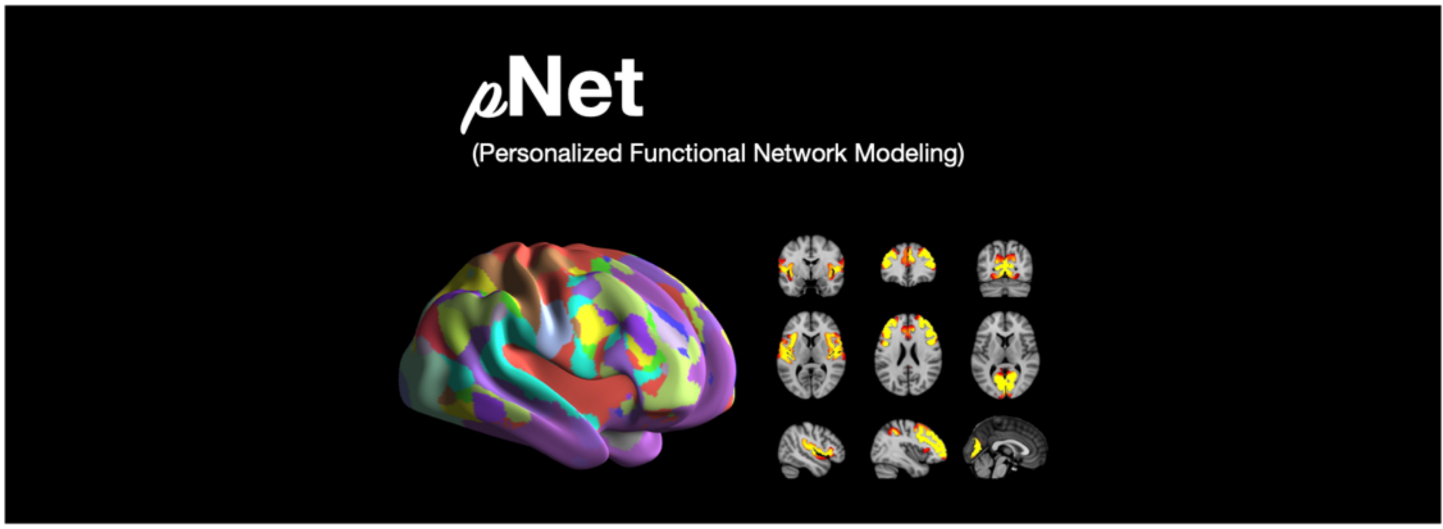


Manual

A guideline for pNet **Python** version

Yuncong Ma

Toolbox Overview

pNet is designed to provide a user-friendly interface to perform personalized functional network (FN) computation and visualization. It is open-source, cross-platform, and expandable. The Python version supports fMRI datasets in volume, surface and surface-volume types. It provides several user-friendly interfaces allowing for different preferences and coding skills, including step-by-step guide, Python script, and cluster computation. It integrates spatial-regularized non-negative matrix factorization (SR-NMF) and group-information-guided independent component analysis (GIG-ICA) to obtain personalized FNs. It outputs spatial and temporal components of both group-level and personalized FNs, quality control measurement, and HTML-based report.

This toolbox can be downloaded from https://github.com/YuncongMa/pNet and https://github.com/MLDataAnalytics/pNet .

