capcalc Documentation

Release 1.3.6

Blaise Frederick

Aug 16, 2023

CONTENTS

1 Citing capcalc **3**

2 Contents **5**

2.1 Introduction 5
2.2 Capcalc 5
2.3 NOTE 5
2.4 Ok, I'm sold. What's in here? 5
2.5 Installation 6
2.6 Docker installation 7
2.7 Singularity installation 8
2.8 Usage information 9
2.9 What's new 13
2.10 Release history 13

3 Indices and tables **17**

capcalc is a suite of python programs used to perform coactivation pattern analysis on time series data. It uses K-Means clustering to find a set of "activation states" that represent the covarying patterns in the data.

HTML documentation is here: http://capcalc.readthedocs.io/en/latest/

CHAPTER

ONE

CITING CAPCALC

Frederick, B, capcalc [Computer Software] (2016-2022). Available from https://github.com/bbfrederick/capcalc. doi:10.5281/zenodo.7035806

CHAPTER

TWO

CONTENTS

2.1 Introduction

2.2 Capcalc

capcalc is a suite of python programs used to perform coactivation pattern analysis on time series data. It uses K-Means clustering to find a set of "activation states" that represent the covarying patterns in the data.

HTML documentation is here: http://capcalc.readthedocs.io/en/latest/

2.3 NOTE

This is an evolving code base. I'm constantly tinkering with it. That said, now that I'm releasing this to the world, I'm being somewhat more responsible about locking down stable release points. In between releases, however, I'll be messing with things. **It's very possible I could break something while doing this, so check back for status updates if you download the code in between releases**. I've finally become a little more modern and started adding automated testing, so as time goes by hopefully the "in between" releases will be somewhat more reliable. Check back often for exciting new features and bug fixes!

2.4 Ok, I'm sold. What's in here?

- **roidecompose** - This program uses an atlas to extract timecourses from a 4D nifti file, producing a text file with the averaged timecourse from each region in the atlas (each integral value in file) in each column. This can be input to capfromtcs. There are various options for normalizing the timecourses.
- **capfromtcs** - This does the actual CAP calculation, performing a k-means cluster analysis on the set of timecourses to find the best representitive set of "states" in the file. Outputs the states found and the dominant state in each timepoint of the timecourse.
- **maptoroi** - The inverse of roidecompose. Give it a set of cluster timecourses and a template file, and it maps the values back onto the rois
- **statematch** - Use this for aligning two state output files. Takes two state timecourse files, and determines which states in the second correspond to which states in the first. Generates a new 'remapped' file with the states in the second file expressed as states in the first.

2.5 Installation

2.5.1 Required dependencies

capcalc requires some external libraries to be installed first:

- Python 3.x
- numpy
- scipy
- matplotlib
- scikit-learn
- statsmodels
- nibabel

2.5.2 Installing from pypi

I've finally gotten pypi deployment working, so the new easiest way to install capcalc is to simply type:

```
pip install capcalc
```

That's it, I think.

2.5.3 Installing with conda

The other simple way to get this all done is to use Anaconda python from Continuum Analytics. It's a free, curated scientific Python distribution that is easy to maintain and takes a lot of headaches out of maintaining a distribution. It also already comes with almost all of the dependancies for capcalc installed by default. You can get it here: https://www.continuum.io. You should download the most recent Python 3 version.

After installing Anaconda python, install the remaining dependency To do this most easily, you should have conda-forge as one of your source channels. To add conda-forge, type:

```
conda config --add channels conda-forge
```

Then install the additional dependencies:

```
conda install pyqtgraph nibabel pillow
```

Done.

2.5.4 Installing capcalc

Once you have installed the prerequisites, cd into the package directory, and type the following:

```
python setup.py install
```

to install all of the tools in the package. You should be able to run them from the command line then (after rehashing).

