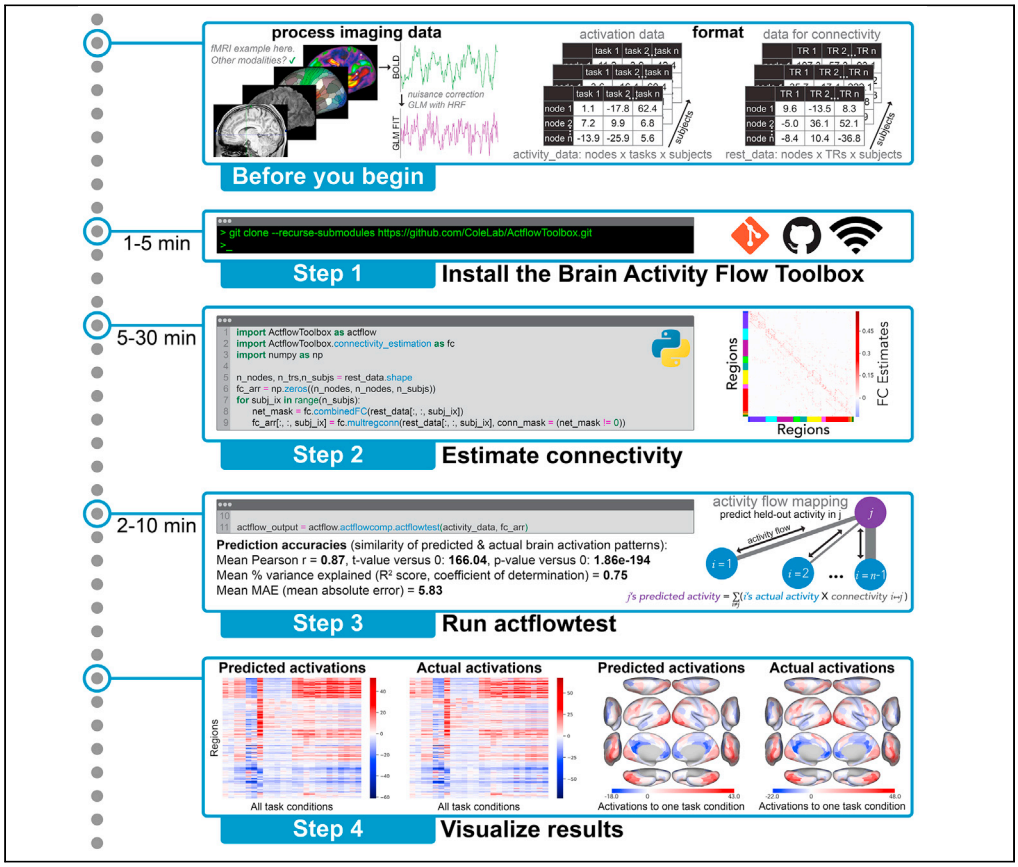
STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol for activity flow mapping of neurocognitive computations using the Brain Activity Flow Toolbox



Traditional cognitive neuroscience uses task-evoked activations to map neurocognitive processes (and information) to brain regions; however, how those processes are generated is unknown. We developed activity flow mapping to identify and empirically validate network mechanisms underlying the generation of neurocognitive processes. This approach models the movement of task-evoked activity over brain connections to predict task-evoked activations. We present a protocol for using the Brain Activity Flow Toolbox (https://colelab.github.io/ActflowToolbox/) to identify network mechanisms underlying neurocognitive processes of interest.

Carrisa V. Cocuzza, Ruben Sanchez-Romero, Michael W. Cole

carrisacocuzza@gmail.com (C.V.C.)
michael.cole@rutgers.edu (M.W.C.)

Highlights

Computational protocol that demonstrates how to use the Brain Activity Flow Toolbox

Model the influence of task-evoked activity flowing over brain connections

Empirically validate network mechanisms generating neurocognitive processes

High prediction accuracy of activity flow mapped activations across cognitive domains

Cocuzza et al., STAR Protocols 3, 101094
March 18, 2022
https://doi.org/10.1016/j.xpro.2021.101094

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol for activity flow mapping of neurocognitive computations using the Brain Activity Flow Toolbox

Carrisa V. Cocuzza,[1,2,3,4,*] Ruben Sanchez-Romero,[1] and Michael W. Cole[1,*]

[1]Center for Molecular and Behavioral Neuroscience, Rutgers University, Newark, NJ 07102, USA

[2]Behavioral and Neural Sciences PhD Program, Rutgers University, Newark, NJ 07102, USA

[3]Technical contact

[4]Lead contact

*Correspondence: carrisacocuzza@gmail.com (C.V.C.), michael.cole@rutgers.edu (M.W.C.)
https://doi.org/10.1016/j.xpro.2021.101094

SUMMARY

Traditional cognitive neuroscience uses task-evoked activations to map neurocognitive processes (and information) to brain regions; however, how those processes are generated is unknown. We developed activity flow mapping to identify and empirically validate network mechanisms underlying the generation of neurocognitive processes. This approach models the movement of task-evoked activity over brain connections to predict task-evoked activations. We present a protocol for using the Brain Activity Flow Toolbox (https://colelab.github.io/ActflowToolbox/) to identify network mechanisms underlying neurocognitive processes of interest.

For complete details on the use and execution of this protocol, please refer to Cole et al., 2021.

BEFORE YOU BEGIN

Utility of the protocol

Identifying brain mechanisms that generate neurocognitive processes is a central problem in neuroscience. Traditionally, brain activations have been simply linked with cognitive processes present during a given task, leaving outstanding questions about the mechanisms underlying those cognitive processes (Mill et al., 2017). Additionally, little is known about the influence of brain interactions on the instantiation of neurocognitive processes, even though there is increasing evidence that neural information is propagated over a distributed functional network architecture (Passingham et al., 2002; Ito et al., 2017; Mars et al., 2018). Here we demonstrate the activity flow mapping procedure, which: (1) builds connectivity-based models that simulate the activity flow processes that likely generate task-evoked activity in a given neural population, and (2) compares the resulting task-evoked activity predictions to the empirically observed task-evoked activity to provide evidence for/against the proposed generative process. This strategy has the benefit of each element of the procedure being empirically based and thus having a less abstract interpretation than other approaches (such as standard neural network or encoding models). Further, a growing body of research implementing activity flow mapping (discussed more extensively in expected outcomes) has observed high prediction accuracy across a variety of research questions, suggesting that the activity flow mapping framework is accurately characterizing many information processing mechanisms. Overall, this supports the conclusion that neurocognitive processes are specified largely by activity flowing over the brain's network architecture (Cole et al., 2016; Ito et al., 2020b). Activity flow mapping is well-suited to address a variety of research questions, such as those regarding the functional relevance of brain connectivity (e.g., a particular connection, a subnetwork, or a connectivity change), or the role of task-evoked

activations in one part of the brain in the generation of activity in other parts of the brain (e.g., prediction of visual cortex activity from prefrontal cortex activity). In the following protocol we provide a step-by-step guide to using the Brain Activity Flow Toolbox (https://colelab.github.io/ActflowToolbox/), which is a publicly available analysis toolbox that principally implements the activity flow mapping procedure. We will demonstrate its use with an example fMRI dataset, recommend best practices when possible, explain supplemental functionality of the toolbox (e.g., built-in functionality for computing connectivity with different methods), and outline inferential advantages and limitations of this approach.

Process imaging data

The activity flow mapping procedure was designed for neuroimaging data such as functional magnetic resonance imaging (fMRI), with future versions anticipated to accommodate electroencephalography/magnetoencephalography (EEG/MEG) (Mill et al., 2021), electrocorticography (ECoG), and spiking data. Throughout the protocol we will delineate activity flow mapping and discuss key considerations and recommendations with respect to fMRI data. Research using data of a different imaging modality should implement best practices in data processing based on the relevant literature. One of the main components of the activity flow mapping procedure is empirically estimated brain connectivity (see format data inputs for the Brain Activity Flow Toolbox for more details). Throughout the protocol we demonstrate this with estimates of functional connectivity (FC), however structural connectivity, which can be estimated using diffusion-weighted magnetic resonance imaging (amongst other approaches), can readily be used in place of functional connectivity (Yan et al., 2021a).

Preprocessing

Timing: Varies based on amount of data, but typically several hours – 1 week

The Brain Activity Flow Toolbox (also termed the Actflow Toolbox) is compatible with a variety of preprocessing pipelines, such as the Human Connectome Project (HCP) minimal preprocessing pipeline (Glasser et al., 2013) or fMRIPrep pipelines (Esteban et al., 2019) (see key resources table for open source analysis tools). FMRI preprocessing utilizes structural data (e.g., T1w and T2w images), functional data (e.g., resting and/or task time series), and references (e.g., field maps and select templates). We recommend preprocessing steps that include motion correction and alignment of functional to anatomical images along with empirically validated approaches for the removal of physiological artifacts such as motion and respiration artifacts (Ciric et al., 2017). The removal or attenuation of physiological artifacts is crucial for estimation of FC, given that these artifacts can introduce spurious dependencies among time series, leading to false inferences regarding brain connectivity (Power et al., 2014).

1. For the removal of motion and physiological artifacts, we recommend a version of the methods empirically validated by Ciric et al. (2017), including:
   a. Modeling and removing motion parameters, their derivatives, and quadratics. Standard motion correction algorithms provide six motion parameters (see key resources table), yielding 24 total nuisance parameters related to motion.
   b. aCompCor (Behzadi et al., 2007) for ventricle and white matter timeseries (i.e., physiological noise); using the first five principal components for each (independently extracted) plus their derivatives and quadratics, yielding 40 total nuisance parameters related to heart rate, motion, and respiration. Note that using aCompCor is equivalent to using global signal regression to better remove motion and respiratory artifacts (Power et al., 2018), but without the circularity introduced by regressing the mean gray matter signal from itself (which can artificially increase the number of negative functional connections; Murphy et al., 2009).

2. Further, general linear models (GLMs) are typically used to model activations, convolved with a hemodynamic response function (Friston et al., 1995; Handwerker et al., 2004; Lindquist et al., 2009). If using task-state data to estimate FC, we recommend using finite impulse response (FIR) regressed task data to estimate task-state FC (Cole et al., 2019) (see format data inputs for the Brain Activity Flow Toolbox for details, and the key resources table for source code information).

Spatial autocorrelation considerations

The Actflow Toolbox contains tools that address potential circularity in prediction accuracy. It is a known issue that fMRI data are spatially smooth (due to local vasculature and additionally from preprocessing smoothing steps), which may artificially inflate the correlation between signals of nearby voxels. Some studies suggest that smoothing due to local vasculature occurs in proximities roughly between 2 mm to 5 mm (Malonek and Grinvald, 1996; Logothetis and Wandell, 2004). This is theoretically problematic for activity-flow-predicted activations involving connectivity that is estimated between to-be-predicted target regions and nearby sources (i.e., the target region's activity may blur into nearby source activity). We recommend not using spatial smoothing as a preprocessing step to reduce this effect, but rather we have included tools to account for this potential confound by masking a conservative 10 mm of vertices around the target (i.e., excluding them from the analysis). This is demonstrated in the step-by-step method details (Part 2) (see Cole et al., 2021 for a specific investigation of this potential confound).

Format data inputs for the Brain Activity Flow Toolbox

Timing: 5–10 min

The two key data inputs for Actflow Toolbox analyses include: (1) task-evoked activation data, and (2) data to be used for connectivity estimation (e.g., resting-state data). We recommend storing these data in Python NumPy arrays for compatibility with Python, taking less memory compared to a Python list, and support of multidimensional data types. Further, NumPy supports a wide variety of mathematical operations and is cross-compatible with many other Python modules (e.g., SciPy). The two data inputs should be formatted with the following dimensions:

3. Task activation data: nodes by task conditions by subjects
   a. Nodes: brain regions, parcels, voxels, or vertices;
   b. Task conditions: cross-event/block averages (e.g., GLM beta estimates), blocks, events, or time points
   c. Subjects: this third dimension is optional and select functions iterate over subjects (detailed throughout this protocol in sample code blocks).

   Note: Given that we recommend iterating the main activity flow mapping steps over subjects, we do not specifically recommend z-scoring (normalizing) data across subjects. However, normalization may be an appropriate preprocessing step in select studies, depending on the research question(s) and data.

4. Data to be used for connectivity estimation: nodes by time points by subjects
   a. Nodes: brain regions, parcels, voxels, or vertices. These should match nodes in the task activation data.
   b. Time points: resting- or task-state time points.
   c. Subjects: this third dimension is optional and select functions iterate over subjects (detailed throughout this protocol in sample code blocks).

Note: If using task-state data for connectivity estimation (i.e., task-state FC), an additional dimension of condition may be used and iterated (looped) over outside the FC estimation function (see the step-by-step method details Part 2 code block). Suggested data format: nodes by time points by condition by subjects. If each condition has a different number of time points, a Python data dictionary is recommended. The difference between task activation data versus task data used to estimate task-state FC is that the former typically consists of the beta coefficient outputs from the task GLM (see before you begin: preprocessing), and the latter typically consists of the GLM residuals. Cole et al., 2019 systematically tested the impact that task-evoked activations have on task-state FC estimates. Accordingly, we recommend the best-performing method to address the possibility of artificially inflated task-state FC estimates, namely using FIR-regressed task data to estimate task-state FC. FIR regression is a flexible method for removing task-evoked responses that may induce spurious FC estimates (see Cole et al., 2019 for more details and the key resources table for source code information).

