). Default is 4 mm for fMRI and 8 mm for dMRI. If you wish to iterate the pipeline across multiple node sizes, separate the list by space (e.g. 2 4 6).

Default: 4

-thr Optionally specify a threshold indicating a proportion of weights to preserve in the graph. Default is no thresholding. If -mst, -dt, or -df flags are not included, than proportional thresholding will be performed

Default: 1.0

-min_thr (hyperparameter): Minimum threshold for multi-thresholding.

-max_thr (hyperparameter): Maximum threshold for multi-thresholding.

-step_thr (hyperparameter): Threshold step value for multi-thresholding. Default is 0.01.

-hp (hyperparameter): Optionally specify high-pass filter values to apply to node-extracted time-series for fMRI. Default is None. If you wish to iterate the pipeline across multiple values, separate the list by space (e.g. 0 0.02 0.1). Safe range: [0-0.15] for resting-state data.

-es Possible choices: sum, mean, median, minimum, maximum, variance, standard_deviation

Include this flag if you are running functional connectometry and wish to specify the name of a specific function (i.e. other than the mean) to reduce the region’s time-series. Options are: sum, mean, median, minimum, maximum, variance, standard_deviation.

Default: “mean”

-k (hyperparameter): Specify a number of clusters to produce. If you wish to iterate the pipeline across multiple values of k, separate the list by space (e.g. 200, 400, 600, 800).

-ct Possible choices: ward, rena, kmeans, complete, average, single, ncut

(hyperparameter): Specify the types of clustering to use. Recommended options are: ward, rena, kmeans, or ncut. Note that imposing spatial constraints with a mask consisting of disconnected components will leading to clustering instability in the case of complete, average, or single clustering. If specifying list of clustering types, separate them by space.
Default: “ward”

**-cm**

(hyperparameter): Specify the path to a Nifti Image mask file to constrained functional clustering. If specifying a list of paths to multiple cluster masks, separate them by space.

**-sm**

(hyperparameter): Intersection distance in mm of a node to an edge. Corresponds to the magnitude of smoothing to be applied to the node-extracted time-series. Default is 0 mm FWHM. Safe range for fMRI is: [0-9]. If you wish to iterate the pipeline across multiple smoothing delimit the list by space (e.g. 2 4 6)

Default: 0

**-ml**

(hyperparameter): Include this flag to manually specify a minimum tract length (mm) for dMRI connectome tracking. Default is 10. If you wish to iterate the pipeline across multiple minimums, separate the list by space (e.g. 10 30 50). Safe range: [0-150]. Depending on the tissue classifier used and the restrictiveness of the parcellation or any way-masking, values >60mm may fail.

Default: 10

**-dg**

Possible choices: det, prob, clos

(hyperparameter): Include this flag to manually specify the statistical approach to tracking for dMRI connectome estimation. Options are: det (deterministic), closest (clos), and prob (probabilistic). Default is det. If you wish to iterate the pipeline across multiple traversal methods, delimit the list by space (e.g. ‘det’, ‘prob’, ‘clos’).

Default: “det”

**-em**

(hyperparameter): Intersection distance in mm of a node to an edge. Corresponds to the extent of parcel-streamline overlap in tractography. If any coordinate in the streamline is within this distance from the center of any voxel in the node. Default is 5 mm. Safe range for dMRI is: [0-15]. If you wish to iterate the pipeline across multiple smoothing delimit the list by space (e.g. 2 4 6)

Default: 5

**-norm**

Possible choices: 0, 1, 2, 3, 4, 5, 6

Include this flag to normalize the resulting graph by (1) maximum edge weight; (2) using log10; (3) using pass-to-ranks for all non-zero edges; (4) using pass-to-ranks for all non-zero edges relative to the number of nodes; (5) using pass-to-ranks with zero-edge boost; and (6) which standardizes the matrix to values [0, 1]. Default is (6).

Default: 1

**-bin**

Include this flag to binarize the resulting graph such that edges are boolean and not weighted.

Default: False

**-dt**

Optionally use this flag if you wish to threshold to achieve a given density or densities indicated by the -thr and -min_thr, -max_thr, -step_thr flags, respectively.

Default: False

**-mst**

Optionally use this flag if you wish to apply local thresholding via the Minimum Spanning Tree approach. -thr values in this case correspond to a target density (if the -dt flag is also included), otherwise a target proportional threshold.
Default: False

-p
Possible choices: 0, 1, 2, 3

(Optional) Include this flag to (0) retain isolated nodes (1) retain only connected components of a minimal size. (2) prune the graph of all but hubs as defined by any of a variety of definitions (see advanced.yaml), or (3) retain only the largest connected component subgraph. Default is (1), which is equivalent to defragmenting only isolated nodes, unless the minimum threshold is >1 (see advanced.yaml).

Default: 1

-df
Optionally use this flag if you wish to apply local thresholding via the disparity filter approach. -thr values in this case correspond to .

Default: False

-mplx
Possible choices: 0, 1, 2

Include this flag to perform multiplex graph analysis across structural-functional connectome modalities. Options include level (1) Create multiplex graphs using mutual information and adaptive thresholding; (2) Additionally perform multiplex graph embedding and analysis. Default is (0) which is no multiplex analysis.

Default: 0

-embed
Optionally use this flag if you wish to embed the ensemble(s) produced into feature vector(s).

Default: False

-spheres
Include this flag to use spheres instead of parcels as nodes.

Default: False

-n

Optionally specify the name of any of the 2017 Yeo-Schaefer RSNs (7-subnet or 17-subnet): Vis, SomMot, DorsAttn, SalVentAttn, Limbic, Cont, Default, VisCent, VisPeri, SomMotA, SomMotB, DorsAttnA, DorsAttnB, SalVentAttnA, SalVentAttnB, LimbicOFC, LimbicTempPole, ContA, ContB, ContC, DefaultA, DefaultB, DefaultC, TempPar. If listing multiple RSNs, separate them by space. (e.g. -n ‘Default’ ‘Cont’ ‘SalVentAttn’).

-vox
Possible choices: 1mm, 2mm

Optionally use this flag if you wish to change the resolution of the images in the workflow. Default is 2mm.

Default: “2mm”

-plt
Optionally use this flag if you wish to activate plotting of adjacency matrices, connectomes, and time-series.

Default: False

-pm
Maximum number of cores and GB of memory, stated as two integers separated by comma. Otherwise, default is auto, which uses all available resources detected on the compute node(s) used for execution.

Default: “auto”
-plug Possible choices: Linear, MultiProc, SGE, PBS, SLURM, SGEgraph, SLURM- graph, LegacyMultiProc
Include this flag to specify a workflow plugin other than the default MultiProc.
Default: “MultiProc”
-v Verbose print for debugging.
Default: False
-noClean Disable post-workflow clean-up of temporary runtime metadata.
Default: False
-config Optionally override advanced configuration parameters. Default is advanced.yaml.
Default: “advanced.yaml”
-work Specify the path to a working directory for pynets to run. Default is /tmp/work.
Default: “/tmp/work”
--version show program’s version number and exit

PyNets BIDS CLI: A Fully-Automated Workflow for Reproducible Ensemble Sampling of Functional and Structural Connectomes

usage: pynets_bids [-h]
[--participant_label PARTICIPANT_LABEL [PARTICIPANT_LABEL ...]]
[--session_label SESSION_LABEL [SESSION_LABEL ...]]
[--run_label RUN_LABEL [RUN_LABEL ...]]
[--push_location PUSH_LOCATION]
[-ua Path to parcellation file in MNI-space [Path to parcellation...
-FILE in MNI-space ...]]
[-cm Cluster mask [Cluster mask ...]]
[-roi Path to binarized Region-of-Interest ROI NiftiImage in,
-template MNI space [Path to binarized Region-of-Interest (ROI NiftiImage in template,
-MNI space ...]]
[-ref Atlas reference file path]
[-way Path to binarized NiftiImage to constrain tractography [Path,]
-to binarized NiftiImage to constrain tractography ...]]
[-config Optional path to a config.json file with runtime settings.]
[-pm Cores,memory] [-plug Scheduler type] [-v] [-clean]
[-work Working directory]
bids_dir output_dir {participant,group} {dwi,func}
[{dwi,func} ...]
Positional Arguments

bids_dir  The directory with the input dataset formatted according to
           the BIDS standard. To use data from s3, just pass
           s3://<bucket>/<dataset> as the input directory.

output_dir  The directory to store pynets derivatives locally.

analysis_level  Possible choices: participant, group
                 Whether to instantiate an individual or group workflow

modality  Possible choices: dwi, func
           Specify data modality to process from bids directory. Options are dwi and func.

