picachooser Documentation

Release 1.4.6

Blaise Frederick

May 07, 2024

CONTENTS

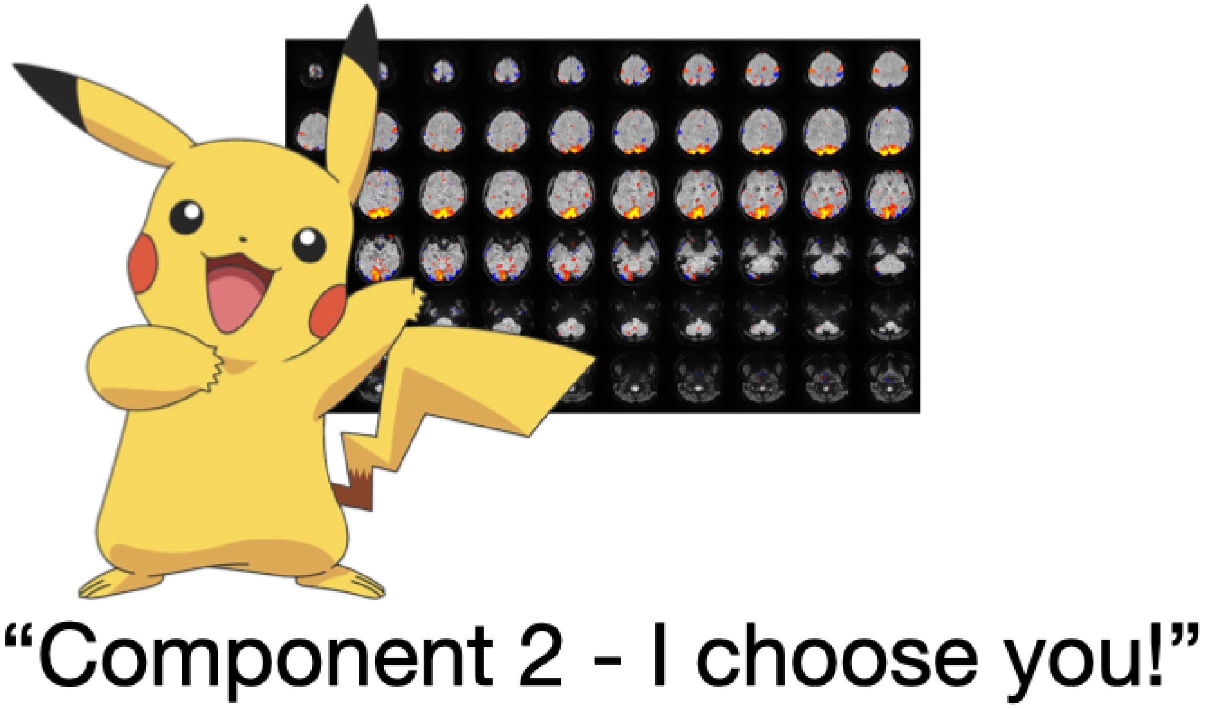
PICAchooser is a suite of simple gui tools for scanning through MELODIC probabalistic ICA runs and quickly making decisions about which components are worth keeping, and what relates to what. These tools each only do one thing, but they do them quickly and easily using only keyboard input. Current programs are PICAchooser, melodicomp, and grader.

CHAPTER

ONE

CONTENTS

1.1 PICAchooser (the package)



A set of simple gui tools for scanning through MELODIC probabalistic ICA runs and quickly making decisions about which components to retain, and what relates to what. These tools each only do one thing, but they do them quickly and easily using only keyboard input. Current programs are PICAchooser, melodicomp, and grader.

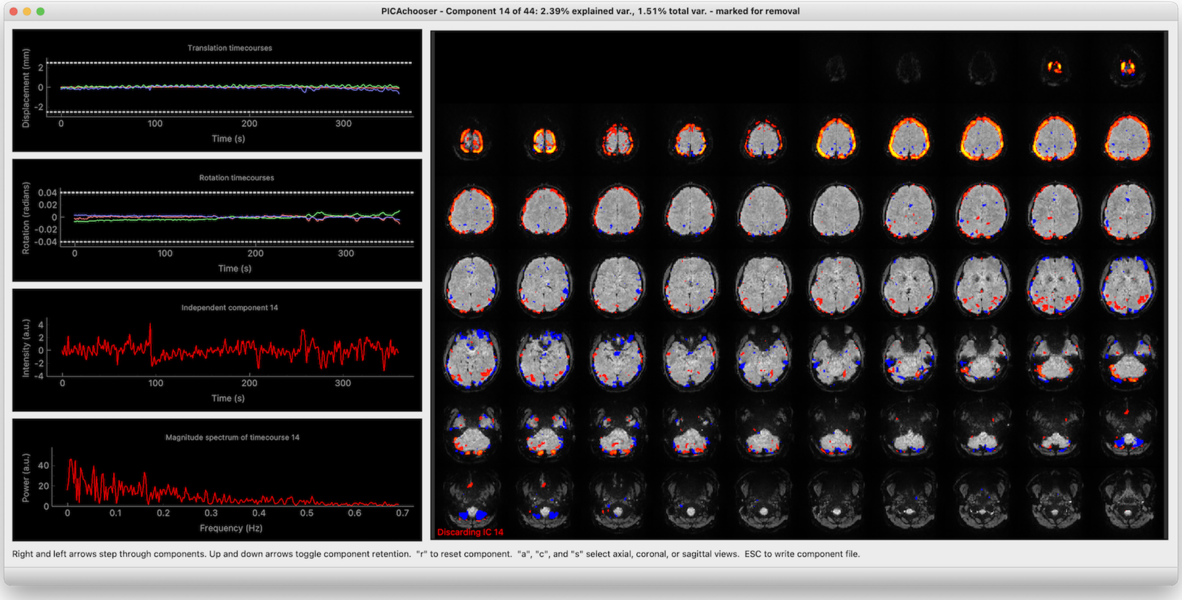
Full documentation is here: https://picachooser.readthedocs.io/en/latest/introduction.html