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
The Human Connectome Project (HCP) S1200 release.	Van Essen et al., 2013 and Marcus et al. (2013)	https://www.humanconnectome.org/study/hcp-young-adult
A portion of the HCP S1200 (n=30), preprocessed and available in the Brain Activity Flow Toolbox (and used throughout this protocol).	Ito et al. (2020a)	https://github.com/ColeLab/ActflowToolbox/tree/master/examples/HCP_example_data
Software and algorithms		
The Brain Activity Flow Toolbox	Cole et al. (2016)	https://doi.org/10.5281/zenodo.5768754
Cole-Anticevic Brain-Wide Network Partition (CAB-NP) Resource	Ji et al. (2019)	https://github.com/ColeLab/ColeAnticevicNetPartition
CombinedFC Toolbox	Sanchez-Romero and Cole (2021)	https://github.com/ColeLab/CombinedFC
Information Transfer Mapping code	Ito et al. (2017)	https://github.com/ColeLab/informationtransfermapping
Code for FIR regression to correct task-state FC confounds	Cole et al., 2019	https://github.com/ColeLab/TaskFCRemoveMeanActivity
HCP minimal preprocessing pipelines: documentation and links to install analysis tools	Glasser et al. (2013)	https://www.humanconnectome.org/software/hcp-mr-pipelines
fMRIPrep minimal preprocessing pipelines: documentation and links to install analysis tools	Esteban et al. (2019)	https://fmriprep.org/en/stable/
Python version 3 or higher	Van Rossum and Drake (2009)	https://www.python.org/
Other		
Demo Jupyter notebook available in the Brain Activity Flow Toolbox.	This paper	https://github.com/ColeLab/ActflowToolbox/blob/master/examples/HCP_example.ipynb

MATERIALS AND EQUIPMENT

- Python version 3 or higher recommended. All code in the toolbox (as well as code blocks throughout this protocol) conforms to PEP 8 style guidelines (https://www.Python.org/dev/peps/pep-0008/).
- Installation of The Brain Activity Flow Toolbox (see key resources table and step-by-step method details).
- Neuroimaging data formatted per before you begin.
- Recommended Python modules (i.e., packages) to install, either via PIP or website download, and then imported at the top of Python script following one of these syntax conventions:

```
# General
import <module_name>

# Importing a specific attribute (functions, classes, variables) from a module
from <module_name> import <attribute>

# Specifying short-hand keyword; to be used throughout script
import <module_name> as <short_hand_name>

# Specifying a short-hand keyword for a module attribute
import <module_name>.<attribute> as <short_hand_name>
```

Note: Modules are listed below in lowercase to follow conventions for installation and importing.

- o numpy version 1.17.0 or higher
- o h5py version 2.9.0 or higher
- o scipy version 1.5.0 or higher
  - scipy.stats
- o sklearn version 0.20.3 or higher
  - sklearn.linear_model.LinearRegression
- o statsmodels version 0.12.2 or higher
- o matplotlib version 3.0.3 or higher
  - matplotlib.pyplot
  - matplotlib.image
- o seaborn version 0.9.0 or higher
- o wbplot version 1.0.11 or higher
  - **Note**: This may require additional installation of nibabel version 3.1.0 or higher.
  - wbplot.pscalar

- Packages typically included with the Python distribution (or with the setuptools library) but recommended to specifically import at the top of the script (note that the versioning of these are linked with the Python version):
  - o sys
  - o math
  - o pkg_resources
  - o time
- Optional software: are linked with the Python version:
  - o Connectome workbench GUI to visualize and map fMRI data (https://www.humanconnectome.org/software/connectome-workbench).

Note: See troubleshooting problem 1 for details on troubleshooting potential issues with importing Python modules.

STEP-BY-STEP METHOD DETAILS

Part 1: Install the Brain Activity Flow Toolbox

Timing: 1–5 min

Installation of the Actflow Toolbox includes cloning (or downloading) the package from GitHub. Software development guidelines and example usage can be found at https://github.com/

ColeLab/ActflowToolbox. We recommend using Git via the command line interface (which requires an internet connection).

1. If not already installed, install Git by following the directions provided at https://git-scm.com/downloads.
2. Open the command line interface (e.g., Terminal application in macOS) and navigate to the appropriate directory:

```
>cd ~/research_project_directory/
```

3. Install Actflow Toolbox with Git:

```
>git clone --recurse-submodules https://github.com/ColeLab/ActflowToolbox.git
```

Part 2: Estimate connectivity

Timing: 5–30 min

This section describes how to estimate FC with the methods provided by the Actflow Toolbox. Brain connectivity is pivotal to the activity flow mapping procedure because it specifies (via estimated connectivity weights) how activity flowing over interacting brain regions contributes to a held out regions' task activity (Cole et al., 2016). Importantly, this builds upon the principle that connectivity is used to transfer information in the brain via activity flow processes (Ito et al., 2020b). We focus on the combinedFC (Sanchez-Romero and Cole 2021; see key resources table) estimation method as this is our current recommended best practice. The toolbox also contains the following options for FC estimation (most of which are detailed elsewhere throughout this protocol): Pearson correlation (corrcoefconn.py), multiple linear regression (multregconn.py), partial correlation (partial_corrconn.py), and principal component regression (pc_multregconn.py). In the following demonstrations we focus on resting-state FC but provide considerations for alternatively utilizing task-state FC. Further, it is worth noting that the connectivity estimate utilized in activity flow mapping does not need to be FC. For example, structural connectivity can be used (see expected outcomes for current examples in the literature).

4. Load data: in a Python compatible text editor (e.g., Atom), integrated development environment (e.g., PyCharm), or development package (e.g., Jupyter Notebook) that sources the same parent directory as /Actflowtoolbox/, load task activation data (Figure 1C) and data to be used in connectivity estimation (see before you begin for formatting) (Figure 1D).
   a. The following code block example loads a subset of N = 30 subject's data from the HCP S1200 release (see key resources table), which can be found in the example notebook included in the Actflow Toolbox (see key resources table).
   b. Further, throughout this protocol we used the multimodal parcellation (MMP) developed by Glasser et al. (2016) to define 180 cortical regions per hemisphere (360 total) (black outlines in Figure 1A). Minimally preprocessed versions of this data and registration details are publicly available at https://www.humanconnectome.org/ (as well as detailed in Glasser et al., 2013; see key resources table).

Note: It is worth noting that the core usage of the Actflow Toolbox does not strictly require data to be organized per this parcellation scheme. The only requirement is that the data inputs are formatted as described in the before you begin section. As described, a node can be at the level of parcels, regions (defined by the atlas of the user's choice), voxels, or vertices; activity flow mapping results will be returned at the same level.

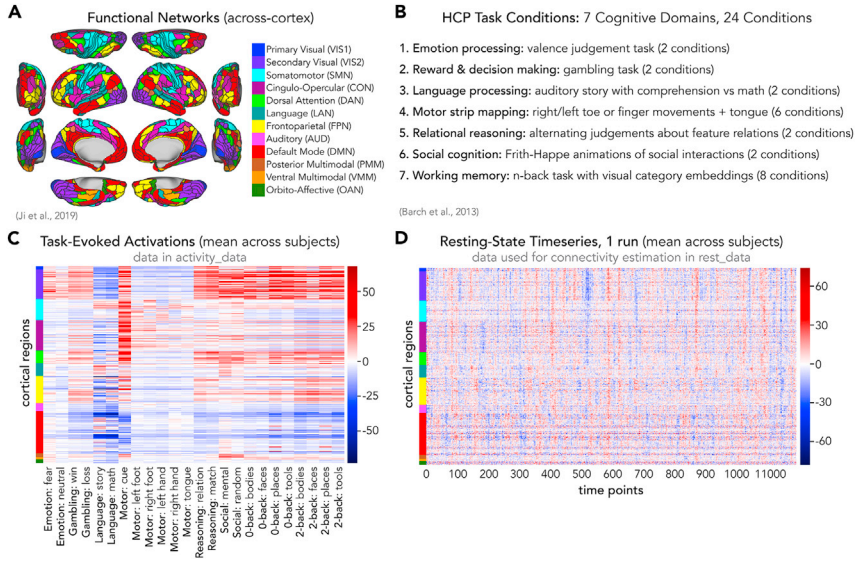


Figure 1. Loaded data, HCP S1200 example

(A) Cortical schematic of the Cole-Anticevic Brain-Wide Network Partition (CAB-NP) and its 12 functional networks from Ji et al. (2019), reproduced with permission. In select portions of this protocol, cortical regions were ordered by these networks (see y-axes in panels C and D). This corresponds to the variable <netorder> used in select code blocks.

(B) The seven cognitive domains (totaling 24 conditions) sampled in the HCP S1200 task fMRI dataset (Barch et al., 2013). These correspond to the x-axis in panel C and the second dimension of the input variable <activity_data>.

(C) Task-evoked activation patterns across 24 task conditions (mean across N = 30 subjects). This was the data contained in the input variable <activity_data>, which was used throughout the protocol, and constitutes the comparison reference for prediction accuracy statistics.

(D) The data utilized for connectivity estimation (mean across N=30 subjects) - which was the resting-state time series for 1 run (used to estimate rest FC) - in the examples used throughout this protocol. This corresponds to the input variable <rest_data>.

Note: The import lines in the corresponding code block specify all modules needed throughout all demonstration code blocks used in this protocol.

```
# Load example data from ~/HCP_example.ipynb
import pkg_resources
import sys
import numpy as np
import h5py
from scipy import stats
import time
import sklearn
from sklearn.linear_model import LinearRegression
import math as math
import statsmodels.api as sm
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
import wbplot
from wbplot import pscalar
import ActflowToolbox as actflow
import ActflowToolbox.connectivity_estimation as fc

# Instantiate variables
actflow_dir = pkg_resources.resource_filename(
  "ActflowToolbox.examples", "HCP_example_data/"
)
```

```
subj_nums = ["100206", "108020", "117930", "126325", "133928", "143224", "153934",
"164636", "174437", "183034", "194443", "204521", "212823", "268749", "322224", "385450",
"463040", "529953", "587664", "656253", "731140", "814548", "877269", "978578", "100408",
"108222", "118124", "126426", "134021", "144832"]
n_trs = 1195

# Load task activations (GLM betas); 360 MMP regions x 24 HCP tasks x 30 HCP subjects
file_path = actflow_dir + "HCP_example_taskactivations_data" + ".h5"
h5f = h5py.File(file_path, "r")
activity_data = h5f["taskbeta"][:]
h5f.close()
n_nodes, n_conditions, n_subjs = activity_data.shape

# Load resting-state fMRI data: 360 MMP regions x one run of resting-state fMRI (1195 time
points) x 30 HCP subjects
rest_data = np.zeros((n_nodes, n_trs, n_subjs))
for subj_ix in range(n_subjs):
  file_path = (
    actflow_dir + "HCP_example_restrun1_subj" + subj_nums[subj_ix] + "_data" + ".h5"
  )
  h5f = h5py.File(file_path, "r")
  rest_data[:, :, subj_ix] = h5f["restdata"][:]
  h5f.close()
```

Note: The corresponding code block demonstrates how to load the entire example dataset provided by the Actflow Toolbox, however the data files can be found in the cloned Actflow-Toolbox directory within the following path: /ActflowToolbox/examples/HCP_example_data/ . A single participant's resting-state data file is named as follows: HCP_example_restrun1_-subj<######>_data.h5 (where <######> is replaced by one of the participant codes listed in the variable in the corresponding code block called <subj_nums>). Participants' task activation data is concatenated in the array inside HCP_example_taskactivations_data.h5. The corresponding code block demonstrates how to extract relevant data from *.h5 files.

Optional: Avoid potential circularity due to spatial smoothness by excluding source vertices that are 10 mm from the target node border (see before you begin for background) (Figure 2). In the present protocol, target nodes refer to the iteratively held-out brain regions (or parcels, vertices, or voxels) (Figure 2 black region) whose task activations are predicted based on modeled contributions from source nodes, which are all other brain regions (excluding the target). For example: if there are 360 regions in total, the first iteration to predict task activations in target region one will hold out region one and use data from source regions two through 360 (359 sources in total) to make the prediction. This is iterated over all 360 target regions. The details of this procedure are described at-length in Part 3, where actflowtest is implemented. Target and source regions can be spatially contiguous, which can potentially raise concerns regarding spatial autocorrelation. To address this, we have added an (optional) step in the iterative process just described that excludes (or masks) vertices 10 mm outside of each target region's border (Figure 2 green vertices) from being part of the source set. That is, contiguous source regions' data will exclude data from vertices in this mask. This effectively adds a 10 mm "buffer" between a given target and all of its contiguous sources. The Actflow Toolbox has built-in functionality to perform this 10 mm masking procedure on fMRI data using the MMP atlas (Glasser et al., 2016) (Figure 2). We recommend the HCP minimal preprocessing pipeline (Glasser et al., 2013; see key resources table) (in addition to our other nuisance regression recommendations in before you begin), as it yields time series data resampled and aligned onto common cortical surface vertices and subcortical voxels, which are more generally referred to as "grayordinates" (note that the data demonstrated in this protocol is from the cortical surface only). Accordingly, we used the MMP atlas because it was developed with HCP grayordinates and vertex-to-region correspondence is readily