Named Arguments

--participant_label  The label(s) of the participant(s) that should be analyzed.
                      The label corresponds to sub-<participant_label> from the BIDS spec (so it
does not include “sub-“). If this parameter is not provided all subjects found
in bids_dir will be analyzed. Multiple participants can be specified with a
space separated list.

--session_label  The label(s) of the session that should be analyzed.
                  The label corresponds to ses-<participant_label> from the BIDS spec (so it
does not include “ses-“). If this parameter is not provided all sessions should
be analyzed. Multiple sessions can be specified
                  with a space separated list.

--run_label  The label(s) of the run, if any, within a given session that
             should be analyzed. The label corresponds to run-<run_label> from the
BIDS spec (so it does not include “run-“). If this parameter is not provided all runs should be analyzed. Specifying multiple runs is not yet supported.

--push_location  Name of folder on s3 to push output data to, if the folder does not exist, it will be
                 created. Format the location as s3://<bucket>/<path>

-ua  Optionally specify a path to a parcellation/atlas Nifti1Image file in MNI152 space.
       Labels should be spatially distinct across hemispheres and ordered with consecutive
       integers with a value of 0 as the background label. If specifying a list of paths to
       multiple user atlases, separate them by space.

-cm  Optionally specify the path to a Nifti1Image mask file to constrained functional
      clustering. If specifying a list of paths to multiple cluster masks, separate them
      by space.

-roi  Optionally specify a binarized ROI mask in template MNI space and retain only
      those nodes of a parcellation contained within that mask for connectome estimation.

-ref  Specify the path to the atlas reference .txt file that maps labels to intensities cor-
      responding to the atlas parcellation file specified with the -ua flag.

-way  Optionally specify a binarized ROI mask in template MNI-space to constrain trac-
      tography in the case of dmri connectome estimation.

-config  Including this flag will override the bids_config.json template in the base directory
       of pynets. See the template ad pynets -h for available settings.
-pm Number of cores to use, number of GB of memory to use for single subject run, entered as two integers separated by comma. Otherwise, default is auto, which uses all resources detected on the current compute node.

Default: “auto”

-plug Possible choices: Linear, MultiProc, SGE, PBS, SLURM, SGEgraph, SLURM-graph, LegacyMultiProc

Include this flag to specify a workflow plugin other than the default MultiProc.

Default: “MultiProc”

-v Verbose print for debugging.

Default: False

-clean Clean up temporary runtime directory after workflow termination.

Default: False

-work Specify the path to a working directory for pynets to run. Default is /tmp/work.

Default: “/tmp/work”

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 scenario, 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 advanced.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:

```bash
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:

```bash
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'
```
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:

```bash
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:

```bash
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.
-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:

```bash
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...
-of the input atlas.
```

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

Building upon the previous examples, let’s say you now wish to create a subject-specific atlas based on the sub-
ject’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.
... 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' dpy/pynets:latest pynets_bids 's3://hnu/HNU '/outputs' participant func participant_label 0025427 --session_label 1 -plug 'MultiProc' -pm '8,12' -work '/tmp' -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 0.028 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'
```

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 0.028 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)
-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:

```bash
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:

```bash
singularity exec_clust_est -B /work:/work /scratch/04171/dPys/pynets_latest-2016-12-04-
...5b74ad9a4c4d.img \
  -B '/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' \
  -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 crash files 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 a 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.

Group Analysis

COMING SOON

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


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:

   ```
   pytest
   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

```
pytest tests/test_*
```

### 1.4.6 API

#### pynets.core package

#### Submodules

##### pynets.core.nodemaker module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017

```python
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.

```python
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.
Create a 3D Nifti1Image atlas parcellation of consecutive integer intensities from an input list of ROI’s.

Parameters

parcels_4d_img
[Nifti1Image] 4D image stack of 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.

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

Parameters

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

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

Returns

parcels_4d
[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_radius
[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.

Check for hemispherically distinct and consecutive labels and rebuild parcellation.
Parameters

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

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

Returns

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

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

Returns

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

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

networks_list
[ ] List of subnet’s and their associated coordinates, if predefined uniquely for a given atlas.

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

pynets.core.nodemaker.gen_network_parcels(parcellation, subnet, labels, dir_path)

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

Parameters

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

subnet
[ ]

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

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

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

Returns
out_path
[st] File path to a new, subnetwork-filtered atlas parcellation Nifti1Image.

`pynets.core.nodemaker.get_names_and_coords_of_parcelss(parcellation, background_label=0)`
Return list of coordinates and max label intensity for a 3D atlas parcellation image.

Parameters

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

Returns

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

atlas
[st] 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_attributes(node_files, emb_shape, atlas='BrainnetomeAtlasFan2016')`

`pynets.core.nodemaker.get_node_membership(subnet, infile, coords, labels, parc, parcels_4d, perc_overlap=0.75, error=4)`
Evaluate the affinity of any arbitrary list of coordinate or parcel nodes for a user-specified subnet based on Yeo-7 or Yeo-17 definitions.

Parameters

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

infile
[st] 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.

parcels_4d
[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 subnet 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 subnet 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 subnet.

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

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

subnet
[str] Resting-state subnet 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.nodemakeet_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]

def mask_roi(dir_path, roi, mask, img_file):
    """Create derivative ROI based on intersection of roi and brain mask.\n    """
    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.

def mmToVox(img_affine, mmcoords):
    """Function to convert a list of mm coordinates to voxel coordinates.\n    """
    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.

def 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.\n    """
    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 subnet’s and their associated coordinates, if predefined uniquely for a given atlas.
    parcellation : str
        File path to atlas parcellation Nifti1Image in MNI template space.
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.

- **parcels_4d**
  - [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.

- **parcellation**
  - [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.

**pynets.core.nodemaker.node_gen_masking(roi, coords, parcels_4d, labels, dir_path, ID, parc, atlas, parcellation, 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**  

- **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.

- **parcels_4d**  
  [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.

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

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

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

pynets.core.nodemaker.parcel_masker

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

Parameters

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

coords
[ ] 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.

parcels_4d
[ ] List of 3D boolean numpy arrays or binarized Nifti1Images corresponding to ROI masks.

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

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

ID
[ ] A subject id or other unique identifier.

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

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

parcel_list_adj
[ ] 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

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.nodemaker.three_to_four_parcellation(parcellation)
Return 4d Nifti1Image 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

parcellation
[<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.thresholding module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017

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.
pynets Documentation, Release 1.30

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


weight

[st] 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

[st] 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
References

[1]

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

Creates a k-nearest neighbour graph.

Parameters

- 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

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

Returns


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 \((conn_matrix, \text{thr}, \text{min}_\text{span}_\text{tree}, \text{dens}_\text{thresh}, \text{disp}_\text{filt})\)

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 \((W, \text{copy}=\text{True})\)

Thresholds the near-zero ranks of a ranked graph.

Parameters


Returns

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

References

[1]
pynets.core.thresholding.thresh_func \((\text{dens}_\text{thresh}, \text{thr}, \text{conn_matrix}, \text{conn}_\text{model}, \text{subnet}, \text{ID}, \text{dir}_\text{path}, \text{roi}, \text{node}_\text{radius}, \text{min}_\text{span}_\text{tree}, \text{smooth}, \text{disp}_\text{filt}, \text{parc}, \text{prune}, \text{atlas}, \text{parcellation}, \text{labels}, \text{coords}, \text{norm}, \text{binary}, \text{hpass}, \text{signal}, \text{check}_\text{consistency}=\text{True})\)

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.

subnet
  [str] Resting-state subnet 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

node_radius
  [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 subnet’ 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.

parcellation
  [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.

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

**Returns**

**conn_matrix_thr**

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

**subnet**
[str] Resting-state subnet 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**

**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.

**parcellation**
[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.

signal
[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, subnet, ID, dir_path, roi, node_radius, min_span_tree, disp_filt, parc, prune, atlas, parcellation, labels, coords, norm, binary, track_type, atlas_for_streams, streams, traversal, 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
[stmt] 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.

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

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

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

roi
node_radius
  [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 subnet’ 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.

parcellation
  [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.

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.

traversal
  [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

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

subnet
[str] Resting-state subnet 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

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.

parcellation
[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.

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.

traversal
[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]

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]

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:

```python
data = np.random.random_sample((10,10))
threshold = 0.25

filtered_data = threshold_proportional(data, threshold)
```

Note this double counts each nonzero element. The 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)

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 subnet, 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

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

pynets.core.utils.as_directory(dir_, remove=False, return_as_path=False)
Convenience function to make a directory while returning it.

Parameters

dir_
[All, 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)

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(subnet, 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

subnet
[str] Resting-state subnet 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

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

**Parameters**

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

**subnet**
- [str] Resting-state subnet 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**

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

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

**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.

**traversal**
- [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

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

**Parameters**

**ID**
- [str] A subject id or other unique identifier.
subnet
[\text{str}] Resting-state subnet based on Yeo-7 and Yeo-17 naming (e.g. ‘Default’) used to filter nodes in the study of brain subgraphs.

conn\_model
[\text{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
[\text{float}] A value, between 0 and 1, to threshold the graph using any variety of methods triggered through other options.

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

dir\_path
[\text{str}] Path to directory containing subject derivative data for given run.

node\_radius
[\text{int}] Spherical centroid node size in the case that coordinate-based centroids are used as ROI’s.

smooth
[\text{int}] Smoothing width (mm fwhm) to apply to time-series when extracting signal from ROI’s.

thr\_type
[\text{str}] Type of thresholding performed (e.g. prop, abs, dens, mst, disp)

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

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

signal
[\text{str}] The name of a valid function used to reduce the time-series region extraction.

Returns

est\_path
[\text{str}] File path to .npy file containing graph with all specified combinations of hyperparameter characteristics.

\textit{pynets.core.utils.create_raw_path_diff}(ID, subnet, conn_model, roi, dir\_path, node\_radius, track\_type, parc, traversal, min\_length, error\_margin)

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

Parameters

ID
[\text{str}] A subject id or other unique identifier.

subnet
[\text{str}] Resting-state subnet based on Yeo-7 and Yeo-17 naming (e.g. ‘Default’) used to filter nodes in the study of brain subgraphs.

conn\_model
[\text{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
[\text{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_radius
  [int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI’s.

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

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

traversal
  [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, subnet, conn_model, roi, dir_path, node_radius, smooth, hpass, parc, signal)

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

Parameters

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

subnet
  [str] Resting-state subnet 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

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

node_radius
  [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.

signal
  [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.

```python
pynets.core.utils.decompress_nifti(infile)
```

```python
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.

```python
pynets.core.utils.dumpstacks(signal, frame)
```

```python
pynets.core.utils.filter_cols_from_targets(df, targets)
```

```python
pynets.core.utils.flatten(l)
```

Flatten list of lists.

```python
pynets.core.utils.get_file()
```

Get a file's base directory path.

```python
pynets.core.utils.get_template_tf(template_name, vox_size)
```

```python
pynets.core.utils.kill_process_family(parent_pid)
```

```python
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.

```python
pynets.core.utils.load_mat_ext(est_path, ID, subnet, conn_model, roi, prune, norm, binary, min_span_tree, dens_thresh, disp_filt)
```

```python
pynets.core.utils.load_runconfig(location=None)
```

```python
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).

```python
pynets.core.utils.mergedicts(dict1, dict2)
```

```python
pynets.core.utils.missing_elements(L)
```
pynets.core.utils.pass_meta_ins(conn_model, est_path, subnet, 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.

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

thr
[ fl ot ] 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

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.

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

thr
[ fl ot ] 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

norm
[ int ] Indicates method of normalizing resulting graph.
**pynets Documentation, Release 1.30**

```python
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 subnet 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 subnet 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.

```python
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 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 of file paths to .npy file containing graph with thresholding applied.

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

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

ID_iterlist
[ ] List of repeated subject id strings.

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

norm_iterlist
[ ] Indicates method of normalizing resulting graph.

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

Returns

conn_model_iterlist
[ ] 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 of file paths to .npy file containing graph with thresholding applied.

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

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

ID_iterlist
[ ] List of repeated subject id strings.

roi_iterlist
[ ] 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.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, subnet='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

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

subnet

[st] Restricted sub-subnet name.

Returns

nodes_path

[st] 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

[st] File path to .npy file containing graph.

fmt

[st] 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, subnet, thr, conn_model, roi, prune, norm, binary)

pynets.core.utils.save_nifti_parcels_map(ID, dir_path, subnet, 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.

subnet

[ str ] Resting-state subnet 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, subnet, ID, dir_path, ts_within_nodes, smooth, hpass, node_radius, signal)

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

**Parameters**

roi


subnet

[ str ] Resting-state subnet 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_radius

[ int ] Spherical centroid node size in the case that coordinate-based centroids are used as ROI’s for time-series extraction.
signal
[\text{str}] The name of a valid function used to reduce the time-series region extraction.

Returns

\text{out\_path\_ts}
[\text{str}] Path to .npy file containing array of fMRI time-series extracted from nodes.

\text{pynets.core.utils.timeout}(\text{seconds})
Timeout function for hung calculations.

class \text{pynets.core.utils.watchdog}
Bases: \text{object}

Methods

\begin{verbatim}
run()
\end{verbatim}

\text{pynets.core.workflows module}

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017

\text{pynets.core.workflows.dmri_connectometry}(\text{ID}, \text{atlas}, \text{subnet}, \text{node\_radius}, \text{roi}, \text{parcellation}, \text{plot\_switch},
\text{parc}, \text{ref\_txt}, \text{procmem}, \text{dwi\_file}, \text{fbval}, \text{fvec}, \text{anat\_file}, \text{thr},
\text{dens\_thresh}, \text{conn\_model}, \text{user\_atlas\_list}, \text{multi\_thr},
\text{multi\_atlas}, \text{max\_thr}, \text{min\_thr}, \text{step\_thr}, \text{node\_radii\_list},
\text{conn\_model\_list}, \text{min\_span\_tree}, \text{use\_parcel\_naming}, \text{disp\_filt},
\text{plugin\_type}, \text{multi\_nets}, \text{prune}, \text{mask}, \text{norm}, \text{binary},
\text{curv\_thr\_list}, \text{step\_list}, \text{track\_type}, \text{min\_length}, \text{maxcrossing},
\text{error\_margin}, \text{traversal}, \text{tiss\_class}, \text{runtime\_dict},
\text{execution\_dict}, \text{multi\_traversal}, \text{template\_name}, \text{vox\_size},
\text{waymask}, \text{min\_length\_list}, \text{error\_margin\_list}, \text{outdir})