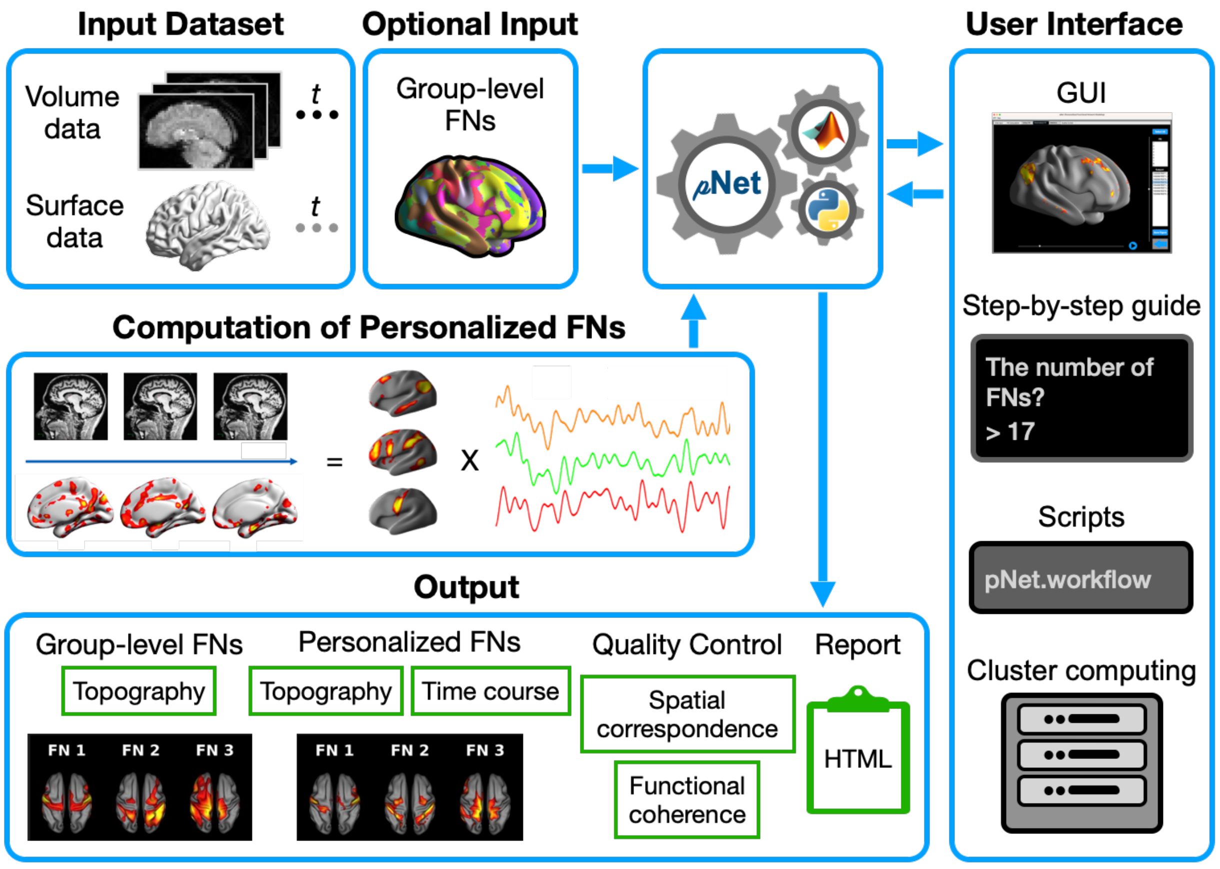
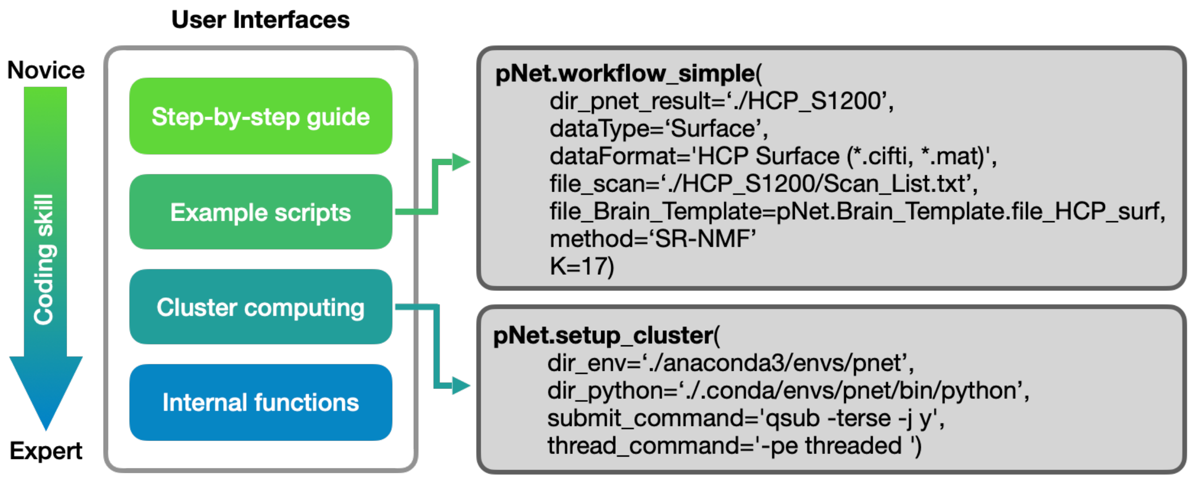