1.2 What's in here?

1.2.1 PICAchooser

Lets you step through the components in an ICA analysis (from many sources), and select which components you want to retain. In addition to showing the spatial ICs, it also displays the componnent timecourses, motion traces, and the correlation between them, to help with your decision making.

Once you launch, you do everything with keyboard commands, and it's been optimized to go as fast as possible, so you aren't waiting around for things.



PICAchooser - Component 14 of 44: 2.39% explained var., 1.51% total var. - marked for removal

Right and left arrows step through components. Up and down arrows toggle component retention. "r" to reset component. "a", "c", and "s" select axial, coronal, or sagittal views. ESC to write component file.

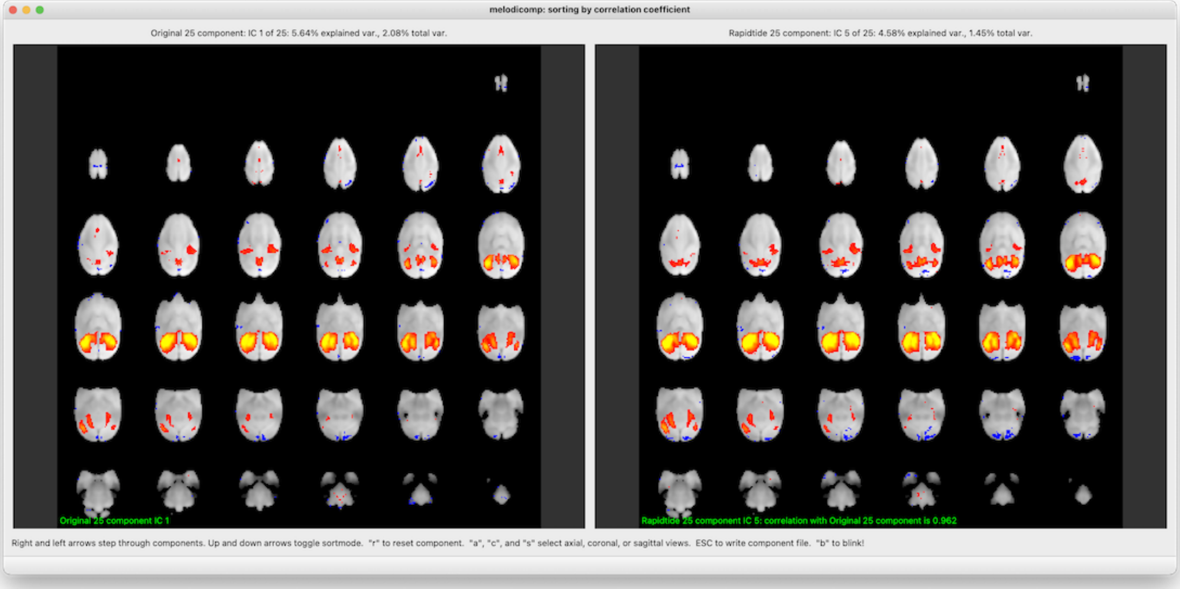
Fig. 1: PICAchooser screenshot

1.2.2 melodicomp

Puts up two melodic IC files side by side. In order to make the comparison meaningful, it first calculates the spatial crosscorrelation between each IC in the first file and each IC in the second. As you step through components in the first file, on the right you see the component with the highest crosscorrelation in the second file. You can sort either by IC order in the first file (i.e. in order of descending variance explained), or in descending correlation coefficient (i.e. best matched components first). When you quit (or hit the escape key), it writes out a file listing the best matched ICs along with their correlation coefficients.

I'm especially proud of the "blink" feature. When you hit the "b" key, the right and left window swap, instantaneously. This lets you see what changes between the two sets of networks in a very natural way. This is inspired by blink comparators, a cool old piece of tech probably long forgotten by most.