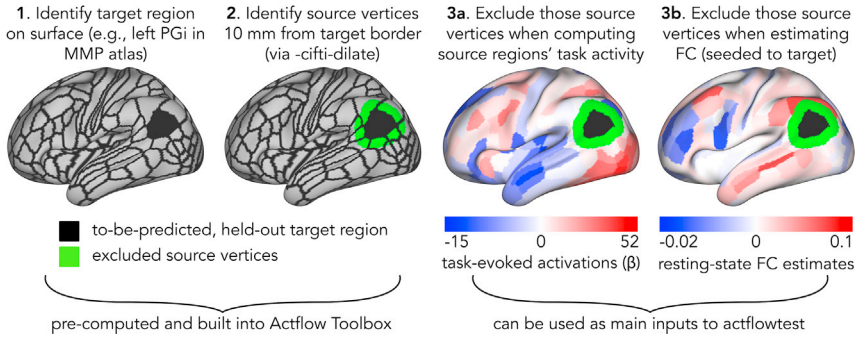


Figure 2. Optional step to avoid circularity due to spatial autocorrelation

All steps are performed iteratively, for each target region. **Step 1**: Identify the to-be-predicted, held-out target region on the MMP atlas surface. The example target region shown (black) is the left hemisphere area PGi (Glasser et al., 2016). Note that all steps are performed for each target region, including right hemisphere regions. The left hemisphere lateral surface is solely visualized for simplicity. **Step 2**: Use the HCP workbench command -cifti-dilate (https://www.humanconnectome.org/software/workbench-command/) to identify source region vertices (also termed grayordinates) that are 10 mm outside the given target region's border (green). Steps 1 and 2 are already computed across all target/source node sets, and the resulting masks are built into the Actflow Toolbox. **Step 3a**: Exclude the source vertices identified in Step 2 from the computation of source regions' task-evoked activations for a given task condition (example task condition "working memory 2-back: body" is shown). This is performed by calcactivity_parcelwise_noncircular. In this step, a source regions' activity is computed by taking the average of all included source vertices belonging to that source region. **Step 3b**: Exclude the source vertices identified in Step 2 from the estimation of FC with the target region. This is performed by calcconn_parcelwise_noncircular. The results of Steps 3a and 3b can be used as inputs into actflowtest, which is detailed in the step-by-step method details: Part 3.

identifiable (from a parcellation that underwent robust validation testing; see Glasser et al., 2016 for details). Currently only 10 mm masks are available in this step. Additionally, this requires the node dimension of input data to be composed of vertices. This can be implemented for both task activation data and connectivity estimation, as follows:

```
# Exclude source vertices 10 mm from target nodes to avoid circularity
# Using preloaded vertex-wise data: activity_data_vertex and rest_data_vertex

# Non-circular task activations
activity_data_noncirc = np.zeros((n_nodes, n_conditions, n_subjs))
for subj_ix in range(n_subjs):
    activity_data_here = activity_data_vertex[:, :, subj_ix]
    activity_data_noncirc_all = fc.calcactivity_parcelwise_noncircular(
        activity_data_here
    )
    for node_ix in range(n_nodes):
        for cond_ix in range(n_conditions):
            extracted_data = activity_data_noncirc_all[node_ix, node_ix, cond_ix]
            activity_data_noncirc[node_ix, cond_ix, subj_ix] = extracted_data.copy()

# Non-circular resting-state FC
fc_arr_noncirc = np.zeros((n_nodes, n_nodes, n_subjs))
for subj_ix in range(n_subjs):
    rest_data_here = rest_data_vertex[:, :, subj_ix]
    # Can specify connmethod = "combinedFC" here
    fc_arr_noncirc[:, :, subj_ix] = fc.calcconn_parcelwise_noncircular(rest_data_here)
```

Note: The foregoing procedure to exclude source vertices 10 mm from the target border may require 30 min - 1 h of computational time, depending on the input data dimensions (e.g., how

many subjects), as well as the computational environment's capabilities. For reference, we ran the last code block on a compute node on Rutgers University's Amarel high performance computing cluster (see here for specifications: https://oarc.rutgers.edu/resources/amarel/) to gauge run times. The data utilized had the following dimensions: <activity_data_vertex> was of shape 59412 vertices by 24 conditions by 30 subjects; <rest_data_vertex> was of shape 59412 vertices by 1195 time points by 30 subjects. The resting-state array was approximately 17 GB, whereas the task activity array was only 0.3 GB. The portion annotated with "Non-circular task activations" took approximately 8 min and the portion annotated with "Non-circular resting-state FC" took approximately 27 min. Altogether these processes used approximately 68% of 20 requested cores per node and 26% of 64 GB requested memory. However, these workload specifications relate to the size of vertex-level input data rather than the Actflow Toolbox functions, and should serve as a reference rather than a recommendation. Users that maintain vertex-level data and are interested in applying the above analysis steps should implement data management practices most suitable to their pipelines and computational environments.

c. The following are additional parameter options available when implementing calcactivity_parcelwise_noncircular (Figure 2, Step 3a), but are set to defaults in the corresponding code block:
   i. **dlabelfile**: This is a string indicating the parcellation reference file. We have provided a default dlabel file in-function. For each vertex it lists the corresponding region number in the MMP atlas (Glasser et al., 2016). While use of the optional step here (applying a 10 mm dilated mask to source vertices) is currently based upon the MMP atlas, this parameter is available as an input in case the load method fails, users may specify this file path directly. We have outlined this use case in troubleshooting: problem 4. This file is a *dlabel.nii, which is CIFTI-2 formatted and readable by most neuroimaging software. We recommend the following resource for more information on CIFTI files: https://wiki.humanconnectome.org/display/PublicData/HCP+Users+FAQ.
   ii. **cortex_only**: This is a Boolean that can be set to either True or False. The default is True, which indicates to assess only cortical regions. If False, both cortical and subcortical regions will be assessed.
   iii. **verbose**: This is a Boolean that is set to False by default. If True, the user will see printed output that tracks the progress of the function.

d. The following are additional parameter options available when implementing calcconn_parcelwise_noncircular (Figure 2, Step 3b), but are set to defaults in the corresponding code block:
   i. **connmethod**: This is a string that indicates which functional connectivity estimation method to use. The options are: 'multreg' (default) (multiple linear regression), 'pearsoncorr' (Pearson correlation), 'pc_multregconn' (principal components regression), and 'combinedFC' (Sanchez-Romero and Cole, 2021).
   ii. **dlabelfile**: The same as listed above for calcactivity_parcelwise_noncircular.
   iii. **precomputedRegularTS**: This is an optional input parameter that is set to None by default. If provided, it should be an array of the mean time series of the original set of regions (i.e., pre-masking) that has the same dimensions as the input variable <data> (i.e., <rest_data> in the corresponding example code block).
   iv. **cortex_only**: The same as listed above for calcactivity_parcelwise_noncircular.
   v. **verbose**: The same as listed above for calcactivity_parcelwise_noncircular.

5. Continuing in the same editor, environment, or notebook, continue on the next line with the following code to estimate connectivity.

Note: This example computes resting-state FC with the combinedFC (Sanchez-Romero and Cole, 2021) method, as our current recommended best practice (Figure 3A). The Actflow

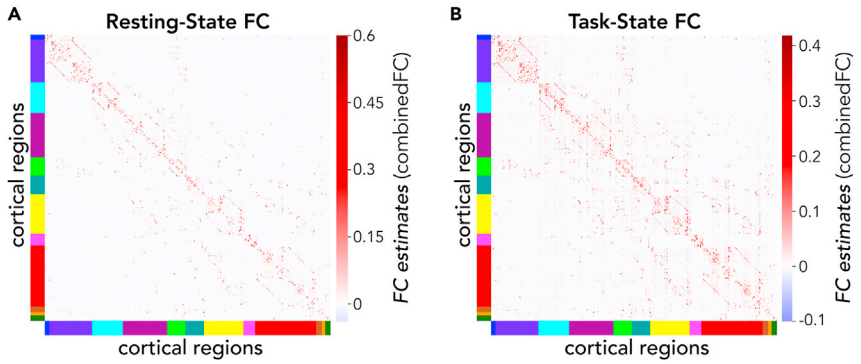


Figure 3. Functional connectivity

(A) The grand average (mean of N = 30 subjects resting-state connectivity matrix estimated via combinedFC) of 360 MMP cortical regions, ordered along each axis per the CAB-NP (see Figure 1A). This represents the connectivity estimates used in this protocol for activity flow mapping.

(B) The same as in panel A, but using task timeseries to estimate FC (mean across N = 30 subjects and all HCP tasks). The use of task-state FC in mapping cognitive computations was assessed in Cole et al., 2021.

Toolbox provides a variety of FC estimation methods; see extended example usage at https://github.com/ColeLab/ActflowToolbox/blob/master/examples/HCP_example.ipynb, which can be applied to either resting-state or task-state data.

Note: This assumes <rest_data> has been formatted per before you begin recommendations: in a NumPy array of shape nodes by TR by subject (note the for-loop iterates over subjects).

```
# Estimate resting-state functional connectivity with combinedFC

n_nodes, n_trs, n_subjs = rest_data.shape
fc_arr = np.zeros((n_nodes, n_nodes, n_subjs))
for subj_ix in range(n_subjs):
    net_mask = fc.combinedFC(rest_data[:, :, subj_ix])
    fc_arr[:, :, subj_ix] = fc.multregconn(
        rest_data[:, :, subj_ix], conn_mask=(net_mask != 0)
    )
```

a. The following are additional parameter options available when implementing combinedFC (see Sanchez-Romero and Cole, 2021 for more details), but are set to defaults in the corresponding code block:
   i. **target_ts**: This is an optional parameter set to None by default. This is recommended for two use cases: (1) When the user wants to estimate connectivity for only one target node (with all other source nodes). This should contain the target node's time series, or, a one dimensional vector of activity across time points. Connectivity will be compared to each source node's time series in the <dataset> input (e.g., <rest_data> in the corresponding code block), and a one dimensional connectivity vector will be returned. (2) When calling the optional parameter of <conn_method='combinedFC'> in the calcconn_parcelwise_noncircular function. The user does not need to change any inputs in this case because the calcconn_parcelwise_noncircular function has built-in handling when <conn_method='combinedFC'>.
   ii. **parcelInt**: This is an optional parameter set to None by default. This is also used by calcconn_parcelwise_noncircular when <conn_method='combinedFC'> which also has built-in handling to manage this input. This input helps index the final connectivity network. If the user would like to use <target_ts> without calcconn_parcelwise_noncircular (e.g., has

another use for separating the source and target node time series), then this is the indexing value of the target node. If there is only one target node, this can be set to 0.

iii. **source_cols**: This is an optional parameter set to None by default. This is the corresponding source node indexing information to <parcelInt> explained above. That is, calcconn_parcelwise_noncircular has built-in handling if <conn_method='combinedFC'>. Otherwise, this is the indexing vector for the source nodes (with the target held out) in the final connectivity network. The following code block is a simple example of how this can be determined:

```
# Determine the indices of source node columns for a given target node (parcelInt); to be used to
index the final connectivity array
parcelInt = 0
num_sources = 360
source_cols = np.arange(num_sources)
source_cols = np.delete(source_cols, parcelInt)
```

iv. **methodCondAsso**: A string that determines which method to use to evaluate conditional associations (i.e., the first portion of the combinedFC procedure). The options are 'partialCorrelation' (default) or 'multipleRegression'.

v. **methodParcorr**: A string that specifies how the partial correlation matrix is computed, and is thus only relevant when <methodCondAsso='partialCorrelation'>. The options are 'inverseCovariance' (default) and 'regression'.

vi. **alphaCondAsso**: A scalar that specifies the alpha significance cut-off for the conditional association evaluation. The default is 0.01.

vii. **methodAsso**: A string specifying the method for the second procedure performed by combinedFC: testing whether edges should be included or excluded in the connectivity matrix (i.e., the unconditional association procedure). The options are 'correlation' (default) and 'simpleRegression'.

viii. **alphaAsso**: A scalar that specifies the alpha significance cut-off for the unconditional association evaluation. The default is 0.01.

ix. **equivalenceTestAsso**: A Boolean that is set to False by default. If set to True, the unconditional association procedure will be performed via equivalence testing. When False, two-sided null hypothesis testing will be performed.

x. **lower_bound**: A scalar that specifies the negative bound for the minimum r value of interest in the equivalence test. The default value is -0.1.

xi. **upper_bound**: A scalar that specifies the positive bound for the minimum r value of interest in the equivalence test. The default value is +0.1.

b. The following are additional parameter options available when implementing multregconn:

i. **target_ts**: This is an optional parameter set to None by default. This is recommended for two use cases: (1) When the user wants to estimate connectivity for only one target node (with all other source nodes). This should contain the target node's time series, or, a one dimensional vector of activity across time points. Connectivity will be compared to each source node's time series in the <activity_matrix> input (e.g., <rest_data> in the corresponding code block), and a one dimensional connectivity vector will be returned. (2) When calling the optional parameter of <conn_method='multreg'> in the calcconn_parcelwise_noncircular function. The user does not need to change any inputs in this case because the calcconn_parcelwise_noncircular function has built-in handling when <conn_method='multreg'>.

ii. **parcelstoexclude_bytarget**: This is an optional parameter set to None by default. This can be a dictionary of lists, where each list includes source regions to exclude for each target parcel. This is not used when <target_ts> is set, and thus not used by calcconn_parcelwise_noncircular. This can be used when there are specific source regions (the entire region) that should be held out from connectivity estimation.

iii. **conn_mask**: This is an optional parameter set to None by default. This is mainly used when calling multregconn after combinedFC (as in the corresponding code block), and specifies a mask to exclude some connections from being fit in the multiple regression procedure by setting them to zero. This should be an array that matches the dimensions of the output <fc_arr> consisting of ones and zeros: ones indicate a connection and zeros indicate no connection. If <target_ts> is set, the dimensions of <conn_mask> should be the number of nodes in the original <activity_matrix> (i.e., <rest_data> in the corresponding code block) by one.