Figure 1. Overview of the pNet toolbox.

The Python version offers a variety of user interfaces to meet different user preferences and coding skills. The step-by-step guide provides detailed information of each parameter, and allows for minimal workflow setups using built-in brain templates. Example scripts provide both minimal and comprehensive setups for running scripts in local computers. pNet offers cluster computing option to take the advantage of large-scale computation resource. This option requires additional setups for the desired cluster environment. Besides, as an open-source toolbox, pNet provides internal functions to perform GIG-ICA, SR-NMF, and visualizations independent to its workflow, allowing experienced users to integrate our code into their own workflows or packages.



```
pNet.workflow_simple(
        dir_pnet_result='./HCP_S1200',
        dataType='Surface',
        dataFormat='HCP Surface (*.cifti, *.mat)',
        file_scan='./HCP_S1200/Scan_List.txt',
        file_Brain_Template=pNet.Brain_Template.file_HCP_surf,
        method='SR-NMF'
        K=17)
```

```
pNet.setup_cluster(
        dir_env='./anaconda3/envs/pnet',
        dir_python='./.conda/envs/pnet/bin/python',
        submit_command='qsub -terse -j y',
        thread_command='-pe threaded ')
```

Figure 2. The Python version offers a variety of user interfaces for users with different coding skills. Both minimal workflow settings and advanced options are available. Examples are shown in functions 'workflow_simple' and 'setup_cluster'.

Content

Installation

This chapter provides guidelines about ways to run this toolbox.

Install in Conda

The Python version is tested using Conda to install the required environment and packages. The whole package including the MATLAB version can be downloaded directly on the GitHub website, or using the terminal command ‘git clone’ (step 1). Then users can create a Conda environment for pNet using the ‘environment.yml’ file (step 2), or install all required additional packages step by step (step 3). Users need to activate the installed environment each time (step 4).

1. Download the whole pNet toolbox and import it to your Python environment

   ```
   $ git clone https://github.com/YuncongMa/pNet <User's directory>
   ```
2. Create a conda environment with all required packages for pNet.

   ```
   $ conda env create --name pnet -f environment.yml
   ```
3. Or install other required packages step by step.

   ```
   $ conda env create --name pnet python=3.8
   $ pip install numpy scipy scikit-learn pandas h5py
   $ conda install -c conda-forge nibabel
   $ conda install pytorch
   $ conda config --add channels conda-forge
   $ conda install mesalib vtk
   $ pip install matplotlib surfplot
   $ python -m pip install -U scikit-image
   ```
4. Activate conda environment

   ```
   $ conda activate pnet
   ```

System and hardware requirement

It is recommended to have at least 4 CPU cores, 16GB memory (RAM), and 10GB free disk space for local computation. The cluster computation may use a significant amount of computation resource for impressive performance. Please be aware that more CPU, memory and disk space will be needed to process bigger data or use parallel computation.

