rapidtide Documentation

Release 2.8.7

Blaise Frederick

Apr 25, 2024

INTRODUCTION:

Rapidtide is a suite of python programs used to perform rapid time delay analysis on functional imaging data to find time lagged correlations between the voxelwise time series and other time series, both in the LFO band (rapditide2) and now in the cardiac band (happy).

CHAPTER

ONE

CITING RAPIDTIDE

Frederick, B, rapidtide [Computer Software] (2016-2024). Available from https://github.com/bbfrederick/rapidtide. doi:10.5281/zenodo.814990

CHAPTER

TWO

CONTENTS

2.1 The rapidtide package

Rapidtide is a suite of Python programs used to model, characterize, visualize, and remove time varying, physiological blood signals from fMRI and fNIRS datasets. The primary workhorses of the package are the rapidtide program, which characterizes bulk blood flow, and happy, which focusses on the cardiac band.

Full documentation is at: http://rapidtide.readthedocs.io/en/latest/

2.1.1 The rapidtide program

Rapidtide is also the name of the first program in the package, which is used to perform rapid time delay analysis on functional imaging data to find time lagged correlations between the voxelwise time series and other time series, primarily in the LFO band.

Why do I want to know about time lagged correlations?

This comes out of work by our group (The Opto-Magnetic group at McLean Hospital - http://www.nirs-fmri.net) looking at the correlations between neuroimaging data (fMRI) and NIRS data recorded simultaneously, either in the brain or the periphery. We found that a large fraction of the "noise" we found at low frequency in fMRI data was due to real, random[*] fluctuations of blood oxygenation and volume (both of which affect the intensity of BOLD fMRI images) in the blood passing through the brain. More interestingly, because these characteristics of blood move with the blood itself, this gives you a way to determine blood arrival time at any location in the brain. This is interesting in and of itself, but also, this gives you a method for optimally modelling (and removing) in band physiological noise from fMRI data (see references below).

After working with this for several years we've also found that you don't need to used simultaneous NIRS to find this blood borne signal - you can get it from blood rich BOLD voxels for example in the superior sagittal sinus, or bootstrap it out of the global mean signal in the BOLD data. You can also track exogenously applied waveforms, such as hypercarbic and/or hyperoxic gas challenges to really boost your signal to noise. So there are lots of times when you might want to do this type of correlation analysis.

As an aside, some of these tools are just generally useful for looking at correlations between timecourses from other sources – for example doing PPI, or even some seed based analyses.

[*] "random" in this context means "determined by something we don't have any information about" - maybe EtCO2 variation, or sympathetic nervous system activity - so not really random.

Correlation analysis is easy - why use this package?

The simple answer is "correlation analysis is easy, but using a prewritten package that handles file I/O, filtering, resampling, windowing, and the rest for you is even easier". A slightly more complex answer is that while correlation analysis is pretty easy to do, it's hard to do right; there are lots and lots of ways to do it incorrectly. Fortunately, I've made most of those mistakes for you over the last 8 years, and corrected my code accordingly. So rather than repeat my boring mistakes, why not make new, interesting mistakes? Explore your own, unique chunk of wrongspace…

2.1.2 Happy

More recently, inspired by Henning Voss' paper on hypersampling of cardiac signals in fMRI, we developed a method to extract and clean cardiac waveforms from fMRI data, even when the fMRI TR is far too long to properly sample cardiac waveforms. This cardiac waveform can then be to track the pulsatile cardiac pressure wave through the brain in somewhat the same way that we track the LFO signals. Among other things, this allows you to get cardiac waveforms during scans even when either 1) you didn't use a plethysmograph, or 2) you did, but the recording was of poor quality, which happens more than you might think.

What does "happy" have to do with any of this?

As to why happy is part of rapidtide, that's partially for practical reasons (the libraries in rapidtide have an awful lot of code that is reused in happy), and partially thematically - rapidtide has evolved from a "let's look at low frequency signals in fMRI data" package to a "let's look at everything in fMRI data EXCEPT neuronal activation", so happy fits right in.

2.2 Why are you releasing this package?

For a number of reasons.

- I want people to use it! I think if it were easier for people to do time delay analysis, they'd be more likely to do it. I don't have enough time or people in my group to do every experiment that I think would be interesting, so I'm hoping other people will, so I can read their papers and learn interesting things.
- It's the right way to do science – I can say lots of things, but if nobody can replicate my results, nobody will believe it (we've gotten that a lot, because some of the implications of what we've seen in resting state data can be a little uncomfortable). We've reached a stage in fMRI where getting from data to results involves a huge amount of processing, so part of confirming results involves being able to see how the data were processed. If you had to do everything from scratch, you'd never even try to confirm anybody's results.
- In any complicated processing scheme, it's quite possible (or in my case, likely) to make dumb mistakes, either coding errors or conceptual errors, and I almost certainly have made some (although hopefully the worst ones have been dealt with at this point). More users and more eyes on the code make it more likely that they will be found. As much as I'm queasy about somebody potentially finding a mistake in my code, I'd rather that they did so, so I can fix it[‡].
- It's giving back to the community. I benefit from the generosity of a lot of authors who have made the open source tools I use for work and play, so I figure I can pony up too.

[‡] or better yet, you, empowered user, can fix it, and push back a fix that benefits everybody…

2.3 Stability, etc.

This is an evolving code base. I'm constantly tinkering with it. That said, now that I've sent this off into 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, although for the most part this will be restricted to the dev branch. **It's very possible that at any given time the dev branch will be very broken, so stay away from it unless you have a good reason to be using it.** 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. That said, my tests routinely fail, even when things actually work. Probably should deal with that. Check back often for exciting new features and bug fixes!

2.4 Python version compatibility

I switched over a while ago to using Python 3 as my daily driver, so I know that everything works there. However, I know that a lot of people can't or won't switch from Python 2x, so I kept Python 2.7 compatibility for quite some time.

That said, the writing is on the wall, and since I depend on a number of packages that have dropped Python 2.x support, as of 2.0, so has rapidtide. However, as of version 1.9.0 I'm also releasing the code in a docker container (frederick-lab/rapidtide), which has everything nicely installed in a fully configured Python 3 environment, so there's really no need for me continue 2.x support. So now it's f-strings all the way, kids!

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

- **rapidtide** - This is the heart of the package - this is the workhorse program that will determine the time lagged correlations between all the voxels in a NIFTI file and a temporal "probe" regressor (which can come from a number of places, including the data itself) - it rapidly determines time delays… There are a truly bewildering array of options, and just about everything can be adjusted, however I've tried to pick a good set of default options for the most basic processing to get you going. At a minimum, it requires a 4D NIFTI file as input, and a root name for all of the output files. It generates a number of 3D NIFTI file maps of various parameters (lag time of maximum correlation, maximum correlation value, a mask of which voxels have valid fits, etc.) and some text files with useful information (significance thresholds, processing timing information, a list of values of configurable options).
- **happy** - This is a companion to rapidtide that focusses on cardiac signals. happy does three things - it attempts to determine the cardiac waveform over the time course of an fMRI dataset using slice selective averaging of fully unprocessed fMRI data. It also cleans up this initial estimate using a deep learning filter to infer what the simultaneously recorded plethysmogram would be. Finally, it uses either the derived or a supplied plethysmogram signal to construct a cardiac pulsation map over a single cycle of the cardiac waveform, a la Voss.
- **showxcorrx** - Like rapidtide, but for single time courses. Takes two text files as input, calculates and displays the time lagged cross correlation between them, fits the maximum time lag, and estimates the significance of the correlation. It has a range of filtering, windowing, and correlation options.
- **rapidtide2x_legacy**, **happy_legacy**, **showxcorr_legacy** - The older versions of the similarly named programs. These use the old calling conventions, for compatibility with older workflows. These will go away eventually, and they don't really get updates or bugfixes, so if you're using them, change to the new ones, and if you're not using them, don't.
- **rapidtide2std** - This is a utility for registering rapidtide output maps to standard coordinates. It's usually much faster to run rapidtide in native space then transform afterwards to MNI152 space. NB: this will only work if you have a working FSL installation.
- **happy2std** - Guess.

- **showtc** - A very simple command line utility that takes timecourses from text files and plots the data in it in a matplotlib window. That's it. A good tool for quickly seeing what's in a file. Has a number of options to make the plot prettier.
- **showxy** - Another simple command line utility that displays the the data contained in text files containing whitespace separated x-y pairs.
- **showhist** - Another simple command line utility that displays the histograms generated by rapidtide.
- **resamp1tc** - takes an input text file at some sample rate and outputs a text file resampled to the specified sample rate.
- **resamplenifti** - takes an input nifti file at some TR and outputs a nifti file resampled to the specified TR.
- **tidepool** - This is a GUI tool for displaying all of the various maps and timecourses generated by rapidtide in one place, overlaid on an anatomic image. This makes it a bit easier to see how all the maps are related to one another, how the probe regressor evolves over the run, and the effect of the filtering parameters. To use it, launch tidepool from the command line, and then select a lag time map - tidepool will figure out the root name and pull in all of the other associated data. Works in native or standard space.

2.6 Financial Support

This code base is being developed and supported by grants from the US NIH (1R01 NS097512, RF1 MH130637-01)

2.7 Additional packages used

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

2.7.1 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://doi.org/10.1109/MCSE.2011.37

2.7.2 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

2.7.3 matplotlib:

1) John D. Hunter. Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95 (2007) | https://doi.org/10.1109/MCSE.2007.55

2.7.4 nibabel:

1) https://github.com/nipy/nibabel | https://doi.org/10.5281/zenodo.591597

2.7.5 scikit-learn:

1) 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., and Duchesnay, E., Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 2011. 12: p. 2825-2830. | https://scikit-learn.org

2.7.6 pandas:

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

2.7.7 nilearn:

1) https://github.com/nilearn/nilearn

2.8 References

Links to PDFs of all papers mentioned here can be found on the OMG website: https://www.nirs-fmri.net/home/publications

2.8.1 General overview of systemic low frequency oscillations in fMRI data

1) Tong Y, Hocke LM, Frederick BB. (2019) Low Frequency Systemic Hemodynamic "Noise" in Resting State BOLD fMRI: Characteristics, Causes, Implications, Mitigation Strategies, and Applications. Front. Neurosci., 14 August 2019 | https://doi.org/10.3389/fnins.2019.00787

2.8.2 Multimodal Cerebral Circulation Imaging

1) Tong Y, Frederick BD. (2010) Time lag dependent multimodal processing of concurrent fMRI and near-infrared spectroscopy (NIRS) data suggests a global circulatory origin for low-frequency oscillation signals in human brain. Neuroimage, 53(2), 553-64.

2) Tong Y, Hocke L, Frederick BD. (2011) Isolating the sources of widespread physiological fluctuations in fNIRS signals. J Biomed Opt. 16(10), 106005.

3) Tong Y, Bergethon PR, Frederick BD. (2011c) An improved method for mapping cerebrovascular reserve using concurrent fMRI and near-infrared spectroscopy with Regressor Interpolation at Progressive Time Delays (RIPTiDe). Neuroimage, 56(4), 2047-2057.

4) Tong Y, Frederick BD. (2012) Concurrent fNIRS and fMRI processing allows independent visualization of the propagation of pressure waves and bulk blood flow in the cerebral vasculature. Neuroimage, Jul 16;61(4): 1419-27.

5) Tong Y, Hocke LM, Licata SC, Frederick BD. (2012) Low frequency oscillations measured in the periphery with near infrared spectroscopy (NIRS) are strongly correlated with blood oxygen level-dependent functional magnetic resonance imaging (BOLD fMRI) signals. J Biomed Opt, 2012;17(10):106004. doi: 10.1117/1.JBO.17.10.106004. PubMed PMID: 23224003; PMCID: 3461094.

6) Tong Y, Hocke LM, Frederick BD. (2013) Short repetition time multiband EPI with simultaneous pulse recording allows dynamic imaging of the cardiac pulsation signal. Magn Reson Med 2014;72(5):1268-76. Epub Nov 22, 2013. doi: 10.1002/mrm.25041. PubMed PMID: 24272768.

7) Tong Y, Frederick B. (2014) Studying the Spatial Distribution of Physiological Effects on BOLD Signals using Ultrafast fMRI. Front Hum Neurosci 2014;5(196). doi: doi: 10.3389/fnhum.2014.00196.

8) Tong Y, Frederick B. (2014) Tracking cerebral blood flow in BOLD fMRI using recursively generated regressors. Hum Brain Mapp. 2014;35(11):5471-85. doi: 10.1002/hbm.22564. PubMed PMID: 24954380; PMCID: PMC4206590.

9) Donahue M, Strother M, Lindsey K, Hocke L, Tong Y, Frederick B. (2015) Time delay processing of hypercapnic fMRI allows quantitative parameterization of cerebrovascular reactivity and blood flow delays. Journal of Cerebral Blood Flow & Metabolism. 2015. PubMed PMID: 26661192. Epub October 19, 2015. doi: 10.1177/0271678X15608643.

10) Hocke L, Cayetano K, Tong Y, Frederick B. (2015) An optimized multimodal fMRI/NIRS probe for ultra-high resolution mapping. Neurophotonics. 2(4), 045004 (Oct-Dec 2015). doi: 10.1117/1.NPh.2.4.0450004.

11) Tong Y, Hocke LM, Fan X, Janes AC, Frederick B (2015). Can apparent resting state connectivity arise from systemic fluctuations? Frontiers in human neuroscience. 2015;9. doi: 10.3389/fnhum.2015.00285.

12) Tong Y, Lindsey KP, Hocke LM, Vitaliano G, Mintzopoulos D, Frederick B. (2016) Perfusion information extracted from resting state functional magnetic resonance imaging. Journal of cerebral blood flow and metabolism : official journal of the International Society of Cerebral Blood Flow and Metabolism. 2016. doi: 10.1177/0271678X16631755. PubMed PMID: 26873885.

2.8.3 Cardiac waveform extraction and refinement

1) Aslan S, Hocke L, Schwarz N, Frederick B. (2019) Extraction of the cardiac waveform from simultaneous multislice fMRI data using slice sorted averaging and a deep learning reconstruction filter. NeuroImage 198, 303–316 (2019).

2.8.4 Physiological noise identification and removal using time delay methods

1) Tong Y, Lindsey KP, Frederick BD. (2011b) Partitioning of physiological noise signals in the brain with concurrent near-infrared spectroscopy (NIRS) and fMRI. J Cereb Blood Flow Metab. 31(12), 2352-62.

2) Frederick BD, Nickerson LD, Tong Y. (2012) Physiological denoising of BOLD fMRI data using Regressor Interpolation at Progressive Time Delays (RIPTiDe) processing of concurrent fMRI and near-infrared spectroscopy (NIRS). Neuroimage, Apr 15;60(3): 1419-27.

3) Tong Y, Hocke LM, Nickerson LD, Licata SC, Lindsey KP, Frederick BB (2013) Evaluating the effects of systemic low frequency oscillations measured in the periphery on the independent component analysis results of resting state networks. NeuroImage. 2013;76C:202-15. doi: 10.1016/j.neuroimage.2013.03.019. PubMed PMID: 23523805; PMCID: PMC3652630.

4) Hocke LM, Tong Y, Lindsey KP, Frederick BB (2016). Comparison of peripheral near-infrared spectroscopy low-frequency oscillations to other denoising methods in resting state functional MRI with ultrahigh temporal

resolution. Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine. 2016. | http://dx.doi.org/10.1002/mrm.26038. PubMed PMID: 26854203.

5) Erdoğan S, Tong Y, Hocke L, Lindsey K, Frederick B (2016). Correcting resting state fMRI-BOLD signals for blood arrival time enhances functional connectivity analysis. Front. Hum. Neurosci., 28 June 2016 | http://dx.doi.org/10.3389/fnhum.2016.00311

6) Tong, Y, Hocke, LM, and Frederick, BB, Low Frequency Systemic Hemodynamic "Noise" in Resting State BOLD fMRI: Characteristics, Causes, Implications, Mitigation Strategies, and Applications. Front Neurosci, 2019. 13: p. 787. | http://dx.doi.org/10.3389/fnins.2019.00787

2.9 Bare metal installation

This gives you the maximum flexibility if you want to look at the code and/or modify things. It may seem a little daunting at first, but it's not that bad. And if you want a simpler path, skip down to the Docker installation instructions

2.9.1 Required dependencies

The processing programs in rapidtide require the following to be installed first:

- Python >= 3.8
- numpy
- scipy
- pandas
- pyarrow
- scikit-image
- scikit-learn
- nibabel
- nilearn
- matplotlib
- pyqt5-sip
- pyqtgraph
- statsmodels
- tqdm

2.9.2 Optional dependencies

The following optional dependencies will be used if present:

- numba (for faster performance)
- pyfftw (faster performance)
- mkl and mkl-service (faster performance on intel CPUs)

If you want to use the deep learning filter in happy, you'll need Keras and some sort of backend. If you want to be able to train filters, you'll probably want GPU support. This is currently an annoying, non-trivial thing to set up, especially on a Mac, which is where I do things, because Apple and Nvidia aren't friends at the moment. If you are on a linux box (or maybe Windows - haven't tested), WITH an Nvidia GPU, install:

- keras
- tensorflow-gpu (This assumes you have all the necessary CUDA libraries. Making this all work together properly is a version dependent moving target. Ask The Google the best way to do it this week - anything I say here will probably be obsolete by the time you read this.)

If you are on linux (or Windows) WITHOUT an Nvidia GPU, install:

- keras
- tensorflow (and make sure it doesn't sneakily try to install the GPU version - that won't work)

If you are on a Mac, you almost certainly have a non-Nvidia GPU, so you should install

- plaidml-keras (it installs Keras and uses PlaidML as the backend rather than tensorflow). You will have to run a configuration step in plaidML to tell it what GPU to use and how. I use the "metal" option with the AMD GPU in my laptop - that seems to be the most stable. Currently, I think you have you have to do this from pypi - I haven't seen a conda version of this.

2.9.3 Installing Python

The simplest way BY FAR 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 many of the dependencies for rapidtide installed by default. You can get it here: https://www.continuum.io. Rapidtide works with Python 3.8 or greater.

After installing Anaconda python, install the remaining dependencies (including some good optional ones:

```
conda install nibabel pyqtgraph pyfftw
```

For the deep learning filter in happy, also do:

```
conda install keras tensorflow-gpu
```

(for Linux or Windows WITH Nvidia GPU)

or:

```
conda install keras tensorflow
```

(for Linux or Windows WITHOUT Nvidia GPU)

or

```
pip install plaidml-keras
```

(on a Mac)

Done.

2.9.4 Installing the rapidtide library

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).

2.9.5 Updating

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

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

2.10 Docker installation

As of 1.9.0, there is now a Docker container with a full rapidtide installation. To use this, first make sure you have docker installed and properly configured, then run the following:

```
docker pull fredericklab/rapidtide:latest-release
```

This will download the docker container from dockerhub. It's around a 3GB download, 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 "latest-release" with the version of the container you want.

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

```
docker pull fredericklab/rapidtide:latest
```

This will use the most recent version on dockerhub, which is built automatically on every git push. NOTE: I don't advise doing this unless you're helping debug something - there's no guarantee that "latest" is functional at any given time.

Now that the file is downloaded, you can run and rapidtide command in the Docker container. For example, to run a simple rapidtide analysis, 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 its own self-contained 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.

```
docker run \
    --mount type=bind,source=INPUTDIRECTORY,destination=/data_in \
    --mount type=bind,source=OUTPUTDIRECTORY,destination=/data_out \
    fredericklab/rapidtide:latest-release \
        rapidtide \
            /data_in/YOURNIFTIFILE.nii.gz \
```

(continues on next page)

(continued from previous page)

```
            /data_out/outputname \
            --filterband lfo \
            --searchrange -15 15 \
            --passes 3
```

NOTE: If you want to run this on the test data, like the examples above for the bare metal installation, the example data is in the Docker container in the /src/rapidtide/rapidtide/data/examples/src directory. So to run the first example, you could just do:

```
docker run \
    --mount type=bind,source=OUTPUTDIRECTORY,destination=/data_out \
    fredericklab/rapidtide:latest-release \
        rapidtide \
            /src/rapidtide/rapidtide/data/examples/src/sub-RAPIDTIDETEST.nii.gz \
            /data_out/dgsr \
            --filterband lfo \
            --searchrange -15 15 \
            --passes 3
```

You can replace the `rapidtide blah blah blah` command with any program in the package - after the fredericklab/rapidtide:latest-release, just specify the command and arguments as you usually would. If you're running a program that displays anything, 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 the following command will work (you can replace 'tidepool' with any of the rapidtide commands that put up windows):

```
docker run \
    --network host\
    --volume=INPUTDIRECTORY:/data_in,OUTPUTDIRECTORY:/data_out \
    -it \
    -e DISPLAY=MYIPADDRESS:0 \
    -u rapidtide \
    fredericklab/rapidtide:latest-release \
        tidepool
```

2.11 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/rapidtide.simg docker://fredericklab/rapidtide:latest-
↪release
```

Running the container is similar to Docker. The "-B" option is used to bind filesystems to mountpoints in the container. For example, to run the simple rapidtide2x analysis above, type the following:

```
singularity run \
    --cleanenv \
    -B INPUTDIRECTORY:/data_in,OUTPUTDIRECTORY:/data_out \
    rapidtide.simg \
        rapidtide \
            /data_in/YOURNIFTIFILE.nii.gz \
            /data_out/outputname \
            --filterband lfo \
            --searchrange -15 15 \
            --passes 3
```

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/zsh)
```

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

2.12 References

1) Erdoğan S, Tong Y, Hocke L, Lindsey K, Frederick B (2016). Correcting resting state fMRI-BOLD signals for blood arrival time enhances functional connectivity analysis. Front. Hum. Neurosci., 28 June 2016 | http://dx.doi.org/10.3389/fnhum.2016.00311

2.13 Support and communication

All bugs, concerns and enhancement requests for this software can be submitted here: https://github.com/bbfrederick/rapidtide/issues.

If you would like to ask a question about usage or rapidtide's outputs, please submit a question to NeuroStars with the rapidtide tag.

We will also attempt to archive certain common questions and associated answers in a Frequently Asked Questions (FAQ) page.

For more information about how the rapidtide library can be used, please see the API page. Common rapidtide workflows can also be called from the command line.

2.14 General points

Before talking about the individual programs, in the 2.0 release and going forward, I've tried to adhere to some common principals, across all program, to make them easier to understand and maintain, and more interoperable with other programs, and to simplify using the outputs.

NB: All commands are shown using backslashes as line continuation characters for clarity to make the commands easier to read. These aren't needed - you can just put all the options on the same line, in any order.

2.14.1 Standardization of interfaces

Wherever possible, I've tried to harmonize the argument parsers between all of the programs in the package. That means I have some standard argument groups, such as ones pertaining to filtering, correlation, correlation fitting, sample rate specification, etc., so if you know how to use one of the programs, that experience should carry over - argument names should be the same, arguments should interact in the same way. Given the wide range of years I've been working on this package, that's not always fully implemented, but I do try to keep all the interfaces up to date as much as possible.

This also means that I have tried to have pretty standard ways of reading and writing data files in terms of formats and naming - every program should read a wide variety of text input types using the same method. All NIFTI file reads and writes are through a standard nibabel interface and should follow pretty standard conventions.

2.14.2 BIDS Outputs

By default, all inputs and outputs are in BIDS compatible formats (this is most true for rapidtide and happy, which get the majority of the work, but the goal is to eventually make all the programs in the package conform to this). The two major ramifications of this are that I have tried to follow BIDS naming conventions for NIFTI, json, and text files containing time series. Also, all text files are by default BIDS continuous timeseries files - data is in compressed, tab separated column format (.tsv.gz), with the column names, sample rate, and start time, in the accompanying .json sidecar file.

2.14.3 Text Inputs

A side effect of moving to BIDS is that I've now made a standardized interface for reading text data into programs in the package to handle many different types of file. In general, now if you are asked for a timeseries, you can supply it in any of the following ways:

A plain text file with one or more columns.

You can specify any subset of columns in any order by adding ":colspec" to the end of the filename. "colspec" is a column specification consisting of one or more comma separated "column ranges". A "column range" is either a single column number or a hyphen separated minimum and maximum column number. The first column in a file is column 0.

For example specifying, "mytextfile.txt:5-6,2,0,10-12"

would return an array containing all the timepoints from columns 5, 6, 2, 0, 10, 11, and 12 from mytextfile.txt, in that order. Not specifying ":colspec" returns all the columns in the file, in order.

If the program in question requires the actual sample rate, this can be specified using the `--samplerate` or `--sampletime` flags. Otherwise 1.0Hz is assumed.

A BIDS continuous file with one or more columns.

BIDS files have names for each column, so these are used in column specification. For these files, "colspec" is a comma separated list of one or more column names:

"thefile_desc-interestingtimeseries_physio.json:cardiac,respiration"

would return the two named columns "cardiac" and "respiration" from the accompanying .tsv.gz file. Not specifying ":colspec" returns all the columns in the file, in order.

Because BIDS continuous files require sample rate and start time to be specified in the sidecar file, these quantities will now already be set. Using the `--samplerate`, `--sampletime` or `--starttime` flags will override any header values, if specified.

2.14.4 Visualizing files

Any output NIFTI file can be visualized in your favorite NIFTI viewer. I like FSLeyes, part of FSL. It's flexible and fast, and has lots of options for displaying 3 and 4D NIFTI files.

While there may be nice, general graphing tools for BIDS timeseries files, I wrote "showtc" many years ago, a matplotlib based file viewer with lots of nice tweaks to make pretty and informative graphs of various rapidtide input and output time series files. It's part of rapidtide, and pretty easy to learn. Just type `showtc --help` to get the options.

As an example, after running happy, if you want to see the derived cardiac waveform, you'd run:

```
showtc \
    happytest_desc-slicerescardfromfmri_timeseries.json:cardiacfromfmri,cardiacfromfmri_
↪dlfiltered \
    --format separate
```

There are some companion programs - `showxy` works on 2D (x, y) data; `showhist` is specificaly for viewing histograms (`_hist` files) generated by several programs in the package, `spectrogram` generates and displays spectrograms of time series data. Each of these is separately documented below.

2.15 rapidtide

2.15.1 Description:

The central program in this package is rapidtide. This is the program that calculates a similarity function between a "probe" signal and every voxel of a BOLD fMRI dataset. It then determines the peak value, time delay, and wi dth of the similarity function to determine when and how strongly that probe signal appears in each voxel.

At its core, rapidtide is simply performing a full crosscorrelation between a "probe" timecourse and every voxel in an fMRI dataset (by "full" I mean over a range of time lags that account for any delays between the signals, rather than only at zero lag, as in a Pearson correlation). As with many things, however, the devil is in the details, and so rapidtide provides a number of features which make it pretty good at this particular task. A few highlights:

- There are lots of ways to do something even as simple as a cross-correlation in a nonoptimal way (not windowing, improper normalization, doing it in the time rather than frequency domain, etc.). I'm pretty sure what rapidtide does by default is, if not the best way, at least a very good and very fast way.
- rapidtide has been optimized and profiled to speed it up quite a bit; it has an optional dependency on numba – if it's installed, some of the most heavily used routines will speed up significantly due to judicious use of @jit.
- The sample rate of your probe regressor and the fMRI data do not have to match - rapidtide resamples the probe regressor to an integral multiple of the fMRI data rate automatically.
- The probe and data can be temporally prefiltered to the LFO, respiratory, or cardiac frequency band with a command line switch, or you can specify any low, high, or bandpass range you want.
- The data can be spatially smoothed at runtime (so you don't have to keep smoothed versions of big datasets around). This is quite fast, so no reason not to do it this way.
- rapidtide can generate a probe regressor from the global mean of the data itself - no externally recorded timecourse is required. Optionally you can input both a mask of regions that you want to be included in the mean, and the voxels that you want excluded from the mean (there are situations when you might want to do one or the other or both).
- Determining the significance threshold for filtered correlations where the optimal delay has been selected is nontrivial; using the conventional formulae for the significance of a correlation leads to wildly inflated p values. rapidtide estimates the spurious correlation threshold by calculating the distribution of null correlation values obtained with a shuffling procedure at the beginning of each run (the default is to use 10000 shuffled correlations), and uses this value to mask the correlation maps it calculates. As of version 0.1.2 it will also handle two-tailed significance, which you need when using bipolar mode.
- rapidtide can do an iterative refinement of the probe regressor by aligning the voxel timecourses in time and regenerating the test regressor.
- rapidtide fits the peak of the correlation function, so you can make fine grained distinctions between close lag times. The resolution of the time lag discrimination is set by the length of the timecourse, not the timestep – this is a feature of correlations, not rapidtide.
- Once the time delay in each voxel has been found, rapidtide outputs a 4D file of delayed probe regressors for using as voxel specific confound regressors or to estimate the strength of the probe regressor in each voxel. This regression is performed by default, but these outputs let you do it yourself if you are so inclined.
- I've put a lot of effort into making the outputs as informative as possible - lots of useful maps, histograms, timecourses, etc.
- There are a lot of tuning parameters you can mess with if you feel the need. I've tried to make intelligent defaults so things will work well out of the box, but you have the ability to set most of the interesting parameters yourself.

2.15.2 Inputs:

At a minimum, rapidtide needs a data file to work on (space by time), which is generally thought to be a BOLD fMRI data file. This can be Nifti1 or Nifti2 (for fMRI data, in which case it is time by up to 3 spatial dimensions) or a whitespace separated text file (for NIRS data, each column is a time course, each row a separate channel); I can currently read (probably) but not write Cifti files, so if you want to use grayordinate files you need to convert them to nifti2 in workbench, run rapidtide, then convert back. As soon as nibabel finishes their Cifti support (EDIT: and I get around to figuring it out), I'll add that.

The file needs one time dimension and at least one spatial dimension. Internally, the array is flattened to a time by voxel array for simplicity.

The file you input here should be the result of any preprocessing you intend to do. The expectation is that rapidtide will be run as the last preprocessing step before resting state or task based analysis. So any slice time correction, motion correction, spike removal, etc. should already have been done. If you use FSL, this means that if you've run preprocessing, you would use the filtered_func_data.nii.gz file as input. Temporal and spatial filtering are the two (partial) exceptions here. Generally rapidtide is most useful for looking at low frequency oscillations, so when you run it, you usually use the `--filterband lfo` option or some other to limit the analysis to the detection and removal of low frequency systemic physiological oscillations. So rapidtide will generally apply it's own temporal filtering on top of whatever you do in preprocessing. Also, you have the option of doing spatial smoothing in rapidtide to boost the SNR of the analysis; the hemodynamic signals rapidtide looks for are often very smooth, so you rather than smooth your functional data excessively, you can do it within rapidtide so that only the hemodynamic data is smoothed at that level.

2.15.3 Outputs:

Outputs are space or space by time NIFTI or text files, depending on what the input data file was, and some text files containing textual information, histograms, or numbers. File formats and naming follow BIDS conventions for derivative data for fMRI input data. 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.