2.5.5 Updating

If you've previously installed capcalc and want to update, cd into the package directory and do a git pull first:

```
git pull
python setup.py install
```

2.6 Docker installation

There is a Docker container with a full capcalc installation. To use this, first make sure you have docker installed and properly configured, then run the following:

```
docker pull fredericklab/capcalc:VERSIONNUMBER
```

This it will download the docker container from dockerhub. It's around 2GB, so it may take some time, but it caches the file locally, so you won't have to do this again unless the container updates. To use a particular version, replace VERSIONNUMBER with the version of the with container you want (currently the newest version is 1.0.0rc2).

If you like to live on the edge, just use:

```
docker pull fredericklab/capcalc:latest
```

This will use the most recent version on dockerhub.

Now that the file is downloaded, you can run any capcalc command in the Docker container. For example, to run capfromtcs itself, you would use the following command (you can do this all in one step - it will just integrate the first pull into the run time if the version you request hasn't already been downloaded).

Docker runs completely in it's own selfcontained environment. If you want to be able to interact with disks outside of container, you map the volume to a mount point in the container using the –volume=EXTERNALDIR:MOUNTPOINT[,ANOTHERDIR:ANOTHERMOUNTPOINT] option to docker.

One complication of Docker - if you're running a program that displays anything (and we do), you'll have to add a few extra arguments to the docker call. Docker is a little weird about X forwarding - the easiest thing to do is find the IP address of the machine you're running on (lets call it MYIPADDRESS), and do the following:

```
xhost +
```

This disables X11 security - this is almost certainly not the best thing to do, but I don't have a better solution at this time, and it works.

If you're on a Mac using Xquartz, prior to this you'll also have to do three more things.

1) In Xquartz, go into the security preferences, and make sure "Allow connections from network hosts" is checked.

2) Tell Xquartz to listen for TCP connections (this is not the default). Go to a terminal window and type:

```
defaults write org.macosforge.xquartz.X11 nolisten_tcp 0
```

3) Log out and log back in again (you only need to do this once - it will stay that way until you change it.)

Then you should be good to go, with the following command:

```
docker run \
    --network host\
    --volume=INPUTDIRECTORY:/data_in,OUTPUTDIRECTORY:/data_out \
    -it \
    -e DISPLAY=MYIPADDRESS:0 \
    -u capcalc \
    fredericklab/capcalc:VERSIONNUMBER \
        capfromtcs \
            -i manyfiles.txt \
            -o output/manyfiles \
            --sampletime=0.72 \
            --varnorm \
            -m \
            -b 4800 \
            -S 1200 \
            --quality \
            -E default \
            --minout=2 \
            [otheroptions]
```

You can replace the capfromtcs blah blah blah command with any other program in the package (currently only "grader", which classifies timecourses) - after the fredericklab/capcalc:latest, just specify the command and arguments as you usually would.

2.7 Singularity installation

Many times you can't use Docker, because of security concerns. Singularity, from LBL, offers containerized computing that runs entirely in user space, so the amount of mischief you can get up to is significantly less. Singularity containers can be created from Docker containers as follows (stealing from the fMRIprep documentation):

```
singularity build /my_images/capcalc-VERSIONNUMBER.simg docker://fredericklab/
↪capcalc:VERSIONNUMBER
```

Running the container is similar to Docker. The "-B" option is used to bind filesystems to mountpoints in the container.

singularity run
–cleanenv -B INPUTDIRECTORY:/data_in,OUTPUTDIRECTORY:/data_out capcalc-VERSIONNUMBER.simg

capfromtcs
-i manyfiles.txt -o output/manyfiles –sampletime=0.72 –varnorm -m -b 4800 -S 1200 –quality -E default –minout=2 [otheroptions]

2.8 Usage information

2.8.1 roidecompose

Description:

Inputs:

Outputs:

Usage:

2.8.2 capfromtcs

Description:

This scripts takes a file containing a number of timecourses, does some preprocessing on them, then calculates the coactivation patterns found in the data.

Inputs:

There are three required inputs:

The input file is a text file containing 2 or more timecourses in columns separated by whitespace. Each row is a time point.