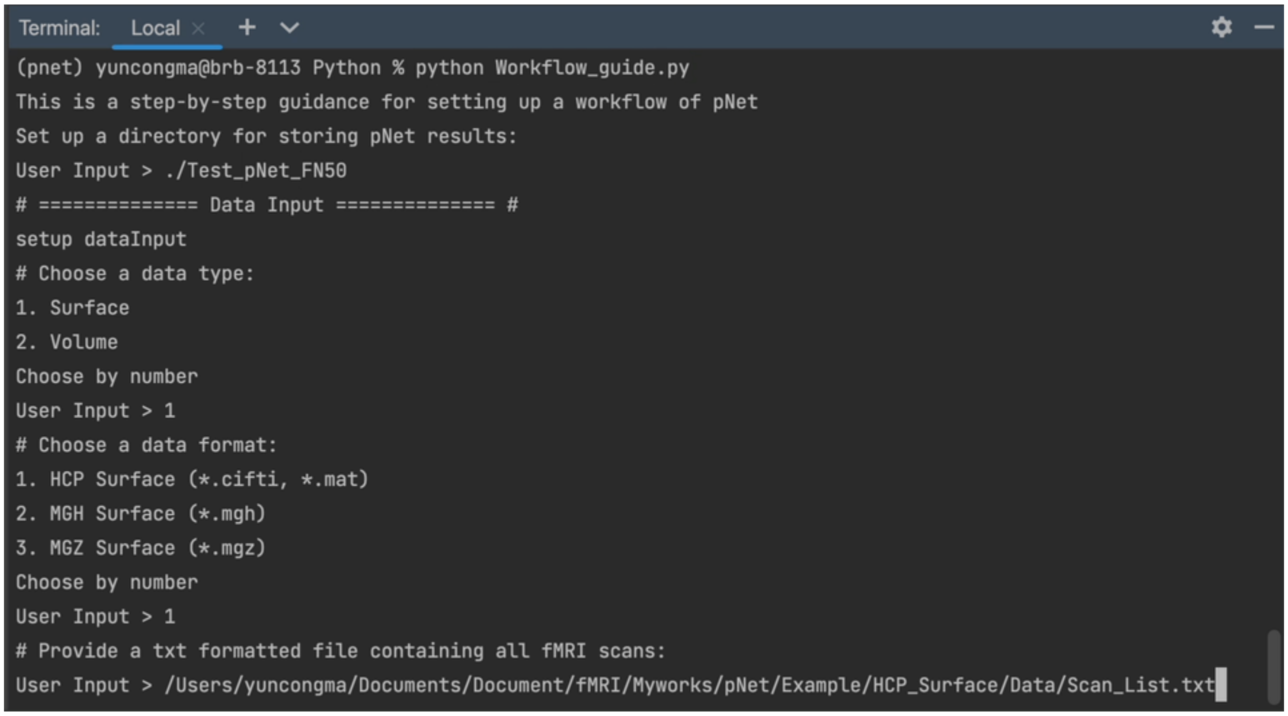
Run the Python version

The Python version can be executed on a local computer using PyCharm, VSCode or similar apps. It also has a server mode to interact with desired server environment for processing large datasets with multi-job execution capability.

pNet offers to a step-by-step guide to setup a workflow for running the Python version on a local computer. To do that, simply run the following terminal command in the conda environment.

```
$ python Workflow_guide.py
```

