The Journal of Open Source Software

NeuroCAPs: A Python Package for Performing Co-Activation Patterns Analyses on Resting-State and Task-Based fMRI Data

Donisha Smith[1]

1 Department of Psychology, Florida International University

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals
Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Co-Activation Patterns (CAPs) is a dynamic functional connectivity technique that clusters similar spatial distributions of brain activity. NeuroCAPs is an open-source Python package that makes CAPs analysis accessible to neuroimaging researchers for preprocessed resting-state or task-based fMRI data. The package is optimized for fMRIPrep-preprocessed data (Esteban et al., 2019), leveraging fMRIPrep's robust, BIDS-compliant outputs.

Background

Numerous fMRI studies employ static functional connectivity (sFC) techniques to analyze correlative activity within and between brain regions. However, these approaches assume functional connectivity patterns, which change within seconds (Jiang et al., 2022), remain stationary throughout the entire data acquisition period (Hutchison et al., 2013).

Unlike sFC approaches, dynamic functional connectivity (dFC) methods enable the analysis of dynamic functional states, which are characterized by consistent, replicable, and distinct periods of time-varying brain connectivity patterns (Rabany et al., 2019). Among these techniques, CAPs analysis aggregates similar spatial distributions of brain activity using a clustering algorithm (i.e. k-means) to capture the dynamic nature of brain activity (Liu et al., 2013, 2018).

Statement of Need

The typical CAPs workflow can be programmatically time-consuming to manually orchestrate as it generally entails several steps:

1. apply spatial dimensionality reduction to timeseries data
2. perform nuisance regression and remove high-motion volumes
3. concatenate timeseries data from multiple subjects into a single matrix
4. implement k-means clustering with optimal cluster selection
5. generate visualizations for CAP interpretation

While excellent CAPs toolboxes exist, they are often implemented in proprietary languages such as MATLAB (TbCAPs (Bolton et al., 2020)), lack comprehensive end-to-end analytical pipelines for both resting-state and task-based fMRI data with temporal dynamic metrics and visualization capabilities (capcalc (Frederick & Drucker, 2022)), or are comprehensive, but generalized toolboxes for evaluating and comparing different dFC methods (pydFC (Torabi et al., 2024)).

NeuroCAPs provides an accessible Python package for end-to-end CAPs analysis, spanning fMRI post-processing through computation of temporal metrics and creation of visualizations. However, many of NeuroCAPs' post-processing functionalities assumes that fMRI data is organized in a BIDS compliant directory (Yarkoni et al., 2019) and is optimized for fMRIPrep (Esteban et al., 2019) or fMRIPrep-like pipelines such as NiBabies (Goncalves et al., 2025). Furthermore, NeuroCAPs is limited to the k-means algorithm for clustering, a choice that aligns with the original CAPs methodology (Liu et al., 2013) and its prevalence in the CAPs literature.

Modules

The core functionalities of NeuroCAPs are concentrated in three modules:

1. `neurocaps.extraction` contains the `TimeseriesExtractor` class, which:

- leverages Nilearn's (contributors, n.d.) `NiftiLabelsMasker` for denoising and spatial dimensionality reduction using deterministic parcellations (e.g., Schaefer (Schaefer et al., 2018), AAL (Tzourio-Mazoyer et al., 2002), etc)
- removes high-motion volumes using fMRIPrep-derived framewise displacement (FD) values
- reports quality control metrics for motion and non-steady state volumes

2. `neurocaps.analysis` contains the `CAP` class for performing the CAPs analysis, as well as standalone functions.

- The `CAP` class:
  - identifies CAPs via k-means clustering (Pedregosa et al., 2011) with optimized cluster selection (e.g., silhouette, elbow (Arvai, 2023), etc)
  - computes subject-level temporal metrics (e.g., temporal fraction, transition probabilities, etc)
  - converts CAPs to NIfTI statistical maps
  - integrates multiple plotting libraries (Gale et al., 2021; Hunter, 2007; Inc., n.d.; Waskom, 2021) for diverse visualizations
- Standalone functions: provides tools for merging timeseries across sessions/tasks and creating group-averaged transition matrices.

3. `neurocaps.utils` contains utility functions for:

- fetching preset parcellation approaches (i.e. 4S, HCPex (Huang et al., 2022), and Gordon (Gordon et al., 2016))
- generating custom parcellation approaches from tabular metadata
- customizing plots and simulating data

Workflow

The following code demonstrates basic usage of NeuroCAPs (with simulated data) to perform CAPs analysis. A version of this example using real data is available on NeuroCAPs' readthedocs.