2.15.4 BIDS Outputs:

Name	Extension(s)	Content	When present
XXX_maxtime_map	.nii.gz, .json	Time of offset of the maximum of the similarity function	Always
XXX_desc-maxtime_hist	.tsv, .json	Histogram of the maxtime map	Always
XXX_maxcorr_map	.nii.gz, .json	Maximum similarity function value (usually the correlation coefficient, R)	Always
XXX_desc-maxcorr_hist	.tsv, .json	Histogram of the maxcorr map	Always
XXX_maxcorrsq_map	.nii.gz, .json	Maximum similarity function value, squared	Always
XXX_desc-maxcorrsq_hist	.tsv, .json	Histogram of the maxcorrsq map	Always
XXX_maxwidth_map	.nii.gz, .json	Width of the maximum of the similarity function	Always
XXX_desc-maxwidth_hist	.tsv, .json	Histogram of the maxwidth map	Always

continues on next page

Table 1 – continued from previous page

Name	Extension(s)	Content	When present
XXX_MTT_map	.nii.gz, .json	Mean transit time (estimated)	Always
XXX_corrfit_mask	.nii.gz	Mask showing where the similarity function fit succeeded	Always
XXX_corrfitfailreason_	.nii.gz, .json	A numerical code giving the reason a peak could not be found (0 if fit succeeded)	Always
XXX_desc-corrfitwindow_info	.nii.gz	Values used for correlation peak fitting	Always
XXX_desc-runoptions_info	.json	A detailed dump of all internal variables in the program. Useful for debugging and data provenance	Always
XXX_desc-lfofilterCleaned_bold	.nii.gz, .json	Filtered BOLD dataset after removing moving regressor	If GLM filtering is enabled (default)
XXX_desc-lfofilterRemoved_bold	.nii.gz, .json	Scaled, voxelwise delayed moving regressor that has been removed from the dataset	If GLM filtering is enabled (default) and `--nolimitoutput` is selected
XXX_desc-lfofilterEVs_bold	.nii.gz, .json	Voxel specific delayed sLFO regressors used as EVs for the GLM	If GLM filtering is enabled (default) and `--nolimitoutput` is selected
XXX_desc-lfofilterCoeff_map	.nii.gz, .json	Magnitude of the delayed sLFO regressor from GLM filter	If GLM filtering is enabled (default)
XXX_desc-lfofilterMean_map	.nii.gz, .json	Mean value over time, from GLM fit	If GLM filtering is enabled (default)
XXX_desc-lfofilterNorm_map	.nii.gz, .json	GLM filter coefficient, divided by the voxel mean over time	If GLM filtering is enabled (default)
XXX_desc-lfofilterR_map	.nii.gz, .json	R value for the GLM fit in the voxel	If GLM filtering is enabled (default)
XXX_desc-lfofilterR2_map	.nii.gz, .json	R value for the GLM fit in the voxel, squared. Multiply by 100 to get percentage variance explained	If GLM filtering is enabled (default)
XXX_desc-lfofilterInbandVariancel	.nii.gz, .json	Mean normalized inband variance in each voxel before GLM filtering	If GLM filtering is enabled (default)
XXX_desc-lfofilterInbandVariance/	.nii.gz, .json	Mean normalized inband variance in each voxel after GLM filtering	If GLM filtering is enabled (default)
XXX_desc-lfofilterInbandVariance(	.nii.gz, .json	Percent change in mean normalized inband variance in each voxel after GLM filtering	If GLM filtering is enabled (default)

continues on next page

Table 1 – continued from previous page

Name	Extension(s)	Content	When present
XXX_desc-CVR_map	.nii.gz, .json	Cerebrovascular response, in units of % BOLD per unit of the supplied regressor (probably mmHg)	If CVR mapping is enabled
XXX_desc-CVRR_map	.nii.gz, .json	R value for the CVR map fit in the voxel	If CVR mapping is enabled
XXX_desc-CVRR2_map	.nii.gz, .json	R value for the CVR map fit in the voxel, squared. Multiply by 100 to get percentage variance explained	If CVR mapping is enabled
XXX_desc-processed_mask	.nii.gz	Mask of all voxels in which the similarity function is calculated	Always
XXX_desc-globalmean_mask	.nii.gz	Mask of voxels used to calculate the global mean signal	This file will exist if no external regressor is specified
XXX_desc-refine_mask	.nii.gz	Mask of voxels used in the last estimate a refined version of the probe regressor	Present if passes > 1
XXX_desc-shiftedtcs_bold	.nii.gz	The filtered input fMRI data, in voxels used for refinement, time shifted by the negated delay in every voxel so that the moving blood component should be aligned.	Present if passes > 1 and `--nolimitoutput` is selected
XXX_desc-despeckle_mask	.nii.gz	Mask of the last set of voxels that had their time delays adjusted due to autocorrelations in the probe regressor	Present if despecklepasses > 0
XXX_desc-corrout_info	.nii.gz	Full similarity function over the search range	Always
XXX_desc-gaussout_info	.nii.gz	Gaussian fit to similarity function peak over the search range	Always
XXX_desc-autocorr_timeseries	.tsv, .json	Autocorrelation of the probe regressor for each pass	Always
XXX_desc-corrdistdata_info	.tsv, .json	Null correlations from the significance estimation for each pass	Present if `--numnull` > 0
XXX_desc-nullsimfunc_hist	.tsv, .json	Histogram of the distribution of null correlation values for each pass	Present if `--numnull` > 0
XXX_desc-plt0p050_mask	.nii.gz	Voxels where the maxcorr value exceeds the p < 0.05 significance level	Present if `--numnull` > 0
XXX_desc-plt0p010_mask	.nii.gz	Voxels where the maxcorr value exceeds the p < 0.01 significance level	Present if `--numnull` > 0
XXX_desc-plt0p005_mask	.nii.gz	Voxels where the maxcorr value exceeds the p < 0.005 significance level	Present if `--numnull` > 0

continues on next page

Table 1 – continued from previous page

Name	Extension(s)	Content	When present
XXX_desc-plt0p001_mask	.nii.gz	Voxels where the maxcorr value exceeds the p < 0.001 significance level	Present if --numnull > 0
XXX_desc-globallag_hist	.tsv, .json	Histogram of peak correlation times between probe and all voxels, over all time lags, for each pass	Always
XXX_desc-initialmovingregressor_	.tsv, .json	The raw and filtered initial probe regressor, at the original sampling resolution	Always
XXX_desc-movingregressor_times	.tsv, .json	The probe regressor used in each pass, at the time resolution of the data	Always
XXX_desc-oversampledmovingreg	.tsv, .json	The probe regressor used in each pass, at the time resolution used for calculating the similarity function	Always
XXX_desc-refinedmovingregressor	.tsv, .json	The raw and filtered probe regressor produced by the refinement procedure, at the time resolution of the data	Present if passes > 1

2.15.5 Usage:

Perform a RIPTiDe time delay analysis on a dataset.

```
usage: rapidtide [-h]
                 [--denoising | --delaymapping | --CVR | --globalpreselect]
                 [--venousrefine | --nirs]
                 [--datatstep TSTEP | --datafreq FREQ] [--noantialias]
                 [--invert] [--interptype {univariate,cubic,quadratic}]
                 [--offsettime OFFSETTIME] [--autosync]
                 [--filterband {None,vlf,lfo,resp,cardiac,hrv_ulf,hrv_vlf,hrv_lf,hrv_hf,
↪hrv_vhf,lfo_legacy}]
                 [--filterfreqs LOWERPASS UPPERPASS]
                 [--filterstopfreqs LOWERSTOP UPPERSTOP]
                 [--filtertype {trapezoidal,brickwall,butterworth}]
                 [--butterorder ORDER] [--padseconds SECONDS]
                 [--permutationmethod {shuffle,phaserandom}] [--numnull NREPS]
                 [--skipsighistfit]
                 [--windowfunc {hamming,hann,blackmanharris,None}]
                 [--zeropadding PADVAL] [--detrendorder ORDER]
                 [--spatialfilt GAUSSSIGMA] [--globalmean]
                 [--globalmaskmethod {mean,variance}]
                 [--globalmeaninclude MASK[:VALSPEC]]
                 [--globalmeanexclude MASK[:VALSPEC]] [--motionfile MOTFILE]
                 [--motderiv] [--confoundfile CONFFILE] [--confoundpowers N]
                 [--confoundderiv] [--noconfoundorthogonalize]
                 [--globalsignalmethod {sum,meanscale,pca}]
                 [--globalpcacomponents VALUE] [--slicetimes FILE]
                 [--numskip SKIP] [--numtozero NUMPOINTS]
                 [--timerange START END] [--nothresh]
```

(continues on next page)

(continued from previous page)

```
                [--oversampfac OVERSAMPFAC] [--regressor FILE]
                [--regressorfreq FREQ | --regressortstep TSTEP]
                [--regressorstart START]
                [--corrweighting {None,phat,liang,eckart,regressor}]
                [--corrtype {linear,circular}]
                [--corrmaskthresh PCT | --corrmask MASK[:VALSPEC]]
                [--similaritymetric {correlation,mutualinfo,hybrid}]
                [--mutualinfosmoothingtime TAU] [--simcalcrange START END]
                [--fixdelay DELAYTIME | --searchrange LAGMIN LAGMAX]
                [--sigmalimit SIGMALIMIT] [--bipolar] [--nofitfilt]
                [--peakfittype {gauss,fastgauss,quad,fastquad,COM,None}]
                [--despecklepasses PASSES] [--despecklethresh VAL]
                [--refineprenorm {None,mean,var,std,invlag}]
                [--refineweighting {None,NIRS,R,R2}] [--passes PASSES]
                [--refineinclude MASK[:VALSPEC]]
                [--refineexclude MASK[:VALSPEC]] [--norefinedespeckled]
                [--lagminthresh MIN] [--lagmaxthresh MAX] [--ampthresh AMP]
                [--sigmathresh SIGMA] [--offsetinclude MASK[:VALSPEC]]
                [--offsetexclude MASK[:VALSPEC]] [--norefineoffset]
                [--pickleft] [--pickleftthresh THRESH]
                [--refineupperlag | --refinelowerlag]
                [--refinetype {pca,ica,weighted_average,unweighted_average}]
                [--pcacomponents VALUE] [--convergencethresh THRESH]
                [--maxpasses MAXPASSES] [--noglm] [--glmsourcefile FILE]
                [--preservefiltering] [--glmderivs NDERIVS] [--nolimitoutput]
                [--savelags] [--histlen HISTLEN] [--saveintermediatemaps]
                [--calccoherence] [--version] [--detailedversion]
                [--noprogressbar] [--checkpoint] [--spcalculation]
                [--dpoutput] [--cifti] [--simulate] [--displayplots]
                [--nonumba] [--nosharedmem] [--memprofile]
                [--mklthreads MKLTHREADS] [--nprocs NPROCS] [--reservecpu]
                [--infotag tagkey tagvalue] [--psdfilter] [--wiener]
                [--corrbaselinespatialsigma SIGMA]
                [--corrbaselinetemphpfcutoff FREQ]
                [--spatialtolerance EPSILON] [--echocancel]
                [--autorespdelete] [--noisetimecourse FILENAME[:VALSPEC]]
                [--noisefreq FREQ | --noisetstep TSTEP] [--noisestart START]
                [--noiseinvert] [--acfix] [--negativegradient]
                [--cleanrefined] [--dispersioncalc] [--tmask FILE] [--debug]
                [--verbose] [--disabledockermemfix] [--alwaysmultiproc]
                [--singleproc_confoundregress] [--singleproc_getNullDist]
                [--singleproc_calcsimilarity] [--singleproc_peakeval]
                [--singleproc_fitcorr] [--singleproc_refine]
                [--singleproc_makelaggedtcs] [--singleproc_glm] [--isatest]
                in_file outputname
```

Positional Arguments

in_file The input data file (BOLD fMRI file or NIRS text file).

outputname The root name for the output files. For BIDS compliance, this can only contain valid BIDS entities from the source data.

Analysis type

Single arguments that set several parameter values, tailored to particular analysis types. Any parameter set by an analysis type can be overridden by setting that parameter explicitly. Analysis types are mutually exclusive with one another.

--denoising Preset for hemodynamic denoising - this is a macro that sets searchrange=(-10.0, 10.0), passes=3, despeckle_passes=4, refineoffset=True, peakfittype=gauss, gausssigma=-1, nofitfilt=True, doglmfilt=True. Any of these options can be overridden with the appropriate additional arguments.

Default: False

--delaymapping Preset for delay mapping analysis - this is a macro that sets searchrange=(-10.0, 30.0), passes=3, despeckle_passes=4, refineoffset=True, pickleft=True, limitoutput=True, doglmfilt=False. Any of these options can be overridden with the appropriate additional arguments.

Default: False

--CVR Preset for calibrated CVR mapping. Given an input regressor that represents some measured quantity over time (e.g. mmHg CO2 in the EtCO2 trace), rapidtide will calculate and output a map of percent BOLD change in units of the input regressor. To do this, this sets: passes=1, despeckle_passes=4, searchrange=(-5.0, 20.0), filterfreqs=(0.0, 0.01), and calculates a voxelwise GLM using the optimally delayed input regressor and the percent normalized, demeaned BOLD data as inputs. This map is output as (XXX_desc-CVR_map.nii.gz). If no input regressor is supplied, this will generate an error. These options can be overridden with the appropriate additional arguments.

Default: False

--globalpreselect Treat this run as an initial pass to locate good candidate voxels for global mean regressor generation. This sets: passes=1, pickleft=True, despecklepasses=0, refinedespeckle=False, limitoutput=True, doglmfilt=False, saveintermediatemaps=False.

Default: False

Macros

Single arguments that change default values for many arguments. Macros override individually set parameters. Macros are mutually exclusive with one another.

--venousrefine This is a macro that sets lagminthresh=2.5, lagmaxthresh=6.0, ampthresh=0.5, and refineupperlag to bias refinement towards voxels in the draining vasculature for an fMRI scan.

Default: False

--nirs This is a NIRS analysis - this is a macro that sets nothresh, refineprenorm=var, ampthresh=0.7, and lagminthresh=0.1.

Default: False

Preprocessing options

--datatstep Set the timestep of the data file to TSTEP. This will override the TR in an fMRI file. NOTE: if using data from a text file, for example with NIRS data, using one of these options is mandatory.

Default: auto

--datafreq Set the timestep of the data file to 1/FREQ. This will override the TR in an fMRI file. NOTE: if using data from a text file, for example with NIRS data, using one of these options is mandatory.

Default: auto

--noantialias Disable antialiasing filter.

Default: True

--invert Invert the sign of the regressor before processing.

Default: False

--interptype Possible choices: univariate, cubic, quadratic

Use specified interpolation type. Options are “cubic”, “quadratic”, and “univariate”. Default is univariate.

Default: “univariate”

--offsettime Apply offset OFFSETTIME to the lag regressors.

Default: 0.0

--autosync Estimate and apply the initial offsettime of an external regressor using the global crosscorrelation. Overrides offsettime if present.

Default: False

--detrendorder Set order of trend removal (0 to disable). Default is 3.

Default: 3

--spatialfilt Spatially filter fMRI data prior to analysis using GAUSSSIGMA in mm. Set GAUSSSIGMA negative to have rapidtide set it to half the mean voxel dimension (a rule of thumb for a good value).

Default: -1

--globalmean Generate a global mean regressor and use that as the reference regressor. If no external regressor is specified, this is enabled by default.

Default: False

--globalmaskmethod Possible choices: mean, variance

Select whether to use timecourse mean or variance to mask voxels prior to generating global mean. Default is “mean”.

Default: “mean”

--globalmeaninclude Only use voxels in mask file NAME for global regressor generation (if VALSPEC is given, only voxels with integral values listed in VALSPEC are used).

--globalmeanexclude Do not use voxels in mask file NAME for global regressor generation (if VALSPEC is given, only voxels with integral values listed in VALSPEC are excluded).

--motionfile Read 6 columns of motion regressors out of MOTFILE file (.par or BIDS .json) (with timepoints rows) and regress them and/or their derivatives out of the data prior to analysis.

--motderiv Toggle whether derivatives will be used in motion regression. Default is True.

Default: True

--confoundfile Read additional (non-motion) confound regressors out of CONFFILE file (which can be any type of multicolumn text file rapidtide reads as long as data is sampled at TR with timepoints rows). Optionally do power expansion and/or calculate derivatives prior to regression.

--confoundpowers Include powers of each confound regressor up to order N. Default is 1 (no expansion).

Default: 1

--confoundderiv Toggle whether derivatives will be used in confound regression. Default is True.

Default: True

--noconfoundorthogonalize Do not orthogonalize confound regressors prior to regressing them out of the data.

Default: True

--globalsignalmethod Possible choices: sum, meanscale, pca

The method for constructing the initial global signal regressor - straight summation, mean scaling each voxel prior to summation, or MLE PCA of the voxels in the global signal mask. Default is "sum."

Default: "sum"

--globalpcacomponents Number of PCA components used for estimating the global signal. If VALUE >= 1, will retain thismany components. If 0.0 < VALUE < 1.0, enough components will be retained to explain the fraction VALUE of the total variance. If VALUE is negative, the number of components will be to retain will be selected automatically using the MLE method. Default is 0.8.

Default: 0.8

--slicetimes Apply offset times from FILE to each slice in the dataset.

--numskip SKIP TRs were previously deleted during preprocessing (e.g. if you have done your preprocessing in FSL and set dummypoints to a nonzero value.) Default is 0.

Default: 0

--numtozero When calculating the moving regressor, set this number of points to zero at the beginning of the voxel timecourses. This prevents initial points which may not be in equilibrium from contaminating the calculated sLFO signal. This may improve similarity fitting and GLM noise removal. Default is 0.

Default: 0

--timerange Limit analysis to data between timepoints START and END in the fmri file. If END is set to -1, analysis will go to the last timepoint. Negative values of START will be set to 0. Default is to use all timepoints.

Default: (-1, -1)

--nothresh Disable voxel intensity threshold (especially useful for NIRS data).

Default: False

Filtering options

--filterband Possible choices: None, vlf, lfo, resp, cardiac, hrv_ulf, hrv_vlf, hrv_lf, hrv_hf, hrv_vhf, lfo_legacy

Filter data and regressors to specific band. Use "None" to disable filtering. Default is "lfo".

Default: "lfo"

--filterfreqs Filter data and regressors to retain LOWERPASS to UPPERPASS. If –filterstopfreqs is not also specified, LOWERSTOP and UPPERSTOP will be calculated automatically.

--filterstopfreqs Filter data and regressors to with stop frequencies LOWERSTOP and UPPERSTOP. LOWERSTOP must be <= LOWERPASS, UPPERSTOP must be >= UPPERPASS. Using this argument requires the use of –filterfreqs.

--filtertype Possible choices: trapezoidal, brickwall, butterworth

Filter data and regressors using a trapezoidal FFT, brickwall FFT, or butterworth bandpass filter. Default is "trapezoidal".

Default: "trapezoidal"

--butterorder Set order of butterworth filter (if used). Default is 6.

Default: 6

--padseconds The number of seconds of padding to add to each end of a filtered timecourse to reduce end effects. Default is 30.0.

Default: 30.0

Significance calculation options

--permutationmethod Possible choices: shuffle, phaserandom

Permutation method for significance testing. Default is "shuffle".

Default: "shuffle"

--numnull Estimate significance threshold by running NREPS null correlations (default is 10000, set to 0 to disable).

Default: 10000

--skipsighistfit Do not fit significance histogram with a Johnson SB function.

Default: True

Windowing options

--windowfunc Possible choices: hamming, hann, blackmanharris, None

Window function to use prior to correlation. Options are hamming, hann, blackmanharris, and None. Default is hamming

Default: "hamming"

--zeropadding Pad input functions to correlation with PADVAL zeros on each side. A PADVAL of 0 does circular correlations, positive values reduce edge artifacts. Set PADVAL < 0 to set automatically. Default is 0.

Default: 0

Correlation options

--oversampfac Oversample the fMRI data by the following integral factor. Set to -1 for automatic selection (default).

Default: -1

--regressor Read the initial probe regressor from file FILE (if not specified, generate and use the global regressor).

--regressorfreq Probe regressor in file has sample frequency FREQ (default is 1/tr) NB: –regressorfreq and –regressortstep) are two ways to specify the same thing.

Default: auto

--regressortstep Probe regressor in file has sample frequency FREQ (default is 1/tr) NB: –regressorfreq and –regressortstep) are two ways to specify the same thing.

Default: auto

--regressorstart The time delay in seconds into the regressor file, corresponding in the first TR of the fMRI file (default is 0.0).

Default: 0.0

--corrweighting Possible choices: None, phat, liang, eckart, regressor

Method to use for cross-correlation weighting. 'None' performs an unweighted correlation. 'phat' weights the correlation by the magnitude of the product of the timecourse's FFTs. 'liang' weights the correlation by the sum of the magnitudes of the timecourse's FFTs. 'eckart' weights the correlation by the product of the magnitudes of the timecourse's FFTs. 'regressor' weights the correlation by the magnitude of the sLFO regressor FFT. Default is "regressor".

Default: "regressor"

--corrtype Possible choices: linear, circular

Cross-correlation type (linear or circular). Default is "linear".

Default: "linear"

--corrmaskthresh Do correlations in voxels where the mean exceeds this percentage of the robust max. Default is 1.0.

Default: 1.0

--corrmask Only do correlations in nonzero voxels in NAME (if VALSPEC is given, only voxels with integral values listed in VALSPEC are used).

--similaritymetric Possible choices: correlation, mutualinfo, hybrid

Similarity metric for finding delay values. Choices are "correlation", "mutualinfo", and "hybrid". Default is correlation.

Default: "correlation"

--mutualinfosmoothingtime Time constant of a temporal smoothing function to apply to the mutual information function. Default is 3.0 seconds. TAU <=0.0 disables smoothing.

Default: 3.0

--simcalcrange Limit correlation calculation to data between timepoints START and END in the fmri file. If END is set to -1, analysis will go to the last timepoint. Negative values of START will be set to 0. Default is to use all timepoints. NOTE: these offsets are relative to the start of the dataset AFTER any trimming done with '–timerange'.

Default: (-1, -1)

Correlation fitting options

--fixdelay Don't fit the delay time - set it to DELAYTIME seconds for all voxels.

--searchrange Limit fit to a range of lags from LAGMIN to LAGMAX. Default is -30.0 to 30.0 seconds.

Default: (-30.0, 30.0)

--sigmalimit Reject lag fits with linewidth wider than SIGMALIMIT Hz. Default is 1000.0 Hz.

Default: 1000.0

--bipolar Bipolar mode - match peak correlation ignoring sign.

Default: False

--nofitfilt Do not zero out peak fit values if fit fails.

Default: True

--peakfittype Possible choices: gauss, fastgauss, quad, fastquad, COM, None

Method for fitting the peak of the similarity function "gauss" performs a Gaussian fit, and is most accurate. "quad" and "fastquad" use a quadratic fit, which is faster, but not as well tested. Default is "gauss".

Default: "gauss"

--despecklepasses Detect and refit suspect correlations to disambiguate peak locations in PASSES passes. Default is to perform 4 passes. Set to 0 to disable.

Default: 4

--despecklethresh Refit correlation if median discontinuity magnitude exceeds VAL. Default is 5.0 seconds.

Default: 5.0

Regressor refinement options

--refineprenorm Possible choices: None, mean, var, std, invlag

Apply TYPE prenormalization to each timecourse prior to refinement. Default is "var".

Default: "var"

--refineweighting Possible choices: None, NIRS, R, R2

Apply TYPE weighting to each timecourse prior to refinement. Default is "None".

Default: "None"

--passes Set the number of processing passes to PASSES. Default is 3.

Default: 3

--refineinclude Only use voxels in file MASK for regressor refinement (if VALSPEC is given, only voxels with integral values listed in VALSPEC are used).

--refineexclude Do not use voxels in file MASK for regressor refinement (if VALSPEC is given, voxels with integral values listed in VALSPEC are excluded).

--norefinedespeckled Do not use despeckled pixels in calculating the refined regressor.

Default: True

--lagminthresh For refinement, exclude voxels with delays less than MIN. Default is 0.5 seconds.

Default: 0.5

--lagmaxthresh For refinement, exclude voxels with delays greater than MAX. Default is 5.0 seconds.

Default: 5.0

--ampthresh For refinement, exclude voxels with correlation coefficients less than AMP (default is 0.3). NOTE: ampthresh will automatically be set to the p<0.05 significance level determined by the –numnull option if NREPS is set greater than 0 and this is not manually specified.

Default: -1.0

--sigmathresh For refinement, exclude voxels with widths greater than SIGMA seconds. Default is 100.0 seconds.

Default: 100.0

--offsetinclude Only use voxels in file MASK for determining the zero time offset value (if VALSPEC is given, only voxels with integral values listed in VALSPEC are used).

--offsetexclude Do not use voxels in file MASK for determining the zero time offset value (if VALSPEC is given, voxels with integral values listed in VALSPEC are excluded).

--norefineoffset Disable realigning refined regressor to zero lag.

Default: True

--pickleft Will select the leftmost delay peak when setting the refine offset.

Default: False

--pickleftthresh Threshhold value (fraction of maximum) in a histogram to be considered the start of a peak. Default is 0.33.

Default: 0.33

--refineupperlag Only use positive lags for regressor refinement.

Default: "both"

--refinelowerlag Only use negative lags for regressor refinement.

Default: "both"

--refinetype Possible choices: pca, ica, weighted_average, unweighted_average

Method with which to derive refined regressor. Default is "pca".

Default: "pca"

--pcacomponents Number of PCA components used for refinement. If VALUE >= 1, will retain this many components. If 0.0 < VALUE < 1.0, enough components will be retained to explain the fraction VALUE of the total variance. If VALUE is negative, the number of components will be to retain will be selected automatically using the MLE method. Default is 0.8.

Default: 0.8

--convergencethresh Continue refinement until the MSE between regressors becomes <= THRESH. By default, this is not set, so refinement will run for the specified number of passes.

--maxpasses Terminate refinement after MAXPASSES passes, whether or not convergence has occured. Default is 15.

Default: 15

GLM noise removal options

--noglm Turn off GLM filtering to remove delayed regressor from each voxel (disables output of fitNorm).

Default: True

--glmsourcefile Regress delayed regressors out of FILE instead of the initial fmri file used to estimate delays.

--preservefiltering Don't reread data prior to performing GLM.

Default: False

--glmderivs When doing final GLM, include derivatives up to NDERIVS order. Default is 0

Default: 0

Output options

--nolimitoutput Save some of the large and rarely used files.

Default: True

--savelags Save a table of lagtimes used.

Default: False

--histlen Change the histogram length to HISTLEN. Default is 101.

Default: 101

--saveintermediatemaps Save lag times, strengths, widths, and mask for each pass.

Default: False

--calccoherence Calculate and save the coherence between the final regressor and the data.

Default: False

Version options

--version Show simplified version information and exit

--detailedversion Show detailed version information and exit

Miscellaneous options

--noprogressbar Will disable showing progress bars (helpful if stdout is going to a file).

Default: True

--checkpoint Enable run checkpoints.

Default: False

--spcalculation Use single precision for internal calculations (may be useful when RAM is limited).

Default: "double"

--dpoutput Use double precision for output files.

Default: "single"

--cifti Data file is a converted CIFTI.

Default: False

--simulate Simulate a run - just report command line options.

Default: False

--displayplots Display plots of interesting timecourses.

Default: False

--nonumba Disable jit compilation with numba.

Default: False

--nosharedmem Disable use of shared memory for large array storage.

Default: True

--memprofile Enable memory profiling - warning: this slows things down a lot.

Default: False

--mklthreads Use no more than MKLTHREADS worker threads in accelerated numpy calls.

Default: 1

--nprocs Use NPROCS worker processes for multiprocessing. Setting NPROCS to less than 1 sets the number of worker processes to n_cpus (unless –reservecpu is used).

Default: 1

--reservecpu When automatically setting nprocs, reserve one CPU for process management rather than using them all for worker threads.

Default: False

--infotag Additional key, value pairs to add to the options json file (useful for tracking analyses).

Experimental options (not fully tested, may not work)

--psdfilter Apply a PSD weighted Wiener filter to shifted timecourses prior to refinement.

Default: False

--wiener Do Wiener deconvolution to find voxel transfer function.

Default: False

--corrbaselinespatialsigma Spatial lowpass kernel, in mm, for filtering the correlation function baseline.

Default: 0.0

--corrbaselinetemphpfcutoff Temporal highpass cutoff, in Hz, for filtering the correlation function baseline.

Default: 0.0

--spatialtolerance When checking to see if the spatial dimensions of two NIFTI files match, allow a relative difference of EPSILON in any dimension. By default, this is set to 0.0, requiring an exact match.

Default: 0.0

--echocancel Attempt to perform echo cancellation on current moving regressor.

Default: False

--autorespdelete Attempt to detect and remove respiratory signal that strays into the LFO band.

Default: False

--noisetimecourse Find and remove any instance of the timecourse supplied from any regressors used for analysis. (if VALSPEC is given, and there are multiple timecourses in the file, use the indicated timecourse.This can be the name of the regressor if it's in the file, or the column number).

--noisefreq Noise timecourse in file has sample frequency FREQ (default is 1/tr) NB: –noisefreq and –noisetstep) are two ways to specify the same thing.

Default: auto

--noisetstep Noise timecourse in file has sample frequency FREQ (default is 1/tr) NB: –noisefreq and –noisetstep) are two ways to specify the same thing.

Default: auto

--noisestart The time delay in seconds into the noise timecourse file, corresponding in the first TR of the fMRI file (default is 0.0).

Default: 0.0

--noiseinvert Invert noise regressor prior to alignment.

Default: False

--acfix Check probe regressor for autocorrelations in order to disambiguate peak location.

Default: False

--negativegradient Calculate the negative gradient of the fmri data after spectral filtering so you can look for CSF flow à la https://www.biorxiv.org/content/10.1101/2021.03.29.437406v1.full.

Default: False

--cleanrefined Perform additional processing on refined regressor to remove spurious components.

Default: False

--dispersioncalc Generate extra data during refinement to allow calculation of dispersion.

Default: False

--tmask Only correlate during epochs specified in MASKFILE (NB: each line of FILE contains the time and duration of an epoch to include.

Debugging options. You probably don't want to use any of these unless I ask you to to help diagnose a problem

--debug Enable additional debugging output.

Default: False

--verbose Enable additional runtime information output.

Default: False

--disabledockermemfix Disable docker memory limit setting.

Default: True

--alwaysmultiproc Use the multiprocessing code path even when nprocs=1.

Default: False

--singleproc_confoundregress Force single proc path for confound regression.

Default: False

--singleproc_getNullDist Force single proc path for getNullDist.

Default: False

--singleproc_calcsimilarity Force single proc path for calcsimilarity.

Default: False

--singleproc_peakeval Force single proc path for peakeval.

Default: False

--singleproc_fitcorr Force single proc path for fitcorr.

Default: False

--singleproc_refine Force single proc path for refine.

Default: False

--singleproc_makelaggedtcs Force single proc path for makelaggedtcs.

Default: False

--singleproc_glm Force single proc path for glm.

Default: False

--isatest This run of rapidtide is in a unit test.

Default: False

2.15.6 Preprocessing for rapidtide

Rapidtide operates on data which has been subjected to "standard" preprocessing steps, most importantly motion correction and slice time correction.

Motion correction - Motion correction is good since you want to actually be looking at the same voxels in each timepoint. Definitely do it. There may be spin history effects even after motion correction, so if you give rapidtide a motion file using `--motionfile FILENAME` (and various other options to tune how it does the motion regression) it can regress out residual motion prior to estimating sLFO parameters. In cases of extreme motion, this will make rapidtide work a lot better. If you choose to regress out the motion signals yourself, that's fine too - rapidtide is happy to work on data that's been run through AROMA (not so much FIX - see a further discussion below).

Slice time correction - Since rapidtide is looking for subtle time differences in the arrival of the sLFO signal, it will absolutely see slice acquisition time differences. If you are doing noise removal, that's not such a big deal, but if you're doing delay mapping, you'll get stripes in your delay maps, which tell you about the fMRI acquisition, but you care about physiology, so best to avoid that. Unfortunately, Human Connectome Project data does NOT have slice time correction applied, and unless you want to rerun the entire processing chain to add it in, you just have to deal with it. Fortunately the TR is quite short, so the stripes are subtle. The geometric distortion correction and alignment steps done in the HCP distort the stripes, but you can certainly see them. If you average enough subjects though, they get washed out.

Spatial filtering - I generally do NOT apply any spatial filtering during preprocessing for a variety of reasons. fmriprep doesn't do it, so I feel validated in this choice. You can always do it later, and rapidtide lets you do spatial smoothing for the purpose of estimating the delayed regressor using the `--gausssigma` parameter. This turns out to stabilize the fits for rapidtide and is usually a good thing, however you probably don't want it for other processing (but that's ok - see below).

Temporal filtering - Rapidtide does all it's own temporal filtering; highpass filtering at 0.01Hz, common in r esting state preprocessing, doesn't affect the frequency ranges rapidtide cares about for sLFOs, so you can do it or not during preprocessing as you see fit (but if you're doing CVR or gas challenge experiments you probably shouldn't).

NOTE: Astute readers will notice that between spatial filtering, motion regression, and other procedures, rapidtide does a lot of it's work of estimating sLFOs on potentially heavily filtered data, which is good for improving the estimation and fitting of the sLFO signal. However, you may or may not want this filtering to have been done for whatever your particular subsequent analysis is. So prior to GLM denoising, rapidtide rereads the unmodified fMRI input file, and regresses the voxel specific sLFO out of *that* - since the filtering process is linear, that's cool - the data you get out is the data you put in, just minus the sLFO signal. If for some reason you *do* want to use the data that rapidtide has abused, simply use the `--preservefiltering` option, but I'd recommend you don't do that.

Working with standard fMRI packages

FSL - At the time I first developed rapidtide, I was using FSL almost exclusively, so some of the assumptions the program makes about the data stem from this. If you want to integrate rapidtide into your FSL workflow, you would typically use the `filtered_func_data.nii.gz` file from your FEAT directory (the result of FSL preprocessing) as input to rapidtide. Note that this is typically in native acquisition space. You can use this, or do the processing in standard space if you've done that alignment - either is fine, but for conventional EPI acquisitions, there are typically far fewer voxels at native resolution, so processing will probably be faster. On the flip side, having everything in standard space makes it easier to combine runs and subjects.

fmriprep - If you do preprocessing in fmriprep, the easiest file to use for input to rapidtide would be either `derivatives/fmriprep/sub-XXX/ses-XXX/func/XXX_desc-preproc_bold.nii.gz` (native space) or `derivatives/fmriprep/sub-XXX/ses-XXX/func/XXX_space-MNI152NLin6Asym_res-2_desc-preproc_bold.nii.gz` (standard space - replace `MNI152NLin6aAsym_res-2` with whatever space and resolution you used if not the FSL compatible one). One caveat - unless this has changed recently, fmriprep does *not* store the transforms needed to go from native BOLD space to standard space, so you'll have to come up with that yourself either by fishing the transform out of the workdir, or redoing the alignment. That's a pretty strong argument for using the standard space. In addition, if you do the analysis in standard space, it makes it easier to use freesurfer parcellations and gray/white/csf segmentations that fmriprep provides for further tuning the rapidtide analysis. See the "Theory of Operation" section for more on this subject.