Again, once you launch, you do everything with keyboard commands, and it's been optimized to go as fast as possible, so you aren't waiting around for things.



Original 25 component: IC 1 of 25: 5.64% explained var., 2.08% total var.

Rapidtide 25 component: IC 5 of 25: 4.58% explained var., 1.45% total var.

Original 25 component IC 1

Rapidtide 25 component IC 5: correlation with Original 25 component is 0.962

Right and left arrows step through components. Up and down arrows toggle sortmode. "r" to reset component. "a", "c", and "s" select axial, coronal, or sagittal views. ESC to write component file. "b" to blink!

Fig. 2: melodicomp screenshot

1.3 A note on component numbering

Astute users will notice that components are numbered differently in different contexts. This is actually intentional. In the GUI, and in any files that work directly with FSL tools, I use whatever convention FSL uses. So for displayed components, the first component is IC1. Output files that will be used by fsl_regfilt also use this convention. However, for any informational output on the terminal that you might use when looking at components in FSLeyes directly, or operating on them with fslmaths or your own python code, the component numbering starts at 0. As the universe intended. If you use matlab, add 1 in your head.

1.4 Support

This code base is being developed and supported by a grant from the US NIH 1R01 NS097512.

1.5 Additional packages used

PICAchooser would not be possible without many additional open source packages. These include:

1.5.1 pyqtgraph:

1) Luke Campagnola. PyQtGraph: Scientific Graphics and GUI Library for Python

1.5.2 nibabel:

1) Nibabel: Python package to access a cacophony of neuro-imaging file formats | https://10.5281/zenodo.591597

1.5.3 numpy:

1) Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011) | https:10.1109/MCSE.2011.37

1.5.4 scipy:

1) Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17, 261–272 (2020) | https://doi.org/10.1038/s41592-019-0686-2

1.5.5 pandas:

1) McKinney, W., pandas: a foundational Python library for data analysis and statistics. Python for High Performance and Scientific Computing, 2011. 14.

1.6 What's new

History of changes

1.6.1 Version 1.4.6 (2/23/24)

- (rtgrader) Added new program - rtgrader - to do quality assessment of rapidtide datasets.
- (Docker) Updated to basecontainer 0.3.2.
- (Docker) Added caching to build.
- Merged of a bunch of dependabot PRs.

1.6.2 Version 1.4.5 (9/13/23)

- Mass merge of a bunch of dependabot PRs.
- (Docker) Updated to basecontainer 0.2.3.

1.6.3 Version 1.4.4 (6/29/23)

- Fixed image pane resizing.

1.6.4 Version 1.4.3 (6/9/23)

- Added reference file matching - if you specify a set of reference components, any IC with a spatial correlation with any reference component above a threshold is retained.

1.6.5 Version 1.4.2 (5/11/23)

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

1.6.6 Version 1.4.1 (2/14/23)

- Upgraded pyqtgraph calls to handle deprecations. NOTE: this only handles versions of pyqtgraph<0.13.
- Made substantial changes to the Dockerfile to handle changes in basecontainer.

1.6.7 Version 1.4.0 (2/9/23)

- Added –version and –detailedversion command line flags to PICAchooser, melodicomp, and grader.
- Accepted several PR's from dependabot for build scripts.
- Updated versioneer.
- Renamed master branch to main.
- Adapted to the new basecontainer.

1.6.8 Version 1.3.1.2 (8/19/22)

- Updated Dockerfile for a newer python distribution.

1.6.9 Version 1.3.1.1 (8/19/22)

- Updated versioneer.

1.6.10 Version 1.3.1 (8/19/22)

- Tweaked pyproject.ml file to hopefully fix documentation build.

1.6.11 Version 1.3.0 (9/8/21)

- Reformatted with black and isort.
- Flipped x axis to display radiological coordinates.
- Harmonized Dockerfile and automated container building methods with rapidtide.
- Fixed some formatting in documentation (thank you DMD!)

1.6.12 Version 1.2.3 (4/6/20)

- Major documentation improvements.
- Finally fixed picachooser.readthedocs.org.
- Added reset component keystroke to PICAchooser and melodicomp.

1.6.13 Version 1.2.2 (4/5/20)

- More fiddling to get deplyment working again.

1.6.14 Version 1.2.1 (4/5/20)

- Fiddled with .gitignore to try to get deplyment working again.
- Added a help line to the bottom of grader GUI window.

1.6.15 Version 1.2.0 (4/5/20)

- Added a new program - melodicomp - to compare ICs between runs.
- Added a help line to the bottom of GUI windows.

1.6.16 Version 1.1.4 (4/3/20)

- Now with pypi! Just `pip install picachooser`, and off you go!

1.6.17 Version 1.1.0 (4/3/20)

- Added groupmelodic runmode, for examining group ICAs.
- Added the ability to select an ROI from the dataset.
- General code cleanup, reformatting with black.
- Updated documentation.

