Yale

Department of Psychology
Wu Tsai Institute
100 College Street
New Haven, CT 06510

Jin Ke
jin.ke@yale.edu

Please find below the README information of the required contents for code and software submission checklist. We provide step-by-step guidance to run the code to fully replicate the main findings of our work entitled "**Ongoing thoughts at rest reflect functional brain organization and behavior**". These instructions include downloading and preprocessing the fMRI data, running the key analyses (brain decoding of ongoing thoughts), and plotting figures for publication.

1. System requirements
MATLAB_R2024a; RStudio Version 2024.04.2+764; Please set up the virtual environment with 'conda env create -f environment.yml' and 'conda activate' it when you run the python scripts.

2. Installation guide
MATLAB: https://www.mathworks.com/help/install/install-products.html
RStudio: https://posit.co/downloads/
Anaconda: https://www.anaconda.com/download
Jupyter: https://jupyter.org/install

Important note: Although most parts of our code could be successfully run on your local device (data analysis code, not fMRI preprocessing), **we highly recommend you run these code with high performance computing (HPC)**. For instance, fMRI preprocessing and analysis in this current project were performed on UChicago-Midway2 and Yale-Milgram.

3. Demo
Complied source code could be downloaded here: https://github.com/jinke828/rest_thoughts/code
Behavioral data could be found here: https://github.com/jinke828/rest_thoughts/data/beh
fMRI data could be downloaded from OpenNeuro: https://openneuro.org/datasets/ds006515.

- Please note that at the moment we only shared anonymous review links to reviewers and will make the data openly available to the public as the paper gets published

Upon downloading the required software, data and code, we suggest that you run the scripts by this following order. Note that most of these scripts have set the default dictionary, for example, `os.chdir('/gpfs/milgram/project/chun/jk2992/rest_thoughts/')`. Please modify it to the path of your downloaded 'rest_thoughts' folder.

Step 0: Preprocess the fMRI data downloaded from OpenNeuro

1) Code path: './rest_thoughts/code/a_preprocessing/'
2) Change the data path to where you store the data, run 'preprocess_session_mot_censor.py'. This is the first step of preprocessing, which computes the 24 motion parameters (6 motion parameters, 6 temporal derivatives, and their squares) and kicks off 'preprocess_scan.py' for each of the 5 fMRI runs. 'preprocess_scan.py' then sends slurm jobs to the cluster,

running multiple preprocessing jobs in AFNI by kicking off 'preprocessing_step1_getmotion_mot_censor.sh'. Note that 'mot_censor' stands for motion censor, censoring TRs with excessive motion (0.25). We did this for the annotated rest and CPT tasks, not for movies, for which you will need to run "preprocess_session.py". This step will take 30 to 90min depending on the computing power of your device (per job, so run many in parallel).

3) After this first step was done, run the second step 'preprocess_session_step2_mot_censor.py'. This step takes the output from the first step and complete the preprocessing steps. This step also takes 30 to 90min (per job, so run many in parallel).
4) If you hope to skip this step and acquire the preprocessed data, please email Jin or Monica.

Below starts data analysis. It includes code to replicate all the key neural findings (Fig. 2 - 5). The code for behavioral analysis is available upon request.
Code path: './rest_thoughts/code/b_analysis/'

Step01: Load the preprocessing fMRI data

1) **'step01_load_netts.m'** loads the preprocessed output: .netts (time series) and .netcc (FC patterns per functional run) files. If not familiar with AFNI or how the files were generated, view here: https://afni.nimh.nih.gov/pub/dist/doc/program_help/3dNetCorr.html
2) Running this code saves the time series of each of the 268 Shen ROIs in each rest run of each subject. The .mat file is saved here: './rest_thoughts/data/brain/rest_ts.mat'. The file size will be around 110.8 MB.
3) This process should take ~15 minutes. The folders of .netts and .netcc files are current hidden here: './rest_thoughts/data/brain' and will be publicly available easier replication of our findings.

Step02: Calculate FC patterns for each trial

1) **'step02_calc_FC.ipynb'** takes in the 'rest_ts.mat' generated from step01 and outputs trial-by-trial FC matrices (see *Methods* in the main text of manuscript). The data is saved here: './rest_thoughts/data/brain/rest_fc.mat' (size ~ 528.9 MB)
2) This step takes 2 to 4 hours. Take a nap and come back later :)

Step03: Representational similarity analysis

1) **'step03_FC-thoughts_RSA.ipynb'** calculates FC similarity with thought similarity (i.e., content or dimension similarity). The code reads in 'rest_fc.mat' to calculate FC similarity, reads in './data/beh/all_ratings.csv' to calculate thought dimension similarity.
2) It also reads in the semantic embeddings of free speech to calculate thought content similarity, however, the speech transcriptions and the embedding extracted from them will not be open sourced due to privacy concerns.
3) The results are saved here: './results/RSA/', which contained four files, one file for each of the four conditions: dimension, within-subject ('dimension_within-sub.mat'); dimension, between-subject ('dimension_fingerprints.mat'); content, within-subject ('content_within-sub.mat'); content, between-subject ('content_fingerprints.mat').

4) Running through the code should take less than 10 minutes.