Alternatives: Estimate FC with task-state data (see before you begin for recommendations and data formatting details) (Figure 3B).

```
# Estimate task-state functional connectivity with combinedFC

# Note: this assumes each task condition contains the same number of time points. If this is not
the case, a data dictionary may be used (instead of a 4D numpy array).
n_nodes, n_trs, n_conditions, n_subjs = task_data.shape
fc_arr = np.zeros((n_nodes, n_nodes, n_conditions, n_subjs))
for subj_ix in range(n_subjs):
    for cond_ix in range(n_conditions):
        net_mask = fc.combinedFC(task_data[:, :, cond_ix, subj_ix])
        fc_arr[:, :, cond_ix, subj_ix] = fc.multregconn(
            task_data[:, :, cond_ix, subj_ix], conn_mask=(net_mask != 0)
        )
```

Optional: Visualize the connectivity matrix with regions sorted along the x- and y-axes per the MMP (Glasser et al., 2016) and CAB-NP (Ji et al., 2019), per the following code (resting-state FC example):

```
# Cole-Anticevic Brain-Wide Network Partition (Ji et al., 2019) variables
networkpartition_dir = pkg_resources.resource_filename(
    "ActflowToolbox.dependencies", "ColeAnticevicNetPartition/"
)
networkdef = np.loadtxt(networkpartition_dir + "/cortex_parcel_network_assignments.txt")
networkorder = np.asarray(sorted(range(len(networkdef)), key=lambda k: networkdef[k]))
netorder = networkorder.copy()

# Visualize FC matrix with Toolbox function
fc_mat = np.mean(fc_arr, axis=2)[netorder, :][:, netorder]
fig = actflow.tools.addNetColors_Seaborn(fc_mat)

# Note: fig can be used to save the figure as follows (change strings as appropriate):
fig_directory = "/my/figure/directory/here/"
fig_file_name = fig_directory + "fc_matrix.png"
fig.savefig(fig_file_name, bbox_inches="tight", format="png", dpi=600)
```

⚠ CRITICAL: Timing depends on computational capabilities of a given system and/or environment. We recommend a test run to assess whether a high performance computing environment will be needed, or, if the analysis should be broken up into smaller groups. The following dimensions can become particularly large in select datasets: nodes (e.g., thousands of vertices), conditions (when estimating task-state FC), and/or subjects (e.g., N > 100). The following code modification can be used as a test run for n = 10 subjects:

```
# Wrapping the time function around the code to assess computation time

n_test_subjs = 10

start_time = time.time()

n_nodes, n_trs, n_subjs = rest_data.shape
fc_arr = np.zeros((n_nodes, n_nodes, n_subjs))
for subj_ix in range(n_test_subjs):
    net_mask = fc.combinedFC(rest_data[:, :, subj_ix])
    fc_arr[:, :, subj_ix] = fc.multregconn(
        rest_data[:, :, subj_ix], conn_mask=(net_mask != 0)
    )

end_time = time.time()
print(
     "Computation time for "
     + str(n_test_subjs)
     + " subjects: "
     + str((end_time - start_time) / 60)
     + " minutes."
)
```

```
Computation time for 10 subjects: 0.08017793099085489 minutes.
```

Note: This test of elapsed computational time was performed on the Rutgers University Amarel compute cluster (https://oarc.rutgers.edu/resources/amarel/) on one compute node with 10 available cores and 64 GB available memory. We additionally ran the same code block on a local machine, which was a Macbook Pro with the following specifications: 2017 model with macOS Catalina 10.15.6, 2.3 GHz dual-core processor, and 16 GB of RAM. The elapsed time was approximately 0.09 min when running this code block in a Python-enabled terminal on that local machine.

Note: As in Yan et al. (2021a), structural connectivity may be used in place of FC. Further, alternate estimates of FC may be used, such as latent FC (as in: McCormick et al., 2021). If using a brain connectivity estimate that is not covered by the Actflow Toolbox, this section (Part 2) can be substituted with project-specific code (or skipped if estimated outside of Python) and the resulting FC data array can be used in Part 3 as long as it follows the formatting convention of: nodes by nodes by subjects.

Part 3: Run actflowtest

Timing: 2–10 min

This section provides instructions for running actflowtest, which is the main mapping procedure in the Actflow Toolbox. We also provide a look "under the hood" and describe the quantitative steps performed within actflowtest (depicted diagrammatically in Figure 4).

6. Run the main function of the toolbox, actflowtest.py, which can be continued from the code above. This is the activity flow mapping procedure (Figure 4), which comes included with predicted-to-actual similarity assessment across multiple conditions and subjects. The following group-level statistics are returned to indicate prediction accuracy: Pearson r with t-testing, percent variance explained via the coefficient of determination, and the mean absolute error (see expected outcomes for details on accuracy indices).

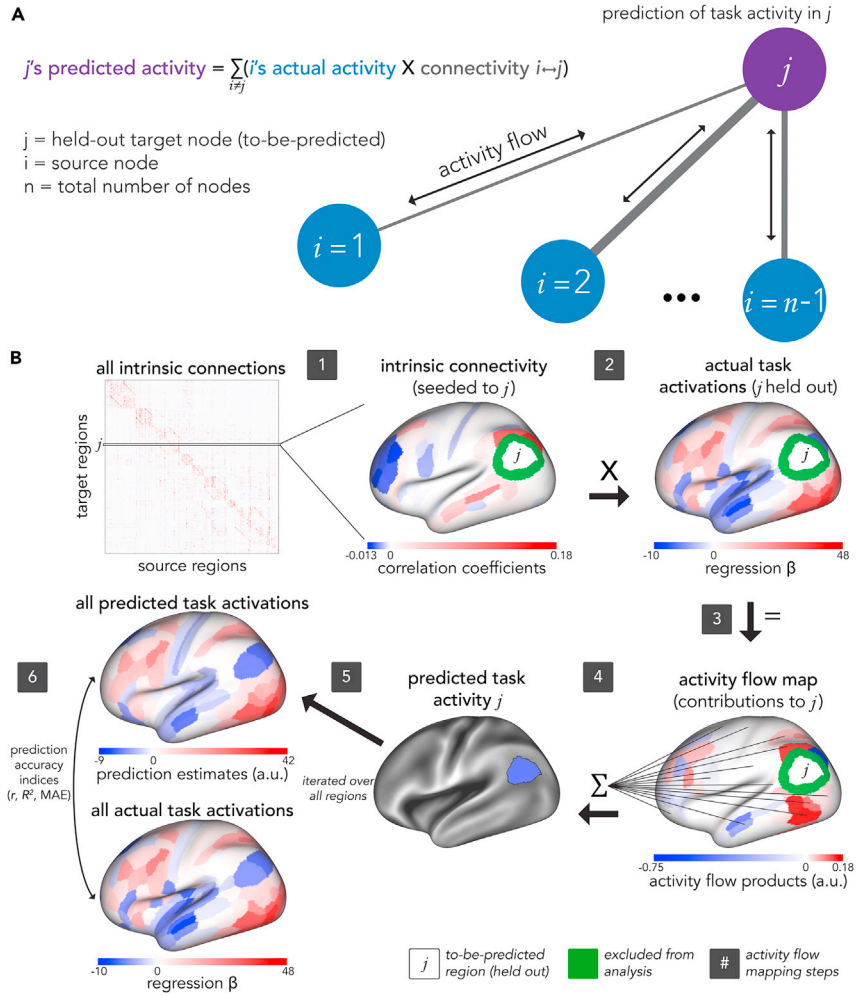


Figure 4. Activity flow mapping procedure performed by actflowtest

(A) Activity flow mapping toy diagram and corresponding formula (adapted from Cole et al., 2016 with permission). Task activity for the held out node, *j* (purple), is predicted by the sum of task activity of all other nodes, *i* (blue) (where *n* = total number of nodes), weighted by their connectivity estimates with *j* (grey).

(B) Activity flow mapping procedure performed by actflowtest with the example HCP S1200 data (N = 30) used throughout this protocol. The computations inside actflowtest are numbered inside dark grey squares, as follows: [1] For held-out target region *j*, connectivity estimates between *j* and all other source regions are [2] multiplied by all other regions' actual task activations (iterated per task). [3] The resulting activity flow map contains the task activations of all source regions weighted by their connectivity estimate with *j*. [4] Flow map values are summed to equal the predicted activity of *j*. [5] Computations 1–4 are iterated over all regions and all tasks, which produces a map of activity-flow predicted task activations across the brain. [6] Predicted activations are compared with actual activations via prediction accuracy indices (see expected outcomes). Excluded source vertices (10 mm from the target region *j*; see before you begin; step-by-step method details part 2; and Figure 2) are masked in green.

Note: This assumes <activity_data> has been formatted per before you begin recommendations, or, in a NumPy array of shape nodes by conditions by subject (note that the last dimension is optional).

```
# Activity flow mapping with resting-state FC estimated via combinedFC
actflow_output = actflow.actflowcomp.actflowtest(activity_data, fc_arr)
```

Note: By default actflowtest prints prediction accuracy results; see expected outcomes for examples.

a. The following are additional parameter options available when implementing actflowtest, but are set to defaults in the corresponding code block:
   i. **actVect_group_test**: This is an optional parameter set to None by default. This can be used to provide independent data to assess prediction accuracy. Some examples include separate runs or separate participants. The dimensions of this should match <actVect_group> (i.e., the variable <activity_data> in the corresponding code block).
   ii. **print_by_condition**: This is a Boolean set to True by default that directs the model_compare function to print the prediction accuracy statistics for each condition separately.
   iii. **full_report**: This is a Boolean set to False by default. If True, the model_compare function will assess prediction accuracy across multiple dimensions of the data. See expected outcomes for a full explanation of these statistics.
   iv. **separate_activations_bytarget**: This is a Boolean set to False by default. If True, this indicates that the input <actVect_group> (i.e., the variable <activity_data> in the corresponding code block) has a single activation vector for the to-be-predicted target node (i.e., to perform activity flow mapping for individual target nodes) and will provide additional handling for this.
   v. **transfer_func**: This is an optional parameter set to None by default, and accepts a string otherwise. If set, activity flow mapping will utilize a transfer function. The options are 'linear', 'relu', 'sigmoid', and 'logit'. This will be applied to the activity of all other nodes (blue nodes in Figure 4A; i.e., source nodes for a given target node) in the activity flow mapping procedure. Usage of a transfer function assumes that the target node's time series is primarily driven by inputs (e.g., local field potentials), such that source time series need to be converted from inputs to outputs via a transfer function.
   vi. **avgthencomp_fixedeffects**: This is a Boolean set to False by default. If True, the model_compare function will assess prediction accuracy after averaging across participants, which is sometimes referred to as a "fixed effects" analysis. We recommend analyses with participants as random effects (i.e., effects for each participant, which can be later averaged to assess group effects), and thus recommend keeping this False (but have included the option for specific use cases).

7. Extract results from the <actflow_output> dictionary with the following code (continued from above):

```
# Extract the predicted and actual activations
predicted_activations_array = actflow_output["actPredVector_bytask_bysubj"]
actual_activations_array = actflow_output["actVect_actual_group"]
```

Optional: The following code can be used to extract prediction accuracy:

```
# Optionally extract prediction accuracies to NumPy arrays
correlations_full_model = actflow_output["model_compare_output"][
    "fullcomp_compthenavg_output"
]["corr_vals"]
expl_variances_full_model = actflow_output["model_compare_output"][
    "fullcomp_compthenavg_output"
]["R2_vals"]
maes_full_model = actflow_output["model_compare_output"]["fullcomp_compthenavg_output"][
    "mae_vals"
]

tstat_full_model = actflow_output["model_compare_output"][
    "tval_ActflowPredAcc_fullcomp"
]
pval_full_model = actflow_output["model_compare_output"]["pval_ActflowPredAcc_fullcomp"]
```

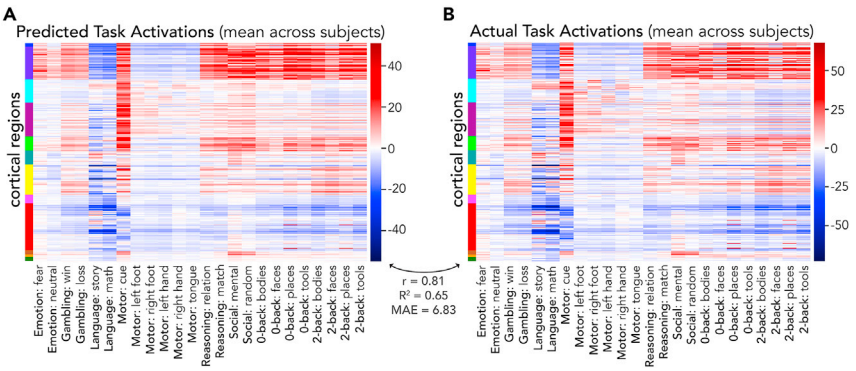


Figure 5. Activity flow mapped task activations compared to actual task activations across all nodes and tasks

(A) Predicted task activation patterns across 24 conditions and all MMP cortical regions, sorted into their CAB-NP functional network assignments (color coded per Figure 1A) (mean of N=30 subjects).

(B) The actual task activation patterns (as in Figure 1C). The predicted activations exhibited high similarity to the actual activations (r = 0.81, R^2 = 0.65, MAE = 6.83; see expected outcomes for more on measuring accuracy).

Optional part 4: Visualize results

Timing: 1–5 min

Once the activity flow mapping procedure is complete, there are a variety of approaches to visualizing results. Here we provide instructions to produce our recommended visualizations, focusing on comparing predicted task activations to actual task activations. Please note that depending on the research question and data, researchers may find the need to modify these figures, or utilize another figure type altogether (see expected outcomes for an overview of the current literature implementing activity flow mapping).