AFNI - Here's a case where you have to take some care - as I mentioned above, rapidtide assumes "FSL-like" data by default. The most important difference between AFNI and FSL preprocessing (assuming you've put your AFNI data into NIFTI format) is that AFNI removes the mean from the preprocessed fMRI data (this is a valid implementation choice - no judgement, but, no, actually - seriously, WTF? WHY WOULD YOU DO THAT???). This makes rapidtide sad, because the mean value of the fMRI data is used for all sorts of things like generating masks. Fortunately, this can be easily accommodated. You have a couple of choices here. You can supply a mean mask and correlation mask explicitly using `--globalmeaninclude FILENAME` and `--corrmask FILENAME`, (FILENAME should definitely be a brain mask for `--corrmask` - it can be more focussed for `--globalmeaninclude` - for example, a gray matter mask, but a brain mask works fine in most cases) which will get rapidtide past the places that zero mean data will confuse it. Alternately, if you don't have a brain mask, you can use `--globalmaskmethod variance` to make a mask based on the variance over time in a voxel rather than than the mean. Rapidtide should then work as normal, although the display in `tidepool` will be a little weird unless you specify a background image explicitly.

SPM - I have no reason to believe rapidtide won't work fine with data preprocessed in SPM. That said, I don't use SPM, so I can't tell you what file to use, or what format to expect the preprocessed data will be in. If you, dear reader, have any insight into this, PLEASE tell me and I'll do what I need to to support SPM data in the code and documentation.

2.15.7 Analysis Examples:

Rapidtide can do many things - as I've found more interesting things to do with time delay processing, it's gained new functions and options to support these new applications. As a result, it can be a little hard to know what to use for a new experiment. To help with that, I've decided to add this section to the manual to get you started. It's broken up by type of data/analysis you might want to do.

NB: To speed up the analysis, adding the argument `--nprocs XX` to any of the following commands will parallelize the analysis to use XX CPUs - set XX to -1 to use all available CPUs. This can result in a speedup approaching a factor of the number of CPUs used.

Removing low frequency physiological noise from fMRI data

This is what I figure most people will use rapidtide for - finding and removing the low frequency (LFO) signal from an existing dataset (including the case where the signal grows over time https://www.biorxiv.org/content/10.1101/2023.09.08.556939v2). This presupposes you have not made a simultaneous physiological recording (well, you may have, but it assumes you aren't using it). For this, you can use a minimal set of options, since the defaults are set to be generally optimal for noise removal.

The base command you'd use would be:

```
rapidtide \
    inputfmrifile \
    outputname \
    --denoising
```

This will do a the default analysis (but each and every particular can be changed by adding command line options). By default, rapidtide will:

1. Temporally prefilter the data to the LFO band (0.009-0.15Hz), and spatially filter with a Gaussian kernel of 1/2 the mean voxel dimension in x, y, and z.
2. Construct a probe regressor from the global mean of the signal in inputfmrifile (default behavior if no regressor or selections masks are specified).
3. Do three passes through the data. In each step, rapidtide will:
    1. Perform a crosscorrelation of each voxel with the probe regressor using the "regressor" weighting.
    2. Estimate the location and strength of the correlation peak using the correlation similarity metric within a range of +/-10 seconds around around the modal delay value.
    3. Generate a new estimate of the global noise signal by:
        1. Aligning all of the voxel timecourses to bring the global signal into phase,
        2. Performing a PCA analysis,
        3. Reconstructing each timecourse using the PCA components accounting for 80% of the signal variance in the aligned voxel timecourses,
        4. Averaging the reconstructed timecourses to produce a new probe regressor,
        5. Applying an offset to the recenter the peak of the delay distribution of all voxels to zero, which should make datasets easier to compare.
4. After the three passes are complete, rapidtide will then use a GLM filter to remove a voxel specific lagged copy of the final probe regressor from the data - this denoised data will be in the file `outputname_desc-lfofilterCleaned_bold.nii.gz`. There will also a number of maps output with the prefix `outputname_` of delay, correlation strength and so on. See the BIDS Output table above for specifics.

Please note that rapidtide plays happily with AROMA, so you don't need to do anything special to process data that's been run through AROMA. While FIX and AROMA both use spatiotemporal analysis of independent components to determine what components to remove, AROMA only targets ICs related to motion, which are quite distinct from the sLFO signal, so they don't interfere with each other. In contrast, FIX targets components that are "bad", for multiple definitions of the term, which includes some purely hemodynamic components near the back of the brain. As a result, FIX denoising impedes the operation of rapidtide. See below.

Removing low frequency physiological noise from fMRI data that has been processed with FIX

There is a special case if you are working on HCP data, which has both minimally processed and a fully processed (including FIX denoising) data files. FIX denoising is a good thing, but it tends to distort the sLFO signals that rapidtide is looking for, so the selection and refinement of the sLFO can wander off into the thicket if applied to FIX processed data. So ideally, you would run rapidtide, and THEN FIX. However, since reprocessing the HCP data is kind of a pain, there's a hack that capitalizes on the fact that all of these operations are linear. You run rapidtide on the minimmally processed data, to accurately assess the sLFO regressor and time delays in each voxel, but you apply the final GLM to the FIX processed data, to remove the data that has the other denoising already done. This works very well! To do this, you use the `--glmsourcefile FILE` option to specify the file you want to denoise. The `outputname_desc-lfofilterCleaned_bold.nii.gz` file is the FIX file, with rapidtide denoising applied.

```
rapidtide \
    minimallyprocessedinputfmrifile \
    outputname \
    --denoising \
    --glmsourcefile FIXprocessedfile
```

Mapping long time delays in response to a gas challenge experiment:

Processing this sort of data requires a very different set of options from the previous case. Instead of the distribution of delays you expect in healthy controls (a slightly skewed, somewhat normal distribution with a tail on the positive side, ranging from about -5 to 5 seconds), in this case, the maximum delay can be extremely long (100-120 seconds is not uncommon in stroke, moyamoya disesase, and atherosclerosis). To do this, you need to radically change what options you use, not just the delay range, but a number of other options having to do with refinement and statistical measures.

For this type of analysis, a good place to start is the following:

```
rapidtide \
    inputfmrifile \
    outputname \
    --numnull 0 \
    --searchrange -10 140 \
    --filterfreqs 0.0 0.01 \
    --ampthresh 0.2 \
    --noglm \
    --nofitfilt
```

The first option (`--numnull 0`), shuts off the calculation of the null correlation distribution. This is used to determine the significance threshold, but the method currently implemented in rapidtide is a bit simplistic - it assumes that all the time points in the data are exchangable. This is certainly true for resting state data (see above), but it is very much NOT true for block paradigm gas challenges. To properly analyze those, I need to consider what time points are 'equivalent', and up to now, I don't, so setting the number of iterations in the Monte Carlo analysis to zero omits this step.

The second option (`--searchrange -10 140`) is fairly obvious - this extends the detectable delay range out to 140 seconds. Note that this is somewhat larger than the maximum delays we frequently see, but to find the correlation peak with maximum precision, you need sufficient additional delay values so that the correlation can come to a peak and then come down enough that you can properly fit it. Obviously adjust this as needed for your experiment, to fit the particulars of your gas challenge waveform and/or expected pathology.

Setting `--filterfreqs 0.0 0.01` is VERY important. By default, rapidtide assumes you are looking at endogenous low frequency oscillations, which typically between 0.009 and 0.15 Hz. However, gas challenge paradigms are usually MUCH lower frequency (90 seconds off, 90 seconds on corresponds to 1/180s = ~0.006Hz). So if you use the default frequency settings, you will completely filter out your stimulus, and presumably, your response. If you are processing one of these experiments and get no results whatsoever, this is almost certainly the problem.

The `--noglm` option disables data filtering. If you are using rapidtide to estimate and remove low frequency noise from resting state or task fMRI data, the last step is to use a glm filter to remove this circulatory signal, leaving "pure" neuronal signal, which you'll use in further analyses. That's not relevant here - the signal you'd be removing is the one you care about. So this option skips that step to save time and disk space.

`--nofitfilt` skips a step after peak estimation. Estimating the delay and correlation amplitude in each voxel is a two step process. First you make a quick estimate (where is the maximum point of the correlation function, and what is its amplitude?), then you refine it by fitting a Gaussian function to the peak to improve the estimate. If this step fails, which it can if the peak is too close to the end of the lag range, or strangely shaped, the default behavior is to mark the point as bad and zero out the parameters for the voxel. The nofitfilt option means that if the fit fails, output the initial estimates rather than all zeros. This means that you get some information, even if it's not fully refined. In my experience it does tend to make the maps for the gas challenge experiments a lot cleaner to use this option since the correlation function is pretty well behaved.

CVR mapping:

This is a slightly different twist on interpreting the strength of the lagged correlation. In this case, you supply an input regressor that corresponds to a measured, calibrated CO2 quantity (for example, etCO2 in mmHg). Rapidtide then does a modified analysis - it still uses the cross-correlation to find when the input regressor is maximally aligned with the variance in the voxel signal, but instead of only returning a correlation strength, it calculates the percentage BOLD change in each voxel in units of the input regressor (e.g. %BOLD/mmHg), which is the standard in CVR analysis.

```
rapidtide \
    inputfmrifile \
    outputname \
    --regressor regressorfile \
    --CVR
```

You invoke this with the `--CVR` option. This is a macro that does a lot of things: I disabled refinement, set `--passes 1`, set `--filterfreqs 0.0 0.01` (for the reasons described above regarding gas challenge experiments), set `--searchrange -5 20`, hijacked the GLM filtering routine, and messed with some normalizations. If you want to refine your regressor estimate, or filter the sLFO signal out of your data, you need to do a separate analysis.

You also need to supply the regressor using `--regressor regressorfile`. If regressorfile is a bids tsv/json pair, this will have the sample rate and offset specified. If the regressor file has sample rate other than the fMRI TR, or a non-zero offset relative to the fMRI data, you will also need to specify these parameters using `--regressorfreq FREQ` or `--regressortstep TSTEP` and/or `--regressorstart START`.

Denoising NIRS data:

Fun face - when we started this whole research effort, I was originally planning to denoise NIRS data, not fMRI data. But one thing led to another, and the NIRS got derailed for the fMRI effort. Now that we have some time to catch our breaths, and more importantly, we have access to some much higher quality NIRS data, this moved back to the front burner. The majority of the work was already done, I just needed to account for a few qualities that make NIRS data different from fMRI data:

- NIRS data is not generally stored in NIFTI files. While there is one now (SNIRF), at the time I started doing this, there was no standard NIRS file format. In the absence of one, you could do worse than a multicolumn text file, with one column per data channel. That's what I did here - if the file has a '.txt' extension rather than '.nii.', '.nii.gz', or no extension, it will assume all I/O should be done on multicolumn text files. However, I'm a firm believer in SNIRF, and will add support for it one of these days.
- NIRS data is often zero mean. This turned out to mess with a lot of my assumptions about which voxels have significant data, and mask construction. This has led to some new options for specifying mask threshholds and data averaging.

- NIRS data is in some sense "calibrated" as relative micromolar changes in oxy-, deoxy-, and total hemoglobin concentration, so mean and/or variance normalizing the timecourses may not be right thing to do. I've added in some new options to mess with normalizations.

2.16 rapidtide2std

2.16.1 Description:

This is a utility for registering rapidtide output maps to standard coordinates. It's usually much faster to run rapidtide in native space then transform afterwards to MNI152 space. NB: this will only work if you have a working FSL installation.

2.16.2 Inputs:

2.16.3 Outputs:

New versions of the rapidtide output maps, registered to either MNI152 space or to the hires anatomic images for the subject. All maps are named with the specified root name with '_std' appended.

2.16.4 Usage:

Register rapidtide output maps to standard space.

```
usage: rapidtide2std [-h] [--all] [--hires] [--linear] [--onefile FILE]
                     [--sequential] [--fake] [--debug]
                     inputfileroot outputdir featdirectory
```

Positional Arguments

inputfileroot	the root name of the input NIFTI files (up to but not including the underscore).
outputdir	The location for the output files
featdirectory	Either a feat directory (x.feat) or an fmriprep derivatives anat directory where the information needed for registration to standard space can be found

Named Arguments

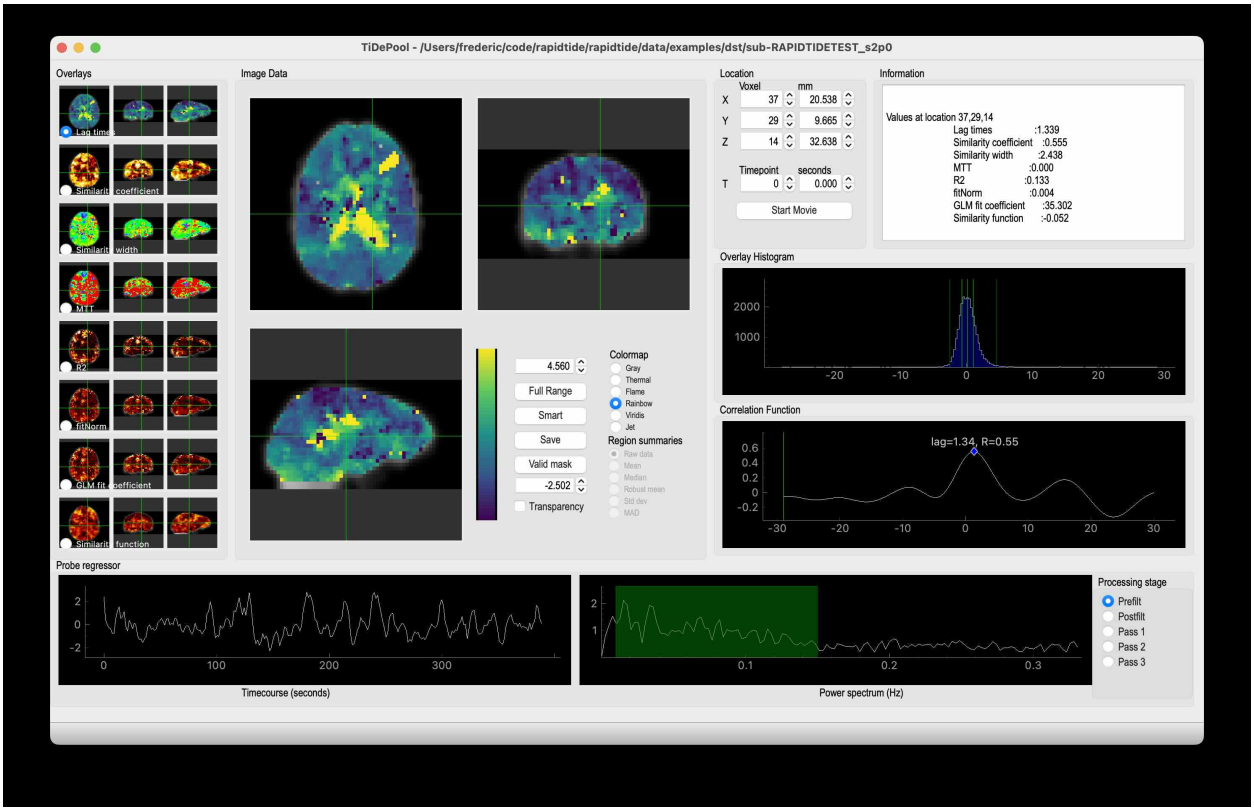
--all	Also transform the corrout file (warning - file may be huge).
	Default: False
--hires	Transform to match the high resolution anatomic image rather than the standard.
	Default: False
--linear	Only do linear transformation, even if warpfile exists.
	Default: False

--onefile	Align a single file, specified by name without extension (ignore INPUTFILE-ROOT).
--sequential	Execute commands sequentially - do not submit to the cluster. Default: False
--fake	Output, but do not execute, alignment commands. Default: False
--debug	Output additional debugging information. Default: False

2.17 tidepool

2.17.1 Description:

Tidepool is a handy tool for displaying all of the various maps generated by rapidtide in one place, overlayed on an anatomic image. This makes it easier to see how all the maps are related to one another. To use it, launch tidepool from the command line, navigate to a rapidtide output directory, and then select a lag time (maxcorr) map. tidpool will figure out the root name and pull in all of the other associated maps, timecourses, and info files. The displays are live, and linked together, so you can explore multiple parameters efficiently. Works in native or standard space.



The main tidepool window with a dataset loaded.

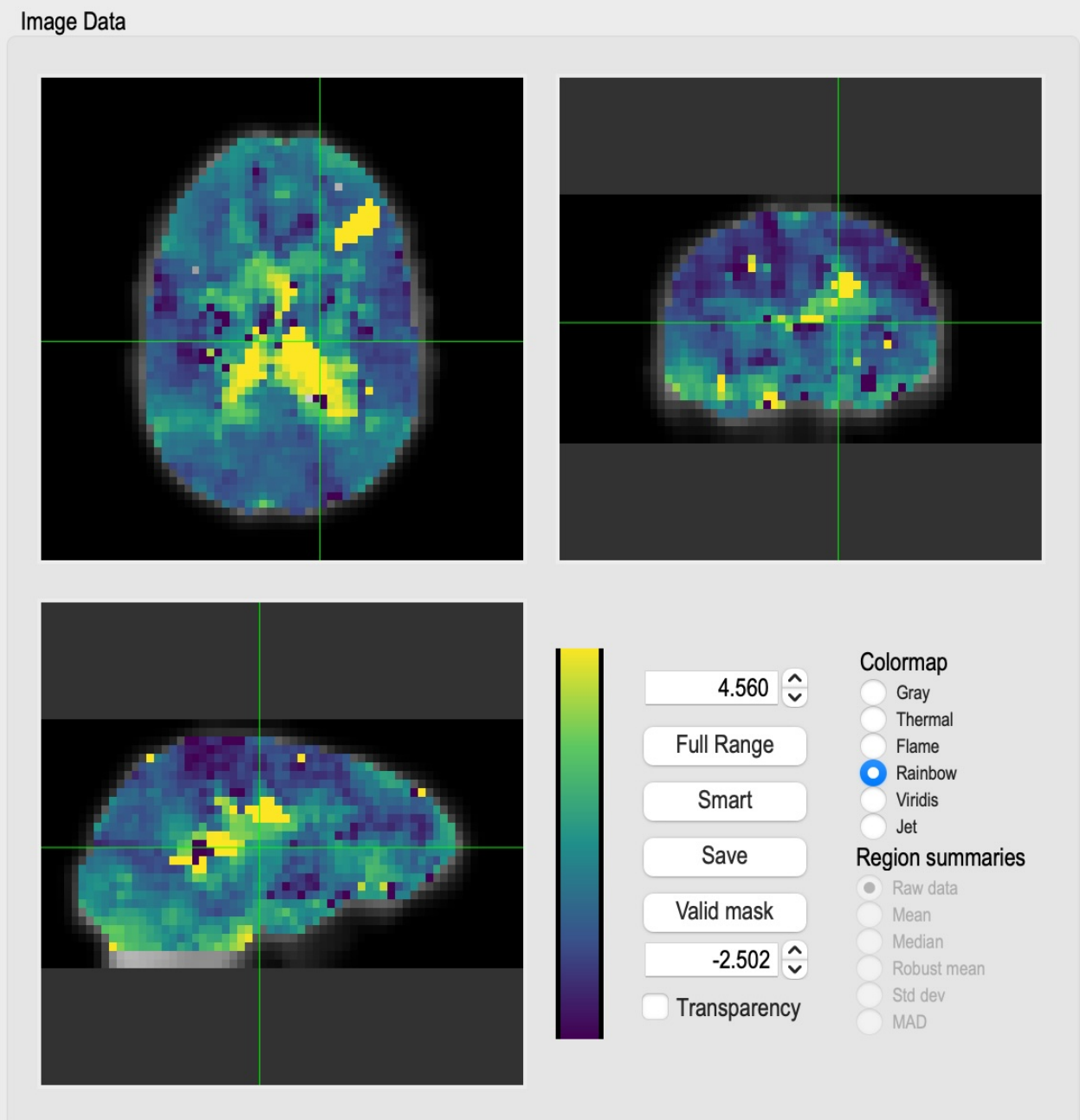
2.17.2 Inputs:

Tidepool loads most of the output files from a rapidtide analysis. The files must all be in the same directory, and use the naming convention and file formats that rapidtide uses.

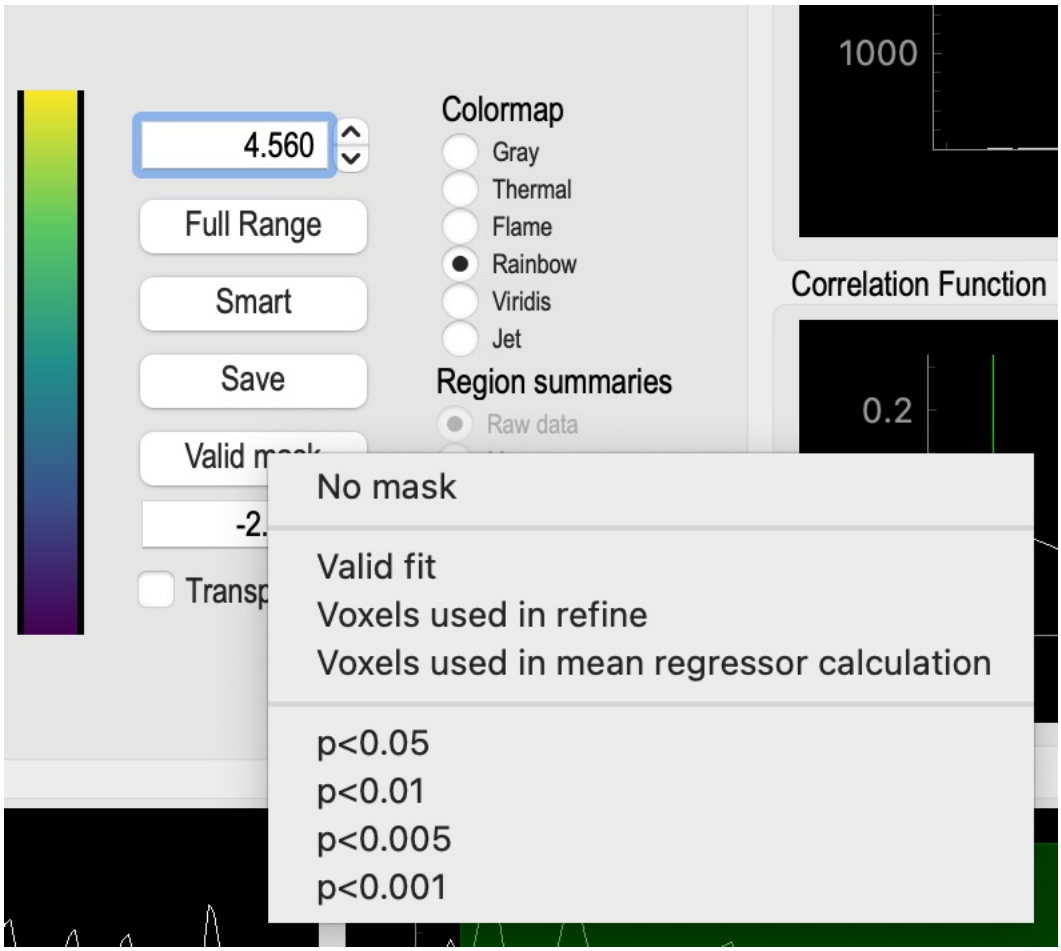
2.17.3 Features:

There are many panels to the tidepool window. They are described in detail below.

Image Data

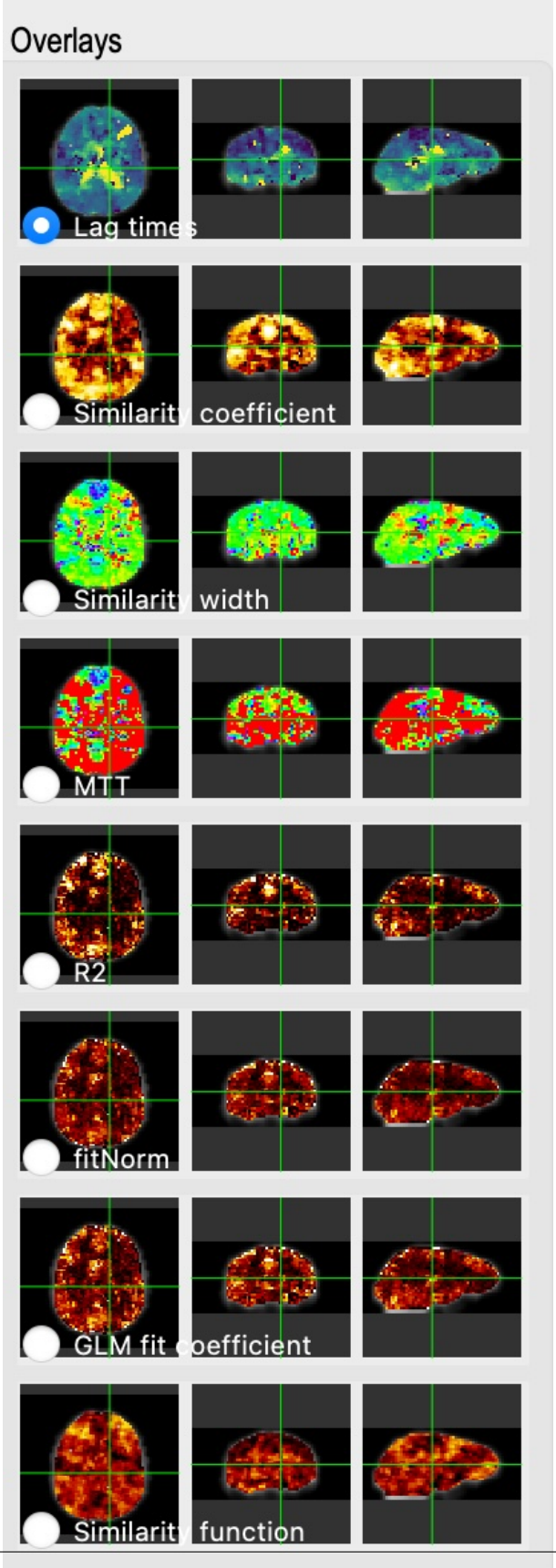


This is the main control of the tidepool window. This shows three orthogonal views of the active map (maxtime in this case) superimposed on an anatomic image (the mean fmri input image to rapidtide by default). Use the left mouse button to select a location in any of the images, and the other two will update to match. The intersecting green lines show the lower left corner of the active location. The lower righthand panel allows you to adjust various parameters, such as the minimum and maximum values of the colormap (set to the "robust range" by default). The "Transparency" button toggles whether values outside of the active range are set to the minimum or maximum colormap value, or are not displayed. The radio buttons in the upper right section of the colormap control panel allow you to change to colormap used from the default values. The "Full Range" button sets the colormap limits to the minimum and maximum values in the map. The "Smart" button sets the colormap limits to the 2% to 98% limits (the "robust range"). The "Save" button saves the three active images to jpeg files. The mask button (below the "Smart" button) indicates what mask to apply when displaying the map. By default, this is the "Valid" mask - all voxels where the rapidtide fit converged. Right clicking on this button gives you a popup window which allows you to select from several other masks, including no mask, the voxels used to set the initial regressor, the voxels used in the final refinement pass, and a range of significance values for the rapidtide fit.



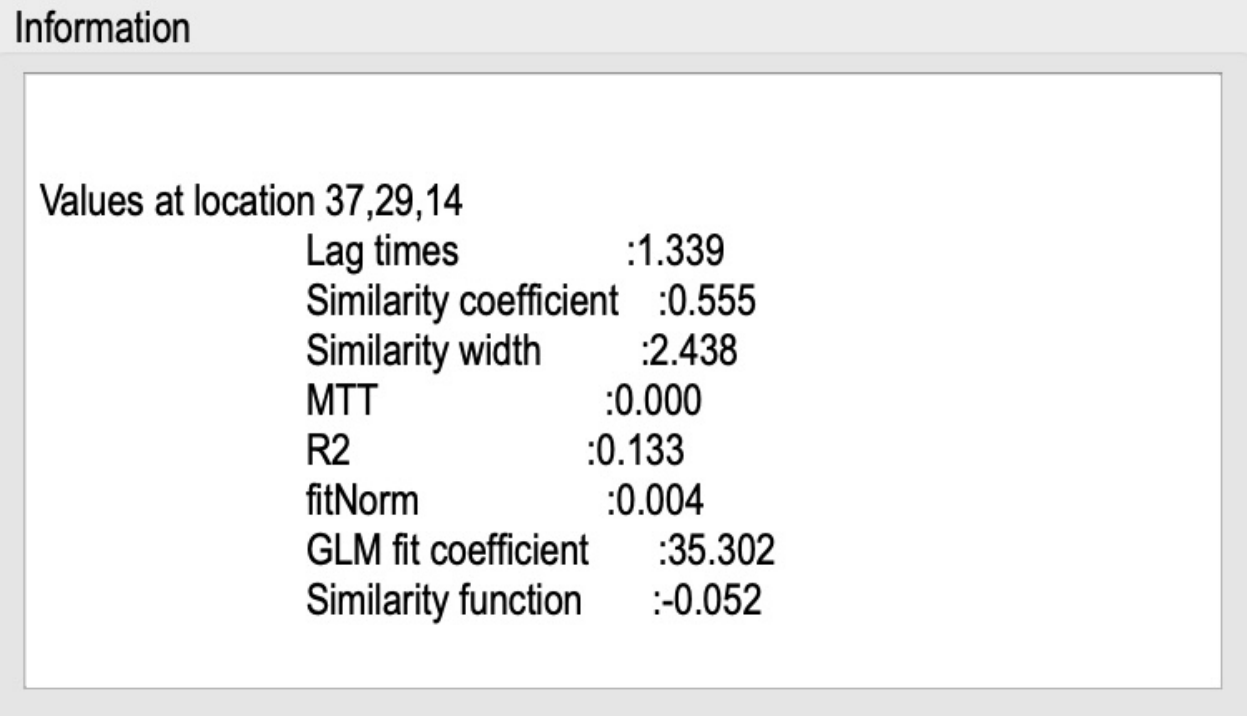
The popup menu for selecting the display mask.

Overlay Selector



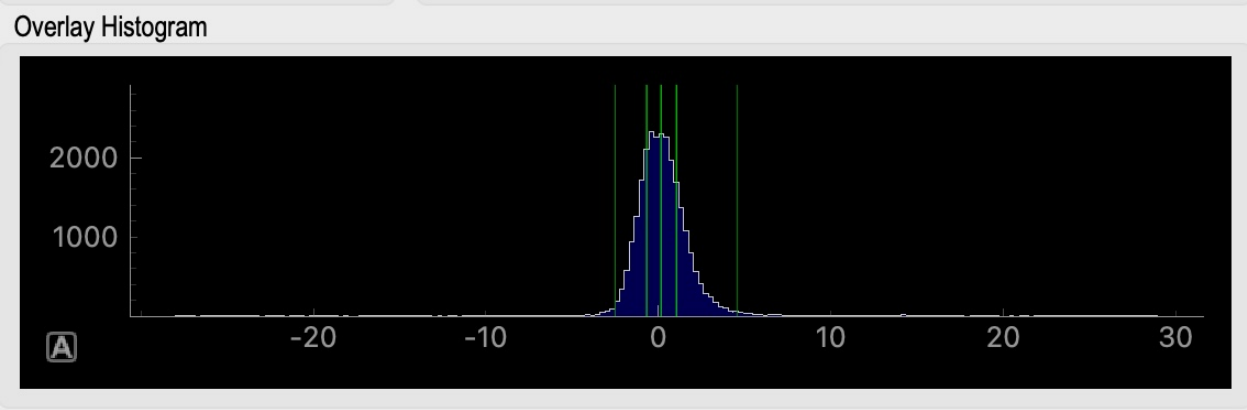
This panel allows you to select which map is displayed in the "Image Data" panel using the radio buttons in the corner of each image. The maps displayed will vary based on the analysis performed. These are all three dimensional maps, with the exception of the bottom map shown - the "Similarity function". This is the full correlation (or other similarity function) used by rapidtide to generate the various maps. When this is loaded, you can use the controls in the "Location" panel to select different time points, or to show the function as a movie.

Information panel



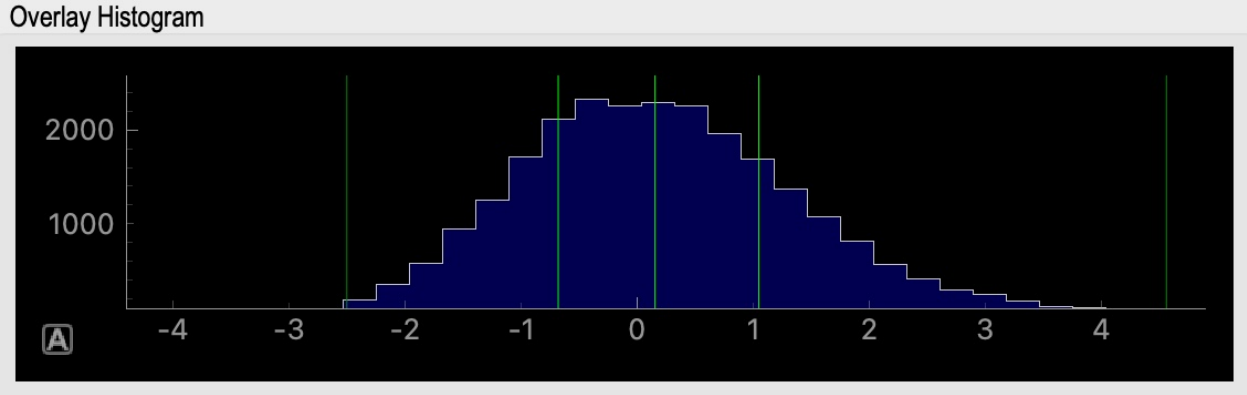
This panel shows the location of the cursor in the "Image Data" panel, and the value of all the loaded maps at that location. If the rapidtide fit failed at that location, all values will be set to zero, and there will be a text description of the reason for the fit failure.