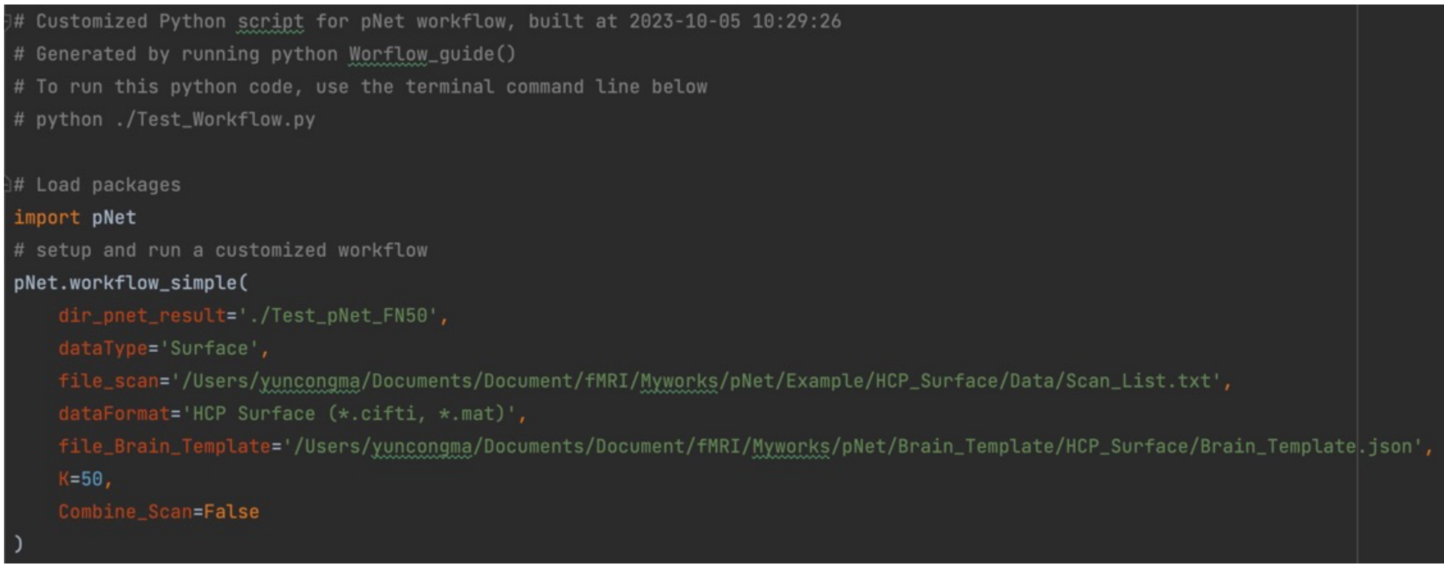
Users will be guided to type in the directory of the result folder, choose data type and so on.



```
(pnet) yuncongma@brb-8113 Python % python Workflow_guide.py
This is a step-by-step guidance for setting up a workflow of pNet
Set up a directory for storing pNet results:
User Input > ./Test_pNet_FN50
# ============== Data Input ============== #
setup dataInput
# Choose a data type:
1. Surface
2. Volume
Choose by number
User Input > 1
# Choose a data format:
1. HCP Surface (*.cifti, *.mat)
2. MGH Surface (*.mgh)
3. MGZ Surface (*.mgz)
Choose by number
User Input > 1
# Provide a txt formatted file containing all fMRI scans:
User Input > /Users/yuncongma/Documents/Document/fMRI/Myworks/pNet/Example/HCP_Surface/Data/Scan_List.txt
```

Screenshot of the step-by-step guide.

A Python script will be generated after finishing all required settings.



```
# Customized Python script for pNet workflow, built at 2023-10-05 10:29:26
# Generated by running python Worflow_guide()
# To run this python code, use the terminal command line below
# python ./Test_Workflow.py

# Load packages
import pNet
# setup and run a customized workflow
pNet.workflow_simple(
    dir_pnet_result='./Test_pNet_FN50',
    dataType='Surface',
    file_scan='/Users/yuncongma/Documents/Document/fMRI/Myworks/pNet/Example/HCP_Surface/Data/Scan_List.txt',
    dataFormat='HCP Surface (*.cifti, *.mat)',
    file_Brain_Template='/Users/yuncongma/Documents/Document/fMRI/Myworks/pNet/Brain_Template/HCP_Surface/Brain_Template.json',
    K=50,
    Combine_Scan=False
)
```