A function interface for generating a dMRI connectometry nested workflow

\text{pynets.core.workflows.fmri_connectometry}(\text{func\_file}, \text{ID}, \text{atlas}, \text{subnet}, \text{node\_radius}, \text{roi}, \text{thr}, \text{parcellation},
\text{conn\_model}, \text{dens\_thresh}, \text{conf}, \text{plot\_switch}, \text{parc}, \text{ref\_txt},
\text{procmem}, \text{multi\_thr}, \text{multi\_atlas}, \text{max\_thr}, \text{min\_thr}, \text{step\_thr}, \text{k},
\text{clust\_mask}, \text{k\_list}, \text{k\_clustering}, \text{user\_atlas\_list},
\text{clust\_mask\_list}, \text{node\_radii\_list}, \text{conn\_model\_list},
\text{min\_span\_tree}, \text{use\_parcel\_naming}, \text{smooth}, \text{smooth\_list},
\text{disp\_filt}, \text{prune}, \text{multi\_nets}, \text{clust\_type}, \text{clust\_type\_list},
\text{plugin\_type}, \text{mask}, \text{norm}, \text{binary}, \text{anat\_file}, \text{runtime\_dict},
\text{execution\_dict}, \text{hp\_pass}, \text{hp\_pass\_list}, \text{template\_name}, \text{vox\_size},
\text{local\_corr}, \text{signal}, \text{signal\_list}, \text{outdir})

A function interface for generating an fMRI connectometry nested workflow

\text{pynets.core.workflows.raw_graph_workflow}(\text{multi\_thr}, \text{thr}, \text{multi\_graph}, \text{graph}, \text{ID}, \text{subnet}, \text{conn\_model},
\text{roi}, \text{prune}, \text{norm}, \text{binary}, \text{min\_span\_tree}, \text{dens\_thresh}, \text{disp\_filt},
\text{min\_thr}, \text{max\_thr}, \text{step\_thr}, \text{wf}, \text{net\_mets\_node}, \text{runtime\_dict})
pynets.core.workflows.workflow_selector(func_file, ID, atlas, subnet, node_radius, roi, thr, parcellation, 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_radii_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, curvThr_list, step_list, track_type, min_length, maxcrossing, error_margin, traversal, tiss_class, runtime_dict, execution_dict, embed, multi_traversal, multimodal, hpass, hpass_list, vox_size, multiplex, waymask, local_corr, min_length_list, error_margin_list, signal, signal_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.utils module

Created on Fri Nov 10 15:44:46 2017 Copyright (C) 2017

pynets.dmri.utils.evaluate_streamline_plausibility(dwi_data, gtab, mask_data, streamlines, affine=array([[1.0, 0.0, 0.0, 0.0], [0.0, 1.0, 0.0, 0.0], [0.0, 0.0, 1.0, 0.0], [0.0, 0.0, 0.0, 1.0]]), sphere='repulsion724')

Linear Fascicle Evaluation (LiFE) takes any connectome and uses a forward modelling approach to predict diffusion measurements in the same brain.

Parameters

dwi_data
[array] 4D array of dwi data.

gtab

mask_data
[array] 3D Brain mask.

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

Returns

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

[1]

pynets.dmri.utils.extract_b0(in_file, b0_ixs, out_path=None)

Extract the $b0$ volumes from a DWI dataset.

Parameters

- **in_file**
  - [str] DWI NIfTI file.

- **b0_ixs**
  - [list] List of B0 indices in `in_file`.

- **out_path**
  - [str] Optionally specify an output path.

Returns

- **out_path**
  - [str] 4D NIfTI file consisting of B0's.

pynets.dmri.utils.generate_seeds(seeds)

Helper function that takes a sequence and returns a generator

Parameters

- **seeds**
  - [sequence] Usually, this would be a list of 2D arrays, representing seeds

Returns

- **generator**

pynets.dmri.utils.generate_sl(streamlines)

Helper function that takes a sequence and returns a generator

Parameters

- **streamlines**
  - [sequence] Usually, this would be a list of 2D arrays, representing streamlines

Returns

- **generator**

pynets.dmri.utils.normalize_gradients(bvecs, bvals, b0_threshold, bvec_norm_epsilon=0.1, b_scale=True)

Normalize b-vectors and b-values.

The resulting b-vectors will be of unit length for the non-zero b-values. The resulting b-values will be normalized by the square of the corresponding vector amplitude.

Parameters

- **bvecs**
  - [m x n 2d array] Raw b-vectors array.

- **bvals**
  - [1d array] Raw b-values float array.

- **b0_threshold**
  - [float] Gradient threshold below which volumes and vectors are considered B0's.
pynets Documentation, Release 1.30

Returns

bvecs

[m x n 2d array] Unit-normed b-vectors array.

bvals

[1d int array] Vector amplitude square normed b-values array.

pynets.dmri.utils.random_seeds_from_mask(mask, seeds_count, affine=\(\begin{bmatrix} 1.0, & 0.0, & 0.0, & 0.0 \\ 0.0, & 1.0, & 0.0, & 0.0 \\ 0.0, & 0.0, & 1.0, & 0.0 \\ 0.0, & 0.0, & 0.0, & 1.0 \end{bmatrix}\), random_seed=1)

Create randomly placed seeds for fiber tracking from a binary mask. Seeds points are placed randomly distributed in voxels of mask which are True. If seed_count_per_voxel is True, this function is similar to seeds_from_mask(), with the difference that instead of evenly distributing the seeds, it randomly places the seeds within the voxels specified by the mask.

Parameters

mask

[binary 3d array_like] A binary array specifying where to place the seeds for fiber tracking.

affine

[array, (4, 4)] The mapping between voxel indices and the point space for seeds. The voxel_to_rasmm matrix, typically from a NIFTI file. A seed point at the center the voxel \([i, j, k]\) will be represented as \([x, y, z]\) where \([x, y, z, 1]\) == \(\text{np.dot(affine, [i, j, k, 1])}\).

seeds_count

[int] The number of seeds to generate. If seed_count_per_voxel is True, specifies the number of seeds to place in each voxel. Otherwise, specifies the total number of seeds to place in the mask.

random_seed

[int] The seed for the random seed generator (numpy.random.seed).

See also:

seeds_from_mask

Raises

ValueError

When mask is not a three-dimensional array

pynets.dmri.estimation module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017

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

Estimate an anisotropic power map image to use for registrations.

Parameters

gtab_file

[estr] File path to pickled DiPy gradient table object.

dwi_file

[estr] File path to diffusion weighted image.

B0_mask

[estr] 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] pyaerts.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] pyaerts.dmri.estimation.csd_mod_est(gtab, data, B0_mask, sh_order=8)
Estimate a Constrained Spherical Deconvolution (CSD) model from dwi data.

**Parameters**

**gtab**

**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
[object] Fitted csd model.

References

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

pynets.dmri.estimation.mcsd_mod_est(gtab, data, B0_mask, wm_in_dwi, gm_in_dwi, vent_csf_in_dwi, sh_order=8, roi_radii=10)

Estimate a Constrained Spherical Deconvolution (CSD) model from dwi data.

Parameters

gtab

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

References

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

pynets.dmri.estimation.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

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.estimation.sfm_mod_est(gtab, data, B0_mask, BACKEND='loky')
Estimate a Sparse Fascicle Model (SFM) from dwi data.

Parameters

gtab

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, conn_model, subnet, node_radius, dens_thres, ID, roi, min_span_tree, disp_fill, parc, prune, atlas, parcellation, labels, coords, norm, binary, traversal, 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 NiftiImage 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').

conn_model
[str] Connectivity reconstruction method (e.g. 'csa', 'tensor', 'csd').
subnet
  [str] Resting-state subnet based on Yeo-7 and Yeo-17 naming (e.g. ‘Default’) used to filter
  nodes in the study of brain subgraphs.

node_radius
  [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

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 subnet’
  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.

parcellation
  [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.

traversal
  [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’).

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

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

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

node_radius
[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

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 subnet’ 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.

parcellation
[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.

traversal
[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.

everror_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
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

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

pynets.dmri.track.create_density_map(fa_img, dir_path, streamlines, conn_model, node_radius, curv_thr_list, step_list, subnet, roi, traversal, 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’).

node_radius
    [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.