8. Visualize the predicted and actual activation patterns across all task conditions (Figure 5).

```
# Visualize predicted and actual task activation patterns across all nodes and tasks

# HCP S1200: names of the 24 conditions (x-axis tick strings)
task_names = ["EMOTION:fear", "EMOTION:neut", "GAMBLING:win", "GAMBLING:loss",
"LANGUAGE:story", "LANGUAGE:math", "MOTOR:cue", "MOTOR:lf", "MOTOR:rf", "MOTOR:lh",
"MOTOR:rh", "MOTOR:t", "REASONING:rel", "REASONING:match", "SOCIAL:mental",
"SOCIAL:rnd", "WM 0bk:body", "WM 0bk:faces", "WM 0bk:places", "WM 0bk:tools", "WM 2bk:body", "WM
2bk:faces", "WM 2bk:places", "WM 2bk:tools"]

# Visualize activity-flow-predicted activation patterns across all 24 HCP S1200 conditions
plt.figure(figsize=[7, 5])
ax = sns.heatmap(
    np.mean(predicted_activations_array, axis=2)[netorder, :],
    center=0,
    cmap="seismic",
    cbar=True,
    yticklabels=100,
    xticklabels=task_names,
)
ax.figure.suptitle("Predicted Task Activations (mean across subjects)")
ax.set(ylabel="cortical regions")
plt.show()

# Visualize actual (empirical) activation patterns across all 24 HCP S1200 conditions
plt.figure(figsize=[7,5])
```

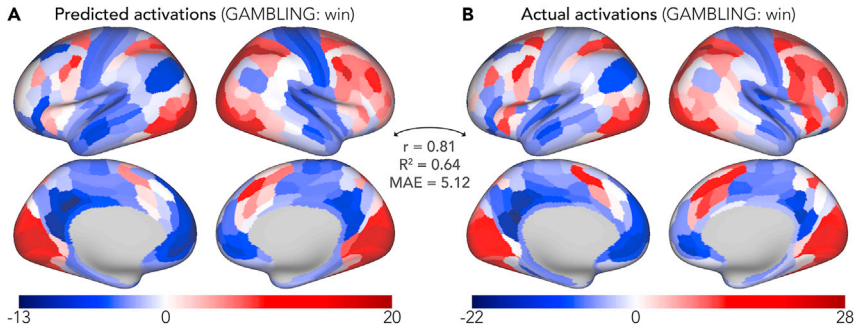


Figure 6. Activity flow mapped task activations compared to actual task activations across all nodes and one task

(A) Activity-flow-predicted task activation patterns for one task condition (the win condition of the gambling task), across all MMP cortical regions (mean of N = 30 subjects).

(B) The actual task activation patterns for the gambling (win) condition, across all MMP cortical regions (mean of N = 30 subjects). The predicted activations exhibited high similarity to the actual activations (r = 0.81, R^2 = 0.64, MAE =5.12; see expected outcomes for more on measuring accuracy with the 'nodewise_compthenavg' flag).

```
ax = sns.heatmap(
    np.mean(actual_activations_array, axis=2)[netorder, :],
    center=0,
    cmap= "seismic",
    cbar=True,
    yticklabels=100,
    xticklabels=task_names,
)
ax.figure.suptitle("Actual Task Activations (mean across subjects)")
ax.set(ylabel="cortical regions")
plt.show()
```

9. Visualize the predicted and actual activation patterns for a select task condition, mapped onto a cortical brain schematic (Figure 6). This uses wbplot (https://github.com/jbburt/wbplot), a Python module (see materials and equipment) built upon Connectome Workbench commands (https://www.humanconnectome.org/software/connectome-workbench) to generate brain schematics in-line, such that wbplot can be called within an IDE or Jupyter Notebook. Functionality is limited and use of the workbench GUI is recommended for full visualization options and HCP integration.

   Note: A pixdim warning message may appear; this can be ignored.

```
# Visualize predicted and actual task activation patterns across all nodes and 1 task

cond_ix = 2 # see task_names for corresponding task
n_parcels = 360 # this should match n_nodes used upstream in code. 360 across-cortex MMP regions; 180 per hemisphere (Glasser et al., 2016)

# Activity-flow-predicted activations for one task condition
inputdata = np.mean(predicted_activations_array, axis=2)[:, cond_ix]

# Flip hemispheres, since CAB-NP is ordered left-to-right, while wbplot uses right-to-left
inputdata_flipped = np.zeros(np.shape(inputdata))
inputdata_flipped[0:int(n_parcels / 2)] = inputdata[
    int(n_parcels / 2):int(n_parcels)
]
```

```
inputdata_flipped[int(n_parcels / 2):int(n_parcels)] = inputdata[
    0:int(n_parcels / 2)
]
file_out = fig_directory + "out.png"

colormap = "seismic" # consider setting to all "reds" if no negative values
pscalar(file_out=file_out, pscalars=inputdata_flipped, cmap=colormap, transparent=True)
img = mpimg.imread(file_out)
plt.figure()
plt.axis("off")
plt.title("Predicted activations (" + task_names[cond_ix] + ")")
plt.imshow(img)

# Actual activations to one task condition
inputdata = np.mean(actual_activations_array, axis=2)[:, cond_ix]

# Flip hemispheres, since CAB-NP is ordered left-to-right, while wbplot uses right-to-left
inputdata_flipped = np.zeros(np.shape(inputdata))
inputdata_flipped[0:int(n_parcels / 2)] = inputdata[
    int(n_parcels / 2):int(n_parcels)
]
inputdata_flipped[int(n_parcels / 2):int(n_parcels)] = inputdata[
    0:int(n_parcels / 2)
]
file_out = fig_directory + "out.png"

colormap = "seismic" # consider setting to all "reds" if no negative values
pscalar(file_out=file_out, pscalars=inputdata_flipped, cmap=colormap, transparent=True)
img = mpimg.imread(file_out)
plt.figure()
plt.axis("off")
plt.title("Actual activations (" + task_names[cond_ix] + ")")
plt.imshow(img)
```

Expected outcomes

Prediction accuracy: model comparison report

The core tool of the Actflow Toolbox, actflowtest (see part 3 in step-by-step method details), by default returns a report of the prediction accuracy indices that assess how similar the activity-flow-predicted activations are to the actual activations (visualized in Figures 5 and 6). These accuracy indices include: (1) Pearson r (with associated t-statistic and p-value), computed with numpy.corrcoef. (2) Variance explained via coefficient of determination (R^2), computed with sklearn.metrics.r2_score. (3) Mean absolute error (MAE) (see Cole et al., 2016), computed with NumPy functions corresponding to the following formula (n = number of subjects):

$$MAE = \frac{\Sigma_{i=1}^{n} |predicted - actual|}{n}$$

A sample report from the Actflow Toolbox example dataset (see key resources table), which assessed n = 30 HCP S1200 subjects and utilized the resting-state FC estimated via combinedFC, printed as follows:

```
===Comparing prediction accuracies between models (similarity between predicted and actual
brain activation patterns)===

==Comparisons between predicted and actual activation patterns, across all conditions and no-
des:==
-Compare-then-average (calculating prediction accuracies before cross-subject averaging):
Each comparison based on 24 conditions across 360 nodes, p-values based on 30 subjects (cross-
subject variance in comparisons)
```

```
Mean Pearson r = 0.81, t-value vs. 0: 64.44, p-value vs. 0: 7.31e-33

Mean % variance explained (R^2 score, coeff. of determination) = 0.65

Mean MAE (mean absolute error) = 6.83

Note: Pearson r and Pearson r^2 are scale-invariant, while R^2 and MAE are not. R^2 units: percentage of the to-be-predicted data's unscaled variance, ranging from negative infinity (because prediction errors can be arbitrarily large) to positive 1. See https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html for more info.
```

By default, these assessments are based on an optional input variable, <comparison_type>, in the function model_compare.py (which is called within actflowtest), which has five possible options specified as strings: **(1) comparison_type = "fullcompare_compthenavg":** This is the default comparison method, where predicted and actual activations are compared across all conditions and nodes simultaneously. This is accomplished by collapsing the data across the condition and node dimensions with numpy.reshape. This process treats the variance between conditions and nodes equally, but independently per subject. The results are summarized (as in the provided sample report) with averages of the cross-subject accuracy indices. **(2) comparison_type = "conditionwise_compthenavg":** This is run separately for (i.e., iterated over) each subject and node, and comparisons are made between the predicted and actual activations across conditions. This can be thought of as characterizing the accuracy of a node's response profile, or a profile of responses across all task conditions. The results are summarized by averaging over both the node and subject dimensions. **(3) comparison_type = "conditionwise_avgthencomp":** Here, predicted and actual activation data are averaged initially across subjects (sometimes called a fixed-effects analysis). Then, the comparisons are run separately for each node, comparing response profiles as in conditionwise_compthenavg. This will boost the signal-to-noise ratio of the activation data, but will reduce the ability to assess the spread of accuracy scores across subjects (which is possible with conditionwise_compthenavg). The results are summarized by averaging over the node dimension. **(4) comparison_type = "nodewise_compthenavg":** This is run separately for each subject and condition, and comparisons are made between the predicted and actual activations across nodes, or the whole-brain (or, whole-cortex, depending on which nodes are assessed with the Actflow Toolbox) activation patterns. These vectors can be thought of as a response profile across the brain to a given condition. The results are summarized by averaging over both the condition and subject dimensions. **(5) comparison_type = "nodewise_avgthencomp":** Here, predicted and actual activations are averaged initially across subjects. Then, comparisons are run separately for each condition, comparing the whole-brain response profiles to each condition. The results are summarized by averaging over the condition dimension.

If a specific comparison type is required, users may specifically call the model_comparison function: see the quantification and statistical analysis section for an example. If all comparison types are of interest, users may turn on another optional input variable in the actflowtest.py function, <full_report>, which is by default turned off, by using the following modification to the actflowtest code block in Part 3 (see Figure 6 for visualization of condition 3):

```
# Return full model comparison report
actflow_output = actflow.actflowcomp.actflowtest(
  activity_data, fc_arr, full_report=True
)
```

```
==Condition-wise comparisons between predicted and actual activation patterns (calculated
for each node separately):==
-Compare-then-average (calculating prediction accuracies before cross-subject averaging):
Each correlation based on N conditions: 24, p-values based on N subjects (cross-subject vari-
ance in correlations): 30

Mean Pearson r = 0.83, t-value vs. 0: 62.45, p-value vs. 0: 1.8026564213102585e-32

Mean % variance explained (R^2 score, coeff. of determination) = 0.35

Mean MAE (mean absolute error) = 6.83

Note: Pearson r and Pearson r^2 are scale-invariant, while R^2 and MAE are not. R^2 units: per-
centage of the to-be-predicted datas unscaled variance, ranging from negative infinity
(because prediction errors can be arbitrarily large) to positive 1. See https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.r2_score.html for more info.

==Node-wise (spatial) correlations between predicted and actual activation patterns (calcu-
lated for each condition separately):==
-Compare-then-average (calculating prediction accuracies before cross-subject averaging):
Each correlation based on N nodes: 360, p-values based on N subjects (cross-subject variance in
correlations): 30
Cross-condition mean r=0.78, t-value vs. 0: 55.40, p-value vs. 0: 5.66e-31
By task condition:
Condition 1: r=0.70, t-value vs. 0: 24.83, p-value vs. 0: 4.351151202867745e-21
Condition 2: r=0.66, t-value vs. 0: 22.49, p-value vs. 0: 6.664353565074144e-20
Condition 3: r=0.81, t-value vs. 0: 49.47, p-value vs. 0: 1.4548795509484526e-29
Condition 4: r=0.81, t-value vs. 0: 44.23, p-value vs. 0: 3.61e-28
Condition 5: r=0.71, t-value vs. 0: 28.35, p-value vs. 0: 1.08-22
Condition 6: r=0.72, t-value vs. 0: 30.75, p-value vs. 0: 1.10e-23
Condition 7: r=0.79, t-value vs. 0: 39.28, p-value vs. 0: 1.07e-26
Condition 8: r=0.69, t-value vs. 0: 24.92, p-value vs. 0: 3.94e-21
Condition 9: r=0.68, t-value vs. 0: 27.53, p-value vs. 0: 2.45e-22
Condition 10: r=0.68, t-value vs. 0: 28.77, p-value vs. 0: 7.10e-23
Condition 11: r=0.67, t-value vs. 0: 32.17, p-value vs. 0: 3.07e-24
Condition 12: r=0.68, t-value vs. 0: 29.39, p-value vs. 0: 3.90e-23
Condition 13: r=0.83, t-value vs. 0: 39.87, p-value vs. 0: 6.95e-27
Condition 14: r=0.84, t-value vs. 0: 44.50, p-value vs. 0: 3.02e-28
Condition 15: r=0.83, t-value vs. 0: 46.05, p-value vs. 0: 1.136e-28
Condition 16: r=0.83, t-value vs. 0: 56.14, p-value vs. 0: 3.86e-31
Condition 17: r=0.82, t-value vs. 0: 40.62, p-value vs. 0: 4.09e-27
Condition 18: r=0.78, t-value vs. 0: 38.98, p-value vs. 0: 1.32e-26
Condition 19: r=0.83, t-value vs. 0: 50.51, p-value vs. 0: 8.01e-30
Condition 20: r=0.83, t-value vs. 0: 42.43, p-value vs. 0: 1.183e-27
Condition 21: r=0.83, t-value vs. 0: 51.47, p-value vs. 0: 4.68e-30
Condition 22: r=0.78, t-value vs. 0: 41.73, p-value vs. 0: 1.90e-27
Condition 23: r=0.81, t-value vs. 0: 49.27, p-value vs. 0: 1.64e-29
Condition 24: r=0.82, t-value vs. 0: 39.27, p-value vs. 0: 1.071e-26
```

Expectations and considerations across research questions

As of the time this protocol was prepared, the Actflow Toolbox (or its precursors) has been successfully applied to multiple datasets and task paradigms related to a variety of cognitive domains. These studies contain in-depth considerations for how activity flow mapping can be applied to various research questions, as well as expectations regarding prediction accuracy and related statistics. We have listed these studies in Table 1 with study specifications highlighted, to operate as a reference guide for expected outcomes of specific research questions (note that future work with activity flow mapping need not be limited by the specifications in Table 1):

> ***Note:*** Unless stated otherwise, these studies in Table 1 utilized fMRI data and resting-state connectivity estimates in the activity flow mapping procedure.