The outputname is the prefix of all output files.

The samplerate (or sampletime) is the frequency (or timestep) of the input data.

Outputs:

Outputs are space or space by time Nifti files (depending on the file), and some text files containing textual information, histograms, or numbers. Output spatial dimensions and file type match the input dimensions and file type (Nifti1 in, Nifti1 out). Depending on the file type of map, there can be no time dimension, a time dimension that matches the input file, or something else, such as a time lag dimension for a correlation map. OUTPUT_normalized.txt

This is a text file containing the input timecourses after applying the selected normalization.

OUTPUT_clustercenters.txt
This is a text file with NUMCLUSTER lines, one for each state, specifying the center location of the cluster (each line has the same number of columns as the number of input timecourses).

OUTPUT_statelabels.txt
This is a text file with one line per timepoint. Each line is the assigned state for the given timepoint in the input data.

OUTPUT_silhouettesegmentstats.csv

OUTPUT_overallsilhouettemean.csv
This text file has one line per cluster, giving the mean of the mean silhouette score of all the segments that spent any time in that state.

OUTPUT_pctsegsinstate.txt
This text file has one line per cluster, indicating what percentage of subjects (segments) spent any time in this state.

OUTPUT_seg_XXXX_instate_YY.txt
This is a text file with one line per timepoint in the segment. The value is 1 if the system is in state YY, 0 otherwise.

OUTPUT_seg_XXXX_statestats.csv
This is a text file with one line per cluster (state), with the following columns:

- percentage of segment spent in state
- number of continuous runs in state
- total number of TRs in state
- minimum number of TRs spent in state
- maximum number of TRs spent in state
- average number of TRs spent in state
- median number of TRs spent in state
- standard deviation of the number of TRs spent in state

OUTPUT_seg_XXXX_statetimestats.csv
This is the equivalent of the statestats file, where the units are time in seconds rather than TRs

OUTPUT_seg_XXXX_statelabels.txt
This is a text file with one line per timepoint. Each line is the assigned state for the given timepoint in the segment.

OUTPUT_seg_XXXX_silhouetteclusterstats.csv
This is a text file with one line per cluster. Each line has four columns:

- the mean silhouette score for that cluster in that segment.
- the median silhouette score for that cluster in that segment.
- the minimum silhouette score for that cluster in that segment.
- the maximum silhouette score for that cluster in thate segment.

OUTPUT_seg_XXXX_rawtransmat.nii.gz
This is a NIFTI file with dimensions n_states by n_states. The number of transitions from state a to state b is in location [a, b]

OUTPUT_seg_XXXX_normtransmat.nii.gz
This is a NIFTI file containing the same information as the rawtransmat file, but each row is normalized to sum to 1, making, so the numbers represent the transition probabilities, rather than the total number of transitions.

OUTPUT_seg_XXXX_offdiagtransmat.nii.gz
This is a NIFTI file containing the same information as the normtransmat file, except that the diagonal elements have been set to zero. This is therefore the relative probability transitioning to each possible destination state in the case where the state does not simply persist.

OUTPUT_seg_XXXX_rawtransmat.csv OUTPUT_seg_XXXX_normtransmat.csv OUTPUT_seg_XXXX_offdiagtransmat.csv

The same data as the above NIFTI files, but as a csv file.