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

roi
traversal
  [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
  vent_csf_in_dwi
  [Nifti1Image] Ventricular CSF tissue segmentation Nifti1Image.
  wm_in_dwi
  tiss_class
  cmc_step_size
  [float] Step size from CMC tissue classification method.

Returns
  tiss_classifier
  [obj] Tissue classifier object.

References
[1],[2]

pynets.dmri.track.run_tracking(step_curv_combinations, recon_shelved, n_seeds_per_iter, traversal, maxcrossing, max_length, pft_back_tracking_dist, pft_front_tracking_dist, particle_count, roi_neighborhood_tol, min_length, track_type, min_separation_angle, sphere, tiss_class, tissue_shelved, verbose=False)
Create a density map of the list of streamlines.

Parameters
  step_curv_combinations
  [list] List of tuples representing all pair combinations of step sizes and curvature thresholds from which to sample streamlines.
recon_path
  [str] File path to diffusion reconstruction model.

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.

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

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

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

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 back tracking 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.

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.

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

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

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

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

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

tiss_class

tissue_shelved
  [str] File path to joblib-shelved 4D T1w tissue segmentations in native diffusion space.

Returns

streamlines
  [ArraySequence] DiPy list/array-like object of streamline points from tractography.
Perform native-space ensemble tractography, restricted to a vector of ROI masks.

**Parameters**

- **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**

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

- **traversal**
  - [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 of 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 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.clustering module

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 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 eigenvectors (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 eigenvectors, 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 eigenvector. 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.clustering.ensemble_parcellate(infiles, k)

pynets.fmri.clustering.indx_1dto3d(idx, sz)

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

Parameters

idx


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.clustering.indx_3dto1d(idx, sz)

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

Parameters

idx


sz

[array] Shape of 3D matrix idx.

Returns

idx1

pynets.fmri.clustering.make_local_connectivity_scorr(func_img, clust_mask_img, thresh)

Constructs a spatially constrained connectivity matrix from an 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.clustering.make_local_connectivity_tcorr(func_img, clust_mask_img, thresh)

Constructs a spatially constrained connectivity matrix from an 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 $j$. 
pynets Documentation, Release 1.30

References

[1] pynets.fmri.clustering.ncut(W, nbEigenValues, offset=0.5, maxiterations=100, eigsErrorTolerance=1e-06, eps=2.2204e-16)

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.clustering.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.clustering.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 renumbered 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
[3D NIFTI file containing a mask, which restricts the voxels used in the analysis.
References

[1]
pynets.fmri.clustering.proportional\((k, \text{voxels\_list})\)
Hagenbach-Bischoff Quota

pynets.fmri.estimation module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017

pynets.fmri.estimation.fill_confound_nans\((\text{confound}, \text{dir\_path}, \text{drop\_thr}=0.5)\)
Fill the NaN values of a confounds dataframe with mean values

pynets.fmri.estimation.get_conn_matrix\((\text{time\_series}, \text{conn\_model}, \text{dir\_path}, \text{node\_radius}, \text{smooth},\)
\quad \text{dens\_thresh}, \text{subnet}, \text{ID}, \text{roi}, \text{min\_span\_tree}, \text{disp\_filt}, \text{parc}, \text{prune},\)
\quad \text{atlas}, \text{parcellation}, \text{labels}, \text{coords}, \text{norm}, \text{binary}, \text{hpss}, \text{signal})\)
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_radius
[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.

subnet
[str] Resting-state subnet 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

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 subnet’ 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.

parcellation
  [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.

signal
  [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_radius
  [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.

subnet
  [str] Resting-state subnet 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
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 subnet’ 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.

parcellation
[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.

signal
[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, cv=5, max_iter=200)
pynets.fmri.estimation.timeseries_bootstrap(tseries, block_size)
   Generates a bootstrap sample derived from the input time-series. Utilizes Circular-block-bootstrapping 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.adjacency module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017

pynets.plotting.adjacency.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.adjacency.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.adjacency.plot_conn_mat_func(conn_matrix, conn_model, atlas, dir_path, ID, subnet, labels, roi, thr, node_radius, smooth, hpass, signal)

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.

subnet
  [str] Resting-state subnet 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

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

node_radius
  [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.

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

pynets.plotting.adjacency.plot_conn_mat_struct(conn_matrix, conn_model, atlas, dir_path, ID, subnet, labels, roi, thr, node_radius, track_type, traversal, 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.

subnet
  [str] Resting-state subnet 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

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

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

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

traversal
   [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.brain module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017

pynets.plotting.brain.create_gb_palette(mat, edge_cmap, coords, labels, node_radius='auto',
   node_cmap=None, prune=True, centrality_type='eig',
   max_node_radius=None, max_edge_size=2,
   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_radius
   [int] Spherical centroid node size in the case that coordinate-based centroids are used as
   ROI’s.

node_radius: 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.brain.plot_all_func(conn_matrix, conn_model, atlas, dir_path, ID, subnet, labels, roi, coords, thr, node_radius, edge_threshold, smooth, prune, parcellation, norm, binary, hpass, signal, 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.

- **subnet**
  - [str] Resting-state subnet 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**

- **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_radius**
  - [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.

- **parcellation**
  - [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.

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

dedge_color_override
[bool] Switch that enables random sequential colormap selection for edges.

\texttt{pynets.plotting.brain.plot_all_struct}(conn\_matrix, conn\_model, atlas, dir\_path, ID, subnet, labels, roi, coords, thr, node\_radius, edge\_threshold, prune, parcellation, norm, binary, track\_type, traversal, min\_length, error\_margin)

Plot adjacency matrix, connectogram, and glass brain for functional connectome.

**Parameters**

\texttt{conn\_matrix}
[array] NxN matrix.

\texttt{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.

\texttt{atlas}
[str] Name of atlas parcellation used.

\texttt{dir\_path}
[str] Path to directory containing subject derivative data for given run.

\texttt{ID}
[str] A subject id or other unique identifier.

\texttt{subnet}
[str] Resting-state subnet based on Yeo-7 and Yeo-17 naming (e.g. ‘Default’) used to filter nodes in the study of brain subgraphs.

\texttt{labels}
[list] List of string labels corresponding to ROI nodes.

\texttt{roi}

\texttt{coords}
[list] List of (x, y, z) tuples corresponding to an a-priori defined set (e.g. a coordinate atlas).

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

\texttt{node\_radius}
[int] Spherical centroid node size in the case that coordinate-based centroids are used as ROI’s.

\texttt{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.

\texttt{prune}
[bool] Indicates whether to prune final graph of disconnected nodes/isolates.

\texttt{parcellation}
[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.

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

traversal
[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.brain.plot_all_struct_func(mG_paths, namer_dir, modality_paths)
Plot adjacency matrix and glass brain for structural-functional multiplex connectome.

Parameters

mG_paths
[tuple] A tuple of two gpickle file paths containing a NetworkX OrderedMultiGraph object and a MultilayerGraph object (See https://github.com/nkoub/multinetx).

namer_dir
[st] Path to output directory for multiplex data.

modality_paths
[tuple] A tuple of file path strings to the raw structural and raw functional connectome graph files (.npy).

pynets.plotting.brain.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.brain.plot_network_clusters(graph, communities, out_path, figsize=(8, 8), node_radius=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
[st] Path to save figure.

figsize
[Tuple of integers] The figure size; it is a pair of float, default (8, 8).

node_radius: 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.brain.plot_timeseries(time_series, subnet, 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.

subnet
[str] Resting-state subnet 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.brain.show_template_bundles(final_streamlines, template_path, fname)

pynets.plotting.brain.view_tractogram(streams, atlas)

Module contents

pynets-registration package

Submodules

pynets-registration-utils module

Module contents

pynets-package

Subpackages

pynets-statistics package

Submodules

pynets-statistics-utils module

Copyright (C) 2017
Module contents

pynets.statistics.group package

Submodules

pynets.statistics.group.prediction module

Copyright (C) 2017

pynets.statistics.group.benchmarking module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2017

Module contents

pynets.statistics.individual package

Submodules

pynets.statistics.individual.algorithms module

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

pynets.statistics.individual.algorithms.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.
pynets Documentation, Release 1.30

References

[1],[2]

pynets.statistics.individual.algorithms.average_shortest_path_length_fast(G, weight='weight')

pynets.statistics.individual.algorithms.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.statistics.individual.algorithms.collect_pandas_df_make(net_mets_csv_list, ID, subnet, plot_switch, embed=False, create_summary=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.

subnet
[str] Resting-state subnet 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.)