1. Extract timeseries data

```python
import numpy as np
from neurocaps.extraction import TimeseriesExtractor
from neurocaps.utils import simulate_bids_dataset

# Set seed
np.random.seed(0)
```

```python
# Generate a BIDS directory with fMRIPrep derivatives
bids_root = simulate_bids_dataset(n_subs=3, n_runs=1, n_volumes=100, task_name="rest")

# Using Schaefer, one of the default parcellation approaches
parcel_approach = {"Schaefer": {"n_rois": 100, "yeo_networks": 7}}

# List of fMRIPrep-derived confounds for nuisance regression
acompcor_names = [f"a_comp_cor_0{i}" for i in range(5)]
confound_names = ["cosine*", "trans*", "rot*", *acompcor_names]

# Initialize extractor with signal cleaning parameters
extractor = TimeseriesExtractor(
    space="MNI152NLin2009cAsym",
    parcel_approach=parcel_approach,
    confound_names=confound_names,
    standardize=False,
    # Run discarded if more than 30% of volumes exceed FD threshold
    fd_threshold={"threshold": 0.90, "outlier_percentage": 0.30},
)

# Extract preprocessed BOLD data
extractor.get_bold(bids_dir=bids_root, task="rest", tr=2, n_cores=1, verbose=False)

# Check QC information
qc_df = extractor.report_qc()
print(qc_df)
```

	Subject_ID	Run	Mean_FD	Std_FD	Frames_Scrubbed	Frames_Interpolated	Mean_High_Motion_Length	Std_High_Motion_Length	N_Dummy_Scans
0	0	run-0	0.516349	0.289657	9	0	1.125000	0.330719	NaN
1	1	run-0	0.526343	0.297550	17	0	1.133333	0.339935	NaN
2	2	run-0	0.518041	0.273964	8	0	1.000000	0.000000	NaN

Figure 1: Quality Control Dataframe.

2. Use k-means clustering to identify the optimal number of CAPs from the data using a heuristic

```python
from neurocaps.analysis import CAP
from neurocaps.utils import PlotDefaults

# Initialize CAP class
cap_analysis = CAP(parcel_approach=extractor.parcel_approach, groups=None)

plot_kwargs = {**PlotDefaults.get_caps(), "figsize": (4, 3), "step": 2}

# Find optimal CAPs (2-20) using silhouette method; results are stored
cap_analysis.get_caps(
    subject_timeseries=extractor.subject_timeseries,
    n_clusters=range(2, 21),
    standardize=True,
    cluster_selection_method="silhouette",
    max_iter=500,
    n_init=10,
    show_figs=True,
    **plot_kwargs,
)
```

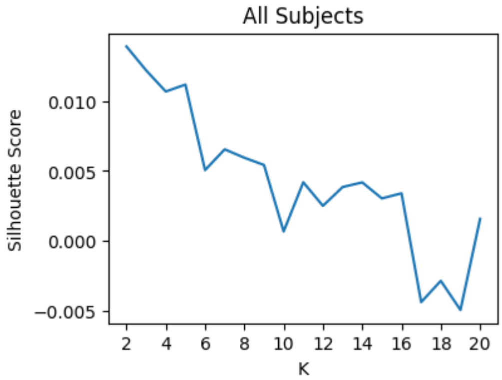

Figure 2: Silhouette Score Plot.

3. Compute temporal dynamic metrics for downstream statistical analyses

```python
# Calculate temporal fraction of each CAP
metric_dict = cap_analysis.calculate_metrics(
    extractor.subject_timeseries, metrics=["temporal_fraction"]
)
print(metric_dict["temporal_fraction"])
```

	Subject_ID	Group	Run	CAP-1	CAP-2
0	0	All Subjects	run-0	0.505495	0.494505
1	1	All Subjects	run-0	0.530120	0.469880
2	2	All Subjects	run-0	0.521739	0.478261

Figure 3: Temporal Fraction Dataframe.

Note that CAP-1 is the dominant brain state across subjects (highest frequency).

4. Visualize CAPs

```python
# Create surface and radar plots for each CAP
surface_kwargs = {**PlotDefaults.caps2surf(), "layout": "row", "size": (500, 100)}

radar_kwargs = {**PlotDefaults.caps2radar(), "height": 400, "width": 485}
radar_kwargs["radialaxis"] = {"range": [0, 0.4], "tickvals": [0.1, "", "", 0.4]}
radar_kwargs["legend"] = {"yanchor": "top", "y": 0.75, "x": 1.15}

cap_analysis.caps2surf(**surface_kwargs).caps2radar(**radar_kwargs)
```

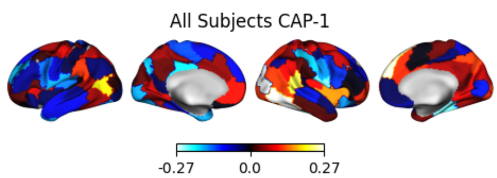

Figure 4: CAP-1 Surface Image.