1.6.18 Version 1.0.2 (11/24/20)

- Added line to help QT compatibility with macOS 11 (Big Sur).

1.6.19 Version 1.0.1 (6/30/20)

- Just bumping the version number to generate an initial Zenodo DOI.

1.6.20 Version 1.0.0 (6/3/20)

- Motion correlation parameters are now properly output to the terminal when switching components.
- Turned down default verbosity.

1.6.21 Version 1.0.0rc13 (6/1/20)

- You can now switch to viewing slices in axial, coronal, or sagittal orientation by pressing the a, c, or s key.
- Fixed the aspect ratio and padding of the images in the display window.

1.6.22 Version 1.0.0rc12 (5/21/20)

- The explained variance and total variance of the component are now displayed in the title bar of the window.

1.6.23 Version 1.0.0rc11 (5/15/20)

- Properly handle the case of the timecourses being shorter than the motion plots (happens when fMRIprep discards (but doesn't really discard) initial timepoints).
- Window resizing works somewhat better (but it's not perfect yet).

1.6.24 Version 1.0.0rc10 (5/7/20)

- All plot linewidths are now settable from the command line.
- Added the –scalemotiontodata option to autoscale motion plots (rather than setting the plot limits by the dashed guide lines.)

1.6.25 Version 1.0.0rc9 (4/27/20)

- keepcolor, discardcolor, transmotlimits and rotmotlimits are now settable via command line arguments. This means docker users can change configuration values.
- Increased some linewidths to make the display more readable, made the widths settable in the config file.

1.6.26 Version 1.0.0rc8 (4/27/20)

- Added fixed (but configurable) “normal” limits to motion plots (thank you to Richard Dinga for the suggestion).

1.6.27 Version 1.0.0rc7 (4/26/20)

- Revamped input file specification to allow for maximum flexibility
- PICAchooser can now read motion out of fmriprep confounds files.
- On bad component file save, print the command needed to refilter the dataset.

1.6.28 Version 1.0.0rc6 (4/25/20)

- Updated the run options for docker and singularity to reflect the current interface

1.6.29 Version 1.0.0rc5 (4/25/20)

- All timecourse colors are now changeable by editing the ${HOME}/.picachooser.json file (the file is created with default values if it doesn’t exist).
- Numerous documentation fixes and updates

1.6.30 Version 1.0.0rc4 (4/25/20)

- Changed help lines to match actual runmode names
- Calculate and print correlation coefficients (and p values) between current component and all motion timecourses

1.6.31 Version 1.0.0rc3 (4/24/20)

- Changed option specification
- Added configuration file to set colors

1.6.32 Version 1.0.0rc2 (4/24/20)

- Fixed docker build issues
- Timecourse and spectrum window now show the component number
- Resolved remaining rapidtide dependencies
- Still having problems with readthedocs

1.6.33 Version 1.0.0rc1 (4/23/20)

- Initial release

1.7 Installation

1.7.1 Required dependencies

PICAchooser requires some external libraries to be installed first:

- Python 3.x
- numpy>=1.16
- scipy
- pandas
- nibabel
- pyqt5
- pyqtgraph
- pillow

1.7.2 Installing from pypi (NEW!)

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

```
pip install picachooser
```

That's it, I think.

1.7.3 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 PICAchooser 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.

1.7.4 Installing PICAchooser

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).

1.7.5 Updating

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

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

1.8 Docker installation

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

```
docker pull fredericklab/picachooser: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/picachooser:latest
```

This will use the most recent version on dockerhub.

Now that the file is downloaded, you can run any picachooser command in the Docker container. For example, to run PICAchooser 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 picachooser \
    fredericklab/picachooser:VERSIONNUMBER \
        PICAchooser \
            RUNMODE \
            --featdir /data_in/FEATDIRECTORY \
            --melodicdir /data_in/MELODICDIRECTORY \
            [otheroptions]
```

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

1.9 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/picachooser-VERSIONNUMBER.simg docker://fredericklab/
↪picachooser: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 picachooser-VERSIONNUMBER.simg

PICAchooser
RUNMODE –featdir /data_in/FEATDIRECTORY –melodicdir /data_in/MELODICDIRECTORY [otheroptions]

To run a GUI application, you need to disable x security on your host (see comment about this above):

```
xhost +
```

then set the display variable to import to the container:

```
setenv SINGULARITY_DISPLAY MYIPADDRESS:0   (if you are using csh)
```

or

```
export SINGULARITY_DISPLAY="MYIPADDRESS:0" (if you are using sh/bash)
```

then just run the gui command with the command given above.

1.10 PICAchooser (the program)

Basically, PICAchooser loads the results of a MELODIC ICA run and lets you step through the components, look at them, and decide if you want to keep them or discard them. After you load the dataset, you get a window showing the active IC component (both the timecourse and the spatial map), the power spectrum of the active IC component, and the motion timecourses for comparison. By default, components are flagged to be kept (and the timecourses are in green*).

1.10.1 Usage

usage: PICAchooser runmode [options]

A program to review (and alter) melodic component selections.

positional arguments:

runmode Analysis mode. Valid choices are "melodic",
"groupmelodic" "aroma", and "fix". In melodic mode, the default output file is named "badcomponents.txt" and will be written to MELODICDIR as comma separated integers. In groupmelodic mode, the default output file is named "goodcomponents.txt" (you are more interested in which components should be retained) and will be written to MELODICDIR as newline separated integers (starting from 0). In aroma mode, the file "classified_motion_ICs.txt" must exist in the parent of MELODICDIR; by default the output will be written to "classified_motion_ICs_revised.txt" in the same directory. In fix mode, the default output file is named "hand_labels_noise.txt" and will be written to MELODICDIR as comma separated integers with square brackets surrounding the line.

optional arguments:

-h, --help show this help message and exit

Standard input file location specification. For certain runmodes, one of these will be sufficient to fully specify all file locations.:

--featdir FEATDIR The FEAT directory associated with this MELODIC run.

--melodicdir MELODICDIR The .ica directory for this MELODIC run.

Nonstandard input file location specification. Setting any of these overrides any location inferred from –melodicdir or –featdir.:

--backgroundfile BGFILE The anatomic file on which to display the ICs (usually found in FEATDIR/reg/example_func.nii.gz),

--funcfile FUNCFILE The functional file to be filtered (usually found in FEATDIR/filtered_func_data.nii.gz),

--motionfile MOTIONFILE The anatomic file on which to display the ICs (usually found in FEATDIR/mc/prefiltered_func_data_mcf.par). If the file has a .tsv extension, assume it is an fmriprep confounds file.

--ICfile ICFILE The independent component file produced by MELODIC (usually found in MELODICDIR/melodic_IC.nii.gz).

--ICmask ICMASK The independent component mask file produced by MELODIC (usually found in MELODICDIR/mask.nii.gz).

--timecoursefile MIXFILE The timecourses of the independant components (usually found in MELODICDIR/melodic_mix),

--ICstatsfile STATSFILE The melodic stats file (usually found in MELODICDIR/melodic_ICstats),

Other arguments:

--initfile INITFILE The name of an initial bad component file (in aroma mode, this overrides the default input file for AROMA).

--outputfile OUTPUTFILE Where to write the bad component file (this overrides the default output file name).

--filteredfile FILTEREDFILE The name of the filtered NIFTI file. If this is set, then when the bad component file is written, the command to generate the filtered file will be printed to the terminal window.

--displaythresh DISPLAYTHRESH z threshold for the displayed ICA components. Default is 2.3.

–spatialroi XMIN XMAX YMIN YMAX ZMIN ZMAX
Only read in image data within the specified ROI. Set MAX to -1 to go to the end of that dimension.

Configuration arguments:

--keepcolor KEEPCOLOR Set the color of timecourses to be kept (default is “g”).

--discardcolor DISCARDCOLOR Set the color of timecourses to discard (default is “r”).

–transmotlimits LOWERLIM UPPERLIM
Override the “normal” limits of translational motion from the values in the configuration file to LOWERLIM-UPPERLIM mm.

–rotmotlimits LOWERLIM UPPERLIM
Override the “normal” limits of rotations motion from the values in the configuration file to LOWERLIM-UPPERLIM radians.

--scalemotiontodata Scale motion plots to the motion timecourse values rather than to the limit lines.

--componentlinewidth LINEWIDTH Override the component line width (in pixels) in the configuration file with LINEWIDTH.

--motionlinewidth LINEWIDTH Override the motion timecourse line widths (in pixels) in the configuration file with LINEWIDTH.

--motionlimitlinewidth LINEWIDTH Override the line widths of the motion limit lines (in pixels) in the configuration file with LINEWIDTH.

Debugging arguments:

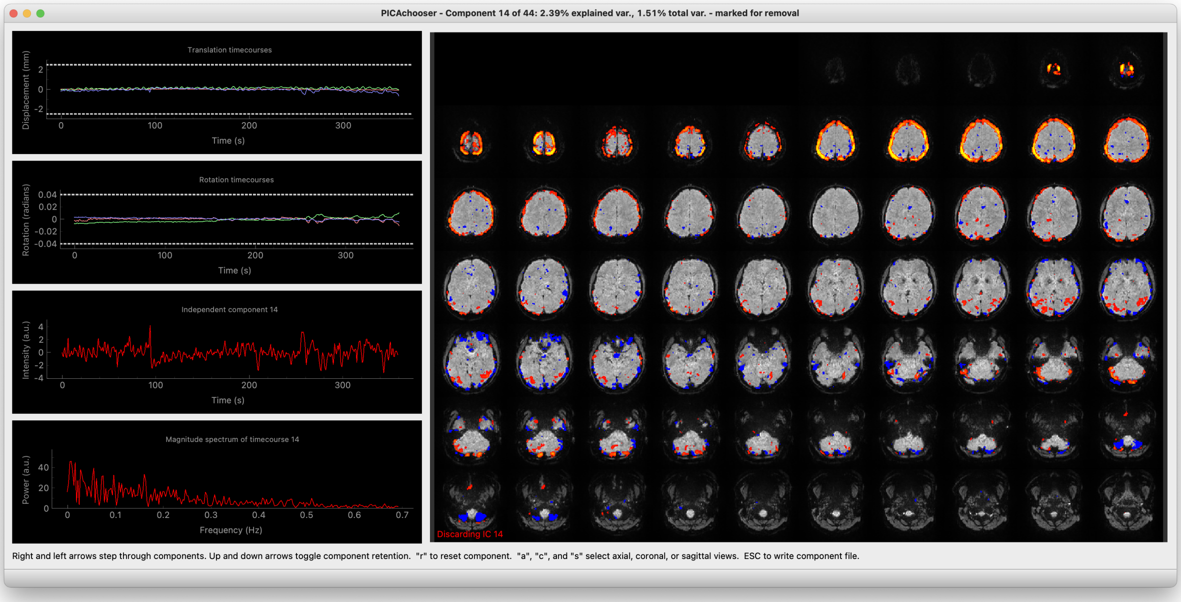
--verbose Output exhaustive amounts of information about the internal workings of PICAchooser. You almost certainly don’t want this.

1.10.2 Example

Run PICAchooser to look at a series of independent components and assign them a rating:

```
PICAchooser RUNMODE --featdir FEATDIRECTORY --melodicdir MELODICDIRECTORY
```

You'll then get a window that looks like this:



PICAchooser - Component 14 of 44: 2.39% explained var., 1.51% total var. - marked for removal

Right and left arrows step through components. Up and down arrows toggle component retention. "r" to reset component. "a", "c", and "s" select axial, coronal, or sagittal views. ESC to write component file.

Fig. 3: PICAchooser screenshot

1.10.3 Controls

To toggle whether the current component should be kept or discarded, press the up or down arrow key. You can change back and forth as much as you want. Components to be discarded are in red, ones to be kept are in green*.

To go to the next (or previous) component, press the right (or left) arrow. You'll wrap around if you hit the end.

Press the escape key at any time to save the current version of the component list. The component list is saved automatically when you quit.

1.10.4 Input file specification

For most datasets, you only need to specify the FEAT directory where the preprocessing was done, and the MELODIC directory where the ICA analysis was performed, and PICAchooser can find all the files it needs to let you do component selection. In some cases, however (looking at you, fmriprep), the files you need to find can be scattered all over the place, with different names (and even different formats). In those cases, you can specify the name and location of every one of the files separately (anything you set with these options will override the default locations calculated from the FEATDIR and MELODICDIR).

1.10.5 Other command line options

`--initfile` lets you read in a bad component file from anywhere to use as a starting point in your classification. It's the normal behavior in aroma mode (reading from MELODICDIR/../classified_motion_ICs.txt), but you can do it in any mode with this flag, and it will override the aroma classifications.

`--outputfile` lets you write the bad component file anywhere you want, rather than just the default location.

`--filteredfile` specifies where the filtered file would go. If this is set, PICAchooser prints the fsl_regfilt command to filter the data using the currently tagged bad components whenever the file is saved (when the escape key is pressed, or when you quit).

`--displaythresh` sets the z-threshold for the component maps.

`--spatialroi XMIN XMAX YMIN YMAX ZMIN ZMAX` lets you zoom in on a cubic ROI within the NIFTI dataset. Useful if you did a constrained ICA on a particular brain region. Set the MAX value to -1 to go to the maximum value for a given dimension. These are voxel indices, with 0 being the first element of each dimension.

1.10.6 * Configuration changes

You can use `--keepcolor`, `--discardcolor`, `--transmotlimits` and `--rotmotlimits` to alter display behavior for the current run (useful if you're using the docker container). To change things semi-permanently, edit the file ${HOME}/.picachooser.json. This file is created with default values if it is not present. You can use any valid python color specification string for color values, e.g. "r", "ff0000", or "FF0000" could all be used for red.

`--componentlinewidth`, `--motionlinewidth`, and `--motionlimitlinewidth` can all be used to specify various linewidths (in pixels) for the various plots. Useful if you want to make a screenshot pretty for a figure.

`--scalemotiontodata` autoscales the motion plots to the motion timecourse values rather than to fixed limits.

The motion plots have two dotted lines to indicate "normal" motion limits (by default +/-2.5 mm for translation and +/-0.04 radians for rotation). The locations of these lines are set by "transmotlimits" and "rotmotlimits" in the configuration file. Setting "motionplotstyle" to 0 will remove the lines, and fix the y range of the plots to the limit values. Set the limit line color using "motionlimitcolor".

1.10.7 Outputs

In melodic mode, the default output file is named "badcomponents.txt"; components flagged for removal will be written to MELODICDIR as comma separated integers. Component numbers start at 1 (for compatibility with fsl_regfilt).

In groupmelodic mode, the default output file is named "goodcomponents.txt"; components which are worth keeping will be written to MELODICDIR as integers, one per line. Component numbers start at 0 (for compatibility with standard NIFTI array indexing).

In aroma mode, the file "classified_motion_ICs.txt" must exist in the parent of MELODICDIR; by default the output will be written to "classified_motion_ICs_revised.txt" in the same directory. Component numbers start at 1 (for compatibility with AROMA numbering convention).

In fix mode, the default output file is named "hand_labels_noise.txt" and will be written to MELODICDIR as comma separated integers with square brackets surrounding the line. Component numbers start at 1 (for compatibility with FIX numbering convention).

1.10.8 Reprocessing fmriprep AROMA analyses

fmriprep reformats things to conform to BIDS standard naming conventions and formatting, so file locations, names, and formats are a little weird. However, you can check components as long as you used an external work directory (you set the "-w" flag during analysis).

A concrete example: I have an analysis in BIDSDIR, and used the option "-w WORKDIR" when I ran fmriprep (with AROMA processing enabled). Say I have a functional run, sub-015_ses-001_task-rest_run-1_bold.nii.gz that I want to redo the AROMA processing on. First off, I need to find my ICfile and IC mask file. They don't get copied into the derivatives directory, as they are intermediate files, not analysis products. It turns out the entire melodic directory does exist in the work directory. In this particular case, if I set:

```
--melodicdir ${WORKDIR}/fmriprep_wf/single_subject_015_wf/func_preproc_ses_001_task_rest_run_1_wf/
ica_aroma_wf/melodic
```

then PICAchooser can find the ICfile and ICmask.

The background file is also in this directory:

```
--backgroundfile ${WORKDIR}/fmriprep_wf/single_subject_015_wf/func_preproc_ses_001_task_rest_run_1_wf/
ica_aroma_wf/melodic/mean.nii.gz
```

Everything else can be found in the functional output directory for this session:

```
FUNCDIR=${BIDSDIR}/derivatives/fmriprep/sub-015/ses-001/func
```

By setting the following options:

```
--initfile ${FUNCDIR}/sub-015_ses-001_task-rest_run-1_AROMAnoiseICs.csv --funcfile
${FUNCDIR}/sub-015_ses-001_task-rest_run-1_space-MNI152NLin6Asym_desc-preproc_bold.nii.gz
--motionfile ${FUNCDIR}/sub-015_ses-001_task-rest_run-1_desc-confounds_regressors.tsv
```

As a bonus, if you also set:

```
--filteredfile ${FUNCDIR}/sub-015_ses-001_task-rest_run-1_space-MNI152NLin6Asym_desc-AROMAnonaggr_bold.
nii.gz
```

Then when you save your bad component file, you'll see the command necessary to refilter your data printed to the terminal window. I haven't investigated far enough to know when the smoothing implied in the name of the exisiting filtered file comes from, so there may be some other steps to get to exactly the output you'd get from fmriprep…

1.11 melodicomp

melodicomp handles a slightly different task - its job is to allow for rapid comparison of two melodic analyses. In this case, you want to match components between the two analyses (the chances that you'll get the same components in the same order between two analyses is basicaly zero, so you need to match them up). We do this with a spatial cross-correlation - higher cross-correlation means the the components look more like each other. This works surprisingly well. We then display the components side by side.

1.11.1 Usage

usage: melodicomp ICfile1 ICfile2 [options] melodicomp: error: the following arguments are required: ICfile1, ICfile2 usage: melodicomp ICfile1 ICfile2 [options]

A program to compare two sets of melodic components.

positional arguments:

ICfile1 The first IC component file. This will be
the exemplar, and for each component, the closest component in ICfile2 will be selected for comparison.

ICfile2 The second IC component file. Components in
this file will be selected to match components in ICfile1.

optional arguments:

-h, --help show this help message and exit

Nonstandard input file location specification. Setting these overrides the locations assumed from ICfile1.:

--backgroundfile BGFILE The anatomic file on which to display the ICs (by default assumes a file called 'mean.nii.gz' in the same directory as ICfile1.))

--maskfile ICMASK The independent component mask file produced by MELODIC (by default assumes a file called 'mask.nii.gz' in the same directory as ICfile1.)

--ICstatsfile1 STATSFILE The melodic stats file (by default called 'melodic_ICstats' in the same directory as ICfile1),

--ICstatsfile2 STATSFILE The melodic stats file (by default called 'melodic_ICstats' in the same directory as ICfile2),

Other arguments:

--corrthresh CORRTHRESH z threshold for the displayed ICA components. Default is 2.3.

--outputfile OUTPUTFILE Where to write the list of corresponding components (default is 'correspondingcomponents.txt' in the same directory as ICfile1

--sortedfile SORTEDFILE Save the components in ICfile2, sorted to match the components of ICfile1, in the file SORTEDFILE.

–spatialroi XMIN XMAX YMIN YMAX ZMIN ZMAX
Only read in image data within the specified ROI. Set MAX to -1 to go to the end of that dimension.

--displaythresh DISPLAYTHRESH z threshold for the displayed ICA components. Default is 2.3.

--label1 LABEL1 Label to give to file 1 components in display. Default is 'File 1'.

--label2 LABEL2 Label to give to file 2 components in display. Default is 'File 2'.

Debugging arguments:

--verbose Output exhaustive amounts of information about the internal workings of melodicomp. You almost certainly don't want this.

1.11.2 Controls