Returns

combination_complete
[bool] If True, then data integration completed successfully.

References

[1]

pynets.statistics.individual.algorithms.community_resolution_selection(G)

pynets.statistics.individual.algorithms.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.statistics.individual.algorithms.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.statistics.individual.algorithms.defragment(G)

pynets.statistics.individual.algorithms.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.statistics.individual.algorithms.get_betweenness_centrality(G_len, metric_list_names, net_met_val_list_final, engine='nx')

pynets.statistics.individual.algorithms.get_clustering(G, metric_list_names, net_met_val_list_final, engine='nx')

pynets.statistics.individual.algorithms.get_comm_centrality(G, metric_list_names, net_met_val_list_final)

pynets.statistics.individual.algorithms.get_community(G, net_met_val_list_final, metric_list_names)

pynets.statistics.individual.algorithms.get_degree_centrality(G, metric_list_names, net_met_val_list_final)

pynets.statistics.individual.algorithms.get_diversity(in_mat, ci, metric_list_names, net_met_val_list_final)

pynets.statistics.individual.algorithms.get_eigen_centrality(G, metric_list_names, net_met_val_list_final, engine='nx')

pynets.statistics.individual.algorithms.get_local_efficiency(G, metric_list_names, net_met_val_list_final)

pynets.statistics.individual.algorithms.get_participation(in_mat, ci, metric_list_names, net_met_val_list_final)

pynets.statistics.individual.algorithms.get_prop_type(value, key=None)

Performstypingandvalueconversionfortheigraph_toolPropertyMapclass. Ifakeyisprovided, italsoensures
thekeyisinformatthatcanbeusedwiththePropertyMap. Returnsatuple,(typename,value,key)

pynets.statistics.individual.algorithms.get_rich_club_coeff(G, metric_list_names, net_met_val_list_final, engine='nx')

pynets.statistics.individual.algorithms.global_efficiency(G, weight='weight', engine='nx')

ReturntheglobalefficiencyoftheG

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 subnet, 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 knowledge may not exist, the scaling factor is not included. If that knowledge 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.statistics.individual.algorithms.iterate_nx_global_measures($G$, metric_list_glob)

pynets.statistics.individual.algorithms.link_communities($W$, type_clustering='single')

The optimal community structure is a subdivision of the subnet 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 subnet 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.statistics.individual.algorithms.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 subnet, 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 knowledge may not exist, the scaling factor is not included. If that knowledge exists, construct the corresponding weighted graph and calculate its local_efficiency to scale the weighted graph.

References

[1], [2]

pynets.statistics.individual.algorithms.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.statistics.individual.algorithms[np2gt](adj)

pynets.statistics.individual.algorithms[nx2gt](nxG)

Converts a networkx graph to a graph-tool graph.

pynets.statistics.individual.algorithms.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

\[P\]
\[P \in \text{Nx1 np.ndarray}\] Participation coefficient

References

[1], [2]

\text{pynets.statistics.individual.algorithms.participation_coef_sign}(W, ci)

Participation coefficient is a measure of diversity of intermodular connections of individual nodes.

Parameters

\[W\]
\[W \in \text{NxN np.ndarray}\] undirected connection matrix with positive and negative weights

\[ci\]
\[ci \in \text{Nx1 np.ndarray}\] community affiliation vector

Returns

\[P_{\text{pos}}\]
\[P_{\text{pos}} \in \text{Nx1 np.ndarray}\] participation coefficient from positive weights

\[P_{\text{neg}}\]
\[P_{\text{neg}} \in \text{Nx1 np.ndarray}\] participation coefficient from negative weights

References

[1], [2]

\text{pynets.statistics.individual.algorithms.prune_small_components}(G, \text{min\_nodes})

Returns a recomposed graph of all connected components of a minimum size

Parameters

\[G\]
\[G \in \text{Obj}\] NetworkX graph with isolated nodes present.

\text{min\_nodes}: \text{int}
Minimum number of nodes permitted in a connected subgraph

Returns

\[G\]
\[G \in \text{Obj}\] NetworkX graph with isolated nodes pruned.

\text{pruned\_nodes}
\[\text{pruned\_nodes} \in \text{list}\] List of indices of nodes that were pruned from \(G\).
pynets Documentation, Release 1.30

References
[1], [2]

`pynets.statistics.individual.algorithms.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_mets`
[float] Value of the graph metric i that was calculated from G.

`pynets.statistics.individual.algorithms.rich_club_coefficient(G, engine='nx')`

`pynets.statistics.individual.algorithms.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:
\[ \omega/\sigma = L_r/L - C/C_l \]
where C and L are respectively the average clustering coefficient/ transitivity and average shortest path length of G. L_r is the average shortest path length of an equivalent random graph and C_l 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 (C_r) and average shortest path length (L_r).

`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 small-world coefficient
pynets Documentation, Release 1.30

References

[1]
pynets.statistics.individual.algorithms.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.statistics.individual.algorithms.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

\[ T = 3 \frac{\# \text{triangles}}{\# \text{triads}}. \]

Parameters

G
[graph]

Returns

out
[float] Transitivity

References

[1], [2]

pynets.statistics.individual.spectral module

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

pynets.statistics.individual.spectral.build_asetomes(est_path_iterlist)
Embeds single graphs using the ASE algorithm.

Parameters

est_path_iterlist
[list] List of file paths to .npy files, each containing a graph.
pynets Documentation, Release 1.30

pynets.statistics.individual.spectral.build_masetome(est_path_iterlist)
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.

References

[1]

pynets.statistics.individual.spectral.build_omnetome(est_path_iterlist)
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.

References

[1], [2]

pynets.statistics.individual.multiplex module

Created on Tue Nov 7 10:40:07 2017 Copyright (C) 2018 @author: Derek Pisner & James Kunert-Graf

pynets.statistics.individual.multiplex.build_multigraphs(est_path_iterlist)
Constructs a multimodal multigraph for each available resolution of vertices.

Parameters

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

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.statistics.individual.multiplex.build_mx_multigraph(func_mat, dwi_mat, name, namer_dir)
It creates a symmetric (undirected) MultilayerGraph object from vertex-aligned structural and functional connectivity matrices.
pynets.statistics.individual.multiplex.matching(paths, atlas, namer_dir)

pynets.statistics.individual.multiplex.mutual_information_2d(x, y, sigma=1, normalized=True)

Computes (normalized) mutual information between two 2D variate from a joint histogram.