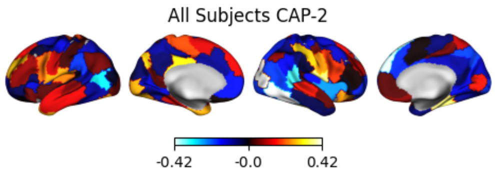

Figure 5: CAP-2 Surface Image.

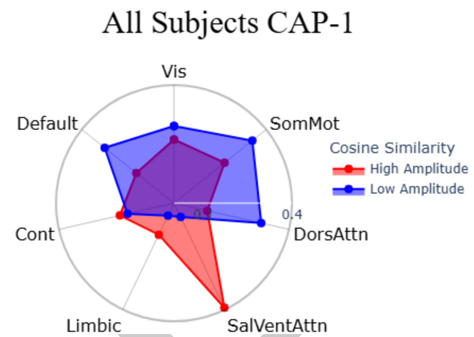

Figure 6: CAP-1 Radar Image.

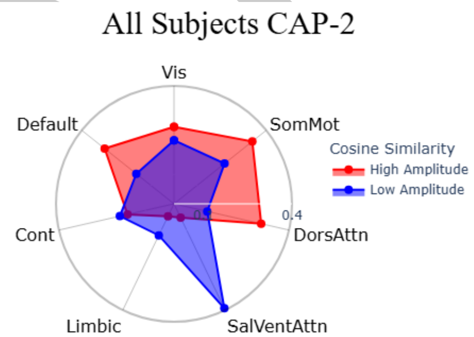

Figure 7: CAP-2 Radar Image.

Radar plots show network alignment (measured by cosine similarity): "High Amplitude" represents alignment to activations (> 0), "Low Amplitude" represents alignment to deactivations (< 0).

Each CAP can be characterized using either maximum alignment (CAP-1: Vis+/SomMot-; CAP-2: SomMot+/Vis-) or predominant alignment ("High Amplitude" – "Low Amplitude"; CAP-1: SalVentAttn+/SomMot-; CAP-2: SomMot+/SalVentAttn-).

```python
import pandas as pd

for cap_name in cap_analysis.caps["All Subjects"]:
    df = pd.DataFrame(cap_analysis.cosine_similarity["All Subjects"][cap_name])
    df["Net"] = df["High Amplitude"] - df["Low Amplitude"]
    df["Regions"] = cap_analysis.cosine_similarity["All Subjects"]["Regions"]
    print(df, "\n")
```

	High Amplitude	Low Amplitude	Net	Regions
0	0.220327	0.199130	0.021197	Vis
1	0.244341	0.154648	0.089693	SomMot
2	0.141356	0.399899	-0.258543	DorsAttn
3	0.300487	0.103134	0.197352	SalVentAttn
4	0.104964	0.092692	0.012272	Limbic
5	0.194957	0.160273	0.034684	Cont
6	0.263373	0.308228	-0.044855	Default

Figure 8: CAP-1 Network Alignment Dataframe.

	High Amplitude	Low Amplitude	Net	Regions
0	0.199130	0.220327	-0.021197	Vis
1	0.154648	0.244341	-0.089693	SomMot
2	0.399899	0.141356	0.258543	DorsAttn
3	0.103134	0.300487	-0.197352	SalVentAttn
4	0.092692	0.104964	-0.012272	Limbic
5	0.160273	0.194957	-0.034684	Cont
6	0.308228	0.263373	0.044855	Default

Figure 9: CAP-2 Network Alignment Dataframe.

Documentation

Comprehensive documentation and tutorials can be found at https://neurocaps.readthedocs.io/ and https://github.com/donishadsmith/neurocaps.

Acknowledgements

Funding from the Dissertation Year Fellowship Program at Florida International University supported NeuroCAPs' refinement and expansion.

References

Arvai, K. (2023). *Kneed*. Zenodo. https://doi.org/10.5281/ZENODO.8127224

Bolton, T. A. W., Tuleasca, C., Wotruba, D., Rey, G., Dhanis, H., Gauthier, B., Delavari, F., Morgenroth, E., Gaviria, J., Blondiaux, E., Smigielski, L., & Van De Ville, D. (2020). TbCAPs: A toolbox for co-activation pattern analysis. *NeuroImage*, *211*, 116621. https://doi.org/10.1016/j.neuroimage.2020.116621

contributors, N. (n.d.). *nilearn*. https://doi.org/10.5281/zenodo.8397156

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., Oya, H., Ghosh, S. S., Wright, J., Durnez, J., Poldrack, R. A., & Gorgolewski, K. J. (2019). fMRIPrep: A robust preprocessing pipeline for functional MRI. *Nature Methods*, *16*(1), 111–116. https://doi.org/10.1038/s41592-018-0235-4

Frederick, B. D., & Drucker, D. M. (2022). *Bbfrederick/capcalc: Version 1.2.2.2 - 8/30/22 deployment bug fix*. Zenodo. https://doi.org/10.5281/ZENODO.7035806