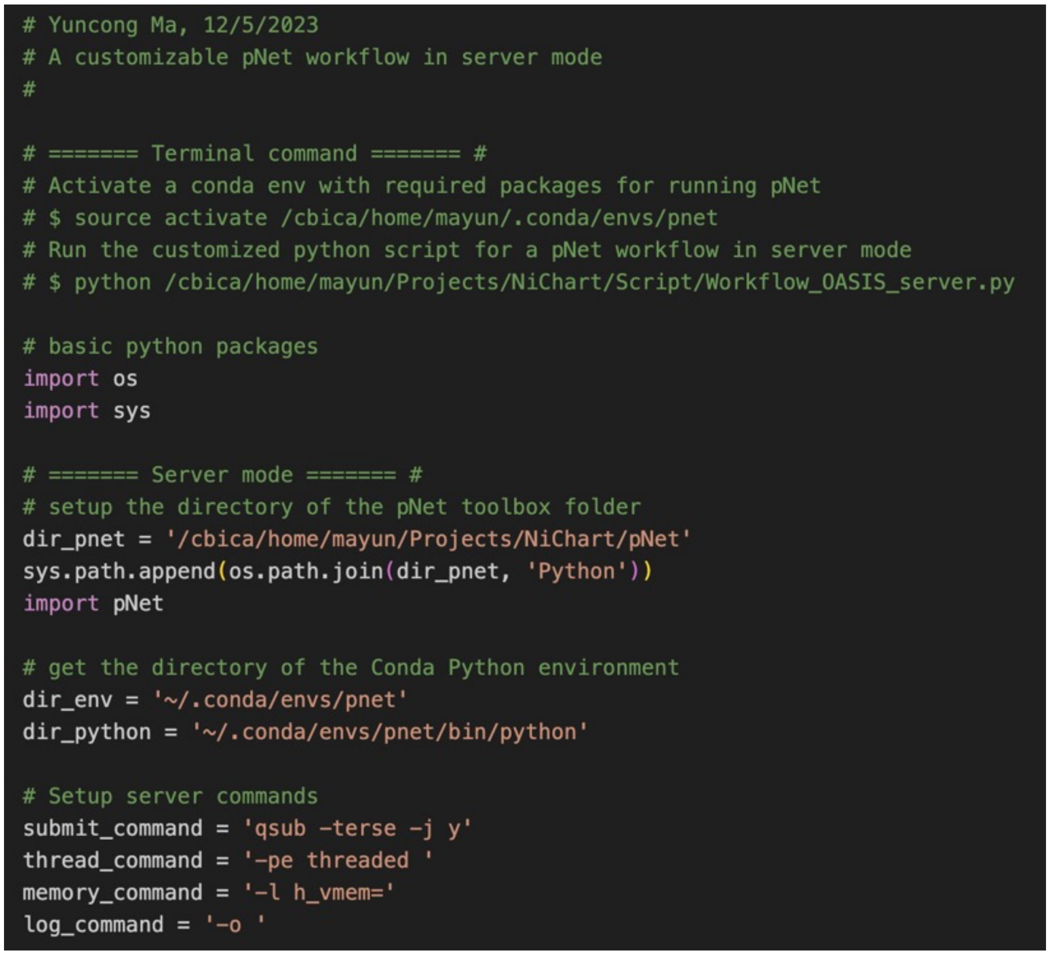
An example Python script generated by the step-by-step guide.

The Python version has a server mode, which has an example Python script (Example_Workflow_Server.py) for fast deployment.

The main difference of this script comparing to the workflow running on a local computer is the settings for the server environment. It includes the following inputs.