**Parameters**

- **x**
  - [1D array] first variable

- **y**
  - [1D array] second variable

- **sigma**: float
  - sigma for Gaussian smoothing of the joint histogram

**Returns**

- **nmi**: float
  - the computed similarity measure

pynets.statistics.individual.multiplex.optimize_mutual_info(dwi_mat, func_mat, bins=20)
BIBLIOGRAPHY


<table>
<thead>
<tr>
<th>Function</th>
<th>Module</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>as_directory()</td>
<td>pynets.core.utils</td>
<td>45</td>
</tr>
<tr>
<td>as_list()</td>
<td>pynets.core.utils</td>
<td>45</td>
</tr>
<tr>
<td>autofix()</td>
<td>pynets.core.thresholding</td>
<td>33</td>
</tr>
<tr>
<td>average_local_efficiency()</td>
<td>pynets.statistics.individual.algorithms</td>
<td>85</td>
</tr>
<tr>
<td>average_shortest_path_length_fast()</td>
<td>pynets.statistics.individual.algorithms</td>
<td>86</td>
</tr>
<tr>
<td>average_shortest_path_length_for_all()</td>
<td>pynets.statistics.individual.algorithms</td>
<td>86</td>
</tr>
<tr>
<td>binarize()</td>
<td>pynets.core.thresholding</td>
<td>34</td>
</tr>
<tr>
<td>build_args_from_config()</td>
<td>pynets.core.utils</td>
<td>45</td>
</tr>
<tr>
<td>build_asetomes()</td>
<td>pynets.statistics.individual.spectral</td>
<td>93</td>
</tr>
<tr>
<td>build_masetome()</td>
<td>pynets.statistics.individual.spectral</td>
<td>93</td>
</tr>
<tr>
<td>build_multigraphs()</td>
<td>pynets.statistics.individual.multiplex</td>
<td>94</td>
</tr>
<tr>
<td>build_mx_multigraph()</td>
<td>pynets.statistics.individual.multiplex</td>
<td>94</td>
</tr>
<tr>
<td>build_omnetome()</td>
<td>pynets.statistics.individual.spectral</td>
<td>94</td>
</tr>
<tr>
<td>check_est_path_existence()</td>
<td>pynets.core.utils</td>
<td>45</td>
</tr>
<tr>
<td>check_template_loads()</td>
<td>pynets.core.utils</td>
<td>46</td>
</tr>
<tr>
<td>checkConsecutive()</td>
<td>pynets.core.utils</td>
<td>45</td>
</tr>
<tr>
<td>collect_pandas_df()</td>
<td>pynets.core.utils</td>
<td>46</td>
</tr>
<tr>
<td>collect_pandas_df_make()</td>
<td>pynets.statistics.individual.algorithms</td>
<td>86</td>
</tr>
<tr>
<td>collectpandasjoin()</td>
<td>pynets.core.utils</td>
<td>46</td>
</tr>
<tr>
<td>community_resolution_selection()</td>
<td>pynets.statistics.individual.algorithms</td>
<td>86</td>
</tr>
<tr>
<td>coords_masker()</td>
<td>pynets.core.nodemaker</td>
<td>24</td>
</tr>
<tr>
<td>countmotifs()</td>
<td>pynets.statistics.individual.algorithms</td>
<td>86</td>
</tr>
<tr>
<td>create_anisopowermap()</td>
<td>pynets.dmri.estimation</td>
<td>60</td>
</tr>
<tr>
<td>create_communities()</td>
<td>pynets.statistics.individual.algorithms</td>
<td>87</td>
</tr>
<tr>
<td>create_csv_path()</td>
<td>pynets.core.utils</td>
<td>46</td>
</tr>
<tr>
<td>create_density_map()</td>
<td>pynets.dmri.track</td>
<td>67</td>
</tr>
<tr>
<td>create_est_path_diff()</td>
<td>pynets.core.utils</td>
<td>46</td>
</tr>
<tr>
<td>create_est_path_func()</td>
<td>pynets.core.utils</td>
<td>47</td>
</tr>
<tr>
<td>create_gb_palette()</td>
<td>pynets.plotting.brain</td>
<td>80</td>
</tr>
<tr>
<td>create_parcel_atlas()</td>
<td>pynets.core.nodemaker</td>
<td>24</td>
</tr>
<tr>
<td>create_raw_path_diff()</td>
<td>pynets.core.utils</td>
<td>48</td>
</tr>
<tr>
<td>create_raw_path_func()</td>
<td>pynets.core.utils</td>
<td>49</td>
</tr>
<tr>
<td>create_spherical_roi_volumes()</td>
<td>pynets.core.nodemaker</td>
<td>25</td>
</tr>
<tr>
<td>csa_mod_est()</td>
<td>pynets.dmri.estimation</td>
<td>61</td>
</tr>
<tr>
<td>csd_mod_est()</td>
<td>pynets.dmri.estimation</td>
<td>61</td>
</tr>
<tr>
<td>decompress_nifti()</td>
<td>pynets.core.utils</td>
<td>50</td>
</tr>
<tr>
<td>defragment()</td>
<td>pynets.statistics.individual.algorithms</td>
<td>87</td>
</tr>
<tr>
<td>density_thresholding()</td>
<td>pynets.core.thresholding</td>
<td>34</td>
</tr>
<tr>
<td>discretisation()</td>
<td>pynets.fmri.clustering</td>
<td>71</td>
</tr>
<tr>
<td>disparity_filter()</td>
<td>pynets.core.thresholding</td>
<td>34</td>
</tr>
<tr>
<td>disparity_filter_alpha_cut()</td>
<td>pynets.core.thresholding</td>
<td>35</td>
</tr>
<tr>
<td>diversity_coef_sign()</td>
<td>pynets.statistics.individual.algorithms</td>
<td>86</td>
</tr>
<tr>
<td>dmri_connectometry()</td>
<td>pynets.core.workflows</td>
<td>57</td>
</tr>
</tbody>
</table>
do_dir_path() (in module pynets.core.utils), 50
drop_badixs_from_parcellation() (in module pynets.core.nodemaker), 25
drop_coords_labels_from_restricted_parcellation() (in module pynets.core.nodemaker), 25
dumpstacks() (in module pynets.core.utils), 50
enforce_hem_distinct_consecutive_labels() (in module pynets.core.nodemaker), 25
ensemble_parcellate() (in module pynets.fmri.estimation), 72
est_density() (in module pynets.core.thresholding), 35
evaluate_streamline_plausibility() (in module pynets.dmri.utils), 58
extract_b0() (in module pynets.dmri.utils), 59
fill_confound_nans() (in module pynets.fmri.estimation), 77
fetch_nilearn_atlas_coords() (in module pynets.core.nodemaker), 27
global_efficiency() (in module pynets.statistics.individual.algorithms), 88
get_degree_centrality() (in module pynets.statistics.individual.algorithms), 88
get_diversity() (in module pynets.statistics.individual.algorithms), 88
get_eigen_centrality() (in module pynets.statistics.individual.algorithms), 88
get_file() (in module pynets.core.utils), 50
get_local_efficiency() (in module pynets.statistics.individual.algorithms), 88
get_node_attributes() (in module pynets.core.nodemaker), 27
get_node_membership() (in module pynets.core.nodemaker), 27
get_optimal_cov_estimator() (in module pynets.fmri.estimation), 77
global_eigens() (in module pynets.core.utils), 36
get_prop_type() (in module pynets.statistics.individual.algorithms), 88
get_betweenness_centrality() (in module pynets.statistics.individual.algorithms), 88
get_community() (in module pynets.statistics.individual.algorithms), 88
get_comm_matrix() (in module pynets.fmri.estimation), 75
get_comm_centralcy() (in module pynets.statistics.individual.algorithms), 88
get_conn_matrix() (in module pynets.statistics.individual.multiplex), 95
get_degree_centrality() (in module pynets.statistics.individual.algorithms), 88
get_diversity() (in module pynets.statistics.individual.algorithms), 88
get_eigen_centrality() (in module pynets.statistics.individual.algorithms), 88
get_file() (in module pynets.core.utils), 50
get_local_efficiency() (in module pynets.statistics.individual.algorithms), 88
get_names_and_coords_of_parcel() (in module pynets.core.nodemaker), 27
get_node_attributes() (in module pynets.core.nodemaker), 27
get_node_membership() (in module pynets.core.nodemaker), 27
get_optimal_cov_estimator() (in module pynets.fmri.estimation), 77
get_participation() (in module pynets.statistics.individual.algorithms), 88
get_prop_type() (in module pynets.statistics.individual.algorithms), 88
get_rich_club_coeff() (in module pynets.statistics.individual.algorithms), 88
get_sphere() (in module pynets.core.nodemaker), 28
get_template_tf() (in module pynets.core.utils), 50
kill_process_family() (in module pynets.core.utils), 50
kill_process_family() (in module pynets.core.utils), 50
kill_process_family() (in module pynets.core.utils), 50
kill_process_family() (in module pynets.core.utils), 50
knn() (in module pynets.core.thresholding), 36
link_communities() (in module pynets.statistics.individual.algorithms), 89
load_mat() (in module pynets.statistics.individual.algorithms), 89
load_mat_ext() (in module pynets.core.utils), 50
load_runconfig() (in module pynets.core.utils), 50
local_efficiency() (in module pynets.statistics.individual.algorithms), 88
local_thresholding_prop() (in module pynets.core.utils), 50
link_communities() (in module pynets.statistics.individual.algorithms), 89
load_mat() (in module pynets.statistics.individual.algorithms), 89
load_mat_ext() (in module pynets.core.utils), 50
load_runconfig() (in module pynets.core.utils), 50
make_local_connectivity_scors() (in module pynets.fmri.clustering), 72
make_local_connectivity_tscore() (in module pynets.fmri.clustering), 73
mask_roi() (in module pynets.core.nodemaker), 29
matching() (in module pynets.statistics.individual.multiplex), 95
mcsd_mod_est() (in module pynets.dmri.estimation), 62
merge_dicts() (in module pynets.core.utils), 50
mergedicts() (in module pynets.core.utils), 50
mmToVox() (in module pynets.core.nodemaker), 29