Step04: Connectome-based modeling to predict thought dimensions

1) **'step04a_SVR_dimensions.ipynb'** reads in FC patterns ('rest_fc.mat') and behavioral ratings ('all_ratings.csv') and builds non-linear support vector regression models to predict the behavioral ratings from the FC patterns.
2) The script outputs the predicted score, predictive accuracy (*r*-value, MSE, and *r*-square), *p*-values (via permutation), as well as the selected FC features in every round of cross-validation, to a matlab file saved here: './results/CPMs/a_awake_features.mat'
3) Note that this example code shows modeling for the awake dimension. Users could easily use it to predict other dimensions by changing the variable name from 'Awake' to any of the 9 dimensions (e.g., 'Image').
4) Running this code can take long time (~40mins for each dimension).
5) When you finish running all the 9 dimensions, the code finally outputs a clean result .csv file to plot the beautiful **Fig. 3c**. Please use **step04b_plot_SVR.R** in RStudio to do so. This R code takes the observed *r*-values ('./results/CPMs/r_actual.csv') and the null *r* distributions ('r_null.csv'). You can also apply the same code on MSE ('mse_actual.csv', 'mse_null.csv') and *r*-square ('rsq_actual.csv', 'rsq_null.csv'), the results of which are saved in the same folder ('./results/CPMs/').
6) To predict imagery in the aphantasic twin, run **step04c_predict_aphantasia.ipynb**. Permutation of 10000 iterations could take ~200 minutes to run. Here our toy code runs 1000 iterations and outputs the results here: "./results/CPMs/predict_aphantasia.mat'

Step05: Connectome-based modeling to classify thought topics

1) Run **'step05_SVC_topics.ipynb'**, which takes in FC patterns ('rest_fc.mat') and human-annotated thought topics ('./data/beh/topics.csv') This code takes 30 – 60 mins to run, and outputs the results to: './results/CPMs/topic_SVC.mat'.

Step06: Generalizing the dimension and topic models to Human Connectome Project (HCP)

1) Run **'step06_crossdataset_prediction.ipynb'** which builds dimensional and topic models in the CAT dataset (in all subjects, without leave-one-out cross-validation) and applied the models to the HCP rest data.
2) This code reads the rest data of 908 HCP subjects from the 1200 Subject Release (S1200) after excluding for excessive motion (see *Methods*; './data/brain/net_hcp.mat'). We are not able to include this file here due to github size limit (~259.9MB), but we'd be happy to share the data upon request. We have included the model outputs for CCA in this folder: './results/predicted/'. Files from 'a_awake.mat' to 'i_detail.mat' and 'topics.mat'
3) This code takes ~30 minutes to run.

Step07: Relating predicted thoughts with behavioral measures using CCA

1) Run **'step07a_CCA_PC.m'** in MATLAB. This code takes in the predicted thought dimensions (e.g., 'a_awake.mat') and topics (i.e., 'topics.mat') generated in Step06. It also

reads several complimentary packages and data, for instance, ‘palm-alpha119’, ‘FSLNets’, and ‘varsQconf.txt’ (confounding variables), which are all stored here: ‘./code/b_analysis/z_CCA_helper/’.

2) Please note that the .txt file of 478 behavioral data in the 908 subjects (‘./code/b_analysis/z_CCA_helper/vars.txt’) for CCA modeling, and a .csv file (‘./data/beh/RESTRICTED_jinke/5_22_2024_13_45_6.csv’) for permutation testing with control of family structure, include restricted behavior in HCP. Thus we are not able to provide these data here. Please apply for the data access via this link: https://www.humanconnectome.org/study/hcp-young -adult/document/restricted-data-usage. Once you have the access to these data, we are happy to send additional instructions on how to format it correctly as the input to CCA.
3) We examined if our results are robust across combinations of thought (2-16) and behavior PCs (10,15,20,25,30,35,40). So we wrote a for loop to do this efficiently (line 81-129), each with its permutation to test significance (10000 iterations). Thus the code outputs 15*7 = 105 results files, one for each condition, which includes the performance of CCA model fit (grotR, grotRpval), canonical variates scores (grotU, grotV) and canonical weights (grotAAd, grotBBd). See *Methods* for more details of the analysis, and this page for an intuitive sense of what CCA does and how it is performed in MATLAB: https://www.mathworks.com/help/stats/canoncorr.html.
4) This code takes < 5 minutes to run. The output files are saved here: ‘./results/CCA/’.
5) To test the unique contributions of thoughts beyond rsFC, we built null thought models which are trained to predict shuffled thought ratings/labels from rsFC patterns. You can find the code in this folder **‘step07b_CCA_null-models’**. We used batch submission scripts (**‘submit_all.sh’**) that launch many null-model jobs at once to HPC cluster. **‘run_null.sh’** is the slurm job script that ‘submit_all.sh’ calls each time with a specified simulation index, and it runs the code in **‘run_null.py’**, which is the actual python code to build the null models.
6) Please note that each round of iteration (i.e., looping through the 9 dimensions and topics) can take 4-8 hours depending on the computation power of your HPC cluster. And the 1000 iterations in our study can take an incredibly long time. So we recommend you to test one iteration on your device, and reasonably allocate the jobs in parallel to minimize the run time and save computational resources for your institution. For your reference, we submitted 240 1-day jobs in total to finish these permutation. Note that you need to check the maximum number of job submission at once. For instance, Yale-Milgram allows 200 at a time.
7) The main texts show an example CCA fit, with 9 thought PCs and 20 behavior PCs (‘./results/CCA/predicted-thoughts_behavior_HCP_9thoughtPC_20behaviorPC.mat’). To prove the theoretical point that the association between rsFC and behavior might reflect differences in ongoing thoughts, we compared the behavioral weights observed in our study with those from Smith et al. (*Nat.Neurosci*., 2015) using **‘step07c_compare_Ke-Smith.R’**. It reads the cleaned-up results file ‘./results/CCA/Compare_Ke-Smith.csv’ and outputs an image (Suppl. Fig. 6b)

Please do not hesitate to reach out to Jin or Monica with questions.

Jin Ke
Graduate student
Department of Psychology
Wu Tsai Institute
Yale University
Email: jin.ke@yale.edu

On behalf of all authors