Histogram



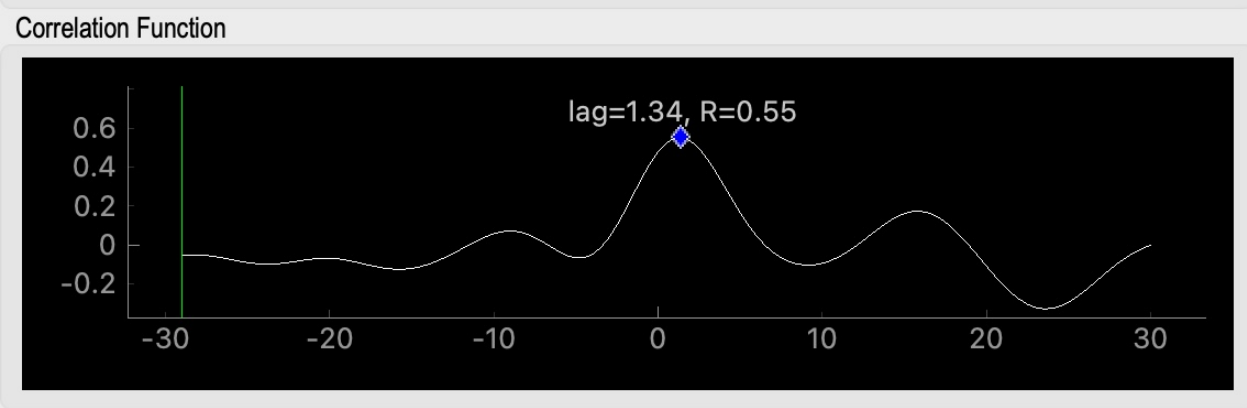
This panel shows the histogram of values displayed (i.e. those selected by the current active mask) in the "Image Data"

panel. By default the range shown is the search range specified during the rapidtide analysis. You can pand and zoom the histogram by clicking and holding the left or right mouse button and moving the mouse. The green bars on the graph show the 2%, 25%, 50%, 75%, and 98% percentile values of the histogram.



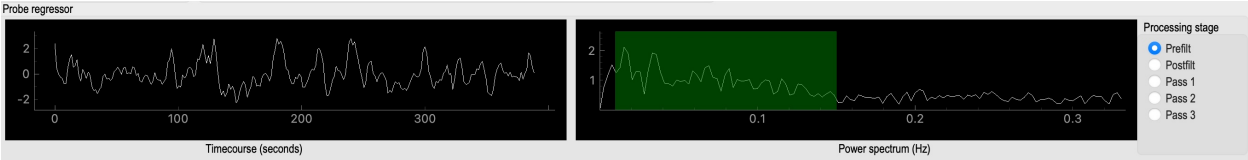
This shows the result of zooming the histogram using the right mouse button. With the mouse in the panel, left click on the "A" in the square box in the lower left of the plot to restore the default display values.

Similarity Function



This panel shows the similarity function (correlation, mutual information) at the location of the cursor in the "Image Data" window. There is a marker showing the maxtime and maxcorr found by the fit (or the text "No valid fit" if the fit failed). This can be used for diagnosing strange fit behavior.

Probe Regressor



This panel shows the probe regressor used in various stages of the rapidtide analysis. The left panel shows the time domain, the right shows the frequency domain, with a translucent green overlay indicating the filter band used in the analysis. The radio buttons on the right select which analysis stage to display: "Prefilt" is the initial probe regressor, either the global mean, or an externally supplied timecourse; "Postfilt" is this regressor after filtering to the active analysis band. "PassX" is the resampled regressor used in each of the analysis passes.

2.17.4 Usage:

If tidepool is called without arguments, a dialog box will appear to allow you to select the maxtime map from the dataset you want to load. This (and other things) can alternately be supplied on the command line as specified below.

2.18 adjustoffset

2.18.1 Description:

Adjust the offset of a rapidtide delay map.

2.18.2 Inputs:

A rapidtide delay map.

2.18.3 Outputs:

A new rapidtide delay map, with a different time offset.

2.18.4 Usage:

Adjust the offset of a rapidtide delay map.

```
usage: adjustoffset [-h] [--includemask MASK[:VALSPEC]]
                    [--excludemask MASK[:VALSPEC]] [--extramask MASK]
                    [--searchrange LAGMIN LAGMAX] [--histbins BINS]
                    [--histonly] [--display] [--pickleft]
                    [--pickleftthresh THRESH] [--setoffset OFFSET]
                    [--norefine] [--debug]
                    inputmap outputroot
```

Positional Arguments

inputmap	The name of the rapidtide maxtime map.
outputroot	The root name for the output files.

Named Arguments

--searchrange Limit fit to a range of lags from LAGMIN to LAGMAX. Default is -10 to 20 seconds.

Default: (-10, 20)

--histbins Number of bins in the entropy histogram (default is 151).

Default: 151

--histonly Only calculate offset histograms - do not perform adjustments.

Default: False

--display Show the delay histogram.

Default: False

--pickleft Choose the leftmost peak of the histogram that exceeds the threshold.

Default: False

--picklefttthresh Fraction of the maximum height that can be considered a peak. Default is 0.33

Default: 0.33

--setoffset Directly set the offset value to OFFSET. Overrides histogram.

--norefine Do not fit the histogram peak.

Default: True

--debug Output debugging information.

Default: False

Masking options

--includemask Only use voxels that are also in file MASK in calculating the offset values (if VALSPEC is given, only voxels with integral values listed in VALSPEC are used).

--excludemask Do not use voxels that are also in file MASK in calculating the offset values (if VALSPEC is given, voxels with integral values listed in VALSPEC are excluded).

--extramask Additional mask to apply to select voxels for adjustment. Zero voxels in this mask will be excluded. If not specified, the corrfit_mask will be used.

2.19 happy

2.19.1 Description:

happy is a new addition to the rapidtide suite. It's complementary to rapidtide - it's focussed on fast, cardiac signals in fMRI, rather than the slow, LFO signals we are usually looking at. It's sort of a Frankenprogram - it has three distinct jobs, which are related, but are very distinct.

The first thing happy does is try to extract a cardiac waveform from the fMRI data. This is something I've been thinking about for a long time. Words go here

The second task is to take this raw estimate of the cardiac waveform, and clean it up using a deep learning filter. The original signal is useful, but pretty gross, but I figured you should be able to exploit the pseudoperiodic nature of the

signal to greatly improve it. This is also a testbed to work on using neural nets to process time domain signals. It seemed like a worthwhile project, so it got grafted in.

The final task (which was actually the initial task, and the reason I wrote happy to begin with) is to implement Henning Voss' totally cool hypersampling with analytic phase projection (guess where the name "happy" comes from). This is fairly straightforward, as Voss describes his method very clearly. But I have lots of data with no simultaneously recorded cardiac signals, and I was too lazy to go find datasets with pleth data to play with, so that's why I did the cardiac waveform extraction part.

2.19.2 Inputs:

Happy needs a 4D BOLD fMRI data file (space by time) as input. This can be Nifti1 or Nifti2. If you have a simultaneously recorded cardiac waveform, it will happily use it, otherwise it will try to construct (and refine) one. NOTE: the 4D input dataset needs to be completely unpreprocessed - gradient distortion correction and motion correction can destroy the relationship between slice number and actual acquisition time, and slice time correction does not behave as expected for aliased signals (which the cardiac component in fMRI most certainly is), and in any case we need the slice time offsets to construct our waveform.

2.19.3 Outputs:

Outputs are space or space by time Nifti or text files, depending on what the input data file was, and some text files containing textual information, histograms, or numbers. File formats and naming follow BIDS conventions for derivative data for fMRI input data. 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.

2.19.4 BIDS Outputs:

Name	Extension(s)	Content	When present
XXX_commandline	.txt	The command line used to run happy	Always
XXX_formattedcomma	.txt	The command line used to run happy, attractively formatted	Always
XXX_desc-rawapp_info	.nii.gz	The analytic phase projection map of the cardiac waveform	Always
XXX_desc-app_info	.nii.gz	The analytic phase projection map of the cardiac waveform, voxelwise minimum subtracted	Always
XXX_desc-normapp_info	.nii.gz	The analytic phase projection map of the cardiac waveform, voxelwise minimum subtracted and normalized	Always
XXX_desc-apppeaks_hist	.tsv.gz, .json	Not sure	Always
XXX_desc-apppeaks_hist_centerof	.txt	Not sure	Always
XXX_desc-apppeaks_hist_peak	.txt	Not sure	Always
XXX_desc-slicerescardfromfmri_ti	.tsv.gz, .json	Cardiac timeseries at the time resolution of slice acquisition ((1/TR * number of slices / multiband factor	Always
XXX_desc-stdrescardfromfmri_tim	.tsv.gz, .json	Cardiac timeseries at standard time resolution (25.O Hz)	Always
XXX_desc-cardpulsefromfmri_tim	.tsv.gz, .json	The average (over time from minimum) of the cardiac waveform over all voxels	Always
XXX_desc-cardiaccyclefromfmri_t	.tsv.gz, .json	The average (over a single cardiac cycle) of the cardiac waveform over all voxels	Always
XXX_desc-cine_info	.nii.gz	Average image of the fMRI data over a single cardiac cycle	Always
XXX_desc-cycleaverage_timeseries	.tsv.gz, .json	Not sure	Always
XXX_desc-maxphase_map	.nii.gz	Map of the average phase where the maximum amplitude occurs for each voxel	Always
XXX_desc-minphase_map	.nii.gz	Map of the average phase where the minimum amplitude occurs for each voxel	Always
XXX_desc-processvoxels_mask	.nii.gz	Map of all voxels used for analytic phase projection	Always
XXX_desc-vessels_map	.nii.gz	Amplitude of variance over a cardiac cycle (large values are assumed to be vessels)	Always
XXX_desc-vessels_mask	.nii.gz	Locations of voxels with variance over a cardiac cycle that exceeds a threshold (assumed to be vessels)	Always
XXX_desc-arteries_map	.nii.gz	High variance vessels with early maximum values within the cardiac cycle	Always
XXX_desc-veins_map	.nii.gz	High variance vessels with late maximum values within the cardiac cycle	Always
XXX_info	.json	Run parameters and derived values found during the run (quality metrics, derived thresholds, etc.)	Always
XXX_memusage	.csv	Memory statistics at multiple checkpoints over the course of the run	Always
XXX_runtimings	.txt	Detailed timing information	Always

2.19.5 Usage:

Hypersampling by Analytic Phase Projection - Yay!.

```
usage: happy [-h] [--cardcalconly] [--skipdlfilter]
             [--usesuperdangerousworkaround] [--slicetimesareinseconds]
             [--model MODELNAME] [--mklthreads NTHREADS] [--numskip SKIP]
             [--motskip SKIP] [--motionfile MOTFILE] [--motionhp HPFREQ]
             [--motionlp LPFREQ] [--nomotorthogonalize] [--motpos]
             [--nomotderiv] [--discardmotionfiltered] [--estmask MASKNAME]
             [--minhr MINHR] [--maxhr MAXHR] [--minhrfilt MINHR]
             [--maxhrfilt MAXHR] [--hilbertcomponents NCOMPS]
             [--envcutoff CUTOFF] [--notchwidth WIDTH] [--invertphysiosign]
             [--cardiacfile FILE[:COL]]
             [--cardiacfreq FREQ | --cardiactstep TSTEP]
             [--cardiacstart START] [--forcehr BPM]
             [--respirationfile FILE[:COL]]
             [--respirationfreq FREQ | --respirationtstep TSTEP]
             [--respirationstart START] [--forcerr BreathsPM] [--spatialglm]
             [--temporalglm] [--stdfreq FREQ] [--outputbins BINS]
             [--gridbins BINS] [--gridkernel {old,gauss,kaiser}]
             [--projmask MASKNAME] [--projectwithraw] [--fliparteries]
             [--arteriesonly] [--version] [--detailedversion]
             [--aliasedcorrelation] [--upsample] [--estimateflow]
             [--noprogressbar] [--infotag tagkey tagvalue] [--debug]
             [--nodetrend DETRENDORDER] [--noorthog] [--disablenotch]
             [--nomask] [--nocensor] [--noappsmooth] [--nophasefilt]
             [--nocardiacalign] [--saveinfoastext] [--saveintermediate]
             [--increaseoutputlevel] [--decreaseoutputlevel]
             fmrifilename slicetimename outputroot
```

Positional Arguments

fmrifilename The input data file (BOLD fmri file or NIRS text file)

slicetimename Text file containing the offset time in seconds of each slice relative to the start of the TR, one value per line, OR the BIDS sidecar JSON file.NB: FSL slicetime files give slice times in fractions of a TR, BIDS sidecars give slice times in seconds. Non-json files are assumed to be the FSL style (fractions of a TR) UNLESS the –slicetimesareinseconds flag is used.

outputroot The root name for the output files

Processing steps

--cardcalconly Stop after all cardiac regressor calculation steps (before phase projection).

Default: False

--skipdlfilter Disable deep learning cardiac waveform filter.

Default: True

--usesuperdangerousworkaround Some versions of tensorflow seem to have some weird conflict with MKL whichI don't seem to be able to fix. If the dl filter bombs complaining about multiple openmp libraries, try rerunning with the secret and inadvisable '–usesuperdangerousworkaround' flag. Good luck!

Default: False

--slicetimesareinseconds If a non-json slicetime file is specified, happy assumes the file is FSL style (slice times are specified in fractions of a TR). Setting this flag overrides this assumption, and interprets the slice time file as being in seconds. This does nothing when the slicetime file is a .json BIDS sidecar.

Default: False

--model Use model MODELNAME for dl filter (default is model_revised - from the revised NeuroImage paper.

Default: "model_revised"

Performance

--mklthreads Use NTHREADS MKL threads to accelerate processing (defaults to 1 - more threads up to the number of cores can accelerate processing a lot, but can really kill you on clusters unless you're very careful. Use at your own risk

Default: 1

Preprocessing

--numskip Skip SKIP tr's at the beginning of the fMRI file (default is 0).

Default: 0

--motskip Skip SKIP tr's at the beginning of the motion regressor file (default is 0).

Default: 0

--motionfile Read 6 columns of motion regressors out of MOTFILE file (.par or BIDS .json) (with timepoints rows) and regress them, their derivatives, and delayed derivatives out of the data prior to analysis.

--motionhp Highpass filter motion regressors to HPFREQ Hz prior to regression.

--motionlp Lowpass filter motion regressors to LPFREQ Hz prior to regression.

--nomotorthogonalize Do not orthogonalize motion regressors prior to regressing them out of the data.

Default: True

--motpos Include motion position regressors.

Default: False

--nomotderiv Do not use motion derivative regressors.

Default: True

--discardmotionfiltered Do not save data after motion filtering.

Default: True

Cardiac estimation tuning

--estmask Generation of cardiac waveform from data will be restricted to voxels in MASKNAME and weighted by the mask intensity. If this is selected, happy will only make a single pass through the data (the initial vessel mask generation pass will be skipped).

--minhr Limit lower cardiac frequency search range to MINHR BPM (default is 40).

Default: 40.0

--maxhr Limit upper cardiac frequency search range to MAXHR BPM (default is 140).

Default: 140.0

--minhrfilt Highpass filter cardiac waveform estimate to MINHR BPM (default is 40).

Default: 40.0

--maxhrfilt Lowpass filter cardiac waveform estimate to MAXHR BPM (default is 1000).

Default: 1000.0

--hilbertcomponents Retain NCOMPS components of the cardiac frequency signal to Hilbert transform (default is 1).

Default: 1

--envcutoff Lowpass filter cardiac normalization envelope to CUTOFF Hz (default is 0.4 Hz).

Default: 0.4

--notchwidth Set the width of the notch filter, in percent of the notch frequency (default is 1.5).

Default: 1.5

--invertphysiosign Invert the waveform extracted from the physiological signal. Use this if there is a contrast agent in the blood.

Default: False

External cardiac waveform options

--cardiacfile Read the cardiac waveform from file FILE. If COL is an integer, and FILE is a text file, use the COL'th column. If FILE is a BIDS format json file, use column named COL. If no file is specified, estimate the cardiac signal from the fMRI data.

--cardiacfreq Cardiac waveform in cardiacfile has sample frequency FREQ (default is 32Hz). NB: –cardiacfreq and –cardiactstep are two ways to specify the same thing.

Default: -32.0

--cardiactstep Cardiac waveform in cardiacfile has time step TSTEP (default is 1/32 sec). NB: –cardiacfreq and –cardiactstep are two ways to specify the same thing.

Default: -32.0

--cardiacstart The time delay in seconds into the cardiac file, corresponding to the first TR of the fMRI file (default is 0.0)

--forcehr Force heart rate fundamental detector to be centered at BPM (overrides peak frequencies found from spectrum). Usefulif there is structured noise that confuses the peak finder.

External respiration waveform options

--respirationfile Read the respiration waveform from file FILE. If COL is an integer, and FILE is a text file, use the COL'th column. If FILE is a BIDS format json file, use column named COL.

--respirationfreq Respiration waveform in respirationfile has sample frequency FREQ (default is 32Hz). NB: –respirationfreq and –respirationtstep are two ways to specify the same thing.

Default: -32.0

--respirationtstep Respiration waveform in respirationfile has time step TSTEP (default is 1/32 sec). NB: –respirationfreq and –respirationtstep are two ways to specify the same thing.

Default: -32.0

--respirationstart The time delay in seconds into the respiration file, corresponding to the first TR of the fMRI file (default is 0.0)

--forcerr Force respiratory rate fundamental detector to be centered at BreathsPM (overrides peak frequencies found from spectrum). Usefulif there is structured noise that confuses the peak finder.

Output processing

--spatialglm Generate framewise cardiac signal maps and filter them out of the input data.

Default: False

--temporalglm Generate voxelwise aliased synthetic cardiac regressors and filter them out of the input data.

Default: False

Output options

--stdfreq Frequency to which the physiological signals are resampled for output. Default is 25.

Default: 25.0

Phase projection tuning

--outputbins Number of output phase bins (default is 32).

Default: 32

--gridbins Width of the gridding kernel in output phase bins (default is 3.0).

Default: 3.0

--gridkernel Possible choices: old, gauss, kaiser

Convolution gridding kernel. Default is kaiser

Default: "kaiser"

--projmask Phase projection will be restricted to voxels in MASKNAME (overrides normal intensity mask.)

--projectwithraw Use fMRI derived cardiac waveform as phase source for projection, even if a plethysmogram is supplied.

Default: False

--fliparteries Attempt to detect arterial signals and flip over the timecourses after phase projection (since relative arterial blood susceptibility is inverted relative to venous blood).

Default: False

--arteriesonly Restrict cardiac waveform estimation to putative arteries only.

Default: False

Version options

--version Show simplified version information and exit

--detailedversion Show detailed version information and exit

Miscellaneous options.

--aliasedcorrelation Attempt to calculate absolute delay using an aliased correlation (experimental).

Default: False

--upsample Attempt to temporally upsample the fMRI data (experimental).

Default: False

--estimateflow Estimate blood flow using optical flow (experimental).

Default: False

--noprogressbar Will disable showing progress bars (helpful if stdout is going to a file).

Default: True

--infotag Additional key, value pairs to add to the info json file (useful for tracking analyses).

Debugging options (probably not of interest to users)

--debug Turn on debugging information.
Default: False

--nodetrend Disable data detrending.
Default: 3

--noorthog Disable orthogonalization of motion confound regressors.
Default: True

--disablenotch Disable subharmonic notch filter.
Default: False

--nomask Disable data masking for calculating cardiac waveform.
Default: True

--nocensor Bad points will not be excluded from analytic phase projection.
Default: True

--noappsmooth Disable smoothing app file in the phase direction.
Default: True

--nophasefilt Disable the phase trend filter (probably not a good idea).
Default: True

--nocardiacalign Disable alignment of pleth signal to fMRI derived cardiac signal.
Default: True

--saveinfoastext Save the info file in text format rather than json.
Default: True

--saveintermediate Save some data from intermediate passes to help debugging.
Default: False

--increaseoutputlevel Increase the number of intermediate output files.
Default: 0

--decreaseoutputlevel Decrease the number of intermediate output files.
Default: 0

2.19.6 Example:

Extract the cardiac waveform and generate phase projections

Case 1: When you don't have a pleth recording

There are substantial improvements to the latest versions of happy. In the old versions, you actually had to run happy twice - the first time to estimate the vessel locations, and the second to actually derive the waveform. Happy now combines these operations interpolation a single run with multiple passes - the first pass locates voxels with high variance, labels them as vessels, then reruns the derivation, restricting the cardiac estimation to these high variance voxels. This gives substantially better results.

Using the example data in the example directory, try the following:

```
happy \
    rapidtide/data/examples/src/sub-HAPPYTEST.nii.gz \
    rapidtide/data/examples/src/sub-HAPPYTEST.json \
    rapidtide/data/examples/dst/happytest
```

This will perform a happy analysis on the example dataset. To see the extracted cardiac waveform (original and filtered), you can use showtc (also part of them rapidtide package):

```
showtc \
    rapidtide/data/examples/src/happytest_desc-slicerescardfromfmri_timeseries.
↪json:cardiacfromfmri,cardiacfromfmri_dlfiltered \
    --format separate
```

2.20 happy2std

2.20.1 Description:

This is a utility for registering happy output maps to standard coordinates. NB: this will only work if you have a working FSL installation.

2.20.2 Inputs:

inputfileroot - the root name of the input NIFTI files (up and including the 'desc' but not the underscore).

outputdir - The location for the output files

featdirectory - Either a feat-like directory (x.feat or x.ica) or an fmriprep derivativesanat directory where the information needed for registration to standard space can be found

2.20.3 Outputs:

The happy maps, transformed to MNI152 space

2.20.4 Usage:

Register happy output maps to standard space.

```
usage: happy2std inputfileroot outputdir featdirectory
```

Positional Arguments

inputfileroot the root name of the input NIFTI files (up and including the 'desc' but not the underscore).

outputdir The location for the output files

featdirectory Either a feat-like directory (x.feat or x.ica) or an fmriprep derivativesanat directory where the information needed for registration to standard space can be found

Named Arguments

--all Also transform the corrout file (warning - file may be huge).

Default: False

--hires Transform to match the high resolution anatomic image rather than the standard.

Default: False

--linear Only do linear transformation, even if warpfile exists.

Default: False

--onefile Align a single file, specified by name without extension (ignore INPUTFILEROOT).

--fake Output, but do not execute, alignment commands.

Default: False

2.21 proj2flow

2.21.1 Description:

2.21.2 Inputs:

2.21.3 Outputs:

2.21.4 Usage:

Convert phase projection movie to velocity map.

```
usage: proj2flow [-h] [--debug] inputfilename outputroot
```

Positional Arguments

inputfilename The name of the input nifti file.

outputroot The root name of the output nifti files.

Named Arguments

--debug Turn on debugging information.

Default: False

2.22 showxcorr_legacy

2.22.1 Description:

Like rapidtide, but for single time courses. Takes two text files as input, calculates and displays the time lagged crosscorrelation between them, fits the maximum time lag, and estimates the significance of the correlation. It has a range of filtering, windowing, and correlation options. This is the old interface - for new analyses you should use showxcorrx.

2.22.2 Inputs:

showxcorr requires two text files containing timecourses with the same sample rate, one timepoint per line, which are to be correlated, and the sample rate.

2.22.3 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.

2.22.4 Usage:

```
usage: showxcorr timecourse1 timecourse2 samplerate [-l LABEL] [-s STARTTIME]␣
↪[-D DURATION] [-d] [-F LOWERFREQ,UPPERFREQ[,LOWERSTOP,UPPERSTOP]] [-V] [-L]␣
↪[-R] [-C] [-t] [-w] [-f] [-z FILENAME] [-N TRIALS]

required arguments:
        timcoursefile1: text file containing a timeseries
        timcoursefile2: text file containing a timeseries
        samplerate:     the sample rate of the timecourses, in Hz

optional arguments:
    -t            - detrend the data
    -w            - prewindow the data
    -l LABEL      - label for the delay value
    -s STARTTIME  - time of first datapoint to use in seconds in the first file
    -D DURATION   - amount of data to use in seconds
```

(continues on next page)

(continued from previous page)

```
    -r RANGE      - restrict peak search range to +/- RANGE seconds (default is
                    +/-15)
    -d            - turns off display of graph
    -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
    -T            - trim data to match
    -A            - print data on a single summary line
    -a            - if summary mode is on, add a header line showing what␣
↪values
                    mean
    -f            - negate (flip) second regressor
    -z FILENAME   - use the columns of FILENAME as controlling variables and
                    return the partial correlation
    -N TRIALS     - estimate significance thresholds by Monte Carlo with TRIALS
                    repetition
```

2.23 showxcorrx

2.23.1 Description:

This is the newest, most avant-garde version of showxcorr. Because it's an x file, it's more fluid and I don't guarantee that it will keep a stable interface (or even work at any given time). But every time I add something new, it goes here. The goal is eventually to make this the "real" version. Unlike rapidtide, however, I've let it drift quite a bit without syncing it because some people here actually use showxcorr and I don't want to disrupt workflows...

2.23.2 Inputs:

showxcorrx requires two text files containing timecourses with the same sample rate, one timepoint per line, which are to be correlated, and the sample rate.

2.23.3 Outputs:

showxcorrx 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.

2.23.4 Usage:

Calculate and display crosscorrelation between two timeseries.

```
usage: showxcorrx [-h] [--samplerate FREQ | --sampletime TSTEP]
                  [--searchrange LAGMIN LAGMAX] [--fixdelay DELAYTIME]
                  [--timerange START END]
                  [--windowfunc {hamming,hann,blackmanharris,None}]
                  [--zeropadding PADVAL]
                  [--filterband {None,vlf,lfo,resp,cardiac,hrv_ulf,hrv_vlf,hrv_lf,hrv_hf,
↪hrv_vhf,lfo_legacy}]
                  [--filterfreqs LOWERPASS UPPERPASS]
                  [--filterstopfreqs LOWERSTOP UPPERSTOP]
                  [--filtertype {trapezoidal,brickwall,butterworth}]
                  [--butterorder ORDER] [--padseconds SECONDS]
                  [--detrendorder ORDER] [--trimdata]
                  [--corrweighting {None,phat,liang,eckart}] [--invert]
                  [--label LABEL] [--partialcorr FILE] [--cepstral]
                  [--calccsd] [--calccoherence]
                  [--permutationmethod {shuffle,phaserandom}]
                  [--numnull NREPS] [--skipsighistfit]
                  [--similaritymetric {correlation,mutualinfo,hybrid}]
                  [--sigmamax SIGMAMAX] [--sigmamin SIGMAMIN]
                  [--mutualinfosmoothingtime TAU] [--outputfile FILE]
                  [--corroutputfile FILE] [--summarymode] [--labelline]
                  [--title TITLE] [--xlabel LABEL] [--ylabel LABEL]
                  [--legend LEGEND] [--legendloc LOC] [--color COLOR]
                  [--nolegend] [--noxax] [--noyax] [--linewidth LINEWIDTH]
                  [--tofile FILENAME] [--fontscalefac FAC] [--saveres DPI]
                  [--noprogressbar] [--nodisplay] [--nonorm] [--nprocs NPROCS]
                  [--debug] [--verbose]
                  infilename1 infilename2
```

Positional Arguments

infilename1 Text file containing a timeseries. Select column COLNUM if multicolumn file

infilename2 Text file containing a timeseries. Select column COLNUM if multicolumn file

General Options

--samplerate Set the sample rate of the data file to FREQ. If neither samplerate or sampletime is specified, sample rate is 1.0.

Default: auto

--sampletime Set the sample rate of the data file to 1.0/TSTEP. If neither samplerate or sampletime is specified, sample rate is 1.0.

Default: auto

--searchrange Limit fit to a range of lags from LAGMIN to LAGMAX. Default is -30.0 to 30.0 seconds.

Default: (-30.0, 30.0)

--fixdelay Don't fit the delay time - set it to DELAYTIME seconds for all voxels.

--timerange Limit analysis to data between timepoints START and END in the input file. If END is set to -1, analysis will go to the last timepoint. Negative values of START will be set to 0. Default is to use all timepoints.

Default: (-1, -1)

Windowing options

--windowfunc Possible choices: hamming, hann, blackmanharris, None

Window function to use prior to correlation. Options are hamming, hann, blackmanharris, and None. Default is hamming

Default: "hamming"

--zeropadding Pad input functions to correlation with PADVAL zeros on each side. A PADVAL of 0 does circular correlations, positive values reduce edge artifacts. Set PADVAL < 0 to set automatically. Default is 0.

Default: 0

Filtering options

--filterband Possible choices: None, vlf, lfo, resp, cardiac, hrv_ulf, hrv_vlf, hrv_lf, hrv_hf, hrv_vhf, lfo_legacy

Filter timecourses to specific band. Use "None" to disable filtering. Default is "lfo".

Default: "lfo"

--filterfreqs Filter timecourses to retain LOWERPASS to UPPERPASS. If –filterstopfreqs is not also specified, LOWERSTOP and UPPERSTOP will be calculated automatically.

--filterstopfreqs Filter timecourses to with stop frequencies LOWERSTOP and UPPERSTOP. LOWERSTOP must be <= LOWERPASS, UPPERSTOP must be >= UPPERPASS. Using this argument requires the use of –filterfreqs.

--filtertype Possible choices: trapezoidal, brickwall, butterworth

Filter timecourses using a trapezoidal FFT, brickwall FFT, or butterworth bandpass filter. Default is "trapezoidal".

Default: "trapezoidal"

--butterorder Set order of butterworth filter (if used). Default is 6.

Default: 6

--padseconds The number of seconds of padding to add to each end of a filtered timecourse to reduce end effects. Default is 30.0.

Default: 30.0

Preprocessing options

--detrendorder Set order of trend removal (0 to disable, default is 1 - linear).

Default: 1

--trimdata Trimming data to match

Default: False

--corrweighting Possible choices: None, phat, liang, eckart

Method to use for cross-correlation weighting. Default is None.

Default: "None"

--invert Invert the second timecourse prior to calculating similarity.

Default: False

--label Label for the delay value.

Default: "None"

--partialcorr Use the columns of FILE as controlling variables and return the partial correlation.

Additional calculations

--cepstral Check time delay using Choudhary's cepstral technique.

Default: False

--calccsd Calculate the cross-spectral density.

Default: False

--calccoherence Calculate the coherence.

Default: False

Similarity function options

--similaritymetric Possible choices: correlation, mutualinfo, hybrid

Similarity metric for finding delay values. Choices are 'correlation' (default), 'mutualinfo', and 'hybrid'.

Default: "correlation"

--sigmamax Reject lag fits with linewidth wider than SIGMAMAX Hz. Default is 1000.0 Hz.

Default: 1000.0

--sigmamin Reject lag fits with linewidth narrower than SIGMAMIN Hz. Default is 0.25 Hz.

Default: 0.25

--mutualinfosmoothingtime Time constant of a temporal smoothing function to apply to the mutual information function. Default is 3.0 seconds. TAU <=0.0 disables smoothing.

Default: 3.0

Output options

--outputfile Save results to FILE.

--corroutputfile Write correlation function to FILE.

--summarymode Print all results on a single line.

Default: "False"

--labelline Print a header line identifying fields in the summary line.

Default: "False"

General plot appearance options

--title Use TITLE as the overall title of the graph.

Default: ""

--xlabel Label for the plot x axis.

Default: ""

--ylabel Label for the plot y axis.

Default: ""

--legend Legends for the timecourse.

--legendloc Integer from 0 to 10 inclusive specifying legend location. Legal values are: 0: best, 1: upper right, 2: upper left, 3: lower left, 4: lower right, 5: right, 6: center left, 7: center right, 8: lower center, 9: upper center, 10: center. Default is 2.

Default: 2

--color Color of the timecourse plot.

--nolegend Turn off legend label.

Default: True

--noxax Do not show x axis.

Default: True

--noyax Do not show y axis.

Default: True

--linewidth Linewidth (in points) for plot. Default is 1.

--tofile Write figure to file FILENAME instead of displaying on the screen.

--fontscalefac Scaling factor for annotation fonts (default is 1.0).

Default: 1.0

--saveres Write figure to file at DPI dots per inch (default is 1000).

Default: 1000

Miscellaneous options

--noprogressbar Will disable showing progress bars (helpful if stdout is going to a file).

Default: True

--nodisplay Do not plot the data (for noninteractive use)

Default: True

--nonorm Will disable normalization of the mutual information function.