Table 1. Reference guide for expected outcomes of activity flow mapping applied to specific research questions.

Citation	Sample(s)	Task paradigm(s)	Connectivity estimation method(s)	Summary of research and outcomes
Cole et al. (2016)	100 healthy adults (HCP 500 subjects release) and simulated data.	Seven cognitive domains: emotion, reward learning, language, motor, relational reasoning, social cognition, and working memory.	Resting-state FC was assessed with Pearson correlation, multiple linear regression, and principal components regression (for vertex-wise analyses).	Introduced activity flow mapping. High prediction accuracy was exhibited by activity flow mapped activations, establishing the utility of this procedure. The relationship between parameters underlying the simulated fMRI data and prediction accuracy was also assessed, helping to validate activity flow mapping for use with fMRI.
Ito et al. (2017)	32 healthy adults.	Rapid instructed task learning paradigm.	Resting-state FC was estimated with multiple linear regression at the brain region level, and with principal components regression at the vertex level.	Introduced information transfer mapping, which utilizes the activity flow principle to assess decodability in the representational geometry of predicted versus actual activation patterns. See key resources table for source code information.
Mill et al. (2020)	101 elderly participants (Adult Children study, Knight Alzheimer's Disease Research Center, Washington University in St. Louis).	The Stroop task and a semantic animacy task.	Resting-state FC was estimated with principal components regression, with a nested cross-validation scheme to identify the optimal number of principal components for repeat reliability of the FC estimates.	Assessed healthy versus at-risk (for Alzheimer's Disease) aging populations with a cross-validation approach, as well as relationships between activity flow mapping results and individual differences in behavior.
Hearne et al. (2021)	36 schizophrenia cohort versus 96 healthy controls.	Spatial working memory task.	Intrinsic FC (including other tasks and rest) was estimated with principal components regression.	Activity flow mapping was used to identify the activity flow abnormalities likely driving the observed abnormal spatial working memory brain activations and associated abnormal behavior in patients. This study also developed a simulated intervention for treating schizophrenia.
Keane et al. (2021)	20 healthy adults.	Visual shape completion task.	Resting-state FC was estimated with multiple regression.	Assessed task activations with a decoding approach plus assessment of functional network contributions.
Schultz et al. (2021)	100 healthy adults.	Rapid instructed task learning paradigm.	Resting-state FC was estimated with Pearson correlation.	Adapted the activity flow mapping procedure to the connectivity fingerprint framework proposed by Passingham et al. (2002) (updated by Mars et al., 2018) to predict task rule representations. This analysis was applied at the parcel level (based on vertex-level multivariate pattern analysis estimates) and additionally assessed functional network differences.

(*Continued on next page*)

Table 1. *Continued*

Citation	Sample(s)	Task paradigm(s)	Connectivity estimation method(s)	Summary of research and outcomes
Mill et al. (2021)	32 healthy adults.	A sensory two-alternative forced choice task.	Resting-state FC was estimated with multivariate autoregression.	This study utilized dense-array EEG data. Resting-state FC predicted future brain activity and motor activations via dynamic activity flow mapping. Further, simulated lesions were implemented to assess functional network contributions to response information flow.
Ito et al. (2021)	100 healthy adults.	Rapid instructed task learning paradigm.	Resting-state FC was estimated with principal components regression at the vertex level.	Decoded task context and stimuli. This study implemented a task-performing, empirically-derived neural network (which was based on activity flow mapping principles) that modeled conjunctive representations to investigate how sensory and task-rule information is integrated in conjunction hubs.
Cole et al., 2021	352 healthy adults (subset of HCP S1200 dataset) with split-half validation.	The same task conditions described in Figure 1 were assessed in this study.	Both task-state and resting-state FC were estimated with Pearson correlation, principal components regression, and multiple linear regression.	This study assessed the contribution of task-state FC (relative to resting-state FC) to shaping task-evoked activity flows underlying task-evoked activations.
Sanchez-Romero et al. (2021)	176 healthy adults (subset of HCP S1200 dataset) and simulated data.	Seven cognitive domains: emotion, reward learning, language, motor, relational reasoning, social cognition, and working memory.	Multiple FC estimation methods, including a directed FC (effective connectivity) method.	Simulated and empirical neuroimaging data were used to compare multiple FC estimation methods.
McCormick et al. (2021)	352 healthy adults (subset of HCP S1200 dataset)	The same task conditions described in Figure 1 were assessed in this study.	A latent factor across multiple FC states termed latent FC; each state estimated with Pearson r.	Assessed whether a latent factor across multiple FC states improves prediction accuracy of activity flow mapping, as well as prediction of a general intelligence score, *g*.
Yan et al. (2021a)	100 unrelated healthy adults from the HCP.	The task paradigms in Figure 1, focusing on the relational reasoning and n-back working memory tasks.	Structural connectivity derived from diffusion tensor imaging.	Structural connectivity was utilized in the activity flow mapping procedure.
Yan et al. (2021b)	80 healthy adults (subset of University of California Los Angeles Consortium for Neuropsychiatric Phenomics LA5c study).	The paired associate memory task.	Connectivity estimates included resting-state FC estimated via multiple linear regression, and structural connectivity derived from diffusion tensor imaging.	The activity flow mapping procedure plus information transfer mapping was applied to assess functional network contributions to episodic memory processes.

QUANTIFICATION AND STATISTICAL ANALYSIS

Further model comparisons

In addition to the model comparison options contained in actflowtest and the various accuracy indices and associated statistics that are reported (see expected outcomes), the Actflow Toolbox contains specific functions that can be used to further compare model variants. Below we provide three example model comparison questions, corresponding code, and sample results:

1. How does multiple regression based rest FC versus Pearson correlation based rest FC impact model performance?

```
# How does FC estimated with multiple regression vs correlation impact model performance?

# Estimate rest FC with multiple regression
fc_arr_mult_reg = np.zeros((n_nodes, n_nodes, n_subjs))
for subj_ix in range(n_subjs):
    fc_arr_mult_reg[:, :, subj_ix] = fc.multregconn(rest_data[:, :, subj_ix])

# Estimate rest FC with Pearson correlation
fc_arr_corr = np.zeros((n_nodes, n_nodes, n_subjs))
for subj_ix in range(n_subjs):
    fc_arr_corr[:, :, subj_ix] = fc.corrcoefconn(rest_data[:, :, subj_ix])

# Model 1: Activity flow mapping with rest FC via multiple regression
actflow_output_mult_reg = actflow.actflowcomp.actflowtest(
    activity_data, fc_arr_mult_reg
)

# Model 2: Activity flow mapping with rest FC via Pearson correlation
actflow_output_corr = actflow.actflowcomp.actflowtest(activity_data, fc_arr_corr)

# Compare the two models, 10 subjects only
model_compare_output = actflow.model_compare(
    target_actvect=activity_data[:, :, :10],
    model1_actvect=actflow_output_mult_reg["actPredVector_bytask_bysubj"][:, :, :10],
    model2_actvect=actflow_output_corr["actPredVector_bytask_bysubj"][:, :, :10],
    full_report=False,
    print_report=True,
)
```

```
===Comparing prediction accuracies between models (similarity between predicted and actual brain activation patterns)===

==Comparisons between predicted and actual activation patterns, across all conditions and nodes:==
-Compare-then-average (calculating prediction accuracies before cross-subject averaging): Each comparison based on 24 conditions across 360 nodes, p-values based on 10 subjects (cross-subject variance in comparisons)

Model1 mean Pearson r=0.78
Model2 mean Pearson r=0.59
R-value difference = 0.20
Model1 vs. Model2 T-value: 12.81, p-value: 4.41e-07

Model1 mean % predicted variance explained R^2=0.58
Model2 mean % predicted variance explained R^2=-595.00
R^2 difference = 595.57

Model1 mean MAE = 7.08
Model2 mean MAE = 277.10
Model1 vs. Model2 mean MAE difference = -270.02

Note: Pearson r and Pearson r^2 are scale-invariant, while R^2 and MAE are not. R^2 units: percentage of the to-be-predicted data's unscaled variance, ranging from negative infinity (because prediction errors can be arbitrarily large) to positive 1. See https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html for more info.
```

a. The following are additional parameter options available when implementing corrcoefconn, but are set to defaults in the corresponding code block:

i. **target_ts**: This is an optional parameter set to None by default. This can be used when the user wants to estimate connectivity for only one target node (with all other source nodes). This should contain the target node's time series, or, a one dimensional vector of activity across time points. Connectivity will be compared to each source node's time series in the <activity_matrix> input (e.g., <rest_data> in the corresponding code block), and a one dimensional connectivity vector will be returned.

b. The following are additional parameter options available when implementing model_compare, some of which are set to defaults in the corresponding code block:
   i. **model2_actvect**: This is an optional parameter set to None by default. This can be used when one wants to compare activity flow mapping results across two models. This is demonstrated here by comparing a model where FC was estimated with multiple regression versus a model where FC was estimated with Pearson correlation. This should contain the array of predicted activation values outputted by actflowtest. Note it does not matter which results are put in <model1_actvect> and which are put into <model2_actvect> (i.e., the order does not matter).
   ii. **full_report**: This is a Boolean set to False by default. If True, prediction accuracy will be assessed across all dimensions. See expected outcomes for a detailed explanation of these statistics.
   iii. **print_report**: This is a Boolean set to True by default, which will print the full model comparison report. If False, printing results will be suppressed.
   iv. **print_by_condition**: This is a Boolean set to True by default and only works when <print_report=True>. If True, the printed report will include results for each condition separately.
   v. **comparison_type**: This is a string specifying how to compute prediction accuracy. The options are extensively described in the expected outcomes section, but in brief the options are: 'fullcompare_compthenavg', 'conditionwise_compthenavg', 'conditionwise_avgthencomp', 'nodewise_compthenavg', and 'nodewise_avgthencomp'.
   vi. **avgthencomp_fixedeffects**: This is a Boolean set to False by default. If True, prediction accuracy will be assessed after averaging across participants, which is sometimes referred to as a "fixed effects" analysis. We recommend analyses with participants as random effects (i.e., effects for each participant), and thus recommend keeping this False (but have included the option for specific use cases).
   vii. **mean_absolute_error**: This is a Boolean set to True by default and determines whether mean absolute error is included as one of the prediction accuracy indices (see expected outcomes for the formula and further details).

2. How does multiple regression based rest FC versus Pearson correlation based rest FC impact model performance, per node (i.e., condition-wise)? (Figure 7A)

```
# Continued from the code block above; assessing all MMP regions
# Assessing generalization across regions (i.e., assessing the cross-condition response profiles of each region)
model_compare_output = actflow.model_compare(
    target_actvect=activity_data[:, :, :10],
    model1_actvect=actflow_output_mult_reg["actPredVector_bytask_bysubj"][:, :, :10],
    model2_actvect=actflow_output_corr["actPredVector_bytask_bysubj"][:, :, :10],
    comparison_type="conditionwise_compthenavg",
    full_report=False,
    print_report=True,
)
```

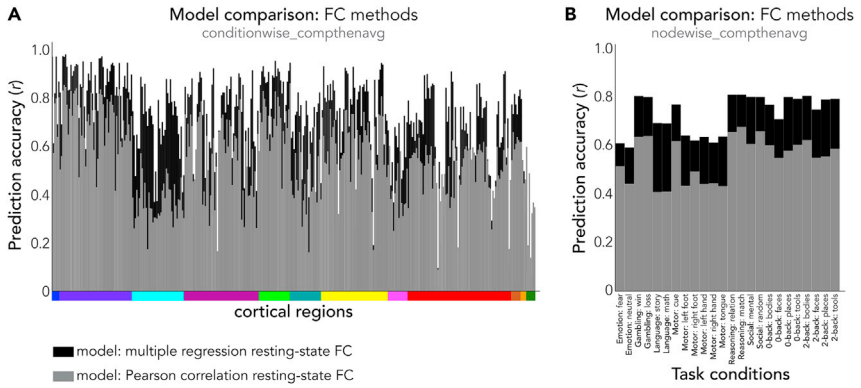


Figure 7. Comparing activity flow model performance with different resting-state FC estimation methods

(A) Prediction accuracy (mean across subjects) of activity flow mappings based on multiple linear regression (black) versus Pearson correlation (grey) estimated rest FC. This corresponds to the model comparison flag 'conditionwise_compthenavg' (see expected outcomes), where response profiles (task activations across 24 HCP conditions) are compared (predicted-to-actual) per node (MMP regions, sorted along the x-axis per functional network assignment in Figure 1A). Across the majority of nodes, the response profile prediction accuracies were higher for the multiple regression model.