As with PICAchooser, this is all keyboard driven. Use the right and left arrows to step through components. In melodicomp, the first file specified on the command line is considered the reference file - we go through all the components of that file and display them on the left, and show the component from the second file that matches best on the right. The number of components in the files do NOT have to match (but their spatial dimensions, voxel sizes, and background images do). Using the up and down arrows toggles between sorting based on the native order of components in file 1, and sorting in descending order of cross-correlation coefficient. Use the “a”, “c”, and “s” keys to switch between axial, coronal, and sagittal views. “b” is for blink - this swaps the right and left images. It takes essentially no time, so it makes it very clear how and where the components are changing. Try it!

By default, pairs of components with correlation coefficients lower than 0.5 are considered poor matches, and are indicated with red text in the annotations. The correlation threshold can be set on the command line.

1.11.3 Output

On exit (or when you hit escape), melodicomp will output a text file with the component of the first melodic file , the matching component from the second file, and the correlation coefficient between them on each line. Component numbers start from 0. The order of lines in the file is the same as the current sort order in the GUI.

1.12 rtgrader

rtgrader is my attempt at letting you quickly run through a bunch of rapidtide analyses and see which ones worked and which did not. This requires you to have a pretty good sense of what you’re looking for (which, as far as I can tell, means you have to be me, but whatever. I wrote it for me).

1.12.1 Usage

usage: rtgrader lagtimes strengths labels [options]

A program to review (and rate) rapidtide analyses.

positional arguments:

lagtimes A 4D NIFTI file of concatenated maxtime_map.nii.gz
files from a group of rapidtide analyses. These obviously all have to be at the same resolution, orentation, and coordinate system (e.g. MNI152NLin6Asm).

strengths A 4D NIFTI file of concatenated maxcorr_map.nii.gz
files from a group of rapidtide analyses. These have to match the lagtimes file in resolution, orientation, coordinate system, and order.

labels A text file, one entry per line, of unique descriptions
of the datasets in the lagtimes and strengths files so you know which dataset is which.

optional arguments:

-h, --help	show this help message and exit

Other arguments:

--startindex INDEX The index to start at (0-based).

--mapmask MASKFILE The 3D or 4D NIFTI to mask all maps.

–spatialroi XMIN XMAX YMIN YMAX ZMIN ZMAX
Only read in image data within the specified ROI. Set MAX to -1 to go to the end of that dimension.

--outputroot OUTPUTROOT Root name for reading and writing the lists of good and bad component indices (default is 'rtgrader', resulting in 'rtgrader_goodindices.txt' and 'rtgrader_badindices.txt')in the current working directory.

–lagrange MINLAG MAXLAG
Lag time range to display. Default is -5.0 to 10.0.

–strengthrange MINSTRENGTH MAXSTRENGTH
Strength range to display. Default is 0.0 to 0.75.

Miscellaneous arguments:

--version show program's version number and exit

--detailedversion show program's version number and exit

Debugging arguments:

--verbose Output exhaustive amounts of information about the internal workings of rtgrader. You almost certainly don't want this.

1.13 Contributing to PICAchooser

This document explains how to set up a development environment for contributing to PICAchooser and code style conventions we follow within the project. For a more general guide to PICAchooser development, please see our contributing guide. Please also remember to follow our code of conduct.

1.13.1 Style Guide

Code

Docstrings should follow numpydoc convention. We encourage extensive documentation.

The code itself should follow PEP8 convention as much as possible, with at most about 500 lines of code (not including docstrings) per file*.

* obviously some of the existing files don't conform to this - working on it…

Pull Requests

We encourage the use of standardized tags for categorizing pull requests. When opening a pull request, please use one of the following prefixes:

- **[ENH]** for enhancements
- **[FIX]** for bug fixes
- **[TST]** for new or updated tests
- **[DOC]** for new or updated documentation
- **[STY]** for stylistic changes
- **[REF]** for refactoring existing code

Pull requests should be submitted early and often! If your pull request is not yet ready to be merged, please also include the **[WIP]** prefix. This tells the development team that your pull request is a "work-in-progress", and that you plan to continue working on it.

CHAPTER TWO

INDICES AND TABLES

- genindex
- search