JOSS The Journal of Open Source Software

Gale, D. J., Vos de Wael., R., Benkarim, O., & Bernhardt, B. (2021). *Surfplot: Publication-ready brain surface figures*. Zenodo. https://doi.org/10.5281/ZENODO.5567926

Goncalves, M., Markiewicz, C. J., Esteban, O., Feczko, E., Poldrack, R. A., & Fair, D. A. (2025). *NiBabies: A robust preprocessing pipeline for infant functional MRI*. Zenodo. https://doi.org/10.5281/ZENODO.14811979

Gordon, E. M., Laumann, T. O., Adeyemo, B., Huckins, J. F., Kelley, W. M., & Petersen, S. E. (2016). Generation and evaluation of a cortical area parcellation from resting-state correlations. *Cerebral Cortex*, *26*(1), 288–303. https://doi.org/10.1093/cercor/bhu239

Huang, C.-C., Rolls, E. T., Feng, J., & Lin, C.-P. (2022). An extended human connectome project multimodal parcellation atlas of the human cortex and subcortical areas. *Brain Structure and Function*, *227*(3), 763–778. https://doi.org/10.1007/s00429-021-02421-6

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Hutchison, R. M., Womelsdorf, T., Allen, E. A., Bandettini, P. A., Calhoun, V. D., Corbetta, M., Della Penna, S., Duyn, J. H., Glover, G. H., Gonzalez-Castillo, J., Handwerker, D. A., Keilholz, S., Kiviniemi, V., Leopold, D. A., De Pasquale, F., Sporns, O., Walter, M., & Chang, C. (2013). Dynamic functional connectivity: Promise, issues, and interpretations. *NeuroImage*, *80*, 360–378. https://doi.org/10.1016/j.neuroimage.2013.05.079

Inc., P. T. (n.d.). *Chart title*. https://plotly.com/python/radar-chart/; Plotly Technologies Inc.

Jiang, F., Jin, H., Gao, Y., Xie, X., Cummings, J., Raj, A., & Nagarajan, S. (2022). Time-varying dynamic network model for dynamic resting state functional connectivity in fMRI and MEG imaging. *NeuroImage*, *254*, 119131. https://doi.org/10.1016/j.neuroimage.2022.119131

Liu, X., Chang, C., & Duyn, J. H. (2013). Decomposition of spontaneous brain activity into distinct fMRI co-activation patterns. *Frontiers in Systems Neuroscience*, *7*. https://doi.org/10.3389/fnsys.2013.00101

Liu, X., Zhang, N., Chang, C., & Duyn, J. H. (2018). Co-activation patterns in resting-state fMRI signals. *NeuroImage*, *180*, 485–494. https://doi.org/10.1016/j.neuroimage.2018.01.041

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Rabany, L., Brocke, S., Calhoun, V. D., Pittman, B., Corbera, S., Wexler, B. E., Bell, M. D., Pelphrey, K., Pearlson, G. D., & Assaf, M. (2019). Dynamic functional connectivity in schizophrenia and autism spectrum disorder: Convergence, divergence and classification. *NeuroImage: Clinical*, *24*, 101966. https://doi.org/10.1016/j.nicl.2019.101966

Schaefer, A., Kong, R., Gordon, E. M., Laumann, T. O., Zuo, X.-N., Holmes, A. J., Eickhoff, S. B., & Yeo, B. T. T. (2018). Local-global parcellation of the human cerebral cortex from intrinsic functional connectivity MRI. *Cerebral Cortex*, *28*(9), 3095–3114. https://doi.org/10.1093/cercor/bhx179

Torabi, M., Mitsis, G. D., & Poline, J.-B. (2024). On the variability of dynamic functional connectivity assessment methods. *GigaScience*, *13*, giae009. https://doi.org/10.1093/gigascience/giae009

Tzourio-Mazoyer, N., Landeau, B., Papathanassiou, D., Crivello, F., Etard, O., Delcroix, N., Mazoyer, B., & Joliot, M. (2002). Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain.

NeuroImage, *15*(1), 273–289. https://doi.org/10.1006/nimg.2001.0978

Waskom, M. L. (2021). Seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Yarkoni, T., Markiewicz, C., De La Vega, A., Gorgolewski, K., Salo, T., Halchenko, Y., McNamara, Q., DeStasio, K., Poline, J.-B., Petrov, D., Hayot-Sasson, V., Nielson, D., Carlin, J., Kiar, G., Whitaker, K., DuPre, E., Wagner, A., Tirrell, L., Jas, M., … Blair, R. (2019). PyBIDS: Python tools for BIDS datasets. *Journal of Open Source Software*, *4*(40), 1294. https://doi.org/10.21105/joss.01294