1. ‘dir_pnet’: directory of the pNet toolbox
2. ‘dir_env’: directory of the desired virtual environment, ex. ‘./anaconda3/envs/pnet’
3. ‘dir_python’: directory of the python in the virtual environment, ex. ‘./anaconda3/envs/pnet/bin/python’
4. ‘submit_command’: the command line to submit a bash job. In SGE system, ‘qsub’ is used to submit bash jobs to the server. An example SGE command is 'qsub -terse -j y'. No space is required at the end of this command string.
5. ‘thread_command’: the command to specify the thread number for a bash job. In SGE system, it is '-pe threaded '. The thread number can be set to ‘4’ for a particular job, which will be set like ‘-pe threaded 4’ in the command line.
6. ‘memory_command’: the command to specify the memory request for a bash job. In SGE system, it is ‘-l h_vmem=’. The memory request can be set to ‘50G’ for a particular job, which will be set like ‘-l h_vmem=50G’ in the command line.

7. ‘log_command’: the command to specify the log file for a bash job. In SGE system, it is ‘-o ’. The log file command line will be like ‘-o ./server_job_pFN.log’.



```
# Yuncong Ma, 12/5/2023
# A customizable pNet workflow in server mode
#

# ======= Terminal command ======= #
# Activate a conda env with required packages for running pNet
# $ source activate /cbica/home/mayun/.conda/envs/pnet
# Run the customized python script for a pNet workflow in server mode
# $ python /cbica/home/mayun/Projects/NiChart/Script/Workflow_OASIS_server.py

# basic python packages
import os
import sys

# ======= Server mode ======= #
# setup the directory of the pNet toolbox folder
dir_pnet = '/cbica/home/mayun/Projects/NiChart/pNet'
sys.path.append(os.path.join(dir_pnet, 'Python'))
import pNet

# get the directory of the Conda Python environment
dir_env = '~/.conda/envs/pnet'
dir_python = '~/.conda/envs/pnet/bin/python'

# Setup server commands
submit_command = 'qsub -terse -j y'
thread_command = '-pe threaded '
memory_command = '-l h_vmem='
log_command = '-o '
```

Screen shot of the example Python script to run a pNet workflow in server mode.

Data Input

This chapter provides guidelines to search and organize fMRI can files, load brain template files.

FN Computation

Statistical Analysis

Quality Control

The quality control is to ensure the spatial correspondence between personalized FNs and corresponding group-level FNs, and highlight the benefit of using personalized functional network modeling in terms of functional coherence. Specifically, the personalized FNs needs to have the highest spatial similarity (defined by Pearson correlation) to their corresponding group-level FNs. The minimum spatial similarity value for all personalized FNs is noted as the min∆Sim for each subject. If min∆Sim is smaller or equal to 0, it means that there is at least one personalized FN mismatches to the group-level FN or has no higher spatial similarity. Since our personalized FN computation method has quality assurance built in, all results will meet the quality control. Here we show one example with two mismatched FNs which will not be obtained from the toolbox.

Report

pNet automatically generates HTML-based reports for fast visual examination without using the MATLAB GUI version. A screenshot of an example web report is shown as below. This report provides description of the dataset, pFN model parameters, and figures for the gFNs, pFNs from a few subjects, quality control, as well as hyperlinks to pFNs of all subjects in the dataset. For faster web browsing experience, low resolution figures (ex. All(Compressed).jpg) are loaded to save memory usage.