(B) Prediction accuracy as in panel A, but for the 'nodewise_compthenavg' model comparison type (see expected outcomes). Per task condition, the cross-node activation patterns are compared (predicted-to-actual). As in the condition-wise model comparison in (A), the multiple regression based rest FC model performed best.

```
===Comparing prediction accuracies between models (similarity between predicted and actual brain activation patterns)===

==Condition-wise comparisons between predicted and actual activation patterns (calculated for each node separately):==
-Compare-then-average (calculating prediction accuracies before cross-subject averaging): Each correlation based on N conditions: 24, p-values based on N subjects (cross-subject variance in correlations): 10

Model1 mean Pearson r=0.81
Model2 mean Pearson r=0.62
R-value difference = 0.19
Model1 vs. Model2 T-value: 12.46, p-value: 5.572065132326254e-07

Model1 mean % predicted variance explained R^2=0.09
Model2 mean % predicted variance explained R^2=-1450.42
R^2 difference = 1450.51

Model1 mean MAE = 7.08
Model2 mean MAE = 277.10
Model1 vs. Model2 mean MAE difference = -270.02

Note: Pearson r and Pearson r^2 are scale-invariant, while R^2 and MAE are not. R^2 units: percentage of the to-be-predicted data's unscaled variance, ranging from negative infinity (because prediction errors can be arbitrarily large) to positive 1. See https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html for more info.
```

3. How does multiple regression based rest FC versus Pearson correlation based rest FC impact model performance, per task condition? (Figure 7B)

```
# Continued from the code block above; assessing the first 4 HCP conditions
# This comparison assesses generalization across task conditions
str_here = "actPredVector_bytask_bysubj"
```

```
model_compare_output = actflow.model_compare(
    target_actvect=activity_data[:, :4, :][:, :, :10],
    model1_actvect=actflow_output_mult_reg[str_here][:, :4, :][:, :, :10],
    model2_actvect=actflow_output_corr[str_here][:, :4, :][:, :, :10],
    comparison_type="nodewise_compthenavg",
    full_report=False,
    print_report=True,
)
```

```
===Comparing prediction accuracies between models (similarity between predicted and actual brain activation patterns)===

==Node-wise (spatial) correlations between predicted and actual activation patterns (calculated for each condition separately):==
-Compare-then-average (calculating prediction accuracies before cross-subject averaging):
Each correlation based on N nodes: 360, p-values based on N subjects (cross-subject variance in correlations): 10
Model1 mean Pearson r=0.74
Model2 mean Pearson r=0.56
R-value difference = 0.18
Model1 vs. Model2 T-value: 21.83, p-value: 4.19e-09
By task condition:
Condition 1: Model 1 r=0.62, Model 2 r=0.52, Model 1 vs. 2 R-value difference =0.10, t-value Model1 vs. Model2: 6.48, p-value vs. 0: 0.0001
Condition 2: Model 1 r=0.62, Model 2 r=0.49, Model 1 vs. 2 R-value difference =0.13, t-value Model1 vs. Model2: 4.50, p-value vs. 0: 0.002
Condition 3: Model 1 r=0.81, Model 2 r=0.66, Model 1 vs. 2 R-value difference =0.15, t-value Model1 vs. Model2: 10.75, p-value vs. 0: 1.95e-06
Condition 4: Model 1 r=0.81, Model 2 r=0.65, Model 1 vs. 2 R-value difference =0.16, t-value Model1 vs. Model2: 11.69, p-value vs. 0: 9.64e-07
```

Non-standard hypothesis testing considerations

Once predicted activations are mapped, researchers may require hypothesis testing procedures beyond prediction accuracy. Below we list recommendations for non-standard use cases ("standard" testing referring to student's t-testing, ANOVA, regression, and more).

Nonparametric data: For research questions where it is unknown if the underlying data follow a normal distribution and/or involving multiple comparisons, we recommend the max-statistic (typically, max-T) nonparametric permutation approach described in Blair and Karniski (1993) and Nichols and Holmes (2002). Max-T testing derives a 95% confidence interval from the maximum T distribution as a critical threshold with an associated p-value equal to one divided by the number of permutations used. This corrects for multiple comparisons (family-wise error) while providing nonparametric p-values. A max-T function is provided as part of the Actflow Toolbox (https://github.com/ColeLab/ActflowToolbox/blob/master/tools/max_t.py).

Assessing decoding accuracy: If using activity flow mapping in a decoding paradigm (as in Ito et al., 2017; Keane et al., 2021; Ito et al., 2021), we recommend following the best practices proposed by Varoquaux et al. (2017). In brief, k-fold cross-validation with at least 20% of the data left out is the most robust approach. If hyperparameters are needed, they should be tuned via nested cross-validation.

LIMITATIONS

The success of activity flow mapping depends on the following factors: (1) The quality of the data, which can be impacted during the initial acquisition and/or during data processing. We have

provided best practice recommendations throughout this protocol for fMRI data. Specifically, we recommend the minimal preprocessing pipeline provided by Glasser et al. (2013), nuisance regression recommendations provided by Ciric et al. (2017), FC estimation with combinedFC (Sanchez-Romero and Cole, 2021), and task regression (when using task-state FC) recommendations provided by Cole et al., 2019. See the key resources table for open source analysis tools for most of the above recommendations. (2) In Cole et al., 2021, we found evidence that prediction accuracy increases with the amount of data utilized for FC estimation, or, how many time points were sampled per state (resting-state overall time points for rest FC; time points per task condition for task-state FC). This effect tended to plateau at around 20 min of data, likely because at the upper bounds of the prediction accuracy index (approaching $R^2 = 1$), differences between compared models shrink ("hit the ceiling").

The inferences based on activity flow mapping results are limited in the following ways: (1) A nonlinear relationship between neural inputs and outputs is assumed by many neuroscientists. However modeling this nonlinearity remains a challenge, often because parameters are introduced that are difficult to estimate empirically and that limit the interpretability of a given model's results. While future work may adapt activity flow mapping to incorporate nonlinearities, the present approach is based on linear estimates of neural interaction. As a preview of future non-linear implementations of activity flow mapping, a non-linearity is used by Ito et al. (2021), demonstrating the importance of non-linearities in implementing rule-guided behavior. (2) The advantages and disadvantages of any given neuroimaging method also apply to activity flow mapping results. For example, fMRI BOLD is a coarse proxy of spiking (and/or local field potential) activity; EEG suffers from the source localization inverse problem. Thus, any inferential limits for the actual task activations also apply to the activity-flow-predicted task activations. This is an important consideration for studies that seek to assess the connectivity patterns relevant to a given region's activity flow predicted activations (e.g., functional network contributions). Disentangling feedforward versus feedback signals remains a challenge (particularly with fMRI), thus, inferences regarding the directionality of connectivity patterns should be made with care.

TROUBLESHOOTING

Problem 1

When estimating FC (Part 2) with the built-in functions multregconn or combinedFC (which was the method outlined in the step-by-step method details section, part 2), the following error message may appear:

```
Exception: More nodes (regressors) than timepoints! Use regularized regression
```

Potential solution

This is because the input data (e.g., <rest_data>) contains more nodes (the first dimension) than time points (the second dimension). Firstly, check that the dimensions of the input variable are as expected (e.g., print(rest_data.shape)). If the data dimensions are as expected, then the error was caused by having too few time points. With the application of multiple linear regression to estimate FC, there needs to be a sufficient number of observations for the target node (or responses) relative to the number of source nodes (regressors or predictors). If more data cannot be added, we recommend a regularized regression approach, such as pc_multregconn provided in the Actflow Toolbox.

Problem 2

An error message appears in the Python coding environment indicating a module cannot be successfully imported (e.g., ModuleNotFoundError) (Part 1). This is likely due to a module being used by the toolbox or one of its dependencies not being installed, or, being installed improperly.

Potential solutions

- Check that the module name and Python syntax are correct.
- Attempt to reinstall the module with PIP (see materials and equipment) and run the Python code again (may require opening a new tab if running Python from the command line).
- It is possible that the Python build environment on a given machine does not include some of the modules listed in this protocol by default (e.g., setuptools and related packages). In this case, we recommend installing and using a Python virtual environment management tool such as pyenv (see here: https://github.com/pyenv/pyenv) or pipenv (see here: https://packaging.Python.org/guides/tool-recommendations/).
- It is possible that a module (particularly those downloaded from a website or imported via Git) may be sourced to a specific directory (e.g., Actflow Toolbox will work this way). Make sure the Python code is either run from the same directory, or points to the directory during the import. For example:

```
# Activity Flow Toolbox was installed via Git under the ~/research_projects directory:
parent_dir = "/Users/myname/research_projects/"
sys.path.insert(0, parent_dir)
import ActflowToolbox as actflow
```

Problem 3

Select data may raise the following error with the principal components regression (https://github.com/ColeLab/ActflowToolbox/blob/master/connectivity_estimation/pc_multregconn.py) method for estimating FC (Part 2):

```
numpy.linalg.linalg.LinAlgError: SVD did not converge
```

Potential solution

This is likely due to the precision limits of sklearn.linear_model.LinearRegression in certain computational environments. We recommend creating a duplicate version of the pc_multregconn.py script (e.g., duplicating and renaming the file with the name pc_multgregconn_alt.py), and trying this alternative package in the appropriate lines, which are listed in the following code block:

```
import statsmodels.api as sm

# Note: lines of the original code are skipped for brevity; see pc_multregconn.py

# The multiple regression step utilized in PC-regression; alternative code
regr = sm.OLS(y, reduced_mat) # Lines: 60, 89, 115, 135
reg = regr.fit() # Lines: 62, 90, 115, 137

# The FC vector utilized in PC-regression; alternative code
betasPCR = pca.inverse_transform(reg.params) # Lines: 92, 117
```

Problem 4

When performing the optional step to avoid potential circularity due to spatial smoothness by excluding source vertices that are 10 mm from the target node border (see before you begin for conceptual details and Part 2 for code), an error message may appear regarding the *dlabel.nii file, such as:

```
FileNotFoundError: No such file or no access:
'~/ActflowToolbox/dependencies/ColeAnticevicNetPartition/
CortexSubcortex_ColeAnticevic_NetPartition_wSubcorGSR_parcels_LR.dlabel.nii'
```

Potential solution

This error is related to the computational environment having trouble accessing the Actflow Toolbox's dependencies. The quickest solution is to first download the *dlabel.nii file called CortexSubcortex_ColeAnticevic_NetPartition_wSubcorGSR_parcels_LR.dlabel.nii to a specific directory path, for example: /my/research/project/directory/CortexSubcortex_ColeAnticevic_NetPartition_wSubcorGSR_parcels_LR.dlabel.nii. This file can be downloaded from here: https://github.com/ColeLab/ColeAnticevicNetPartition/blob/master/CortexSubcortex_ColeAnticevic_NetPartition_wSubcorGSR_parcels_LR.dlabel.nii. Second, to use the optional parameter of <dlabelfile> in either calcconn_parcelwise_noncircular or calccactivity_parcelwise_noncircular, in the code block from Part 2 as follows:

```
# Exclude source vertices 10 mm from target nodes to avoid circularity
# Using preloaded vertex-wise data: activity_data_vertex and rest_data_vertex

defaultdlabelfile =
'/my/research/project/directory/CortexSubcortex_ColeAnticevic_
NetPartition_wSubcorGSR_parcels_LR.dlabel.nii'

# Non-circular task activations
activity_data_noncirc = np.zeros((n_nodes, n_conditions, n_subjs))
for subj_ix in range(n_subjs):
    activity_data_here = activity_data_vertex[:, :, subj_ix]
    activity_data_noncirc_all = fc.calcactivity_parcelwise_noncircular(
        activity_data_here,
        dlabelfile=defaultdlabelfile,
    )
    for node_ix in range(n_nodes):
        for cond_ix in range(n_conditions):
            extracted_data = activity_data_noncirc_all[node_ix, node_ix, cond_ix]
            activity_data_noncirc[node_ix, cond_ix, subj_ix] = extracted_data.copy()

# Non-circular resting-state FC with combinedFC
fc_arr_noncirc = np.zeros((n_nodes, n_nodes, n_subjs))
for subj_ix in range(n_subjs):
    rest_data_here = rest_data_vertex[:, :, subj_ix]
    # Can specify connmethod = "combinedFC" here
    fc_arr_noncirc[:, :, subj_ix] = fc.calcconn_parcelwise_noncircular(
        rest_data_here,
        dlabelfile=defaultdlabelfile,
  )
```

Problem 5

The printed output of actflowtest (Part 3) lists data dimensions not consistent with your input data. For example, you wish to perform activity flow mapping for the first target node only, and otherwise your task activation data contains 24 conditions and 30 subjects. You run the following:

```
# Activity flow mapping with resting-state FC estimated via combinedFC; attempting to run activity flow mapping for the first target node only (index 0)
actflow_output = actflow.actflowcomp.actflowtest(activity_data[0,:,:], fc_arr)
```

But you receive this printed output

```
===Comparing prediction accuracies between models (similarity between predicted and actual brain activation patterns)===

==Comparisons between predicted and actual activation patterns, across all conditions and nodes:==
-Compare-then-average (calculating prediction accuracies before cross-subject averaging):
```

```
Each comparison based on 1 conditions across 24 nodes, p-values based on 30 subjects (cross-subject variance in comparisons)

Mean Pearson r = 0.02, t-value vs. 0: 0.32, p-value vs. 0: 0.7484007039676109

Mean % variance explained (R^2 score, coeff. of determination) = -0.30

Mean MAE (mean absolute error) = 23.63
```

This incorrectly lists 1 condition across 24 nodes.