Default: True

--nprocs Use NPROCS worker processes for multiprocessing. Setting NPROCS to less than 1 sets the number of worker processes to n_cpus - 1.

Default: 1

Debugging options

--debug Enable additional debugging output.

Default: False

--verbose Print out more debugging information

Default: False

2.24 showtc

2.24.1 Description:

A very simple command line utility that takes a text file and plots the data in it in a matplotlib window. That's it. A good tool for quickly seeing what's in a file. Has some options to make the plot prettier.

2.24.2 Inputs:

Text files containing time series data

2.24.3 Outputs:

None

2.24.4 Usage:

Plots the data in text files.

```
usage: showtc [-h] [--samplerate FREQ | --sampletime TSTEP]
              [--displaytype {time,power,phase}]
              [--format {overlaid,separate,separatelinked}] [--waterfall]
              [--voffset OFFSET] [--transpose] [--normall] [--title TITLE]
              [--xlabel LABEL] [--ylabel LABEL]
              [--legends LEGEND[,LEGEND[,LEGEND...]]] [--legendloc LOC]
              [--colors COLOR[,COLOR[,COLOR...]]] [--nolegend] [--noxax]
              [--noyax] [--linewidth LINEWIDTH[,LINEWIDTH[,LINEWIDTH...]]]
              [--tofile FILENAME] [--fontscalefac FAC] [--saveres DPI]
              [--starttime START] [--endtime END] [--numskip NUM] [--debug]
              [--version] [--detailedversion]
              textfilenames [textfilenames ...]
```

Positional Arguments

textfilenames One or more input files, with optional column specifications

Named Arguments

--samplerate Set the sample rate of the data file to FREQ. If neither samplerate or sampletime is specified, sample rate is 1.0.

Default: auto

--sampletime Set the sample rate of the data file to 1.0/TSTEP. If neither samplerate or sampletime is specified, sample rate is 1.0.

Default: auto

--displaytype Possible choices: time, power, phase

Display data as time series (default), power spectrum, or phase spectrum.

Default: “time”

--format Possible choices: overlaid, separate, separatelinked

Display data overlaid (default), in individually scaled windows, or in separate windows with linked scaling.

Default: “overlaid”

--waterfall Display multiple timecourses in a waterfall plot.

Default: False

--voffset Plot multiple timecourses with OFFSET between them (use negative OFFSET to set automatically).

Default: 0.0

--transpose Swap rows and columns in the input files.

Default: False

--normall Normalize all displayed timecourses to unit standard deviation and zero mean.

Default: False

--starttime Start plotting at START seconds (default is the start of the data).

--endtime Finish plotting at END seconds (default is the end of the data).

--numskip Skip NUM lines at the beginning of each file (to get past header lines).

Default: 0

--debug Output additional debugging information.

Default: False

General plot appearance options

--title Use TITLE as the overall title of the graph.

Default: “”

--xlabel Label for the plot x axis.

Default: “”

--ylabel Label for the plot y axis.

Default: “”

--legends Comma separated list of legends for each timecourse.

--legendloc Integer from 0 to 10 inclusive specifying legend location. Legal values are: 0: best, 1: upper right, 2: upper left, 3: lower left, 4: lower right, 5: right, 6: center left, 7: center right, 8: lower center, 9: upper center, 10: center. Default is 2.

Default: 2

--colors Comma separated list of colors for each timecourse.

--nolegend Turn off legend label.

Default: True

--noxax Do not show x axis.

Default: True

--noyax Do not show y axis.

Default: True

--linewidth A comma separated list of linewidths (in points) for plots. Default is 1.

--tofile Write figure to file FILENAME instead of displaying on the screen.

--fontscalefac Scaling factor for annotation fonts (default is 1.0).

Default: 1.0

--saveres Write figure to file at DPI dots per inch (default is 1000).

Default: 1000

Version options

--version Show simplified version information and exit

--detailedversion Show detailed version information and exit

2.25 showxy

2.25.1 Description:

2.25.2 Inputs:

2.25.3 Outputs:

2.25.4 Usage:

Plots xy data in text files.

```
usage: showxy [-h] [--xlabel XLABEL] [--ylabel YLABEL] [--xrange XMIN XMAX]
              [--yrange YMIN YMAX] [--title TITLE] [--outputfile FILENAME]
              [--fontscalefac FAC] [--saveres DPI]
              [--legends LEGEND[,LEGEND[,LEGEND...]]] [--legendloc LOC]
              [--colors COLOR[,COLOR[,COLOR...]]] [--blandaltman] [--usex]
              [--noannotate] [--usepoints] [--dobars] [--debug]
              textfilenames [textfilenames ...]
```

Positional Arguments

textfilenames One or more text files containing whitespace separated x y data, one point per line.

Named Arguments

--xlabel Use XLABEL on the x axis.

--ylabel Use YLABEL on the y axis.

--xrange Limit x display range to XMIN to XMAX.

--yrange Limit y display range to YMIN to YMAX.

--title Use TITLE at the top of the graph.

--outputfile Save plot to FILENAME rather than displaying on the screen.

--fontscalefac Scaling factor for annotation fonts (default is 1.0).

Default: 1.0

--saveres Write figure to file at DPI dots per inch (default is 1000).

Default: 1000

--legends Comma separated list of legends for each timecourse.

--legendloc	Integer from 0 to 10 inclusive specifying legend location. Legal values are: 0: best, 1: upper right, 2: upper left, 3: lower left, 4: lower right, 5: right, 6: center left, 7: center right, 8: lower center, 9: upper center, 10: center. Default is 2. Default: 2
--colors	Comma separated list of colors for each timecourse.
--blandaltman	Make a Bland-Altman plot. Default: False
--usex	Use x instead of (y + x)/2 in Bland-Altman plot. Default: False
--noannotate	Hide annotation on Bland-Altman plots. Default: True
--usepoints	Plot as individual values (do not connect points) Default: False
--dobars	Plot bars rather than lines. Default: False
--debug	Print additional internal information. Default: False

2.26 showhist

2.26.1 Description:

Another simple command line utility that displays the histograms generated by rapidtide.

2.26.2 Inputs:

A textfile generated by rapidtide containing histogram information

2.26.3 Outputs:

None

2.26.4 Usage:

Plots xy histogram data in text file.

```
usage: showhist [-h] [--xlabel XLABEL] [--ylabel YLABEL] [--title TITLE]
                [--outputfile FILENAME] [--dobars] [--calcdist] [--debug]
                infilename
```

Positional Arguments

infilename	a text file containing histogram data

Named Arguments

--xlabel	Use XLABEL on the x axis.
--ylabel	Use YLABEL on the y axis.
--title	Use TITLE at the top of the graph.
--outputfile	Save plot to FILENAME rather than displaying on the screen.
--dobars	Plot bars rather than lines. Default: False
--calcdist	Make a histogram out of the data. Default: False
--debug	Print additional internal information. Default: False

2.27 spectrogram

2.27.1 Description:

2.27.2 Inputs:

2.27.3 Outputs:

2.27.4 Usage:

Computes and shows the spectrogram of a text file.

```
usage: spectrogram [-h] [--nperseg NPERSEG] [--debug] textfilename samplerate
```

Positional Arguments

textfilename	The input data file (text file containing a timecourse, one point per line).
samplerate	Sample rate in Hz.

Named Arguments

--nperseg The number of points to include in each spectrogram (default is 128).

Default: 128

--debug Enable additional debugging output.

Default: False

2.28 glmfilt

2.28.1 Description:

Uses a GLM filter to remove timecourses (1D text files or 4D NIFTI files) from 4D NIFTI files.

2.28.2 Inputs:

2.28.3 Outputs:

2.28.4 Usage:

Fits and removes the effect of voxel specific and/or global regressors.

```
usage: glmfilt [-h] [--numskip NUMSKIP] [--evfile EVFILE [EVFILE ...]]
               [--dmask DATAMASK] [--limitoutput]
               inputfile outputroot
```

Positional Arguments

inputfile The name of the 3 or 4 dimensional nifti file to fit.

outputroot The root name for all output files.

Named Arguments

--numskip The number of points to skip at the beginning of the timecourse when fitting. Default is 0.

Default: 0

--evfile One or more files (text timecourse or 4D NIFTI) containing signals to regress out.

--dmask Use DATAMASK to specify which voxels in the data to use.

--limitoutput Only save the filtered data and the R value.

Default: True

2.29 atlasaverage

2.29.1 Description:

Average data within atlas regions.

2.29.2 Inputs:

datafile - The name of the 3 or 4D nifti file with the data to be averaged over atlas regions. templatefile - The name of the atlas region NIFTI file outputroot - The root name of the output files.

2.29.3 Outputs:

A csv file containing the summary metrics for each region in the atlas.

2.29.4 Usage:

Average data within atlas regions.

```
usage: atlasaverage [-h] [--normmethod {none,pct,var,std,p2p}]
                    [--summarymethod {mean,median,sum}] [--ignorezeros]
                    [--regionlistfile REGIONLISTFILE]
                    [--includemask MASK[:VALSPEC]]
                    [--excludemask MASK[:VALSPEC]] [--extramask MASK]
                    [--headerline] [--datalabel LABEL] [--debug]
                    datafile templatefile outputroot
```

Positional Arguments

datafile The name of the 3 or 4D nifti file with the data to be averaged over atlas regions.

templatefile The name of the atlas region NIFTI file

outputroot The root name of the output files.

Named Arguments

--normmethod Possible choices: none, pct, var, std, p2p

Normalization to apply to input timecourses (in addition to demeaning) prior to combining. Choices are 'none' (no normalization, default), 'pct' (divide by mean), 'var' (unit variance), 'std' (unit standard deviation), and 'p2p' (unit range).

Default: "none"

--summarymethod Possible choices: mean, median, sum

Method to summarize a region. Choices are 'mean' (default), 'median', and 'sum'.

Default: "mean"

--ignorezeros Zero value voxels will not be included in calculation of summary statistics.

Default: False

--regionlistfile The name of of a text file containing the integer region numbers to summarize, one per line. Values that do not exist in the atlas will return NaNs.

--includemask Only use atlas voxels that are also in file MASK in calculating the region summaries (if VALSPEC is given, only voxels with integral values listed in VALSPEC are used).

--excludemask Do not use atlas voxels that are also in file MASK in calculating the region summaries (if VALSPEC is given, voxels with integral values listed in VALSPEC are excluded).

--extramask Additional mask to apply select voxels for region summaries. Zero voxels in this mask will be excluded.

--headerline Add a header line to the text output summary of 3D files.

Default: False

--datalabel Label to add to the beginning of the text summary line.

--debug Output additional debugging information.

Default: False

2.30 ccorrica

2.30.1 Description:

Find temporal crosscorrelations between the columns of an input file

2.30.2 Inputs:

A text file containing one or more timeseries columns. Use [:COLSPEC] to select which column(s) to use, where COLSPEC is an integer, a column separated list of ranges, or a comma separated set of column names (if input file is BIDS). Default is to use all columns.

2.30.3 Outputs:

Various NIFTI and text files with the correlation information.

2.30.4 Usage:

Find temporal crosscorrelations between a set of timecourses

```
usage: ccorrica [-h] [--samplerate FREQ | --sampletstep TSTEP]
                [--windowfunc {hamming,hann,blackmanharris,None}]
                [--zeropadding PADVAL] [--searchrange LAGMIN LAGMAX]
                [--filterband {None,vlf,lfo,resp,cardiac,hrv_ulf,hrv_vlf,hrv_lf,hrv_hf,
```

(continues on next page)

(continued from previous page)

```
↪hrv_vhf,lfo_legacy}]
               [--filterfreqs LOWERPASS UPPERPASS]
               [--filterstopfreqs LOWERSTOP UPPERSTOP]
               [--corrweighting {None,phat,liang,eckart}]
               [--detrendorder DETRENDORDER]
               [--oversampfactor OVERSAMPFACTOR] [--debug]
               timecoursefile outputroot
```

Positional Arguments

timecoursefile Text file containing one or more timeseries columns. Use [:COLSPEC] to select which column(s) to use, where COLSPEC is an integer, a column separated list of ranges, or a comma separated set of column names (if input file is BIDS). Default is to use all columns

outputroot Root name for the output files

Named Arguments

--samplerate Timecourses in file have sample frequency FREQ (default is 1.0Hz) NB: –samplerate and –sampletstep) are two ways to specify the same thing.

Default: auto

--sampletstep Timecourses in file have sample timestep TSTEP (default is 1.0s) NB: –samplerate and –sampletstep) are two ways to specify the same thing.

Default: auto

--searchrange Limit fit to a range of lags from LAGMIN to LAGMAX. Default is -30.0 to 30.0 seconds.

Default: (-30.0, 30.0)

--corrweighting Possible choices: None, phat, liang, eckart

Method to use for cross-correlation weighting. Default is None.

Default: “None”

--detrendorder Detrending order (default is 1 - linear). Set to 0 to disable

Default: 1

--oversampfactor Factor by which to oversample timecourses prior to correlation. Default is 1. If set negative, factor will be set automatically.

Default: 1

--debug Enable additional debugging output.

Default: False

Windowing options

--windowfunc Possible choices: hamming, hann, blackmanharris, None

Window function to use prior to correlation. Options are hamming, hann, blackmanharris, and None. Default is hamming

Default: "hamming"

--zeropadding Pad input functions to correlation with PADVAL zeros on each side. A PADVAL of 0 does circular correlations, positive values reduce edge artifacts. Set PADVAL < 0 to set automatically. Default is 0.

Default: 0

Filtering options

--filterband Possible choices: None, vlf, lfo, resp, cardiac, hrv_ulf, hrv_vlf, hrv_lf, hrv_hf, hrv_vhf, lfo_legacy

Filter timecourses to specific band. Use "None" to disable filtering. Default is "lfo".

Default: "lfo"

--filterfreqs Filter timecourses to retain LOWERPASS to UPPERPASS. If –filterstopfreqs is not also specified, LOWERSTOP and UPPERSTOP will be calculated automatically.

--filterstopfreqs Filter timecourses to with stop frequencies LOWERSTOP and UPPERSTOP. LOWERSTOP must be <= LOWERPASS, UPPERSTOP must be >= UPPERPASS. Using this argument requires the use of –filterfreqs.

2.31 tcfrom2col

2.31.1 Description:

A simple command line that takes an FSL style 2 column regressor file and generates a time course (waveform) file. FSL 3 column files are text files containing one row per "event". Each row has three columns: start time in seconds, duration in seconds, and waveform value. The output waveform is zero everywhere that is not covered by an "event" in the file.

2.31.2 Inputs:

An FSL style two column text file (start time, duration)

2.31.3 Outputs:

A single column text file containing the waveform

2.31.4 Usage:

Plots the data in text files.

```
usage: tcfrom2col [-h] [--debug] infilename timestep numpoints outfilename
```

Positional Arguments

infilename	the name of the input two column file
timestep	the time step of the output time course in seconds
numpoints	the number of output time points
outfilename	the name of the output time course file

Named Arguments

--debug	turn on additional debugging output Default: False

2.32 tcfrom3col

2.32.1 Description:

A simple command line that takes an FSL style 3 column regressor file and generates a time course (waveform) file. FSL 3 column files are text files containing one row per "event". Each row has three columns: start time in seconds, duration in seconds, and waveform value. The output waveform is zero everywhere that is not covered by an "event" in the file.

2.32.2 Inputs:

An FSL style three column text file (start time, duration, value)

2.32.3 Outputs:

A single column text file containing the waveform

2.32.4 Usage:

Plots the data in text files.

```
usage: tcfrom3col [-h] [--debug] infilename timestep numpoints outfilename
```

Positional Arguments

infilename	the name of the input three column file
timestep	the time step of the output time course in seconds
numpoints	the number of output time points
outfilename	the name of the output time course file

Named Arguments

--debug	turn on additional debugging output Default: False

2.33 pixelcomp

2.33.1 Description:

A program to compare voxel values in two 3D NIFTI files. You give pixelcomp two files, each with their own mask. Any voxel that has a nonzero mask in both files gets added to a list of xy pairs, with the value from the first file being x, and the value from the second file being y. Pixelcomp then: 1) Makes and displays a 2D histogram of all the xy values. 2) Does a linear fit to x and y, and outputs the coefficients (slope and offset) to a XXX_linfit.txt file. 3) Writes all the xy pairs to a tab separated text file, and 4) Makes a Bland-Altman plot of x vs y

2.33.2 Inputs:

Two 3D NIFTI image files, the accompanying mask files, and the root name for the output files.

2.33.3 Outputs:

None

2.33.4 Usage:

Compare two nifti files, voxel by voxel, in a contour plot

```
usage: pixelcomp [-h] [--scatter] [--fitonly] [--nodisplay] [--fitorder ORDER]
                 [--usex] [--histbins NUM]
                 inputfilename1 maskfilename1 inputfilename2 maskfilename2
                 outputroot
```

Positional Arguments

inputfilename1	The name of the first input image nifti file.
maskfilename1	The name of the first input mask nifti file.
inputfilename2	The name of the second input image nifti file.
maskfilename2	The name of the second input mask nifti file.
outputroot	The root name of the output files.

Named Arguments

--scatter	Do a scatter plot intstead of a contour plot.
	Default: False
--fitonly	Perform fit only - do not generate graph.
	Default: False
--nodisplay	Save graphs to file only - do not display.
	Default: True
--fitorder	Order of line fit - default is 1 (linear).
	Default: 1
--usex	Use x instead of (y + x)/2 in Bland-Altman plot.
	Default: False
--histbins	Number of bins per dimension for the contour plot -Default is 51.
	Default: 51

2.34 atlastool

2.34.1 Description:

A utility to manipulate nifti atlas files

2.34.2 Inputs:

A template file which is either a 3D NIFTI with different regions specified by integer values, or a 4D NIFTI with nonzero values indicating that a voxel is in the region indicated by the 4th dimension.

2.34.3 Outputs:

A new NIFTI template file which has been processed using the options specified

2.34.4 Usage:

A utility to manipulate nifti atlas files

```
usage: atlastool [-h] [--3d] [--4d] [--split] [--maskthresh FILE]
                 [--labelfile FILE] [--xfm FILE] [--targetfile TARGETFILE]
                 [--maskfile MASK] [--removeemptyregions] [--RtoL] [--debug]
                 [--maxval MAXVAL]
                 inputtemplatename outputtemplatename
```

Positional Arguments

inputtemplatename Input NIFTI file name. Must be either a 3D file with different regions specified by integer values, or a 4D file with nonzero values indicating that a voxel is in the region.

outputtemplatename Output NIFTI file name.

Named Arguments

--3d Return a 3d file with regions encoded as integers

Default: False

--4d Return a 4d file with one region per volume

Default: False

--split Split regions along the midline into left and right subregions

Default: False

--maskthresh Threshhold for autogenerated mask (default is 0.25).

Default: 0.25

--labelfile Labels for the source atlas

--xfm Transform file to go to the reference.

--targetfile Match the resolution of TARGET

--maskfile Mask the final atlas with the 3D mask specified (if using a target file, mask must match it).

--removeemptyregions Remove regions with no voxels, so that label values are consecutive. Adjust the label file as necessary.

Default: False

--RtoL Reverse left/right assignment of labels (default is LtoR). Change if the default output is wrong.

Default: True

2.35 applydlfilter

2.35.1 Description:

Apply happy's deep learning filter to a noisy cardiac timecourse to get a high quality synthetic plethysmogram

2.35.2 Inputs:

A text file containing a noisy cardiac timecourse

2.35.3 Outputs:

A cleaned cardiac timecourse text file

2.35.4 Usage:

2.36 calctexticc

2.36.1 Description:

Calculate per-column ICC(3,1) on a set of text files.

2.36.2 Inputs:

One or more two dimensional text files, with quantities in the columns, and subjects in the rows

2.36.3 Outputs:

One dimensional text files with ICC, r, and e for each input quantity

2.36.4 Usage:

Calculate per-column ICC(3,1) on a set of text files.

```
usage: calctexticc [-h] [--demedian] [--demean] [--nocache] [--debug]
                   [--deepdebug]
                   datafile measurementlist outputroot
```

Positional Arguments

datafile A comma separated list of 1 or more 2 dimensional text files. Each column is a distinct quantity. Each line in the file is a measurement on a subject.

measurementlist A multicolumn value file of integers specifying how to group measurements. Each row is a subject, each column specifies the line numbers of the repeated measurement. Subject and measurement numbering starts at 0.

outputroot The root name for the output text files. Each distinct quantity will be in a separate row corresponding to the input file(s) columns.

Named Arguments

--demedian Subtract the median value from each map prior to ICC calculation.

Default: False

--demean Subtract the mean value from each map prior to ICC calculation.

Default: False

--nocache Disable caching for the ICC calculation. This is a terrible idea. Don't do this.

Default: False

--debug Print out additional debugging information.

Default: False

--deepdebug Print out insane additional debugging information.

Default: False

2.37 diffrois

2.37.1 Description:

Create matrices showing the difference in values between ROIs in a CSV file.

2.37.2 Inputs:

The name of the CSV file containing the ROI data. Assumes a 1 line header row.

2.37.3 Outputs:

2.37.4 Usage:

Create matrices showing the difference in values between ROIs in a CSV file.

```
usage: diffrois [-h] [--keyfile KEYFILE] [--maxlines MAXLINES] [--debug]
                datafile outputroot
```

Positional Arguments

datafile	The name of the CSV file containing the ROI data. Assumes a 1 line header row.
outputroot	The root name for the output files.

Named Arguments

--keyfile	A file containing the region labels in the desired order. The axes of the output files will be in that order, rather than the order in which they occur in the source CSV file.
--maxlines	Only process the first MAXLINES lines of the CSV file.
--debug	Output debugging information. Default: False

2.38 endtidalproc

2.38.1 Description:

2.38.2 Inputs:

2.38.3 Outputs:

2.38.4 Usage:

Process a gas trace to generate the endtidal waveform.

```
usage: endtidalproc [-h] [--isoxygen] [--samplerate FREQ | --sampletime TSTEP]
                    [--starttime START] [--endtime END] [--thresh PCT]
                    [--debug]
                    infilename outfilename
```

Positional Arguments

infilename	a text file containing a single gas trace, one timepoint per line
outfilename	a text file for the interpolated data

Named Arguments

--isoxygen	Assume the trace is oxygen, fits the bottom of the waveform, not the top.
	Default: False
--samplerate	The sample rate of the input data is FREQ Hz (default is 1Hz).
--sampletime	The sample rate of the input data is 1/TSTEP Hz (default is 1Hz).
--starttime	Start plot at START seconds.
	Default: -1000000.0
--endtime	Finish plot at END seconds.
	Default: 1000000.0
--thresh	Amount of fall (or rise) needed, in percent, to recognize a peak (or trough).
	Default: 1.0
--debug	Print additional internal information.
	Default: False

2.39 filtnifti

2.39.1 Description:

Temporally filters a NIFTI file.

2.39.2 Inputs:

A 4D NIFTI file

2.39.3 Outputs:

The filtered 4D NIFTI file

2.39.4 Usage:

Temporally filters a NIFTI file.

```
usage: filtnifti [-h] inputfilename outputfilename lowestfreq highestfreq
```

Positional Arguments

inputfilename	The name of the input nifti file.
outputfilename	The name of the output nifti file.
lowestfreq	The low passband frequency limit in Hz (set less than zero to disable HPF).
highestfreq	The high passband frequency limit in Hz (set less than zero to disable LPF)

2.40 filttc

2.40.1 Description:

2.40.2 Inputs:

2.40.3 Outputs:

2.40.4 Usage:

Filter timecourse data in text files

```
usage: filttc [-h] [--samplerate FREQ | --sampletstep TSTEP]
              [--filterband {None,vlf,lfo,resp,cardiac,hrv_ulf,hrv_vlf,hrv_lf,hrv_hf,hrv_
↪vhf,lfo_legacy}]
              [--filterfreqs LOWERPASS UPPERPASS]
              [--filterstopfreqs LOWERSTOP UPPERSTOP]
              [--normmethod {None,percent,variance,stddev,z,p2p,mad}]
              [--normfirst] [--debug]
              inputfile outputfile
```

Positional Arguments

inputfile Text file containing one or more timeseries columns. Use [:COLSPEC] to select which column(s) to use, where COLSPEC is an integer, a column separated list of ranges, or a comma separated set of column names (if input file is BIDS). Default is to use all columns

outputfile Name of the output text file.

Named Arguments

--samplerate Timecourses in file have sample frequency FREQ (default is 1.0Hz) NB: –samplerate and –sampletstep) are two ways to specify the same thing.

Default: auto

--sampletstep Timecourses in file have sample timestep TSTEP (default is 1.0s) NB: –samplerate and –sampletstep) are two ways to specify the same thing.

Default: auto

--normfirst Normalize before filtering, rather than after.

Default: False

--debug Enable additional debugging output.

Default: False

Filtering options

--filterband Possible choices: None, vlf, lfo, resp, cardiac, hrv_ulf, hrv_vlf, hrv_lf, hrv_hf, hrv_vhf, lfo_legacy

Filter timecourses to specific band. Use “None” to disable filtering. Default is “lfo”.

Default: “lfo”

--filterfreqs Filter timecourses to retain LOWERPASS to UPPERPASS. If –filterstopfreqs is not also specified, LOWERSTOP and UPPERSTOP will be calculated automatically.

--filterstopfreqs Filter timecourses to with stop frequencies LOWERSTOP and UPPERSTOP. LOWERSTOP must be <= LOWERPASS, UPPERSTOP must be >= UPPERPASS. Using this argument requires the use of –filterfreqs.

Normalization options

--normmethod Possible choices: None, percent, variance, stddev, z, p2p, mad

Demean and normalize timecourses using one of the following methods: “None” - demean only; “percent” - divide by mean; “variance” - divide by variance; “stddev” or “z” - divide by standard deviation; “p2p” - divide by range; “mad” - divide by median absolute deviation. Default is “None”.

Default: “None”

2.41 histtc

2.41.1 Description:

2.41.2 Inputs:

2.41.3 Outputs:

2.41.4 Usage:

Generate a histogram of the values in a timecourse

```
usage: histtc [-h] [--numbins BINS] [--minval MINVAL] [--maxval MAXVAL]
              [--robustrange] [--nozero] [--nozerothresh THRESH] [--normhist]
              [--debug]
              inputfilename outputroot
```

Positional Arguments

inputfilename Text file containing one or more timeseries columns. Use [:COLUMN] to select which column to use, where COLUMN is an integer or a column name (if input file is BIDS).

outputroot Name of the output text file.

Named Arguments

--numbins Number of histogram bins (default is 101)

Default: 101

--minval Minimum bin value in histogram.

--maxval Maximum bin value in histogram.

--robustrange Set histogram limits to the data's robust range (2nd to 98th percentile).

Default: False

--nozero Do not include zero values in the histogram.

Default: False

--nozerothresh Absolute values less than this are considered zero. Default is 0.01.

Default: 0.01

--normhist Return a probability density instead of raw counts.

Default: False

--debug Print additional debugging information.

Default: False

2.42 histnifti

2.42.1 Description:

A command line tool to generate a histogram for a nifti file

2.42.2 Inputs:

A nifti file

2.42.3 Outputs:

A text file containing the histogram information

None

2.42.4 Usage:

Generates a histogram of the values in a NIFTI file.

```
usage: histnifti [-h] [--histlen LEN] [--minval MINVAL] [--maxval MAXVAL]
                 [--robustrange] [--transform] [--nozero]
                 [--nozerothresh THRESH] [--normhist] [--maskfile MASK]
                 [--nodisplay]
                 inputfile outputroot
```

Positional Arguments

inputfile	The name of the input NIFTI file.
outputroot	The root of the output file names.

Named Arguments

--histlen	Set histogram length to LEN (default is to set automatically).
--minval	Minimum bin value in histogram.
--maxval	Maximum bin value in histogram.
--robustrange	Set histogram limits to the data's robust range (2nd to 98th percentile).
	Default: False
--transform	Replace data value with it's percentile score.
	Default: False
--nozero	Do not include zero values in the histogram.
	Default: False

--nozerothresh	Absolute values less than this are considered zero. Default is 0.01.
	Default: 0.01
--normhist	Return a probability density instead of raw counts.
	Default: False
--maskfile	Only process voxels within the 3D mask MASK.
--nodisplay	Do not display histogram.
	Default: True

2.43 resamplenifti

2.43.1 Description:

Takes an input NIFTI file at some TR and outputs a NIFTI file resampled to the specified TR. Downsampling is antialiased unless disabled.

2.43.2 Inputs:

The 4D NIFTI file to resample.

2.43.3 Outputs:

A 4D NIFTI file resampled to the new TR.

2.43.4 Usage:

Resamples a nifti file to a different TR.

```
usage: resamplenifti [-h] [--noantialias] [--normalize] [--debug]
                     inputfile outputfile outputtr
```

Positional Arguments

inputfile	The name of the input nifti file, including extension
outputfile	The name of the output nifti file, including extension
outputtr	The target TR, in seconds

Named Arguments

--noantialias Disable antialiasing filter

Default: True

--normalize Normalize data and save as UINT16

Default: False

--debug Print debugging information

Default: False

2.44 resampletc

2.44.1 Description:

Takes an input text file at some sample rate and outputs a text file resampled to the specified sample rate. If downsampling, antialiasing is applied unless disabled.

2.44.2 Inputs:

A text file with one or more columns. If there are more than column, only the specified column is resampled.

2.44.3 Outputs:

A text file, resampled to the specified sample rate.

2.44.4 Usage:

Resample a timeseries file

```
usage: resampletc [-h] [--nodisplay] [--noantialias]
                  inputfile insamplerate outputfile outsamplerate
```

Positional Arguments

inputfile Text file containing one or more timeseries columns. Use [:COLSPEC] to select which column(s) to use, where COLSPEC is an integer, a column separated list of ranges, or a comma separated set of column names (if input file is BIDS). Default is to use all columns

insamplerate Output sample rate in Hz.

outputfile Name of the output text file.

outsamplerate Output sample rate in Hz.

Named Arguments

--nodisplay Do not display data.

Default: True

--noantialias Enable additional debugging output.

Default: True

2.45 aligntcs

2.45.1 Description:

Given two timecourses:

- Resample the second timecourse to the samplerate of the first timecourse
- Use crosscorrelation to determine the time delay between the sequences
- Apply a time shift to align the second timecourse with the first timecourse

2.45.2 Inputs:

Two text files containing timecourses

2.45.3 Outputs:

A text file containing the second input timecourse, resampled and time shifted to match the first timecourse

2.45.4 Usage:

Resample and align two time series.

```
usage: aligntcs [-h] [--nodisplay] [--verbose] [--searchrange LAGMIN LAGMAX]
                [--filterband {None,vlf,lfo,resp,cardiac,hrv_ulf,hrv_vlf,hrv_lf,hrv_hf,
↪hrv_vhf,lfo_legacy}]
                [--filterfreqs LOWERPASS UPPERPASS]
                [--filterstopfreqs LOWERSTOP UPPERSTOP]
                infile1[:COLNUM] insamplerate1 infile2[:COLNUM] insamplerate2
                outputfile
```

Positional Arguments

infile1[:COLNUM] text file containing a timeseries. Select column COLNUM if multicolumn file.

insamplerate1 The input data file (BOLD fmri file or NIRS text file)

infile2[:COLNUM] text file containing a timeseries. Select column COLNUM if multicolumn file.

insamplerate2 The input data file (BOLD fmri file or NIRS text file)

outputfile The name of the output file

Named Arguments

--nodisplay Do not plot the data (for noninteractive use)

Default: True

--verbose Print out more debugging information

Default: False

--searchrange Limit fit to a range of lags from LAGMIN to LAGMAX. Default is -30.0 to 30.0 seconds.

Default: (-30.0, 30.0)

Filtering options

--filterband Possible choices: None, vlf, lfo, resp, cardiac, hrv_ulf, hrv_vlf, hrv_lf, hrv_hf, hrv_vhf, lfo_legacy

Filter timecourses to specific band. Use "None" to disable filtering. Default is "lfo".

Default: "lfo"

--filterfreqs Filter timecourses to retain LOWERPASS to UPPERPASS. If –filterstopfreqs is not also specified, LOWERSTOP and UPPERSTOP will be calculated automatically.

--filterstopfreqs Filter timecourses to with stop frequencies LOWERSTOP and UPPERSTOP. LOWERSTOP must be <= LOWERPASS, UPPERSTOP must be >= UPPERPASS. Using this argument requires the use of –filterfreqs.

2.46 temporaldecomp

2.46.1 Description:

2.46.2 Inputs:

2.46.3 Outputs:

2.46.4 Usage:

Perform PCA or ICA decomposition on a data file in the time dimension.

```
usage: temporaldecomp [-h] [--dmask DATAMASK] [--ncomp NCOMPS]
                      [--smooth SIGMA] [--type {pca,sparse,ica}] [--nodemean]
                      [--novarnorm]
                      datafile outputroot
```

Positional Arguments

datafile The name of the 3 or 4 dimensional nifti file to fit

outputroot The root name for the output nifti files

Named Arguments

--dmask Use DATAMASK to specify which voxels in the data to use.

--ncomp The number of PCA/ICA components to return (default is to estimate the number).

Default: -1.0

--smooth Spatially smooth the input data with a SIGMA mm kernel.