Usage:

```
capfromtcs - calculate and cluster coactivation patterns for a set of␣
↪timecourses

usage: capfromtcs -i timecoursefile -o outputfile --samplefreq=FREQ --
↪sampletime=TSTEP
                  [--nodetrend] [-s STARTTIME] [-D DURATION]
                  [-F LOWERFREQ,UPPERFREQ[,LOWERSTOP,UPPERSTOP]] [-V] [-L] [-
↪R] [-C]
                  [-m] [-n NUMCLUSTER] [-b BATCHSIZE] [-S SEGMENTSIZE] [-E␣
↪SEGMENTTYPE] [-I INITIALIZATIONS]
                  [--noscale] [--nonorm] [--pctnorm] [--varnorm] [--stdnorm] [-
↪-ppnorm] [--quality]
                  [--pca] [--ica] [-p NUMCOMPONENTS]

required arguments:
    -i, --infile=TIMECOURSEFILE  - text file mulitple timeseries
    -o, --outfile=OUTNAME        - the root name of the output files

    --samplefreq=FREQ            - sample frequency of all timecourses is FREQ
           or
    --sampletime=TSTEP           - time step of all timecourses is TSTEP
                                   NB: --samplefreq and --sampletime are two␣
↪ways to specify
                                   the same thing.

optional arguments:

  Data selection/partition:
    -s STARTTIME                 - time of first datapoint to use in seconds␣
↪in the first file
    -D DURATION                  - amount of data to use in seconds
    -S SEGMENTSIZE,[SEGSIZE2,...SEGSIZEN]
                                 - treat the timecourses as segments of length␣
↪SEGMENTSIZE for preprocessing.
    -E SEGTYPE,SEGTYPE2[,...SEGTYPEN]
                                 - group subsegments for summary statistics. ␣
↪All subsegments in the same group must be the same length
                                   If there are multiple, comma separated␣
↪numbers, treat these as subsegment lengths.
                                   Default segmentsize is the entire length
  Clustering:
    -m                           - run MiniBatch Kmeans rather than␣
↪conventional - use with very large datasets
    -n NUMCLUSTER                - set the number of clusters to NUMCLUSTER␣
↪(default is 8)
    -b BATCHSIZE                 - use a batchsize of BATCHSIZE if doing␣
↪MiniBatch - ignored if not.  Default is 1000
    --dbscan                     - perform dbscan clustering
    --hdbscan                    - perform hdbscan clustering
    -I INITIALIZATIONS           - Restart KMeans INITIALIZATIONS times to␣
↪find best fit (default is 1000)
```

(continues on next page)

(continued from previous page)

```
  Preprocessing:
    -F                          - filter data and regressors from LOWERFREQ␣
↪to UPPERFREQ.
                                  LOWERSTOP and UPPERSTOP can be specified,␣
↪or will be calculated automatically
    -V                          - filter data and regressors to VLF band
    -L                          - filter data and regressors to LFO band
    -R                          - filter data and regressors to respiratory␣
↪band
    -C                          - filter data and regressors to cardiac band
    --nodetrend                 - do not detrend the data before correlation
    --noscale                   - don't perform vector magnitude scaling
    --nonorm                    - don't normalize timecourses
    --pctnorm                   - scale each timecourse to it's percentage of␣
↪the mean
    --varnorm                   - scale each timecourse to have a variance of␣
↪1.0 (default)
    --stdnorm                   - scale each timecourse to have a standard␣
↪deviation of 1.0
    --ppnorm                    - scale each timecourse to have a peak to␣
↪peak range of 1.0
    --pca                       - perform PCA dimensionality reduction prior␣
↪to analysis
    --ica                       - perform ICA dimensionality reduction prior␣
↪to analysis
    -p NUMCOMPONENTS            - set the number of p/ica components to␣
↪NUMCOMPONENTS (default is 8).  Set to -1 to estimate
    --noscale                   - do not apply standard scaler befor cluster␣
↪fitting

  Other:
    --GBR                       - apply gradient boosting regressor testing␣
↪on clusters
    -d                          - display some quality metrics
    --quality                   - perform a silhouette test to evaluate fit␣
↪quality
    -v                          - turn on verbose mode
```

These options are somewhat self-explanatory. I will be expanding this section of the manual going forward, but I want to put something here to get this out here.

2.8.3 maptoroi

Description:

maptoroi takes ROI values from a text file and maps them back onto a NIFTI image for display.

Inputs:

maptoroi requires an input text file with 1 column per region giving the value of the ROI. If there are multiple rows, each row corresponds to a time point. It also requires a template NIFTI file.