pNet Report

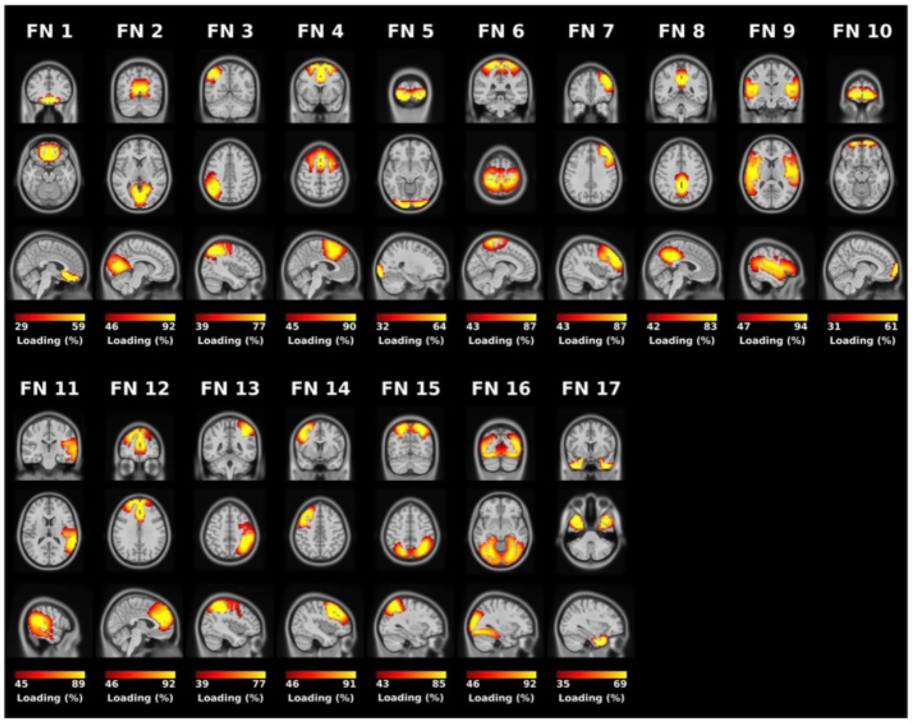
This is a brief report summarizing the results obtained using pNet.
pNet is an open-source toolbox (GitHub link) to obtain personalized functional networks from fMRI data.
pNet is developed by Yong's lab (lab link).
Report is generated at 2023-12-19 09:38:44

Essential settings for this pNet workflow

The personalized functional network modeling is based on the method named as SR-NMF.
The number of FNs is set to 17.
The fMRI data type is 'Volume' with format as 'Volume (*.nii, *.nii.gz, *.mat).'
The whole fMRI dataset contains 1754 scans from 521 subjects.
Parameters for the data input are stored in './Data_Input/Setting.json'
Parameters for the personalized FN modeling are stored in './FN_Computation/Setting.json'

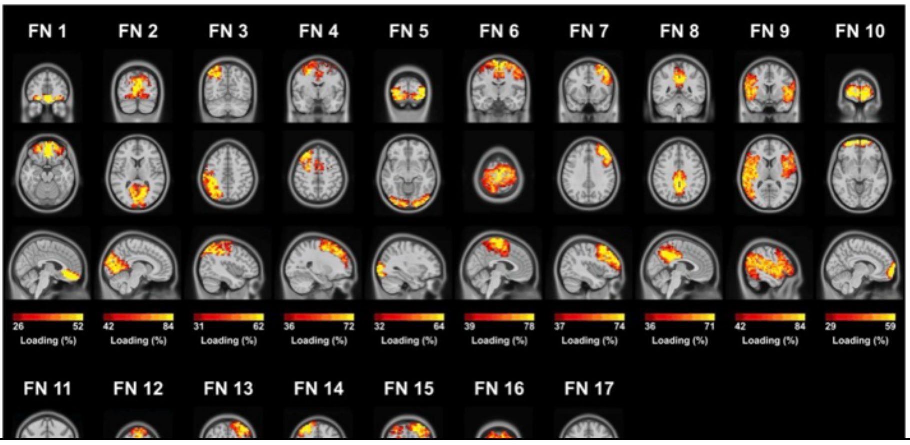
Group Functional Networks (gFNs)

The group FNs are derived using the whole fMRI dataset



Personalized Functional Networks (pFNs)

A few examples of personalized FNs derived from several subjects are shown as below.
Links to all pFNs are at the end of this web report.



Screenshot of a web-based report

Example