Default: 0.0

--type Possible choices: pca, sparse, ica

Type of decomposition to perform. Default is pca.

Default: "pca"

--nodemean Do not demean data prior to decomposition.

Default: True

--novarnorm Do not variance normalize data prior to decomposition.

Default: True

2.47 spatialdecomp

2.47.1 Description:

2.47.2 Inputs:

2.47.3 Outputs:

2.47.4 Usage:

Perform PCA or ICA decomposition on a data file in the spatial dimension.

```
usage: spatialdecomp [-h] [--dmask DATAMASK] [--ncomp NCOMPS] [--smooth SIGMA]
                     [--type {pca,sparse,ica}] [--nodemean] [--novarnorm]
                     datafile outputroot
```

Positional Arguments

datafile The name of the 3 or 4 dimensional nifti file to fit

outputroot The root name for the output nifti files

Named Arguments

--dmask Use DATAMASK to specify which voxels in the data to use.

--ncomp The number of PCA/ICA components to return (default is to estimate the number).

Default: -1.0

--smooth Spatially smooth the input data with a SIGMA mm kernel.

Default: 0.0

--type Possible choices: pca, sparse, ica

Type of decomposition to perform. Default is pca.

Default: "pca"

--nodemean Do not demean data prior to decomposition.

Default: True

--novarnorm Do not variance normalize data prior to decomposition.

Default: True

2.48 polyfitim

2.48.1 Description:

Fit a spatial template to a 3D or 4D NIFTI file.

2.48.2 Inputs:

A 3D or 4D NIFTI file, with mask, to be fit A 3D or 4D template file, with mask, to fit to the data file (a spatial EV)

2.48.3 Outputs:

NIFTI files of the polynomial fit of the template and residuals to the data to the specified order Text files giving the coefficients of the fit, and the R values

2.48.4 Usage:

Fit a spatial template to a 3D or 4D NIFTI file.

```
usage: polyfitim [-h] [--regionatlas ATLASFILE] [--order ORDER]
                 datafile datamask templatefile templatemask outputroot
```

Positional Arguments

datafile	The name of the 3 or 4 dimensional nifti file to fit.
datamask	The name of the 3 or 4 dimensional nifti file valid voxel mask (must match datafile).
templatefile	The name of the 3D nifti template file (must match datafile).
templatemask	The name of the 3D nifti template mask (must match datafile).
outputroot	The root name for all output files.

Named Arguments

--regionatlas	Do individual fits to every region in ATLASFILE (3D NIFTI file).
--order	Perform fit to ORDERth order (default (and minimum) is 1) Default: 1

2.49 mergequality

2.49.1 Description:

Merge rapidtide quality check data from several runs.

2.49.2 Inputs:

One or more json files containing rapidtide dataset quality metrics (outputs of `runqualitycheck` or other tools).

2.49.3 Outputs:

A csv file with one row per input file, and one column per quality metric, and one histogram file per metric showing the distribution of values over all subjects.

2.49.4 Usage:

Merge rapidtide quality check data from several runs.

```
usage: mergequality [-h] --input [INPUT ...] --outputroot OUTPUTROOT
                    [--keyfile KEYFILE] [--showhists] [--addgraymetrics]
                    [--addwhitemetrics] [--debug]
```

Named Arguments

--input

--outputroot

--keyfile

--showhists Display the histograms of the tracked quantities.

Default: False

--addgraymetrics Include gray matter only metrics.

Default: False

--addwhitemetrics Include white matter only metrics.

Default: False

--debug Output additional debugging information.

Default: False

2.50 pairproc

2.50.1 Description:

2.50.2 Inputs:

2.50.3 Outputs:

2.50.4 Usage:

Compare the even and odd volumes of a 4D nifti file.

```
usage: pairproc [-h] [--dmask DATAMASK] [--getdist] [--demean] [--debug]
                inputfile outputroot
```

Positional Arguments

inputfile The name of the input nifti file, including extension

outputroot The base name of the output files

Named Arguments

--dmask Use DATAMASK to specify which voxels in the data to use.

--getdist Get the distribution of false correlations

Default: False

--demean Remove the mean from each image prior to processing

Default: False

--debug Print debugging information

Default: False

2.51 pairwisemergenifti

2.51.1 Description:

2.51.2 Inputs:

2.51.3 Outputs:

2.51.4 Usage:

Merges adjacent timepoints in a nifti file.

```
usage: pairwisemergenifti [-h] [--maskmerge] [--debug]
                          inputfile inputmask outputfile
```

Positional Arguments

inputfile The name of the input nifti file, including extension

inputmask The name of the mask nifti file, including extension

outputfile The name of the output nifti file, including extension

Named Arguments

--maskmerge Input is a mask
Default: False

--debug Print debugging information
Default: False

2.52 physiofreq

2.52.1 Description:

2.52.2 Inputs:

2.52.3 Outputs:

2.52.4 Usage:

Finds the dominant frequency in a cardiac or respiratory waveform.

```
usage: physiofreq [-h] [--display] [--samplerate SAMPLERATE]
                  [--lowestbpm LOWESTBPM] [--highestbpm HIGHESTBPM]
                  [--disablesmoothing]
                  textfilename
```

Positional Arguments

textfilename A text input files, with optional column specification

Named Arguments

--display display the fit spectrum
Default: False

--samplerate sample rate of the waveform in Hz
Default: 1.0

--lowestbpm Lowest allowable frequency in cycles per minute
Default: 6.0

--highestbpm Highest allowable frequency in cycles per minute
Default: 20.0

--disablesmoothing Do not apply Savitsky-Golay filter to spectrum
Default: False

2.53 plethquality

2.53.1 Description:

2.53.2 Inputs:

2.53.3 Outputs:

2.53.4 Usage:

Calculate quality metrics from a cardiac text file.

```
usage: plethquality [-h] [--samplerate SAMPLERATE] [--nodisplay]
                    infilename outfilename
```

Positional Arguments

infilename Text file containing one or more timeseries columns. Use [:COLUMN] to select which column to use, where COLUMN is an integer or a column name (if input file is BIDS).

outfilename The name of the output text file.

Named Arguments

--samplerate Sample rate of timecourse, in Hz (if not specified in input file).

--nodisplay Do not plot the data (for noninteractive use).

Default: True

2.54 rankimage

2.54.1 Description:

Convert a 3D or 4D NIFTI file into a map of each voxel's rank in the map values.

2.54.2 Inputs:

A 3D or 4D NIFTI file.

2.54.3 Outputs:

A 3D or 4D NIFTI file of voxel ranks scaled from 0-100. If 4D, each timepoint is considered separately.

2.54.4 Usage:

Convert a 3D or 4D nifti image into a percentile map.

```
usage: rankimage [-h] [--debug] inputfilename maskfilename outputroot
```

Positional Arguments

inputfilename The name of the input image nifti file.

maskfilename The name of the input mask nifti file.

outputroot The root name of the output files.

Named Arguments

--debug Output additional debugging information.

Default: False

2.55 runqualitycheck

2.55.1 Description:

2.55.2 Inputs:

2.55.3 Outputs:

2.55.4 Usage:

Run a quality check on a rapidtide dataset.

```
usage: runqualitycheck [-h] [--graymaskspec MASK[:VALSPEC]]
                       [--whitemaskspec MASK[:VALSPEC]] [--debug]
                       inputfileroot
```

Positional Arguments

inputfileroot The root of the rapidtide dataset name (without the underscore.)

Named Arguments

--graymaskspec The name of a gray matter mask that matches the input dataset. If VALSPEC is given, only voxels with integral values listed in VALSPEC are used. If using an aparc+aseg file, set to APARC_GRAY.

--whitemaskspec The name of a white matter mask that matches the input dataset. If VALSPEC is given, only voxels with integral values listed in VALSPEC are used. If using an aparc+aseg file, set to APARC_WHITE.

--debug Output additional debugging information.

Default: False

2.56 variabilityizer

2.56.1 Description:

2.56.2 Inputs:

2.56.3 Outputs:

2.56.4 Usage:

Transform a nifti fmri file into a temporal variability file.

```
usage: variabilityizer [-h] inputfilename outputfilename windowlength
```

Positional Arguments

inputfilename The name of the input nifti file.

outputfilename The name of the output nifti file.

windowlength The size of the temporal window in seconds.

2.57 fdica

2.57.1 Description:

2.57.2 Inputs:

2.57.3 Outputs:

2.57.4 Usage:

Fit a spatial template to a 3D or 4D NIFTI file.

```
usage: fdica [-h] [--spatialfilt GAUSSSIGMA] [--pcacomponents NCOMP]
             [--icacomponents NCOMP] [--debug]
             datafile datamask outputroot
```

Positional Arguments

datafile	The name of the 3 or 4 dimensional nifti file to fit.
datamask	The name of the 3 dimensional nifti file voxel mask (must match datafile).
outputroot	The root name for all output files.

Named Arguments

--spatialfilt	Spatially filter fMRI data prior to analysis using GAUSSSIGMA in mm. Set GAUSSSIGMA negative to have rapidtide set it to half the mean voxel dimension (a rule of thumb for a good value). Default: 0.0
--pcacomponents	Use NCOMP components for PCA fit of phase Default: 0.9
--icacomponents	Use NCOMP components for ICA decomposition
--debug	Output additional debugging information. Default: False

2.58 gmscalc

2.58.1 Description:

2.58.2 Inputs:

2.58.3 Outputs:

2.58.4 Usage:

Calculate the global mean signal, and filtered versions

```
usage: gmscalc [-h] [--dmask DATAMASK]
               [--normmethod {None,percent,variance,stddev,z,p2p,mad}]
               [--normfirst] [--smooth SIGMA] [--debug]
               datafile outputroot
```

Positional Arguments

datafile The name of a 4 dimensional nifti files (3 spatial dimensions and a subject dimension).

outputroot The root name for the output nifti files.

Named Arguments

--dmask Use DATAMASK to specify which voxels in the data to use.

--normfirst Normalize before filtering, rather than after.

Default: False

--smooth Spatially smooth the input data with a SIGMA mm kernel prior to calculation.

Default: 0.0

--debug Print out additional debugging information.

Default: False

Normalization options

--normmethod Possible choices: None, percent, variance, stddev, z, p2p, mad

Demean and normalize timecourses using one of the following methods: “None” - demean only; “percent” - divide by mean; “variance” - divide by variance; “stddev” or “z” - divide by standard deviation; “p2p” - divide by range; “mad” - divide by median absolute deviation. Default is “None”.

Default: “None”

2.59 roisummarize

2.59.1 Description:

2.59.2 Inputs:

2.59.3 Outputs:

2.59.4 Usage:

Extract summary timecourses from the regions in an atlas

```
usage: roisummarize [-h] [--samplerate FREQ | --sampletstep TSTEP]
                    [--numskip NPTS]
                    [--filterband {None,vlf,lfo,resp,cardiac,hrv_ulf,hrv_vlf,hrv_lf,hrv_
↪hf,hrv_vhf,lfo_legacy}]
                    [--filterfreqs LOWERPASS UPPERPASS]
                    [--filterstopfreqs LOWERSTOP UPPERSTOP]
                    [--normmethod {None,percent,variance,stddev,z,p2p,mad}]
                    [--debug]
                    inputfilename templatefile outputfile
```

Positional Arguments

inputfilename Text file containing one or more timeseries columns. Use [:COLSPEC] to select which column(s) to use, where COLSPEC is an integer, a column separated list of ranges, or a comma separated set of column names (if input file is BIDS). Default is to use all columns

templatefile Text file containing one or more timeseries columns. Use [:COLSPEC] to select which column(s) to use, where COLSPEC is an integer, a column separated list of ranges, or a comma separated set of column names (if input file is BIDS). Default is to use all columns

outputfile Name of the output text file.

Named Arguments

--samplerate Timecourses in file have sample frequency FREQ (default is 1.0Hz) NB: –samplerate and –sampletstep) are two ways to specify the same thing.

Default: auto

--sampletstep Timecourses in file have sample timestep TSTEP (default is 1.0s) NB: –samplerate and –sampletstep) are two ways to specify the same thing.

Default: auto

--numskip Skip NPTS initial points to get past T1 relaxation.

Default: 0

--debug Enable additional debugging output.

Default: False

Filtering options

--filterband Possible choices: None, vlf, lfo, resp, cardiac, hrv_ulf, hrv_vlf, hrv_lf, hrv_hf, hrv_vhf, lfo_legacy

Filter timecourses to specific band. Use "None" to disable filtering. Default is "None".

Default: "None"

--filterfreqs Filter timecourses to retain LOWERPASS to UPPERPASS. If –filterstopfreqs is not also specified, LOWERSTOP and UPPERSTOP will be calculated automatically.

--filterstopfreqs Filter timecourses to with stop frequencies LOWERSTOP and UPPERSTOP. LOWERSTOP must be <= LOWERPASS, UPPERSTOP must be >= UPPERPASS. Using this argument requires the use of –filterfreqs.

Normalization options

--normmethod Possible choices: None, percent, variance, stddev, z, p2p, mad

Demean and normalize timecourses using one of the following methods: "None" - demean only; "percent" - divide by mean; "variance" - divide by variance; "stddev" or "z" - divide by standard deviation; "p2p" - divide by range; "mad" - divide by median absolute deviation. Default is "None".

Default: "None"

2.60 simdata

2.60.1 Description:

Generates a simulated rapidtide dataset.

2.60.2 Inputs:

2.60.3 Outputs:

2.60.4 Usage:

Generate simulated fMRI data with known correlation parameters

```
usage: simdata [-h] [--lfopctfile FILE] [--lfolagfile FILE]
               [--lforegressor FILE] [--lfosamprate SAMPRATE]
               [--lfostarttime STARTTIME] [--resppctfile FILE]
               [--resplagfile FILE] [--respregressor FILE]
```

(continues on next page)

(continued from previous page)

```
                [--respsamprate SAMPRATE] [--respstarttime STARTTIME]
                [--cardiacpctfile FILE] [--cardiaclagfile FILE]
                [--cardiacregressor FILE] [--cardiacsamprate SAMPRATE]
                [--cardiacstarttime STARTTIME] [--numskip SKIP]
                [--globalnoiselevel LEVEL] [--voxelnoiselevel LEVEL] [--debug]
                fmrifilename immeanfilename outputroot slicetimefile
```

Positional Arguments

fmrifilename	Input NIFTI file name. An exemplar BOLD fMRI file with the target dimensions
immeanfilename	Input NIFTI file name. 3D file with the mean value for each voxel
outputroot	Root name for the output files.
slicetimefile	Slice acquisition time file, either FSL format or BIDS sidecar.

Named Arguments

--lfopctfile	3D NIFTI file with the lfo amplitude in percent of mean at every point
--lfolagfile	3D NIFTI file with the lfo delay value in seconds at every point
--lforegressor	The LFO regressor text file
--lfosamprate	The sample rate of the LFO regressor file, in Hz
--lfostarttime	The time delay, in seconds, into the lfo regressor file that matches the start time of the fmrifile. Default is 0.0
--resppctfile	3D NIFTI file with the resp amplitude in percent of mean at every point
--resplagfile	3D NIFTI file with the resp delay value in seconds at every point
--respregressor	The LFO regressor text file
--respsamprate	The sample rate of the LFO regressor file, in Hz
--respstarttime	The time delay, in seconds, into the resp regressor file that matches the start time of the fmrifile. Default is 0.0
--cardiacpctfile	3D NIFTI file with the cardiac amplitude in percent of mean at every point
--cardiaclagfile	3D NIFTI file with the cardiac delay value in seconds at every point
--cardiacregressor	The LFO regressor text file
--cardiacsamprate	The sample rate of the LFO regressor file, in Hz
--cardiacstarttime	The time delay, in seconds, into the cardiac regressor file that matches the start time of the fmrifile. Default is 0.0
--numskip	Use to simulate tr periods deleted during preprocessing Default: 0
--globalnoiselevel	The variance of the noise common to every voxel. Default is 0.0 Default: 0.0

--voxelnoiselevel The variance of the voxel specific noise. Default is 0.0

Default: 0.0

--debug Enable additional debugging output.

Default: False

2.61 spatialfit

2.61.1 Description:

2.61.2 Inputs:

2.61.3 Outputs:

2.61.4 Usage:

Fit a 3D or 4D NIFTI file to a spatial template.

```
usage: spatialfit [-h] [--datamask DATAMASK] [--templatemask TEMPLATEMASK]
                  [--order ORDER] [--debug]
                  datafile templatefile outputroot
```

Positional Arguments

datafile The name of the 3D or 4D input NIFTI file.

templatefile The name of the 3D template NIFTI file.

outputroot The root of the output file names.

Named Arguments

--datamask DATAMASK specifies which voxels in the data to use.

--templatemask TEMPLATEMASK specifies which voxels in the template to use.

--order The order of the fit to the template.

Default: 1

--debug Enable additional debugging output.

Default: False

2.62 spatialmi

2.62.1 Description:

2.62.2 Inputs:

2.62.3 Outputs:

2.62.4 Usage:

Calculate the localized spatial mutual information between two images

```
usage: spatialmi [-h] [--noprebin] [--nonorm] [--radius RADIUS]
                 [--sigma SIGMA] [--kernelwidth WIDTH] [--spherical]
                 [--index1 INDEX1] [--index2 INDEX2] [--debug]
                 inputfilename1 maskfilename1 inputfilename2 maskfilename2
                 outputroot
```

Positional Arguments

inputfilename1	The name of the first input image nifti file.
maskfilename1	The name of the first input mask nifti file.
inputfilename2	The name of the second input image nifti file.
maskfilename2	The name of the second input mask nifti file.
outputroot	The root name of the output files.

Named Arguments

--noprebin	Dynamically calculate histogram bins for each voxel (slower). Default: True
--nonorm	Do not normalize neighborhood by the variance. Default: True
--radius	Radius of the comparison, in voxels. If not spherical, comparison neighborhood is cubic with (2 * RADIUS + 1)^3 voxels. Must be 1.0 or greater. Default is 2.0 Default: 2.0
--sigma	Width, in voxels, of a gaussian smoothing filter to apply to each input dataset. Default is no filteriing.
--kernelwidth	Kernel width, in voxels, of gaussian neighborhood limit. Default is no kernel.
--spherical	Use a spherical (rather than cubical) neighborhood (much slower). Default: False

--index1 If input file 1 is 4 dimensional, select timepoint INDEX1 for spatial mutual information calculation.If not specified, the first image will be used.

Default: 0

--index2 If input file 2 is 4 dimensional, select timepoint INDEX2 for spatial mutual information calculation.If not specified, the first image will be used.

Default: 0

--debug Print additional internal information.

Default: False

2.63 localflow

2.63.1 Description:

2.63.2 Inputs:

2.63.3 Outputs:

2.63.4 Usage:

Calculate local sources of signal.

```
usage: localflow [-h] [--npasses NPASSES] [--radius RADIUS]
                 [--minlagdiff MINLAGDIFF] [--ampthresh AMPTHRESH]
                 [--gausssigma GAUSSSIGMA] [--oversampfac OVERSAMPFAC]
                 [--dofit] [--detrendorder ORDER] [--nosphere]
                 [--filterband {None,vlf,lfo,resp,cardiac,hrv_ulf,hrv_vlf,hrv_lf,hrv_hf,
↪hrv_vhf,lfo_legacy}]
                 [--filterfreqs LOWERPASS UPPERPASS]
                 [--filterstopfreqs LOWERSTOP UPPERSTOP]
                 [--filtertype {trapezoidal,brickwall,butterworth}]
                 [--butterorder ORDER] [--padseconds SECONDS]
                 [--windowfunc {hamming,hann,blackmanharris,None}]
                 [--zeropadding PADVAL] [--noprogressbar] [--debug]
                 inputfilename outputroot
```

Positional Arguments

inputfilename The name of the input nifti file.

outputroot The root name of the output nifti files.

Named Arguments

--npasses The number of passes for reconstruction. Default is 20

Default: 20

--radius The radius around the voxel to check correlations. Default is 10.5

Default: 10.5

--minlagdiff The minimum lagtime difference threshold to select which diffs to include in reconstruction. Default is 0.0

Default: 0.0

--ampthresh The correlation threshold to select which diffs to include in reconstruction. Default is 0.3

Default: 0.3

--gausssigma Spatially filter fMRI data prior to analysis using GAUSSSIGMA in mm. Set GAUSSSIGMA negative to set it to half the mean voxel dimension (a rule of thumb for a good value).

Default: 0.0

--oversampfac Oversample the fMRI data by the following integral factor. Set to -1 for automatic selection (default).

Default: -1

--dofit Turn on correlation fitting.

Default: False

--detrendorder Set order of trend removal (0 to disable). Default is 3.

Default: 3

--nosphere Use rectangular rather than spherical reconstruction kernel.

Default: True

Filtering options

--filterband Possible choices: None, vlf, lfo, resp, cardiac, hrv_ulf, hrv_vlf, hrv_lf, hrv_hf, hrv_vhf, lfo_legacy

Filter data and regressors to specific band. Use "None" to disable filtering. Default is "lfo".

Default: "lfo"

--filterfreqs Filter data and regressors to retain LOWERPASS to UPPERPASS. If –filterstopfreqs is not also specified, LOWERSTOP and UPPERSTOP will be calculated automatically.

--filterstopfreqs Filter data and regressors to with stop frequencies LOWERSTOP and UPPERSTOP. LOWERSTOP must be <= LOWERPASS, UPPERSTOP must be >= UPPERPASS. Using this argument requires the use of –filterfreqs.

--filtertype Possible choices: trapezoidal, brickwall, butterworth

Filter data and regressors using a trapezoidal FFT, brickwall FFT, or butterworth bandpass filter. Default is "trapezoidal".

Default: "trapezoidal"

--butterorder Set order of butterworth filter (if used). Default is 6.

Default: 6

--padseconds The number of seconds of padding to add to each end of a filtered timecourse to reduce end effects. Default is 30.0.

Default: 30.0

Windowing options

--windowfunc Possible choices: hamming, hann, blackmanharris, None

Window function to use prior to correlation. Options are hamming, hann, blackmanharris, and None. Default is hamming

Default: "hamming"

--zeropadding Pad input functions to correlation with PADVAL zeros on each side. A PADVAL of 0 does circular correlations, positive values reduce edge artifacts. Set PADVAL < 0 to set automatically. Default is 0.

Default: 0

Miscellaneous options

--noprogressbar Will disable showing progress bars (helpful if stdout is going to a file).

Default: True

--debug Turn on debugging information.

Default: False

2.64 synthASL

2.64.1 Description:

2.64.2 Inputs:

2.64.3 Outputs:

2.64.4 Usage:

Use rapidtide output to predict ASL image.

```
usage: synthASL [-h] [--tagoffset SECS] [--pld SECS] [--labelduration SECS]
                [--bloodT1 SECS]
                dataset outputfilename
```

Positional Arguments

dataset The name of the rapidtide dataset.

outputfilename The name of the output nifti file.

Named Arguments

--tagoffset The assumed time of tagging, relative to the peak of the lag histogram (default is 2.945).

Default: 2.945

--pld The postlabel delay (default is 1.8).

Default: 1.8

--labelduration The length of the labelling period (default is 1.8).

Default: 1.8

--bloodT1 The T1 of blood at this field strength (default is 1.841, for 3T).

Default: 1.841

2.65 Legacy interface:

For compatibility with old workflows, rapidtide can be called using legacy syntax by using "rapidtide2x_legacy". Although the underlying code is the same, not all options are settable from the legacy interface. This interface is deprecated and will be removed in a future version of rapidtide, so please convert existing workflows.

```
usage:  rapidtide2x_legacy  datafilename outputname
[-r LAGMIN,LAGMAX] [-s SIGMALIMIT] [-a] [--nowindow] [--phat] [--liang] [--
↪eckart] [-f GAUSSSIGMA] [-O oversampfac] [-t TSTEP] [--datatstep=TSTEP] [--
↪datafreq=FREQ] [-d] [-b] [-V] [-L] [-R] [-C] [-F LOWERFREQ,UPPERFREQ[,
↪LOWERSTOP,UPPERSTOP]] [-o OFFSETTIME] [--autosync] [-T] [-p] [-P] [-B] [-h␣
↪HISTLEN] [-i INTERPTYPE] [-I] [-Z DELAYTIME] [--nofitfilt] [--
↪searchfrac=SEARCHFRAC] [-N NREPS] [--motionfile=MOTFILE] [--pickleft] [--
↪numskip=SKIP] [--refineweighting=TYPE] [--refineprenorm=TYPE] [--
↪passes=PASSES] [--refinepasses=PASSES] [--excluderefine=MASK] [--
↪includerefine=MASK] [--includemean=MASK] [--excludemean=MASK][--
↪lagminthresh=MIN] [--lagmaxthresh=MAX] [--ampthresh=AMP] [--
↪sigmathresh=SIGMA] [--corrmask=MASK] [--corrmaskthresh=PCT] [--refineoffset]␣
↪[--pca] [--ica] [--weightedavg] [--avg] [--psdfilter] [--noprogressbar] [--
↪despecklethresh=VAL] [--despecklepasses=PASSES] [--dispersioncalc] [--
↪refineupperlag] [--refinelowerlag] [--nosharedmem] [--tmask=MASKFILE] [--
↪limitoutput] [--motionfile=FILENAME[:COLSPEC] [--softlimit] [--
↪timerange=START,END] [--skipsighistfit] [--accheck] [--acfix][--
↪numskip=SKIP] [--slicetimes=FILE] [--glmsourcefile=FILE] [--
↪regressorfreq=FREQ] [--regressortstep=TSTEP][--regressor=FILENAME] [--
↪regressorstart=STARTTIME] [--usesp] [--peakfittype=FITTYPE] [--
↪mklthreads=NTHREADS] [--nprocs=NPROCS] [--nirs] [--venousrefine]

Required arguments:
    datafilename               - The input data file (BOLD fmri file or NIRS)
```

(continues on next page)

(continued from previous page)

```
    outputname                  - The root name for the output files

Optional arguments:
    Arguments are processed in order of appearance.  Later options can␣
↪override ones earlier on
    the command line

Macros:
    --venousrefine              - This is a macro that sets --
↪lagminthresh=2.5, --lagmaxthresh=6.0,
                                  --ampthresh=0.5, and --refineupperlag to␣
↪bias refinement towards
                                  voxels in the draining vasculature for an␣
↪fMRI scan.
    --nirs                      - This is a NIRS analysis - this is a macro␣
↪that sets --nothresh,
                                  --preservefiltering, --refinenorm=var, --
↪ampthresh=0.7,
                                  and --lagminthresh=0.1.

Preprocessing options:
    -t TSTEP,                   - Set the timestep of the data file to␣
↪TSTEP (or 1/FREQ)
      --datatstep=TSTEP,          This will override the TR in an fMRI file.
      --datafreq=FREQ             NOTE: if using data from a text file, for␣
↪example with
                                  NIRS data, using one of these options is␣
↪mandatory.
    -a                          - Disable antialiasing filter
    --detrendorder=ORDER        - Set order of trend removal (0 to disable,␣
↪default is 1 - linear)
    -I                          - Invert the sign of the regressor before␣
↪processing
    -i                          - Use specified interpolation type (options␣
↪are 'cubic',
                                  'quadratic', and 'univariate (default)')
    -o                          - Apply an offset OFFSETTIME to the lag␣
↪regressors
    --autosync                  - Calculate and apply offset time of an␣
↪external regressor from
                                  the global crosscorrelation.  Overrides␣
↪offsettime if specified.
    -b                          - Use butterworth filter for band splitting␣
↪instead of
                                  trapezoidal FFT filter
    -F  LOWERFREQ,UPPERFREQ[,LOWERSTOP,UPPERSTOP]
                                - 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
```

(continues on next page)

(continued from previous page)

```
    -L                                - Filter data and regressors to LFO band
    -R                                - Filter data and regressors to respiratory␣
↪band
    -C                                - Filter data and regressors to cardiac band
    --padseconds=SECONDS              - Set the filter pad time to SECONDS␣
↪seconds.  Default
                                        is 30.0
    -N NREPS                          - Estimate significance threshold by␣
↪running NREPS null
                                        correlations (default is 10000, set to 0␣
↪to disable).  If you are
                                        running multiple passes, 'ampthresh' will␣
↪be set to the 0.05 significance.
                                        level unless it is manually specified␣
↪(see below).
    --permutationmethod=METHOD        - Method for permuting the regressor for␣
↪significance estimation.  Default
                                        is shuffle
    --skipsighistfit                  - Do not fit significance histogram with a␣
↪Johnson SB function
    --windowfunc=FUNC                 - Use FUNC window funcion prior to␣
↪correlation.  Options are
                                        hamming (default), hann, blackmanharris,␣
↪and None
    --nowindow                        - Disable precorrelation windowing
    -f GAUSSSIGMA                     - Spatially filter fMRI data prior to␣
↪analysis using
                                        GAUSSSIGMA in mm
    -M                                - Generate a global mean regressor and use␣
↪that as the
                                        reference regressor
    --globalmeaninclude=MASK[:VALSPEC]
                                      - Only use voxels in NAME for global␣
↪regressor generation (if VALSPEC is
                                        given, only voxels with integral values␣
↪listed in VALSPEC are used.)
    --globalmeanexclude=MASK[:VALSPEC]
                                      - Do not use voxels in NAME for global␣
↪regressor generation (if VALSPEC is
                                        given, only voxels with integral values␣
↪listed in VALSPEC are used.)
    -m                                - Mean scale regressors during global mean␣
↪estimation
    --slicetimes=FILE                 - Apply offset times from FILE to each␣
↪slice in the dataset
    --numskip=SKIP                    - SKIP tr's were previously deleted during␣
↪preprocessing (e.g. if you
                                        have done your preprocessing in FSL and␣
↪set dummypoints to a
                                        nonzero value.) Default is 0.
    --timerange=START,END             - Limit analysis to data between timepoints␣
↪START
```

(continues on next page)

(continued from previous page)

```
                                    and END in the fmri file. If END is set␣
↪to -1,
                                    analysis will go to the last timepoint. ␣
↪Negative values
                                    of START will be set to 0. Default is to␣
↪use all timepoints.
    --nothresh                    - Disable voxel intensity threshold␣
↪(especially useful
                                    for NIRS data)
    --motionfile=MOTFILE[:COLSPEC] - Read 6 columns of motion regressors out␣
↪of MOTFILE text file.
                                    (with timepoints rows) and regress their␣
↪derivatives
                                    and delayed derivatives out of the data␣
↪prior to analysis.
                                    If COLSPEC is present, use the comma␣
↪separated list of ranges to
                                    specify X, Y, Z, RotX, RotY, and RotZ, in␣
↪that order.  For
                                    example, :3-5,7,0,9 would use columns 3,␣
↪4, 5, 7, 0 and 9
                                    for X, Y, Z, RotX, RotY, RotZ,␣
↪respectively
    --motpos                      - Toggle whether displacement regressors␣
↪will be used in motion regression.
                                    Default is False.
    --motderiv                    - Toggle whether derivatives will be used␣
↪in motion regression.
                                    Default is True.
    --motdelayderiv               - Toggle whether delayed derivative ␣
↪regressors will be used in motion regression.
                                    Default is False.

Correlation options:
    -O OVERSAMPFAC                - Oversample the fMRI data by the following␣
↪integral
                                    factor.  Setting to -1 chooses the factor␣
↪automatically (default)
    --regressor=FILENAME          - Read probe regressor from file FILENAME␣
↪(if none
                                    specified, generate and use global␣
↪regressor)
    --regressorfreq=FREQ          - Probe regressor in file has sample␣
↪frequency FREQ
                                    (default is 1/tr) NB: --regressorfreq and␣
↪--regressortstep
                                    are two ways to specify the same thing
    --regressortstep=TSTEP        - Probe regressor in file has sample time␣
↪step TSTEP
                                    (default is tr) NB: --regressorfreq and --
↪regressortstep
                                    are two ways to specify the same thing
```

(continues on next page)

(continued from previous page)

```
    --regressorstart=START          - The time delay in seconds into the␣
↪regressor file, corresponding
                                      in the first TR of the fmri file (default␣
↪is 0.0)
    --phat                          - Use generalized cross-correlation with␣
↪phase alignment
                                      transform (PHAT) instead of correlation
    --liang                         - Use generalized cross-correlation with␣
↪Liang weighting function
                                      (Liang, et al, doi:10.1109/IMCCC.2015.283)
    --eckart                        - Use generalized cross-correlation with␣
↪Eckart weighting function
    --corrmaskthresh=PCT            - Do correlations in voxels where the mean␣
↪exceeeds this
                                      percentage of the robust max (default is␣
↪1.0)
    --corrmask=MASK                 - Only do correlations in voxels in MASK␣
↪(if set, corrmaskthresh
                                      is ignored).
    --accheck                       - Check for periodic components that␣
↪corrupt the autocorrelation

Correlation fitting options:
    -Z DELAYTIME                    - Don't fit the delay time - set it to␣
↪DELAYTIME seconds
                                      for all voxels
    -r LAGMIN,LAGMAX                - Limit fit to a range of lags from LAGMIN␣
↪to LAGMAX
    -s SIGMALIMIT                   - Reject lag fits with linewidth wider than␣
↪SIGMALIMIT
    -B                              - Bipolar mode - match peak correlation␣
↪ignoring sign
    --nofitfilt                     - Do not zero out peak fit values if fit␣
↪fails
    --searchfrac=FRAC               - When peak fitting, include points with␣
↪amplitude > FRAC * the
                                      maximum amplitude.
                                      (default value is 0.5)
    --peakfittype=FITTYPE           - Method for fitting the peak of the␣
↪similarity function
                                      (default is 'gauss'). 'quad' uses a␣
↪quadratic fit. Other options are
                                      'fastgauss' which is faster but not as␣
↪well tested, and 'None'.
    --despecklepasses=PASSES        - detect and refit suspect correlations to␣
↪disambiguate peak
                                      locations in PASSES passes
    --despecklethresh=VAL           - refit correlation if median discontinuity␣
↪magnitude exceeds
                                      VAL (default is 5s)
    --softlimit                     - Allow peaks outside of range if the␣
↪maximum correlation is
```