Module

pynets, 95
pynets.core, 58
pynets.core.nodemaker, 24
pynets.core.thresholding, 33
pynets.core.utils, 45
pynets.core.workflows, 57
pynets.dmri, 71
pynets.dmri.estimation, 60
pynets.dmri.track, 67
pynets.dmri.utils, 58
pynets.fmri, 78
pynets.fmri.clustering, 71
pynets.fmri.estimation, 75
pynets.plotting, 84
pynets.plotting.adjacency, 78
pynets.plotting.brain, 80
pynets.statistics, 85
pynets.statistics.group, 85
pynets.statistics.group.benchmarking, 85
pynets.statistics.group.prediction, 85
pynets.statistics.individual, 95
pynets.statistics.individual.algorithms, 85
pynets.statistics.individual.multiplex, 94
pynets.statistics.individual.spectral, 93
pynets.statistics.utils, 84

Most important() (in module pynets.statistics.individual.algorithms), 90

Mutual information 2d() (in module pynets.statistics.individual.multiplex), 95

N

Ncut() (in module pynets.fmri.clustering), 74
nilearn_atlas_helper() (in module pynets.core.nodemaker), 29
node_gen() (in module pynets.core.nodemaker), 29
node_gen_masking() (in module pynets.core.nodemaker), 30
normalize() (in module pynets.core.thresholding), 36
normalize_gradients() (in module pynets.dmri.utils), 59
np2gt() (in module pynets.statistics.individual.algorithms), 90
nx2gt() (in module pynets.statistics.individual.algorithms), 90

O

Optimize mutual info() (in module pynets.statistics.individual.multiplex), 95

P

Parcel masker() (in module pynets.core.nodemaker), 32
Parcellocate() (in module pynets.fmri.clustering), 74
Parcellocate_ncut() (in module pynets.fmri.clustering), 74
Participation coef() (in module pynets.statistics.individual.algorithms), 90
Participation coef_sign() (in module pynets.statistics.individual.algorithms), 91
Pass meta ins() (in module pynets.core.utils), 50
Pass meta_ins_multi() (in module pynets.core.utils), 52
Pass meta outs() (in module pynets.core.utils), 53
Perform thresholding() (in module pynets.core.thresholding), 37
Plot all_func() (in module pynets.plotting.brain), 80
Plot all_struct() (in module pynets.plotting.brain), 82
Plot all_struct_func() (in module pynets.plotting.brain), 83
Plot community conn mat() (in module pynets.plotting.adjacency), 78
Plot conn mat() (in module pynets.plotting.adjacency), 78
Plot conn_mat_func() (in module pynets.plotting.adjacency), 78
Plot conn_mat_struct() (in module pynets.plotting.adjacency), 79
Plot graph measure hists() (in module pynets.plotting.brain), 83
Plot network clusters() (in module pynets.plotting.brain), 83
Plot timeseries() (in module pynets.plotting.brain), 84
Prep tissues() (in module pynets.dmri.track), 68
Proportional() (in module pynets.fmri.clustering), 75
Prune small components() (in module pynets.statistics.individual.algorithms), 91
Prune suffices() (in module pynets.core.utils), 55
Pynets module, 95
Pynets.core module, 58
Pynets.core.nodemaker module, 24
Pynets.core.thresholding module, 24
Pynets.core.utils module, 45
Pynets.core.workflows module, 57
Pynets.dmri module, 71
pynets.dmri.estimation module, 60
pynets.dmri.track module, 67
pynets.dmri.utils module, 58
pynets.dmri.utils module, 78
pynets.fmri module, 75
pynets.fmri.clustering module, 71
pynets.fmri.estimation module, 75
pynets.plotting module, 84
pynets.plotting.adjacency module, 78
pynets.plotting.brain module, 80
pynets.statistics module, 85
pynets.statistics.group module, 85
pynets.statistics.group.benchmarking module, 85
pynets.statistics.group.prediction module, 85
pynets.statistics.individual module, 95
pynets.statistics.individual.algorithms module, 85
pynets.statistics.individual.multiplex module, 94
pynets.statistics.individual.spectral module, 93
pynets.statistics.utils module, 84

R
random_seeds_from_mask() (in module pynets.dmri.utils), 60
raw_graph_workflow() (in module pynets.core.workflows), 57
raw_mets() (in module pynets.statistics.individual.algorithms), 92
reconstruction() (in module pynets.dmri.estimation), 62
rich_club_coefficient() (in module pynets.statistics.individual.algorithms), 92
run() (pynets.core.utils.watchdog method), 57
run_tracking() (in module pynets.dmri.track), 68

S
save_3d_to_4d() (in module pynets.core.utils), 55
save_4d_to_3d() (in module pynets.core.utils), 55
save_coords_and_labels_to_json() (in module pynets.core.utils), 55
save_mat() (in module pynets.core.utils), 55
save_mat_thresholded() (in module pynets.core.utils), 55
save_nifti_parcel_map() (in module pynets.core.utils), 56
save_ts_to_file() (in module pynets.core.utils), 56
sfm_mod_est() (in module pynets.dmri.estimation), 63
show_template_bundles() (in module pynets.plotting.brain), 84
smallworldness() (in module pynets.statistics.individual.algorithms), 92
standardize() (in module pynets.core.thresholding), 37
streams2graph() (in module pynets.dmri.estimation), 63
subgraph_number_of_cliques_for_all() (in module pynets.statistics.individual.algorithms), 93

T
tens_mod_est() (in module pynets.dmri.estimation), 66
tens_mod_fa_est() (in module pynets.dmri.estimation), 66
thr2prob() (in module pynets.core.thresholding), 37
three_to_four_parcellation() (in module pynets.core.nodemaker), 33
thresh_func() (in module pynets.core.thresholding), 37
thresh_raw_graph() (in module pynets.core.thresholding), 40
thresh_struct() (in module pynets.core.thresholding), 40
threshold_absolute() (in module pynets.core.thresholding), 43
threshold_proportional() (in module pynets.core.thresholding), 43
timeseries_bootstrap() (in module pynets.fmri.estimation), 77
track_ensemble() (in module pynets.dmri.track), 69

V
view_tractogram() (in module pynets.plotting.brain), 84
VoxTomm() (in module pynets.core.nodemaker), 24

W
watchdog (class in pynets.core.utils), 57
weight_conversion() (in module pynets.core.thresholding), 44
weight_to_distance() (in module pynets.core.thresholding), 45
weighted_transitivity() (in module pynets.statistics.individual.algorithms), 93
workflow_selector()  (in module pynets.core.workflows), 57