Potential solution

This is because the activity flow mapping procedure is designed to iteratively hold out target nodes' task activations and use source nodes' activations, weighted by their connectivity with the target, to predict the target's activity (see Figure 4). This procedure currently relies on the target and source sets including the same number of nodes. However, given that this procedure is iterative, the activity-flow-predicted activations that are returned were computed per target node. Thus, the output can be indexed instead to obtain the predicted activations of one target node, as follows:

```
# Activity flow mapping with resting-state FC estimated via combinedFC
actflow_output = actflow.actflowcomp.actflowtest(activity_data, fc_arr)

# Extract first target nodes' results only
predicted_activations_one_target = actflow_output["actPredVector_bytask_bysubj"][
  0, :, :
]
```

Note: When the data was handled properly, the cross-condition prediction accuracies for this first target node were actually (across-subject average): $r = 0.99$, $R^2 = 0.90$, and MAE = 6.07. This can be extracted as follows (as in Part 3, but specifying condition-wise results):

```
# Activity flow mapping with resting-state FC estimated via combinedFC
actflow_output = actflow.actflowcomp.actflowtest(
    activity_data, fc_arr, full_report=True
)

# Extract first target nodes' results only
predicted_activations_one_target = actflow_output["actPredVector_bytask_bysubj"][
    0, :, :
]

# Extract and print prediction accuracy statistics for that first target node (across-subject means)
model_compare_dict = actflow_output["model_compare_output"]
corr_conditionwise_compthenavg_bynode = model_compare_dict[
    "corr_conditionwise_compthenavg_bynode"
]
R2_conditionwise_compthenavg_bynode = model_compare_dict[
    "R2_conditionwise_compthenavg_bynode"
]
mae_conditionwise_compthenavg_bynode = model_compare_dict[
    "mae_conditionwise_compthenavg_bynode"
]
print("r = " + str(np.nanmean(corr_conditionwise_compthenavg_bynode, axis=1)[0]))
print("R-squared = " + str(np.nanmean(R2_conditionwise_compthenavg_bynode, axis=1)[0]))
print("MAE = " + str(np.nanmean(mae_conditionwise_compthenavg_bynode, axis=1)[0]))
```

```
r = 0.986819103980586
R-squared = 0.9046279146038344
MAE = 6.07483973061582
```

Note: please submit issues in GitHub following these conventions: https://docs.github.com/en/issues/tracking-your-work-with-issues/creating-an-issue. Additionally, we encourage users to join the ColeNeuroLab Users Group to stay up to date with any major updates to this toolbox (see: https://groups.google.com/forum/#!forum/coleneurolab_users).

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact: Carrisa V. Cocuzza (carrisacocuzza@gmail.com).

Materials availability

This study did not generate new unique reagents.

Data and code availability

All code is publicly available on GitHub: https://github.com/ColeLab/ActflowToolbox (https://doi.org/10.5281/zenodo.5768754).

ACKNOWLEDGEMENTS

The authors acknowledge the support of the US National Institutes of Health (NIH) under grants R01-AG055556 and R01-MH109520 and the Behavioral and Neural Sciences Graduate Program at Rutgers, The State University of New Jersey. Data were provided by the Human Connectome Project, WU-Minn Consortium (Principal Investigators: D. Van Essen and K. Ugurbil; Grant 1U54-MH091657) funded by the 16 NIH institutes and centers that support the NIH Blueprint for Neuroscience Research; and by the McDonnell Center for Systems Neuroscience at Washington University. We also thank the Office of Advanced Research Computing at Rutgers, The State University of New Jersey, for providing access to the Amarel cluster and associated research computing resources that have contributed to the results reported here. The content is solely the responsibility of the authors and does not necessarily represent the official views of any of the funding agencies.

AUTHOR CONTRIBUTIONS

Conceptualization: C.V.C. and M.W.C.; Methodology: M.W.C. and R.S.R.; Software: M.W.C., C.V.C., and R.S.R.; Validation: M.W.C., C.V.C., and R.S.R.; Formal Analysis: M.W.C., C.V.C., and R.S.R.; Investigations: M.W.C., C.V.C., and R.S.R.; Writing: C.V.C., M.W.C., and R.S.R.; Visualization: C.V.C.; Supervision: M.W.C.; Funding acquisition: M.W.C.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

Barch, D.M., Burgess, G.C., Harms, M.P., Petersen, S.E., Schlaggar, B.L., Corbetta, M., Glasser, M.F., Curtiss, S., Dixit, S., Feldt, C., et al. (2013). Function in the human connectome: task-fMRI and individual differences in behavior. NeuroImage *80*, 169–189.

Behzadi, Y., Restom, K., Liau, J., and Liu, T.T. (2007). A component based noise correction method (CompCor) for BOLD and perfusion based fMRI. NeuroImage *37*, 90–101.

Blair, R.C., and Karniski, W. (1993). An alternative method for significance testing of waveform difference potentials. Psychophysiology *30*, 518–524.

Ciric, R., Wolf, D.H., Power, J.D., Roalf, D.R., Baum, G.L., Ruparel, K., Shinohara, R.T., Elliott, M.A., Eickhoff, S.B., Davatzikos, C., et al. (2017). Benchmarking of participant-level confound regression strategies for the control of motion artifact in studies of functional connectivity. NeuroImage *154*, 174–187.

Cole, M.W., Ito, T., Bassett, D.S., and Schultz, D.H. (2016). Activity flow over resting-state networks shapes cognitive task activations. Nat. Neurosci. *19*, 1718–1726.

Cole, M.W., Ito, T., Cocuzza, C.V., and Sanchez-Romero, R. (2021). The functional relevance of task-state functional connectivity. J. Neurosci. *41*, 2684–2702.

Cole, M.W., Ito, T., Schultz, D., Mill, R., Chen, R., and Cocuzza, C.V. (2019). Task activations

produce spurious but systematic inflation of task functional connectivity estimates. NeuroImage *189*, 1–18.

Esteban, O., Markiewicz, C.J., Blair, R.W., Moodie, C.A., Isik, A.I., Erramuzpe, A., Kent, J.D., Goncalves, M., DuPre, E., Snyder, M., et al. (2019). fMRIPrep: a robust preprocessing pipeline for functional MRI. Nat. Methods *16*, 111–116.

Friston, K.J., Holmes, A.P., Poline, J.B., Grasby, P.J., Williams, S.C., Frackowiak, R.S., and Turner, R. (1995). Analysis of fMRI time-series revisited. NeuroImage *2*, 45–53.

Glasser, M.F., Coalson, T.S., Robinson, E.C., Hacker, C.D., Harwell, J., Yacoub, E., Ugurbil, K., Andersson, J., Beckmann, C.F., Jenkinson, M., et al. (2016). A multi-modal parcellation of human cerebral cortex. Nature *536*, 171–178.

Glasser, M.F., Sotiropoulos, S.N., Wilson, J.A., Coalson, T.S., Fischl, B., Andersson, J.L., Xu, J., Jbabdi, S., Webster, M., Polimeni, J.R., et al. (2013). The minimal preprocessing pipelines for the Human Connectome Project. NeuroImage *80*, 105–124.

Handwerker, D.A., Ollinger, J.M., and D'Esposito, M. (2004). Variation of BOLD hemodynamic responses across subjects and brain regions and their effects on statistical analyses. NeuroImage *21*, 1639–1651.

Hearne, L.J., Mill, R.D., Keane, B.P., Repovš, G., Anticevic, A., and Cole, M.W. (2021). Activity flow underlying abnormalities in brain activations and cognition in schizophrenia. Sci. Adv. *7*, eabf2513.

Ito, T., Kulkarni, K.R., Schultz, D.H., Mill, R.D., Chen, R.H., Solomyak, L.I., and Cole, M.W. (2017). Cognitive task information is transferred between brain regions via resting-state network topology. Nat. Commun. *8*, 1027.

Ito, T., Brincat, S.L., Siegel, M., Mill, R.D., He, B.J., Miller, E.K., Rotstein, H.G., and Cole, M.W. (2020a). Task-evoked activity quenches neural correlations and variability across cortical areas. PLoS Comput. Biol. *16*, e1007983.

Ito, T., Hearne, L., Mill, R., Cocuzza, C., and Cole, M.W. (2020b). Discovering the computational relevance of brain network organization. Trends Cogn. Sci. *24*, 25–38.

Ito, T., Yang, G.R., Laurent, P., Schultz, D.H., and Cole, M.W. (2021). Constructing neural network models from brain data reveals representational transformations underlying adaptive behavior. bioRxiv preprint. https://doi.org/10.1101/2020.12.24.424353.

Ji, J.L., Spronk, M., Kulkarni, K., Repovš, G., Anticevic, A., and Cole, M.W. (2019). Mapping the human brain's cortical-subcortical functional network organization. NeuroImage *185*, 35–57.

Keane, B.P., Barch, D.M., Mill, R.D., Silverstein, S.M., Krekelberg, B., and Cole, M.W. (2021). Brain network mechanisms of visual shape completion. NeuroImage *236*, 118069.

Lindquist, M.A., Meng Loh, J., Atlas, L.Y., and Wager, T.D. (2009). Modeling the hemodynamic response function in fMRI: efficiency, bias and mis-modeling. NeuroImage *45*, S187–S198.

Logothetis, N.K., and Wandell, B.A. (2004). Interpreting the BOLD signal. Annu. Rev. Physiol. *66*, 735–769.

Malonek, D., and Grinvald, A. (1996). Interactions between electrical activity and cortical microcirculation revealed by imaging spectroscopy: implications for functional brain mapping. Science *272*, 551–554.

Marcus, D.S., Harms, M.P., Snyder, A.Z., Jenkinson, M., Wilson, J.A., Glasser, M.F., Barch, D.M., Archie, K.A., Burgess, G.C., Ramaratnam, M., et al. (2013). Human Connectome Project informatics: quality control, database services, and data visualization. NeuroImage *80*, 202–219.

Mars, R.B., Passingham, R.E., and Jbabdi, S. (2018). Connectivity fingerprints: from areal descriptions to abstract spaces. Trends Cogn. Sci. *22*, 1026–1037.

McCormick, E.M., Arnemann, K.L., Ito, T., Hanson, S.J., and Cole, M.W. (2021). Latent functional connectivity underlying multiple brain states. bioRxiv preprint. https://doi.org/10.1101/2021.04.05.438534.

Mill, R.D., Gordon, B.A., Balota, D.A., and Cole, M.W. (2020). Predicting dysfunctional age-related task activations from resting-state network alterations. NeuroImage *221*, 117167.

Mill, R.D., Hamilton, J.L., Winfield, E.C., Lalta, N., Chen, R.H., and Cole, M.W. (2021). Causal emergence of task information from dynamic network interactions in the human brain. bioRxiv preprint. https://doi.org/10.1101/2021.01.26.428276.

Mill, R.D., Ito, T., and Cole, M.W. (2017). From connectome to cognition: the search for mechanism in human functional brain networks. Neuroimage *160*, 124–139.

Murphy, K., Birn, R.M., Handwerker, D.A., Jones, T.B., and Bandettini, P.A. (2009). The impact of global signal regression on resting state correlations: are anti-correlated networks introduced? NeuroImage *44*, 893–905.

Nichols, T.E., and Holmes, A.P. (2002). Nonparametric permutation tests for functional neuroimaging: a primer with examples. Hum. Brain Mapp. *15*, 1–25.

Passingham, R.E., Stephan, K.E., and Kötter, R. (2002). The anatomical basis of functional localization in the cortex. Nat. Rev. Neurosci. *3*, 606–616.

Power, J.D., Mitra, A., Laumann, T.O., Snyder, A.Z., Schlaggar, B.L., and Petersen, S.E. (2014). Methods to detect, characterize, and remove motion artifact in resting state fMRI. NeuroImage *84*, 320–341.

Power, J.D., Plitt, M., Gotts, S.J., Kundu, P., Voon, V., Bandettini, P.A., and Martin, A. (2018). Ridding fMRI data of motion-related influences: removal of signals with distinct spatial and physical bases in multiecho data. Proc. Natl. Acad. Sci. U S A *115*, E2105–E2114.

Sanchez-Romero, R., and Cole, M.W. (2021). Combining multiple functional connectivity methods to improve causal inferences. J. Cogn. Neurosci. *33*, 180–194.

Sanchez-Romero, R., Ito, T., Mill, R.D., Hanson, S.J., and Cole, M.W. (2021). Causally informed activity flow models provide mechanistic insight into the emergence of cognitive processes from brain network interactions. bioRxiv preprint. https://doi.org/10.1101/2021.04.16.440226.

Schultz, D.H., Ito, T., and Cole, M.W. (2021). The human brain's intrinsic network architecture is organized to represent diverse cognitive task information. bioRxiv preprint. https://doi.org/10.1101/2021.01.25.428141.

Van Essen, D.C., Smith, S.M., Barch, D.M., Behrens, T.E.J., Yacoub, E., and Ugurbil, K.; WU-Minn HCP Consortium (2013). The WU-Minn human connectome project: an overview. NeuroImage *80*, 62–79.

Van Rossum, G., and Drake, F.L. (2009). Python 3 Reference Manual (CreateSpace).

Varoquaux, G., Raamana, P.R., Engemann, D.A., Hoyos-Idrobo, A., Schwartz, Y., and Thirion, B. (2017). Assessing and tuning brain decoders: cross-validation, caveats, and guidelines. NeuroImage *145*, 166–179.

Yan, T., Liu, T., Ai, J., Shi, Z., Zhang, J., Pei, G., and Wu, J. (2021a). Task-induced activation transmitted by structural connectivity is associated with behavioral performance. Brain Struct. Funct. *226*, 1437–1452.

Yan, T., Wang, G., Wang, L., Liu, T., Li, T., Wang, L., Suo, D., Funahashi, S., Chen, D., Wang, B., et al. (2021b). Episodic memory in aspects of brain information transfer by resting-state network topology. bioRxiv preprint. https://doi.org/10.1101/2021.02.28.433300.