(continues on next page)

(continued from previous page)

```
                                    at an edge of the range.

Regressor refinement options:
    --refineprenorm=TYPE            - Apply TYPE prenormalization to each␣
↪timecourse prior
                                    to refinement (valid weightings are 'None
↪',
                                    'mean' (default), 'var', and 'std'
    --refineweighting=TYPE          - Apply TYPE weighting to each timecourse␣
↪prior
                                    to refinement (valid weightings are 'None
↪',
                                    'R', 'R2' (default)
    --passes=PASSES,                - Set the number of processing passes to␣
↪PASSES
     --refinepasses=PASSES          (default is 1 pass - no refinement).
                                    NB: refinepasses is the wrong name for␣
↪this option -
                                    --refinepasses is deprecated, use --
↪passes from now on.
    --refineinclude=MASK[:VALSPEC] - Only use nonzero voxels in MASK for␣
↪regressor refinement (if VALSPEC is
                                    given, only voxels with integral values␣
↪listed in VALSPEC are used.)
    --refineexclude=MASK[:VALSPEC] - Do not use nonzero voxels in MASK for␣
↪regressor refinement (if VALSPEC is
                                    given, only voxels with integral values␣
↪listed in VALSPEC are used.)
    --lagminthresh=MIN              - For refinement, exclude voxels with␣
↪delays less
                                    than MIN (default is 0.5s)
    --lagmaxthresh=MAX              - For refinement, exclude voxels with␣
↪delays greater
                                    than MAX (default is 5s)
    --ampthresh=AMP                 - For refinement, exclude voxels with␣
↪correlation
                                    coefficients less than AMP (default is 0.
↪3).  NOTE: ampthresh will
                                    automatically be set to the p<0.05␣
↪significance level determined by
                                    the -N option if -N is set greater than 0␣
↪and this is not
                                    manually specified.
    --sigmathresh=SIGMA             - For refinement, exclude voxels with␣
↪widths greater
                                    than SIGMA (default is 100s)
    --refineoffset                  - Adjust offset time during refinement to␣
↪bring peak
                                    delay to zero
    --pickleft                      - When setting refineoffset, always select␣
↪the leftmost histogram peak
    --pickleftthresh=THRESH         - Set the threshold value (fraction of␣
```

(continues on next page)

(continued from previous page)

```
↪maximum) to decide something is a
                                peak in a histogram.  Default is 0.33.
    --refineupperlag            - Only use positive lags for regressor␣
↪refinement
    --refinelowerlag            - Only use negative lags for regressor␣
↪refinement
    --pca                       - Use pca to derive refined regressor␣
↪(default is
                                unweighted averaging)
    --ica                       - Use ica to derive refined regressor␣
↪(default is
                                unweighted averaging)
    --weightedavg               - Use weighted average to derive refined␣
↪regressor
                                (default is unweighted averaging)
    --avg                       - Use unweighted average to derive refined␣
↪regressor
                                (default)
    --psdfilter                 - Apply a PSD weighted Wiener filter to␣
↪shifted
                                timecourses prior to refinement

Output options:
    --limitoutput               - Don't save some of the large and rarely␣
↪used files
    -T                          - Save a table of lagtimes used
    -h HISTLEN                  - Change the histogram length to HISTLEN␣
↪(default is
                                100)
    --glmsourcefile=FILE        - Regress delayed regressors out of FILE␣
↪instead of the
                                initial fmri file used to estimate delays
    --noglm                     - Turn off GLM filtering to remove delayed␣
↪regressor
                                from each voxel (disables output of␣
↪fitNorm)
    --preservefiltering         - don't reread data prior to GLM

Miscellaneous options:
    --noprogressbar             - Disable progress bars - useful if saving␣
↪output to files
    --wiener                    - Perform Wiener deconvolution to get voxel␣
↪transfer functions
    --usesp                     - Use single precision for internal␣
↪calculations (may
                                be useful when RAM is limited)
    -c                          - Data file is a converted CIFTI
    -S                          - Simulate a run - just report command line␣
↪options
    -d                          - Display plots of interesting timecourses
    --nonumba                   - Disable jit compilation with numba
    --nosharedmem               - Disable use of shared memory for large␣
```

(continues on next page)

(continued from previous page)

```
↪array storage
    --memprofile                    - Enable memory profiling for debugging -␣
↪warning:
                                      this slows things down a lot.
    --multiproc                     - Enable multiprocessing versions of key␣
↪subroutines.  This
                                      speeds things up dramatically.  Almost␣
↪certainly will NOT
                                      work on Windows (due to different forking␣
↪behavior).
    --mklthreads=NTHREADS           - Use no more than NTHREADS worker threads␣
↪in accelerated numpy calls.
    --nprocs=NPROCS                 - Use NPROCS worker processes for␣
↪multiprocessing.  Setting NPROCS
                                      less than 1 sets the number of worker␣
↪processes to
                                      n_cpus - 1 (default).  Setting NPROCS␣
↪enables --multiproc.
    --debug                         - Enable additional information output
    --saveoptionsasjson             - Save the options file in json format␣
↪rather than text.  Will eventually
                                      become the default, but for now I'm just␣
↪trying it out.

Experimental options (not fully tested, may not work):
    --cleanrefined                  - perform additional processing on refined␣
↪regressor to remove spurious
                                      components.
    --dispersioncalc                - Generate extra data during refinement to␣
↪allow calculation of
                                      dispersion.
    --acfix                         - Perform a secondary correlation to␣
↪disambiguate peak location
                                      (enables --accheck).  Experimental.
    --tmask=MASKFILE                - Only correlate during epochs specified in
                                      MASKFILE (NB: if file has one colum, the␣
↪length needs to match
                                      the number of TRs used.  TRs with nonzero␣
↪values will be used
                                      in analysis.  If there are 2 or more␣
↪columns, each line of MASKFILE
                                      contains the time (first column) and␣
↪duration (second column) of an
                                      epoch to include.)
```

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.

When using the legacy interface, file names will be output using the old, non-BIDS names and formats. rapidtide can be forced to use the old style outputs with the `--legacyoutput` flag.

2.66 Equivalence between BIDS and legacy outputs:

BIDS style name	Legacy name
XXX_maxtime_map(.nii.gz, .json)	XXX_lagtimes.nii.gz
XXX_desc-maxtime_hist(.tsv, .json)	XXX_laghist.txt
XXX_maxcorr_map(.nii.gz, .json)	XXX_lagstrengths.nii.gz
XXX_desc-maxcorr_hist(.tsv, .json)	XXX_strengthhist.txt
XXX_maxcorrsq_map(.nii.gz, .json)	XXX_R2.nii.gz
XXX_desc-maxcorrsq_hist(.tsv, .json)	XXX_R2hist.txt
XXX_maxwidth_map(.nii.gz, .json)	XXX_lagsigma.nii.gz
XXX_desc-maxwidth_hist(.tsv, .json)	XXX_widthhist.txt
XXX_MTT_map(.nii.gz, .json)	XXX_MTT.nii.gz
XXX_corrfit_mask.nii.gz	XXX_fitmask.nii.gz
XXX_corrfitfailreason_map(.nii.gz, .json)	XXX_failreason.nii.gz
XXX_desc-corrfitwindow_info.nii.gz	XXX_windowout.nii.gz
XXX_desc-runoptions_info.json	XXX_options.json
XXX_desc-lfofilterCleaned_bold(.nii.gz, .json)	XXX_filtereddata.nii.gz
XXX_desc-lfofilterRemoved_bold(.nii.gz, .json)	XXX_datatoremove.nii.gz
XXX_desc-lfofilterCoeff_map.nii.gz	XXX_fitcoeff.nii.gz
XXX_desc-lfofilterMean_map.nii.gz	XXX_meanvalue.nii.gz
XXX_desc-lfofilterNorm_map.nii.gz	XXX_fitNorm.nii.gz
XXX_desc-lfofilterR2_map.nii.gz	XXX_r2value.nii.gz
XXX_desc-lfofilterR_map.nii.gz	XXX_rvalue.nii.gz
XXX_desc-processed_mask.nii.gz	XXX_corrmask.nii.gz
XXX_desc-globalmean_mask.nii.gz	XXX_meanmask.nii.gz
XXX_desc-refine_mask.nii.gz	XXX_refinemask.nii.gz
XXX_desc-despeckle_mask.nii.gz	XXX_despecklemask.nii.gz
XXX_desc-corrout_info.nii.gz	XXX_corrout.nii.gz
XXX_desc-gaussout_info.nii.gz	XXX_gaussout.nii.gz
XXX_desc-autocorr_timeseries(.tsv, .json)	XXX_referenceautocorr_passN.txt
XXX_desc-corrdistdata_info(.tsv, .json)	XXX_corrdistdata_passN.txt
XXX_desc-nullsimfunc_hist(.tsv, .json)	XXX_nullsimfunchist_passN.txt
XXX_desc-plt0p050_mask.nii.gz	XXX_p_lt_0p050_mask.nii.gz
XXX_desc-plt0p010_mask.nii.gz	XXX_p_lt_0p010_mask.nii.gz
XXX_desc-plt0p005_mask.nii.gz	XXX_p_lt_0p005_mask.nii.gz
XXX_desc-plt0p001_mask.nii.gz	XXX_p_lt_0p001_mask.nii.gz
XXX_desc-globallag_hist(.tsv, .json)	XXX_globallaghist_passN.txt
XXX_desc-initialmovingregressor_timeseries(.tsv, .json)	XXX_reference_origres.txt, XXX_reference_origres_prefilt.txt
XXX_desc-movingregressor_timeseries(.tsv, .json)	XXX_reference_fmrires_passN.txt
XXX_desc-oversampledmovingregressor_timeseries(.tsv .json)	XXX_reference_resampres_passN.txt
XXX_desc-refinedmovingregressor_timeseries(.tsv, .json)	XXX_unfilteredrefinedregressor_passN.txt, XXX_refinedregressor_passN.txt
XXX_commandline.txt	XXX_commandline.txt
XXX_formattedcommandline.txt	XXX_formattedcommandline.txt
XXX_memusage.csv	XXX_memusage.csv
XXX_runtimings.txt	XXX_runtimings.txt

2.67 API

rapidtide

2.67.1 rapidtide

2.68 Contributing to rapidtide

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

2.68.1 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.

2.68.2 A note on current coding quality and style

This code has been in active development since June of 2012. This has two implications. The first is that it has been tuned and refined quite a bit over the years, with a lot of optimizations and bug fixes - most of the core routines have been tested fairly extensively to get rid of the stupidest bugs. I find new bugs all the time, but most of the showstoppers seem to be gone. The second result is that the coding style is all over the place. When I started writing this, I had just moved over from C, and it was basically a mental port of how I would write it in C (and I do mean just "C". Not C++, C#, or anything like that. You can literally say my old code has no Class (heh heh)), and was extremely unpythonic (I've been told by a somewhat reliable source that looking over some of my early python efforts "made his eyes bleed"). Over the years, as I've gone back and added functions, I periodically get embarrassed and upgrade things to a somewhat more modern coding style. I even put in some classes - that's what the cool kids do, right? But the pace of that effort has to be balanced with the fact that when I make major architectural changes, I tend to break things. So be patient with me, and keep in mind that you get what you pay for, and this cost you nothing! Function before form.

2.69 Theory of operation

If you're bored enough or misguided enough to be reading this section, you are my intended audience!

2.69.1 rapidtide

2.69.2 What is rapidtide trying to do?

Rapidtide attempts to separate an fMRI or NIRS dataset into two components - a single timecourse that appears throughout the dataset with varying time delays and intensities in each voxel, and everything else. We and others have observed that a large proportion of the "global mean signal", commonly referred to as "physiological noise" seen throughout in vivo datasets that quantify time dependant fluctuations in hemodynamic measures can be well modelled by a single timecourse with a range of time shifts. This has been seen in fMRI and NIRS data recorded throughout the brain and body, with time lags generally increasing at locations farther from the heart along the vasculature. This appears to be a signal carried by the blood, as changes in blood oxygenation and/or volume that propagate with bulk blood flow. The source of the signal is not known, being variously attributed to cardiac and respiratory changes over time, changes in blood CO2, gastric motility, and other sources (for a survey, see [Tong2019].) As biology is complicated, it's probably some mixture of these sources and others that we may not have considered. No matter what the source of the signal, this model can be exploited for a number of purposes.

If you're interested in hemodynamics, using rapidtide to get the time delay in every voxel gives you a lot of information that's otherwise hard or impossible to obtain noninvasively, namely the arrival time of blood in each voxel, and the fraction of the variance in that voxel that's accounted for by that moving signal, which is related to regional CBV (however there's also a factor that's due to blood oxygenation, so you have to interpret it carefully). You can use this information to understand the blood flow changes arising from vascular pathology, such as stroke or moyamoya disease, or to potentially see changes in blood flow due to a pharmacological intervention. In this case, the moving signal is not noise - it's the signal of interest. So the various maps rapidtide produces can be used to describe hemodynamics.

However, if you are interested in local rather than global hemodynamics, due to, say, neuronal activation, then this moving signal constitutes rather pernicious in-band noise. Global mean regression is often used to remove it, but this is not optimal - in fact it can generate spurious anticorrelations, which are not at all helpful. Rapidtide will regress out the moving signal, appropriately delayed in each voxel. This removes significantly more variance, and also avoids generating spurious correlations. For a detailed consideration of this, look here [Erdogan2016].

2.69.3 What is the difference between RIPTiDe and rapidtide?

RIPTiDe (Regressor Interpolation at Progressive Time Delays) is the name of the technique used for finding and removing time lagged physiological signals in fMRI data. In the original RIPTiDe papers, we generated a set of regressors over a range of different time shifts (starting from a regressor recorded outside of the brain), and then ran a GLM in FSL using the entire set of regressors. We realized that this 1) doesn't give you the optimal delay value directly, which turns out to be a useful thing to know, 2) burns degrees of freedom unnecessarily, since having one optimally delayed regressor in each voxel gets you pretty much the same degree of noise removal (this is assuming that in each voxel there is one and only one pool of delayed blood, which while not true, is true enough, since almost every voxel is dominated by a single pool of delayed blood), 3) is slow, since you're doing way more calculation than you need to, and 4) doesn't necessarily get you the best noise removal, since the systemic noise signal recorded outside the brain has its own characteristics and noise mechanisms that may make it diverge somewhat from what is actually getting into the brain (although on the plus side, it is inarguably non-neuronal, so you don't have to have any arguments about slow neuronal waves).

In contrast rapidtide (lets say it means Rapid Time Delay) is the newer faster, self-contained python program that implements an updated version of the RIPTiDe algorithm which estimates delay in every voxel and recursively refines an estimate of the "true" systemic noise signal propagating through the brain by shifting and merging the voxel timecourses to undo this effect. This refinement procedure is shown in Figure 5 of Tong, 2019 (reference 6 in the Physiology section below). In recent years, I've personally become more interested in estimating blood flow in the brain than denoising resting state data, so a lot of the documentation talks about that, but the two procedures are tightly coupled, and as the final step, rapidtide does regress the optimally delayed refined estimate of the systemic noise signal out of the data. We have found that it works quite well for resting state noise removal while avoiding the major problems of global signal regression (which we refer to as "static global signal regression" as opposed to "dynamic global signal regression", which is what rapidtide does). For a detailed exploration of this topic, we refer you again to [Erdogan2016] (also in the Physiology section below).

2.69.4 How does rapidtide work?

In order to perform this task, rapidtide does a number of things:

1. Obtain some initial estimate of the moving signal.
2. Preprocess this signal to selectively emphasize the bloodborne component.
3. Analyze the signal to find and correct, if possible, non-ideal properties that may confound the estimation of time delays.
4. Preprocess the incoming dataset to determine which voxels are suitable for analysis, and to emphasize the bloodborne component.
5. Determine the time delay in each voxel by finding the time when the voxel timecourse has the maximum similarity to the moving signal.
6. Optionally use this time delay information to generate a better estimate of the moving signal.
7. Repeat steps 3-7 as needed.
8. Parametrize the similarity between the moving signal and each voxels' timecourse, and save these metrics.
9. Optionally regress the voxelwise time delayed moving signal out of the original dataset.

Each of these steps (and substeps) has nuances which will be discussed below.

Generation of Masks

By default, rapidtide calculates masks dynamically at run time. There are 5 masks used: 1) the global mean mask, which determines which voxels are used to generate the initial global mean regressor, 2) The correlation mask, which determines which voxels you actually calculate rapidtide fits in (what you are describing here), 3) the refine mask, which selects which voxels are used to generate a refined regressor for the next fitting pass, 4) the offset mask, which determines which voxels are used to estimate the "zero" time of the delay distribution, and 5) the GLM mask, which determines which voxels have the rapidtide regressors removed.

Below is a description of how this works currently. NB: this is not how I THOUGHT is worked - until I just looked at the code just now. It built up over time, and evolved into something that was not quite what I designed. I'm going to fix it up, but this what it's doing as of 2.6.1, which works most of the time, but may not be what you want.

The default behavior is to first calculate the correlation mask using nilearn.masking.compute_epi_mask with default values. This is a complicated function, which I'm using as a bit of a black box. Documentation for it is here: https://nilearn.github.io/stable/modules/generated/nilearn.masking.compute_epi_mask.html#nilearn.masking.compute_epi_mask. If you have standard, non-zero-mean fMRI data, it seems to work pretty well, but you can specify your own mask using –corrmask NAME[:VALSPEC] (include any non-zero voxels in the file NAME in the mask. If VALSPEC is provided, only include voxels with integral values specified by VALSPEC in the mask). VALSPEC is a comma separated list of integers (1,2,7,12) and/or integer ranges (2-7,12-15) so you can make masks of complicated combinations of regions from an atlas. So for example –corrmask mymask.nii.gz:1,7-9,54 would include any voxels in mymask with values of 1, 7, 8, 9, or 54, whereas –corrmask mymask.nii.gz would include any non-zero voxels in mymask.

For the global mean mask: If –globalmeaninclude MASK[:VALSPEC] is specified, include all voxels selected by MASK[:VALSPEC]. If it is not specified, include all voxels in the mask. Then, if –globalmeanexclude MASK[:VALSPEC] is specified, remove any voxels selected by MASK[:VALSPEC] from the mask. If it is not specified, don't change the mask.

For the refine mean mask: If –refineinclude MASK[:VALSPEC] is specified, include all voxels selected by MASK[:VALSPEC]. If it is not specified, include all voxels in the correlation mask mask. Then if –refineexclude MASK[:VALSPEC] is specified, remove any voxels selected by MASK[:VALSPEC] from the mask. If it is not specified, don't change the mask. Then multiply by corrmask, since you can't use voxels where rapidtide was not run to do refinement.

For the offset mask If –offsetinclude MASK[:VALSPEC] is specified, include all voxels selected by MASK[:VALSPEC]. If it is not specified, include all voxels in the correlation mask. Then if –offsetexclude MASK[:VALSPEC] is specified, remove any voxels selected by MASK[:VALSPEC] from the mask. If it is not specified, don't change the mask. Then multiply by corrmask, and use the voxels within the mask to generate a histogram of delay values. Calculate the offset of the peak of the delay histogram, and subtract this value from all delay values within the correlation mask.

For the GLM mask: Include all voxels, unless you are calculating a CVR map, in which caserates other than the TR. Therefore the first step in moving regressor processing is to resample the moving regressor estimate to match the (oversampled) data sample rate.

Temporal filtering: By default, all data and moving regressors are temporally bandpass filtered to 0.009-0.15Hz (our standard definition of the LFO band). This can be overridden with `--filterband` and `--filterfreqs` command line options.

Depending on your data (including pathology), and what you want to accomplish, using the default correlation mask is not ideal. For example, if a subject has obvious pathology, you may want to exclude these voxels from being used to generate the initial global mean signal estimate, or from being used in refinement.

Initial Moving Signal Estimation

You can stabilize and improve rapidtide's delay estimation quite a bit by making sure you have a good starting regressor, estimating the global mean signal from "good" brain regions that don't have wacky delay structures. While just using the whole brain works well in young, healthy subjects (like the HCP-YA dataset), as people get older, their delays become weird - my working theory is that over time various routine vascular insults and unhealthy habits accumulate, leading to increasing heterogeneity between vascular territories (which I like to call "vascular personality"). So the global mean may be made up of several pools of blood, delayed by up to several seconds relative to each other, leading to weird autocorrelation in the global mean (essentially, confusing echoes of the moving signal) that can confuse my delay finding algorithm, because it invalidates my assumption that the global mean is a good initial estimate of the "true" moving regressor. One way to combat this is to limit the brain region that you get your initial regressor from, so that you are only sampling a single "pool" of delays. For example, you could use a gray matter mask for the global regresor estimation, since white matter has a smaller contribution from the moving blood signal, and tends to get blood much later than gray matter anyway. Just add the option `--globalmeaninclude graymask.nii.gz` to your rapidtide command line. If you are using fmriprep, you can get a gray matter mask using:

```
fslmaths \
    BIDSHOME/derivatives/fmriprep/sub-XXX/anat/sub-YYY_space-MNI152NLin6Asym_res-2_label-
↪GM_probseg.nii.gz \
    -s 3 \
    -thr 0.25 \
    -bin \
    graymask
```

If you want to be even more proactive, you could select a more focal brain region that you think has unperturbed circulation. For an Alzheimer's study that I am currently working on, we ended up starting only from blood in right and left cerebellar gray matter (freesurfer aparc+aseg regions 8 and 47) on the theory that if circulation in your cerebellum is too messed up, you're dead, so would not be in the dataset. That made our delay estimates work a lot better. So we used the freesurfer parcellations from fmriprep, transformed to standard space, to do that preselection, using the option `--globalmeaninclude standardspaceaparcasegfilename.nii.gz:8,47`.

fmriprep does not provide a standard space aparc+aseg file - it's in T1 native space at 1mm resolution (because that's the space freesurfer works in). Resampling to standard space is easy, BUT you must remember to use NearestNeighbor interpolation, or you'll get smeared, averaged boundaries between brain regions, which you REALLY don't want. This command should get you a `standardspaceaparcasegfilename.nii.gz` (you need to have ANTs installed for this):

```
antsApplyTransforms \
    -d 3 \
    -i BIDSHOME/derivatives/sub-XXX/anat/sub-XXX_desc-aparcaseg_dseg.nii.gz \
    -o BIDSHOME/derivatives/sub-XXX/anat/mymnispace_desc-aparcaseg_dseg.nii.gz \
    -r BIDSHOME/derivatives/sub-XXX/anat/sub-XXX_space-MNI152NLin6Asym_res-2_desc-
↪preproc_T1w.nii.gz \
    --interpolation NearestNeighbor \
    --transform BIDSHOME/derivatives/sub-XXX/anat/sub-XXX_from-T1w_to-MNI152NLin6Asym_
↪mode-image_xfm.h5
```

Moving Signal Preprocessing

Before launching into in each pass, we process our moving regressor to make it more amenable to calculations. This includes the following operations:

Oversampling: In order to simplify delay calculation, rapidtide performs all delay estimation operations on data with a sample rate of 2Hz or faster. Since most fMRI is recorded with a TR > 0.5s, this is achieved by oversampling the data. The oversampling factor can be specified explicitly (using the `--oversampfac` command line argument), but if it is not given, for data with a sample rate of less than 2Hz, all data and regressors are internally upsampled by the lowest integral factor that results in a sample rate >= 2Hz.

Regressor resampling: In the case where we are using the global mean signal as the moving signal, the moving signal estimate and the fMRI data have the same sample rate, but if we use external recordings, such as NIRS or etCO2 timecourses, these will in general have sample rates other than the TR, and may start before and/or end after the fMRI acquisition. Therefore the first step in moving regressor processing is to resample the moving regressor estimate to match the (oversampled) data sample rate and time range.

Temporal filtering: The moving regressor is then filtered to the appropriate frequency range - by default the LFO band (0.009-0.15Hz).

Detrending and normalization: The regressor is detrended to Nth order (N=3 by default), demeaned, and divided by the standard deviation over time.

Windowing: We apply a window function to the regressor to improve the correlation properties. By default, this is a Hamming window, but you can also select Hann, Blackman-Harris, or None, with the `--windowfunc` argument.

Zero padding: The regressor is zero padded on each end to twice its length, so that we will be doing a linear rather than circular correlation (you can select circular correlation with `--corrtype`, but I wouldn't recommend it.

Moving Signal Massaging

Because the moving signal is "noise", we can't select or specify its properties, and sometimes the sLFO signal you end up with is problematic for one reason or another. Rapidtide attempts to correct, where possible, problems in the moving signal that will impair proper delay estimation. Again, if you're just doing signal denoising, these are not that important to you.

Pseudoperiodicity: The first potential problem in the sLFO regressor is pseudoperiodicity. From time to time, signal energy in the 0.009-0.15 Hz band will be strongly concentrated in one or more spectral peaks. This can be completely random, or it can arise due to some pathological or congenital condition that affects circulation. It seems for the most part to be purely by chance, as you occasionally see it when looking at multiple runs in the same subject, where one run is pseudoperiodic while the rest are not. The effect of this is to cause the crosscorrelation between the probe signal and voxel timecourses to have more than one strong correlation peak. This means that in the presence of noise, or extreme spectral concentration of the sLFO, the wrong crosscorrelation peak can appear larger, leading to an incorrect delay estimation. This is particularly problematic if the pseudoperiod is shorter than the reciprocal of the search window (for example, if the search window for correlation peaks is between -5 and +5 seconds, and the sLFO has a strong spectral component at 0.1Hz or higher, more than one correlation peak will occur within the search window). As the width of the search range increases, the spectral range of potentially confounding spectral peaks covers more of the sLFO frequency band.

Implications of pseudoperiodicity: The extent to which pseudoperiodicity is a problem depends on the application. In the case of noise removal, where the goal is to remove the global sLFO signal, and leave the local or networked neuronal signal variance, it turns out not to be much of a problem at all. If the sLFO signal in a given voxel is sufficiently periodic that that the correctly delayed signal is indistinguishable from the signal one or more periods away, then it doesn't matter which signal is removed – the resulting denoised signal is the same. As the Host in Westworld asked - "Well if you can't tell, does it matter?" In this case, no. Sadly, for those of you care more about hemodynamics than neuronal activation (raises hand), this is NOT ok, and we have to figure out how to deal with it.

Mitigation of pseudoperiodicity: While we continue to work on fully resolving this issue, we have a number of hackish ways of dealing with this. First of all, spectral analysis of the sLFO signal allows us to determine if the signal may be problematic. Rapidtide checks the autocorrelation function of the sLFO signal for large sidelobes with periods within the delay search window and issues a warning when these signals are present. Then after delay maps are calculated, they are processed with an iterative despeckling process analogous to phase unwrapping. The delay of each voxel is compared to the median delay of its neighbors. If the voxel delay differs by the period of an identified problematic sidelobe, the delay is switched to the "correct" value, and refit. This procedure greatly attenuates, but does not completely solve, the problem of bad sidelobes. A more general solution to the problem of non-uniform spectra will likely improve the correction.

Correlation weighting: Another method I've recently implemented is "regressor weighting" the correlation function - since we do correlation in the spectral domain, you can normalize the power spectrum magnitude by the power spectrum of the sLFO regressor - this deemphasizes spectral peaks. It helps, but it's not a magic wand.

Echo cancellation: One thing that I keep thinking about is that in the case of pathology causing disparate delay pools, we are essentially looking at an echo cancellation problem. We have a driving signal, and it is corrupted by delayed copies of itself being added in. This is a problem that Bell Labs solved in the 60s or 70s (well digitally - I think analog echo cancellation existed long before that). It seems like I should be able to dust off some audio library somewhere that would fix this right up, but I haven't found anything yet. Any bored audio engineers looking to get involved in a FOSS neuroimaging project :-) ?

Most of the options languishing in the "experimental" group of command line options are partially implemented versions of various regressor fixes.

Dataset Preprocessing

Prior to processing, I do a few things to the fMRI dataset:

Spatial filtering: While the moving signal can contribute up to 50% of the low frequency variance in gray matter voxels, it's often MUCH less than that, especially in white matter. So anything you can do to boost your SNR is a plus. Spatial filtering works for that - for the most part, the delay time varies quite smoothly over space, since capillary blood (to which we are most sensitive) moves in a pretty orderly fashion. Even a small amount of smoothing is sufficient to boost the quality of the delay maps a lot. A Gaussian kernel with a radius of ~1/2 the average voxel dimension in all three axes turns out to be pretty good. Use `--spatialfilt SIGMA` to set the filtering. Set SIGMA to -1 to have it set automatically as described above (default), or set SIGMA to the kernel size in mm. SIGMA=0 turns spatial filtering off.

Mask, trim to size and reshape: Select only the voxels and timpoints that are going to be processed, as specified by the spatial masks, and the `--numskip` and `--timerange` options, and reformat the remaining data into a voxel by time array. This simplifies all of the subsequent processing. Spatial filtering (done previously) and despeckling (managed by mapping lag data back to x, y, z space to check against neighbors) are the only operations that require us to know the spatial relationship between voxels.

Significance threshold estimation

This step is placed where it is done in the processing stream, but involves procedures described below.

Estimating the significance threshold for the fitted crosscorrelation measurements done below is not straightforward. While there is a standard relationship to convert correlation coefficient R to p for a given timecourse length, this assumes that you performing a Pearsonn correlation of truly random signals (i.e. Gaussian random signals with white noise power spectra). But the sLFO signals are severely band limited, so if you use these formulae, you will dramatically overestimate the significance of your correlations. Moreover, we are selecting the peak of a crosscorrelation over a range of delays, which will further inflate the values. There are analytical ways of adjusting for this, but they are tedious - Monte Carlo simulation by performing and fitting a set of crosscorrelations of the sLFO regressor with scrambled, filtered versions of itself are more straightforward (this is described in [Hocke2016]). Prior to each pass, we do NREPS of these sham correlations (NREPS=10000 by default - adjust with `--numnull NREPS`. Set to 0 to disable significance

estimation). The p<0.05, p<0.01, and p<0.005 significance thresholds are estimated by fitting the set of null correlations to a Johnson SB distribution (the functional form which we empirically found best fits the data).

Time delay determination

This is the core of the program, that actually does the delay determination. It's currently divided into two parts - calculation of a time dependant similarity function between the sLFO regressor and each voxel (currently using one of three methods), and then a fitting step to find the peak time delay and strength of association between the two signals.

Signal preparation

Prior to processing, each timecourse is processed in the same way as the moving regressor (oversampling, filtering, detrending, applying the same window function used on the reference regressor, and zeropadding the ends.)

Types of similarity function

Crosscorrelation: The most straightforward way to calculate similarity between two timecourses is crosscorrelation. It has several advantages - interpretation is easy - the magnitude of the function ranges from 0 (no similarity) to 1 (timecourses are identical). Negative magnitudes mean that the one timecourse is inverted relative to the other. It is also extremely fast to calculate in the spectral domain (O(2Nlog2N) rather than O(N2)). For signals of the length of typical fMRI scans, calculation in the spectral domain is substantially faster than in the time domain. However, it does have drawbacks. First, it assumes the relationship between the signals is linear. In practice, this is generally ok for our purposes, but is not ideal. More problematic is unpredictable behavior when the SNR is low (as it is in voxels with lower blood content, such as white matter), which can make the signal harder to quantify (more below). Use `--similaritymetric correlation` to select crosscorrelation (default).

Mutual information: Mutual information (MI) is a very different method of quantifying similarity. It is a measure of the amount of information you can gain about one signal from the other (yes, I know the definition is about "random variables", but for our purposes, we mean timecourses). So, there is no assumption of linearity (or in fact any assumption whatsoever about the functional form of the relationship). That's cool, because it really frees you up in terms of what you can look at (as an aside, I'm not sure why this isn't used more in task based analyses - it seems like it could get past having to know the exact form of the hemodynamic response function). MI is especially useful in image registration, for example, lining T2 weighted functional images up with T1 weighted anatomics. The cross-MI has some nice properties.

- It tends to give sharp peaks when signals are aligned, even in cases where the source data is lowpass filtered.
- As mentioned above, it really doesn't care how signals are related, only that they are. So you aren't restricted to linear relationships between signals.

So why don't we use it for everything? A couple of reasons.