Outputs:

showxcorr outputs everything to standard out, including the Pearson correlation, the maximum cross correlation, the time of maximum cross correlation, and estimates of the significance levels (if specified). There are no output files.

Usage:

```
usage: maptoroi inputfile templatefile outputroot

required arguments:
    inputfile        - the name of the file with the roi values to be mapped␣
↪back to image space
    templatefile     - the name of the template region file
    outputfile       - the name of the output nifti file
```

2.8.4 roidecompose

Description:

Inputs:

Outputs:

Usage:

2.9 What's new

2.10 Release history

2.10.1 Version 1.3.6 (5/11/23)

- (Docker) Updated to python 3.11 basecontainer.
- (Docker) Fixed testdocker.sh.
- (package) Modernized install procedure.

2.10.2 Version 1.3.5 (1/11/23)

- (capfromtcs) Fixed summary statistics outputs.

2.10.3 Version 1.3.4 (1/10/23)

- (capfromtcs) Revert change to max_iter when specifying initialcenters.

2.10.4 Version 1.3.3 (1/10/23)

- (capfromtcs) Changed intialization of kmeans from initialcenters, save more intermediate results.
- (utils.py) Factored out some transition array calculations.

2.10.5 Version 1.3.2 (11/15/22)

- (capfromtcs) Fixed a typo in specifying the initial k-means++ method.
- (io) Updated libraries to match some changes in rapidtide.

2.10.6 Version 1.3.1 (11/10/22)

- (capfromtcs) Added new option "–initialcenters" to allow specification of k-means cluster centers. This lets you reuse the CAPS as previous runs, but gets around the difficulty of reading back old models run under a different version of python or scikit-learn.

2.10.7 Version 1.3.0 (11/10/22)

- (package) Removed rapidtide as a dependency by copying over the necessary support routines.
- (Docker) Updated basecontainer to latest, switched over to using pip rather than mamba for env.
- (docs) Corrected a problem that was causing readthedocs build to fail.

2.10.8 Version 1.2.3 (8/31/22)

- (capfromtcs, clustercomp) Used newer, non-deprecated method to access nifti files with nibabel.
- (package) Added docker and singularity test scripts.
- (package) Reformatted several files with black.
- (package) Made many (unsuccesful) attempts to get the documentation to build.

2.10.9 Version 1.2.2.5 (8/30/22)

- (package) Bump to trigger github deployment.

2.10.10 Version 1.2.2.4 (8/30/22)

- (package) Convert README.md to README.rst

2.10.11 Version 1.2.2.3 (8/30/22)

- (package) Fixed Development status in setup.py

2.10.12 Version 1.2.2.2 (8/30/22)

- (package) Syncing with rapidtide to try to get pypi deployment to work

2.10.13 Version 1.2.2.1 (8/29/22)

- (package) Fixed versioneer installation.

2.10.14 Version 1.2.2 (8/29/22)

- (package) Updated pyproject.toml and versioneer to try to fix pypi deployment.

2.10.15 Version 1.2.1 (8/22/22)

- (Docker) Fixed Dockerfile error.
- (package) Updated pypi authentication information.

2.10.16 Version 1.2.0 (8/22/22)

- (Docker) Added Docker compatibility.
- (package) Updated to match the current version of rapidtide.
- (package) Added "refresh" script to simplify updates.
- (package) Formatting changes.
- (capfromtcs) Fixed import of joblib, updated to new NonCausalFilter calling conventions.
- (clusternifti) Major overhaul. Added normalization, PCA and ICA dimensionality reduction, switched to argparse, added repeats.
- (clustersort) New program to harmonize multiple cluster solutions.
- (clusternifti, clustersort, supercluster) Harmonized mask specification arguments and internal variable names.

2.10.17 Version 1.1.0 (8/20/21)

- (package) Move to versioneer.

2.10.18 Version 1.0.0 (2/15/17)

- First release

CHAPTER

THREE

INDICES AND TABLES

- genindex
- search