- It's much more computationally expensive than correlation (O(N2) at least). My implementation of a cross-MI function (which is actually pretty fast) still takes about 10x as long to calculate as crosscorrelation for typical fMRI data.
- It does not have as straightforward an interpretation as crosscorrelation - while there are "normalized" calculations, "1" does not mean identical, "0" does not mean unrelated, and it's positive definite. The MI of a signal with itself is the same as the MI of -1 times itself. For cross-MI, you can really only rely on the fact that you get a maximum when the signals are most aligned.

Use `--similaritymetric mutualinfo` to select MI.

Hybrid similarity: I'm kind of proud of this one. Crosscorrelation is fast and interpretable, but has the problem of ambiguous time delay values, whereas cross-MI is very slow and hard to interpret, but quite unambiguous in selecting

the best match. Enter "hybrid similarity" - Use the crosscorrelation to identify candidate peaks, then calculate the MI only at those peak locations, pick the one that has the higher MI, and then proceed to the fitting step for full quantification. This is almost as fast as straight correlation, but does tend to be more stable. Use `--similaritymetric hybrid` to select hybrid similarity.

Peak fitting and quantification

The second part of this process is peak fitting and quantification. For most of this discussion, I'll refer to crosscorrelation, since its what I usually use.

To first approximation, fitting isn't necessary. The crosscorrelation function will always have a maximum somewhere, and if you've chosen your search range to cover the range of time lags that blood will have, it will fall within that range. However, that's not a great way to do things. If you do this, your delay values will be quantized, either to TR, or in our case, TR divided by the oversampling factor (which is why we oversampled to begin with). The delay range in healthy young adults runs from about -2 to +4 seconds, and is strongly peaked near 0. Using our default oversampling, which makes the effective TR 0.5 seconds, that gives you at most 13 possible delay values, with most of them in a more restricted range of 5 or so values. While somewhat useful, this is throwing away a lot of information unnecessarily.

Remember that the sLFO signal is bandlimited to 0.009 to 0.15Hz, which means the highest frequency component in the data has a period of about 6.67 seconds. So at a minimum, the correlation peak will be several seconds across, so in addition to the peak location, there will be several points on either side that carry information about the peak location, height, and width. If you fit all the points around the peak, you'll get a much better estimate of the true delay and correlation value.

Correlation peaks can be a little messy; low pass filtering, weird autocorrelation properties due to nonuniform power spectra, window function choices, and baseline roll can lead to incorrect peak identification. This makes the peak fitting process complicated.

Despeckling

As mentioned above, your correlation function may be pseudoperiodic due to an unfortunate power spectrum. At this point, the delay maps are subjected to a multipass despeckling operation, where voxels that look like they may have had incorrect fits are refit to be more consistent with their neighbors.

Generating a better moving signal estimate (refinement)

Now that we have an estimate of when the moving regressor arrives at every voxel, we can make a better estimate of the driving signal.

Voxel selection

First we pick the voxels we want to use to generate the new estimate. We can set the starting mask explicitly using the `--refineinclude MASKFILE:VALSPEC` and `--refineexclude MASKFILE:VALSPEC` command line options. If left unset, we use all voxels with valid correlation fits. We can further tune which voxels are excluded from refinement with the `--norefinedespeckled`, `--lagminthresh`, `--lagmaxthresh`, and `--sigmathresh` options. By default, we also exclude voxels with correlation strengths less than the p<0.05 threshold found using the significance threshold estimation step above, or we can override this threshold using `--ampthresh`.

Timecourse alignment

In each of the voxels selected for refinement, we first negate the time delay in every voxel and timeshift the voxel by that amount. This will have the effect of bringing the portion of the signal in each voxel due to the moving sLFO signal into alignment.

Prescaling

We then prenormalize the voxels to use in the fit using their mean, variance, or standard deviation over time, the inverse of the lag time, or leave them unscaled. Selection is via the `--refineprenorm` option. The default is to do no prenormalization.

New timecourse generation

The new timecourse is then generated from the set of aligned, scaled timecourses using a method specified with `--refinetype`:

pca (default): Perform a principal component analysis on the timecourses, reprojecting them onto a reduced set of components (specified by `--pcacomponents` - the default is the set explaining >=80% of total variance). Average the result.

ica: Perform an independent component analysis on the timecourses, reprojecting them onto a reduced set of components (specified by `--pcacomponents` - the default is the set explaining >=80% of total variance). Average the result.

weighted_average: Each voxel is scaled with either the correlation strength from the current pass, the square of the correlation strength, or is left unscaled. This is selected with the `--refineweighting` option - the default is "R2". The timecourses are then averaged.

unweighted_average: Average the voxels.

Lather, Rinse, Repeat

Now that there is a new starting regressor, repeat the entire process some number of times. This can be a fixed number of passes, specified by `--passes NUMPASSES`. The default is to do 3 passes. Alternatively, by specifying `--convergencethresh THRESH`, the process is repeated until either the MSE between the new sLFO regresssor and the regressor from the previous pass falls below THRESH, or the number of passes reaches MAX, specified by `--maxpasses MAX` (default is 15).

Regress Out the Moving Signal

Now that we have optimized the moving blood signal and have final estimates of blood arrival time at each voxel, we can do the final regression to (intelligently) remove the sLFO signal from the data. By default, this is done on the original, unmodified data - i.e. none of the spatial or temporal filtering, masking, confound regression, or anything else has been done. The reason for this is that some of the operations may be needed to get a good sLFO regressor estimate, or a good delay map, but they could interfere with whatever further analysis you might want to do after sLFO removal. You can always do them later if you want. Also, if you really want to keep all those manipulations, you can choose to by selecting `--preservefiltering`. But don't.

Alternately, instead of loading the original file, you can load a _different_ file, and denoise that instead. Why would you want to do that? This is here for a very particular reason. HCP data uses FIX, a really spiffy ICA noise removal tool that cleans things up quite a bit. However, as mentioned above in the rapidtide usage section, it does tend to remove a lot of hemodynamic signal in some regions, particularly around the superior sagittal sinus. That makes rapidtide's

sLFO estimation and refinement process a lot less stable. So you really want to do that estimation on non-FIX'ed data (the "minimally processed" data). Ideally, you would then run FIX on the rapidtide cleaned data, but that's a lot of computation that you don't necessarily want to do. So a cheat is to regress the voxel specific noise regressors out of the FIX cleaned data. Since the operations are linear, the order shouldn't matter (waves hands to distract from the fact that FIX has probably generated some spurious negative correlations by regressing out hemodynamic signal at the wrong time delay). Anyway, while it's not perfect, it's better than not doing it this way.

Finally, if you don't want to do glm filtering at all (i.e. you only care about time delays, and want to minimize storage space), you can shut off the glm filtering with `--noglm`.

References

2.70 Release history

2.70.1 Version 2.8.7 (4/17/24)

- (rapidtide) Spatial filtering with the default kernel (1/2 the average voxel dimension) is now the default.
- (rapidtide) Rewrote the lag time rank order calculation to use scipy.signal.rankdata.
- (rapidtide) Moved some command line options to the "experimental" section, where they should have been to begin with.
- (rapidtide) Moved GLM options to a new "GLM" section.
- (rapidtide) Do not automatically disable windowing for linear correlation.
- (rapidtide, happy) Reorganized, expanded and rationalized confound regression code.
- (tidepool) Replaced the flame colormap option with plasma and jet with turbo (similar, to what they replaced, but perceptually uniform).
- (package) Made some proactive fixes for numpy 2.0.0 compatibility.
- (package) Merged some dependabot PRs.
- (tests) Removed some superfluous metadata from test data files.
- (docs) Further extensive revisions.

2.70.2 Version 2.8.6 (4/5/24)

- (rapidtide) Tweaked the behavior of the `--CVR` flag.
- (rapidtide) Made major improvements to motion regression.
- (rapidtide) Consolodated all glm filtering into a single multiprocessing routine, added some new capabilities to rapidtide GLM filtering.
- (resampletc) Renamed from resamp1tc to make the program names more consistent.
- (package) Made pyfftw an optional dependency, since it seems to be the only thing blocking python 3.12 compatibility.
- (package) Added some new tests.
- (package) Major reorganization of most scripts to make them workflows (this is to make the calling interfaces and documentation consistent.)
- (package) Incorporated dependabot changes.

- (docs) Further cleanup of the program documentation section, especially rapidtide usage.
- (docs) Expanded the Theory of Operation section.
- (docs) Fixed many typos.

2.70.3 Version 2.8.5.1 (4/1/24)

- (docs) Extensive revision of the rapidtide program usage section.

2.70.4 Version 2.8.5 (3/30/24)

- (rapidtide) Adjusted some default parameters to reflect the current best picks.
- (rapidtide) Fixed the –denoising option (it didn't actually do anything!)
- (package) Partially implemented a major reorganization of all the programs in the package to make them workflows with standard interfaces.
- (docs) Started the process of standardizing the program descriptions and properly using autodoc.

2.70.5 Version 2.8.4 (3/28/24)

- (rapidtide) Output some .json sidecars that I had neglected.
- (glmsim) New program to help develop instructional tools re: how rapidtide works. This is a WIP.
- (docs) Major revisions to the rapidtide usage instructions.
- (package) Accepted several dependabot changes.

2.70.6 Version 2.8.3 (3/7/24)

- (rapidtide) Fixed the logic for saving lagregressors - they only exist if you do GLM or CVR analysis, so if you set nolimitoutput, check for existance first (thanks to Laura Murray for finding this bug).
- (rapidtide) Changed the name of the file containing the voxel specific EVs that are regressed out by the GLM from "lagregressors_bold" to "lfofilterEVs_bold" (thanks to Tianye Zhai for flagging this).
- (localflow) Added a new program to test a hunch.
- (fit) Gracefully handle singular matrices in mlregress.
- (reference) Corrected the abbreviated name for the MLSR region in the JHU level 1 atlas xml file (oops!).
- (docs) Added description of the lfofilterEVs_bold and shiftedtcs_bold output files to the usage section.

2.70.7 Version 2.8.2 (2/26/24)

- (rapidtide) Added a lot more internal debugging resources, and fixed a bug where zero range time data that was included due to explicit masks would put NaN's in the maps.
- (rapidtide) Implemented multiprocessing to speed up motion regression.

2.70.8 Version 2.8.1 (2/19/24)

- (cloud) Now using an s3 working folder.

2.70.9 Version 2.8.0 (2/18/24)

- (Docker) Set to basecontainer_plus:latest-release to simplify builds.

2.70.10 Version 2.7.9 (2/18/24)

- (runqualitycheck) Added new tests.
- (reference) Added MNI152NLin2009cAsym versions of the HCP reference maps.
- (cloud) Added support for selecting the dataset (currently HCPA and ABCD are supported).

2.70.11 Version 2.7.8 (1/31/24)

- (rapidtide) Added new feature - '–numtozero NUMPOINTS' allows you to specify how many points at the beginning of the data set to set to zero prior to processing. This means that the sLFO fit does not get contaminated by synchronous noise at the beginning of the timecourse (such as T1 decay). The initial timepoints are filled in from timecourses shifted forward in time. This means better correlation of timecourses with the sLFO, and better noise removal with the GLM.
- (rapidtide) Fixed a bug in how setting timerange and simcalc range interacted (and in range checking). It all works now, and simcalc range is specified relative to the restricted time range.
- (runqualitycheck) Fixed some masking bugs.

2.70.12 Version 2.7.7 (1/31/24)

- (runqualitycheck) Added new tests, and the ability to optionally do tests restricted to gray and/or white matter.
- (package) The makeandsavehistogram routine now saves some useful histogram stats in the json file.
- (package) Added the ability to specify APARC_WHITE and APARC_ALLBUTCSF macros to a mask specification if you have an aparc+aseg file.
- (RapidtideDataset) You can now optionally include gray and white matter masks in the dataset.

2.70.13 Version 2.7.6 (1/29/24)

- (rapidtide) Added the ability to calculate delays over a limited time range, but still GLM filter the entire timecourse.
- (rapidtide) Fixed a very old bug in null significance distribution estimation. Multiple worker processes all start with the same random seed (unless you explicitly fix that). Who knew?
- (rapidtide) Improved significance distribution model fitting for mutualinfo similarity metric. The distribution is a Gaussian, not a Johnson distribution (as it is for selected correlation).
- (runqualitycheck) Added an automated quality assessment tool. This will likely evolve quite a bit over time.
- (rapidtide2std) Updated for new maps; also copied over timecourses and options so you can load a rapidtide2std dataset into tidepool.
- (atlasaverage) Set output NIFTI size properly for 3D templates.
- (testing) Parallelized tests on CircleCI for a significant speedup.
- (package) Updated copyright messages, made headers more consistent, removed some old SCCS tags.

2.70.14 Version 2.7.5 (1/13/24)

- (rapidtide) Moved lagtc generation out of fitcorr into its own module. This will help with implementation of new, secret evil plans.
- (rapidtide) Altered voxel selection logic for multiproc correlation fitting. Now singleproc and multiproc outputs are the same.
- (rapidtide) Fixed a multiprocessing bug that's been there since multiprocessing was added - any job with an integral multiple of 50000 tasks would die (don't ask).
- (rapidtide) Fixed a bug that allowed NaNs into the lfoCleanedR2 map.
- (rapidtide) General code cleanup.
- (package) Accepted some dependabot PRs for security updates.

2.70.15 Version 2.7.4 (1/10/24)

- (rapidtide) Fixed a crash when despeckling is turned off. (thank you to Wesley Richerson for finding this).
- (rapidtide) Adjusted the regreessor frequency setting logic.
- (rapidtide) Adjusted the default absminsigma to 0.05s.
- (rapidtide) Moved motion regression before global mean correction.
- (rapidtide) Properly reinitialize the motion regression output file if you have a previous run.

2.70.16 Version 2.7.3.3 (12/18/23)

- (rapidtide-cloud) Another bump to improve NDA access.

2.70.17 Version 2.7.3.2 (12/18/23)

- (rapidtide-cloud) Another bump to improve NDA access.

2.70.18 Version 2.7.3.1 (12/18/23)

- (rapidtide-cloud) Redoing push to fix a regression due to *somebody* not testing before deploying (tsk tsk).

2.70.19 Version 2.7.3 (12/18/23)

- (correlate) Added a new correlation weighting - "regressor", that whitens the correlation spectrum relative to the probe regressor.
- (rapidtide) Add support for the "regressor" correlation weighting.
- (rapidtide) Linear (rather than circular) correlations are now the default.
- (rapidtide) Add infrastructure to support baseline correction of correlation function during lag estimation.
- (rapidtide) Added lag rank map (each voxel is the percentile within the lag time distribution)
- (showarbcorr) Numerous bugfixes and functionality improvements.
- (tidepool) Support new lag rank map.
- (rankimage) Convert lag maps to lag rank maps.
- (rapidtide-cloud) Added tools for NDA download.

2.70.20 Version 2.7.2 (12/12/23)

- (Docker) Bumped to basecontainer_plus v0.0.3.
- (Docker) Removed push of new containers to ECR.

2.70.21 Version 2.7.1 (12/12/23)

- (Docker) Fixed deployment of new containers to ECR.

2.70.22 Version 2.7.0 (12/11/23)

- (Docker) Added caching to build.
- (Docker) Switched to basecontainer_plus to pick up some FSL utilities.

2.70.23 Version 2.6.9.1 (12/11/23)

- (package) Fixing some mysterious deploy errors.

2.70.24 Version 2.6.9 (12/11/23)

- (filter) Updated predefined filter bands to include hrv frequency ranges.
- (happy) Tried a new approach for aliased correlation. Not really done yet.
- (docs) Updated installation instructions.
- (docs) Fixed a build problem.
- (Docker) Update to basecontainer v0.3.0.

2.70.25 Version 2.6.8 (11/21/23)

- (rapidtide) Rapidtide is now less chatty by default.
- (rapidtide) Put the significance estimation command line options in their own subsection.
- (tidepool) Updated to support the newest pyqtgraph (>=0.13.0).
- (Docker) Update to basecontainer v0.2.9.1.
- (package) Increased test coverage to 49.26%.
- (docs) Fully documented tidepool.

2.70.26 Version 2.6.7 (10/31/23)

- (Docker) Update to basecontainer v0.2.7.
- (rapidtide) Added the option to delete noise signals from the probe regressor (mostly due to slow breathing). Currently not working.
- (rapidtide) All outputs from rapidtide are in BIDS derivative format. The ability to select legacy outputs has been removed.
- (happy) All outputs from happy are in BIDS derivative format. The ability to select legacy outputs has been removed.
- (rapidtide2x_legacy) The legacy version of rapidtide (rapidtide2x_legacy) has been removed.
- (happy_legacy) The legacy version of happy (happy_legacy) has been removed.
- (showtc) Fixed a very old bug that caused some timecourses to not be properly aligned if they had different start times.
- (showtc) Added ability to normalize all timecourses to make displaying them together more informative.
- (package) Added selfcontained routines to do glm filtering (with or without polynomial expansion, and to align timecourses.
- (package) Added macros for selecting all the gray matter values in an aparc+aseg file.
- (package) Accepted dependabot changes.
- (rapidtide-cloud) Added basecontainer to AWS.
- (rapidtide-cloud) Various tweaks and changes to AWS authentication procedures to deal with NDA.

- (docs) Some updates to theory of operation.

2.70.27 Version 2.6.6 (10/7/23)

- (adjustoffset) New tool to alter overall delay offset in maxtime maps.
- (Docker, package) Really, truly, actually fixed version reporting.
- (rapidtide) Added debugging option to disable docker memory limit "fix".

2.70.28 Version 2.6.5 (10/4/23)

- (rapidtide) Report version on startup. Resolves https://github.com/bbfrederick/rapidtide/issues/91.
- (Docker, package) Fixed version tagging and reporting. Resolves https://github.com/bbfrederick/rapidtide/issues/96.
- (Docker) Moved some time consuming installations into basecontainer to make building new containers MUCH faster.
- (package) Merged some dependabot security PRs.
- (diffrois) Fixed handling of missing values.

2.70.29 Version 2.6.4 (9/28/23)

- Mass merge of more dependabot PRs.
- (diffrois) Added a new program to make "vasculomes" - measuring delay differences between ROIs. This is still in flux.
- (fingerprint, atlasaverage) Implemented a standard masking method with atlas indexed include and exclude masks, and an extra geometric mask.
- (fingerprint) Bug fixes.

2.70.30 Version 2.6.3 (9/13/23)

- Mass merge of a bunch of dependabot PRs.
- (rapidtide) Fixed return values from findavailablemem() when running in a Docker container with cgroups v1. Thank you to Jeffrey N Stout for finding this. Should resolve https://github.com/bbfrederick/rapidtide/issues/122.
- (Docker) Updated to basecontainer 0.2.3.

2.70.31 Version 2.6.2 (8/29/23)

- (atlastool) Add ability to use ANTs alignments.
- (atlasaverage) Add ability to restrict statistics to non-zero voxels.
- (documentation) Started beefing up the "Theory of operation" section.
- (Docker) Set memory limits on resource use when running in Docker containers so you don't get silent out of memory failures.

2.70.32 Version 2.6.1 (8/17/23)

- (rapidtide) Fixed crash when using –acfix option. Thanks to Jakub Szewczyk for spotting this. Should resolve https://github.com/bbfrederick/rapidtide/issues/115.
- (atlasaverage) Added text region summary outputs.
- (atlastool) Enhancing spatial registration options.
- (package) Initial steps to implementing a more flexible way of applying external registration tools to data.
- (package) Moving closer to a single pyproject.toml file with all the packaging information in it.:
- (Docker) Updated to basecontainer 0.2.1 and added new cleanup operations - the container is now ~30% smaller.

2.70.33 Version 2.6.0 (8/10/23)

- (rapidtide) Added new "–CVR" analysis type to generate calibrated CVR maps when given a CO2 regressor as input. Thanks to Kristina Zvolanek for the suggestion to add it!
- (rapidtide) Fixed calculation and output of variance change after GLM filtering.
- (happy) Moved support functions into a separate file.
- (simdata) Added separate voxel level and volume level noise specification, and a test script.
- (documentation) Added information on CVR mapping outputs, updated funding information.
- (package) Made ArgumentParser initialization uniform to make automatic documentation easier.
- (package) Removed Python 3.7 support (mostly because it doesn't support all the features of f-strings I use.)

2.70.34 Version 2.5.8 (8/3/23)

- (rapidtide) –nofitfilt now actually works. Thank you to https://github.com/poeplau for finding (and fixing) the problem! Resolves https://github.com/bbfrederick/rapidtide/issues/114

2.70.35 Version 2.5.7 (5/15/23)

- (glmfilt) Added ability to specify a mask, and to limit output files.

2.70.36 Version 2.5.6 (5/14/23)

- (niftidecomp) Made some major internal changes to allow processing multiple files at once.
- (gmscalc) New program to do some global mean signal calculations within the package (so we can do them on AWS).

2.70.37 Version 2.5.5 (5/11/23)

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

2.70.38 Version 2.5.4 (5/10/23)

- (rapidtide) Default to using N processors rather than N-1 when nprocs=-1. You can toggle old behavior with –reservecpu.
- (rapidtide-cloud) Rapidtide will record the instance type if running on AWS in the options file (AWS_instancetype).

2.70.39 Version 2.5.3.1 (5/9/23)

- (rapidtide, happy) Fixed a crash when you DIDN'T specify informational tags (SMH).

2.70.40 Version 2.5.3 (5/9/23)

- (rapidtide, happy) Added the ability to save arbitrary informational tags to the run options (or info) json files using the –infotag command line argument.

2.70.41 Version 2.5.2 (5/8/23)

- (rapidtide) Now allow the global mean mask to be completely outside of the correlation mask (the fact this previously wasn't allowed was a bug). Thank you to Daniele Marinazzo for finding this.
- (rapidtide) Fixed a bug in formatting run timings.
- (filttc) Now allow normalization before or after filtering.
- (showxcorrx) Made fit width limits configurable.
- (calcicc) Moved main calculations into niftistats, made two shells to calculate either icc or ttests.
- (package) Disabled numba because of multiple bugs and incompatibility with py3.11 and ARM.
- (package) Made some updates to rapidtide cloud routines to make things a bit more stable.

2.70.42 Version 2.5.1.2 (4/28/23)

- (package) New release to trigger ECR upload.

2.70.43 Version 2.5.1.1 (4/28/23)

- (package) New release to trigger ECR upload.

2.70.44 Version 2.5.1 (4/28/23)

- (package) New release to trigger ECR upload.

2.70.45 Version 2.5 (4/28/23)

- (package) Fixed and upgraded tests both locally and on CircleCI.
- (package) Fixed coverage calculation and upload and increased coverage to 52%.
- (package) Made some changes to support our new AWS account (dmd).
- (reference) Added XML files for the JHU level 2 arterial atlases.

2.70.46 Version 2.4.5.1 (4/10/23)

- (docs) Removed duplicate funding source. Hopefully this will resolve the Pypi upload issue.

2.70.47 Version 2.4.5 (4/10/23)

- (docs) Addded some new sections to theory.
- (package) Completely changed the way I handle and distribute test data. This makes the package much smaller (~17M), which should fix pypi deployment. This involved several changes to the Docker and circleCI workflows, which I think are now stable.

2.70.48 Version 2.4.4 (3/30/23)

- (examples) Separated essential test data from developer test data to make the installation much smaller.
- (package) Major modernization to package build and test files.

2.70.49 Version 2.4.3 (3/30/23)

- (rapidtide) Some work on phase permutation for null correlation calculation.
- (happy) Put in the skeletons of some new features (upsampling, optical flow calculation).
- (OrthoImageItem.py) Removed the last of the obsolete pyqtgraph calls.
- (package) Attempting to modernize the packaging scripts to avoid deprecated behavior.
- (package) Several changes to fix the build environment.

2.70.50 Version 2.4.2 (2/8/23)

- (rapidtide) Added ability set a threshold value for "equivalence" of spatial dimensions of NIFTI files (rather than requiring an exact match) using the –spatialtolerance option.
- (rapidtide) Added "offset masks" to set the region that defines "zero" time offset.
- (fingerprint) Added several new summary statistics for each region.
- (fingerprint) Allow the use of 3D masks with 4D data.
- (tidepool) Resolve a deprecation in Qt.
- (tidepool) Made tidepool less chatty by default (default verbosity is now 0).
- (Docker) Cleaned up the container build.
- (package) Critical bug fix for multiprocessing in python versions 3.10 and above ("10" < "8"! Who knew?)
- (package) Added ".csv" files as a supported text file type.
- (package) Improved version handling when in a container.
- (reference) Added XML files to make the JHU arterial atlases loadable in FSLeyes.

2.70.51 Version 2.4.1 (10/12/22)

- (package) Spruced up all the progress bars with tqdm.
- (deployment) Improved the testing structure to cache environment builds.
- (Docker) Build python environment with pip rather than conda now.

2.70.52 Version 2.4.0 (10/6/22)

- (rapidtide) Added enhanced variance removal assesment.
- (rapidtide) Fixed a rare crashing bug in proctiminglogfile.
- (rapidtide) Output some files indicating run status.
- (package) Fixed a deprecation warning in pearsonr.
- (Docker) Now build amd64 and arm64 containers.

2.70.53 Version 2.3.1 (9/27/22)

- (Dockerfile) Some tweaks to package versions to try to eliminate error messages.
- (Dockerfile) Add some AWS libraries to facilitate using S3 volumes.
- (Dockerfile) Moved timezone data loading earlier in the file to accomodate the new libraries.
- (reference) Added HCP_negmask_2mm to improve map display.
- (github) Updated codeql actions to v2.

2.70.54 Version 2.3.0 (9/23/22)

- (rapidtide) Fixed option setting for nirs mode and did some tests on real data.
- (Dockerfile) Rolled back the version of pyfftw in the container to avoid the annoying (and erroneous) warnings about the plan file.
- (package) Made some changes to the reference files and dispatcher to help with upcoming AWS deployment.

2.70.55 Version 2.2.9 (9/21/22)

- (showarbcorr) Now show the correlation function, fixed a typo
- (reference) Added slicetimes files for some large datasets

2.70.56 Version 2.2.8.1 (8/29/22)

- (package) Fixed versioneer installation.

2.70.57 Version 2.2.8 (8/29/22)

- (happy) Some under the hood tweaks to phase analysis to prep for multidimensional phase projection.
- (rapidtide) Exploring the use of complex PCA.
- (util) Added tcfrom2col.
- (rapidtide) Added explicit selection of linear and circular correlations.
- (package) Updated python environment for Docker.
- (package) Updated versioneer.
- (package) Changed some config files to try to fix documentation builds.

2.70.58 Version 2.2.7.1 (6/30/22)

- (Dockerfile) Updated to a consistent python environment.

2.70.59 Version 2.2.7 (6/29/22)

- (rapidtide) Fixed GLM noise removal in CIFTI files.
- (rapidtide) Initial support for linear rather than circular correlation.
- (happy) Fixed some pretty broken masking logic when you want to process more (rather than less) voxels than the brain.

2.70.60 Version 2.2.6 (5/17/22)

- (fingerprint) Various fixes to mask handling.
- (package) Staged some prep work on updating the setup/installation files.

2.70.61 Version 2.2.5 (4/26/22)

- (rapidtide) Postprocess timing information to make it more useful.
- (rapidtide) Reenabled numba by default.
- (fingerprint) Fixed handling of 4D atlases, empty regions, and 4D masks. Added "constant" template, and allow 0th order processing (mean).
- (atlastood) Fixed 4D atlas handling. Now mask atlas after collapsing to 3D.
- (histnifti) Added –transform flag to map values to percentiles.

2.70.62 Version 2.2.4 (4/11/22)

- (fingerprint) Now works properly for 3D input files.
- (tidepool) Turned the default level of verbosity way down, but gave you the ability to crank it back up.
- (RapidtideDataset.py) Fixed the default type of "numberofpasses".

2.70.63 Version 2.2.3 (4/1/22)

- (rapidtide) Added a new feature, –globalmeanselect, to try to locate a good, uniform, short delay pool of voxels to use for the initial global mean signal. This is an attempt to fix the "poison regressor" problem - if the initial regressor contains data from multiple, distinct pools of voxels with different delays, the initial global regressor is strongly autocorrelated, and delay fits become ambiguous. This cannot be corrected by refinement, so better to avoid it altogether. This option selects only voxels with clear, short delays, after a single pass with despeckling disabled. The result is a mask (XXXdesc-globalmeanpreselect_mask.nii.gz) that can be used with –globalmeanincludemask for a subsequent run.
- (rapidtide) Fixed a nasty bug that caused offsettime and lagminthresh to interact incorrectly, sometimes leading to almost no voxels for refinement.
- (happy) Moved some code around, changed some internal names, and added secret bits to support future, secret, features.
- (tidepool) Trying to add a little more clarity to the user about image orientation (the image's affine transform is correct, so the mapping between voxel and MNI coordinate is correct, but currently it's not clear if displayed images are radiological or neurological orientation.
- (fingerprint) Added the JHU atlases as options.
- (package) Added slightly modified version of the JHU arterial territorial atlases to the reference section (Paper: https://doi.org/10.1101/2021.05.03.442478, Download: https://www.nitrc.org/projects/arterialatlas).
- (Docker) Fixed a dependency problem for pyfftw (resolves https://github.com/bbfrederick/rapidtide/issues/79)
- (pony) One time offer, today only - every user gets a pony upon request!

2.70.64 Version 2.2.2 (3/16/22)

- (happy, happy_legacy, simdata) This release corrects a flaw (or perhaps more accurately an ambiguity) in slice time specification. In FSL slicetime files, slicetimes are specified in fractions of a TR. In .json sidecars, they are specified in seconds. This is now detected on file read, and slicetime files are now converted to seconds. Until now, happy and simdata assumed all slice times were in seconds. This will fix behavior when FSL-style (fractional TR) slicetime files are used. Behavior with .json sidecars is not changed. Non-json files are assumed to be the FSL style (fractions of a TR) UNLESS the –slicetimesareinseconds flag is used.

2.70.65 Version 2.2.1 (3/16/22)

- (rapidtide) Tweaked mask checking logic to address a bug introduced by despeckling changes.
- (histtc, histnifti) Harmonized options between the programs.
- (Docker) Updated Dockerfile to fix a bug that caused automatic build to fail.

2.70.66 Version 2.2.0 (3/11/22)

- (rapidtide) Major rethink of despeckling. Despeckling no longer causes negative correlation values when bipolar fitting is not enabled, and voxel parameters are only updated in a despeckled voxel if the correlation fit succeeds. This results in better fits without mysteriously unfit voxels.
- (showxy) Bland-Altman plots can now use alternative formatting, as per Krouwer, J. S. Why Bland–Altman plots should use X, not (Y+X)/2 when X is a reference method. Stat Med 27, 778–780 (2008).
- (fingerprint) This program is now substantially more useful, working on 4D input files. Output files are more convenient as well.
- (cleandirs) Cleandirs now keeps cleaning until it runs out of old installations to remove.

2.70.67 Version 2.1.2 (1/10/22)

- (calcicc, calctexticc) Some fixes to indexing.

2.70.68 Version 2.1.1 (11/4/21)

- (spatialmi, calcicc) Major improvements in performance, stability, and flexibility.
- (showtc) Added support for files with large star time offsets.
- (showxy) Some appearance tweaks.
- (niftidecomp) Improved mask generation.
- (variabilityizer) New progam to transform fMRI datasets to variability measures.

2.70.69 Version 2.1.0 (9/21/21)

- (spatialmi) Added new program to calculate local mutual information between 3D images.
- (calcicc) Tool to calculate ICC(3,1) - quickly - for a set of 3D images.
- (correlate.py) Fixed the reference for mutual_info_2d.
- (package) Simplified and cleaned up release process.

2.70.70 Version 2.0.9 (8/26/21)

- (rapidtide) Fixed a strange edge case that could lead to "hot pixels" in the maxcorr map.
- (io) Added a "tolerance" for spatial mapping of niftis to account for rounding errors in header creation.

2.70.71 Version 2.0.8 (8/20/21)

- (rapidtide) Disabled processing of abbreviated arguments.
- (showtc) Suppressed some unnecessary output when not in debug mode.
- (Docker) Added automatic build and push as a github action.

2.70.72 Version 2.0.7 (8/19/21)

- (reference) Include the new JHU digital arterial territory atlas.
- (Docker) Updated container to Python 3.9.
- (package) General cleanup of imports.

2.70.73 Version 2.0.6 (8/16/21)

- (package) Merged Taylor Salo's PR that fixes the documentation builds on readthedocs (THANK YOU!) and cleans up and centralizes requirement specifications.

2.70.74 Version 2.0.5 (8/9/21)

- (package) Further Windows compatibility fixes.
- (documentation) Updated USAGE.rst for the current naming and syntax of rapidtide.

2.70.75 Version 2.0.4 (7/28/21)

- (package) Fixed a problem where any program using util.py wouldn't run on Windows.
- (roisummarize) New addition to the package.
- (CI) Fixed a bug in the document environment.

2.70.76 Version 2.0.3 (7/16/21)

- (spatialdecomp, temporaldecomp) Improved the consistency between the programs.
- (showxcorrx) Fixed some command line options.
- (package) Began to clean up and unify text output formatting.
- (package) Addressed some numpy deprecation warnings.
- (all scripts) Corrected file permissions (this may matter on Windows installations).
- (docs) Fixed some typos.
- (showtc) Enhanced debugging output.
- (testing) Tweaked circleci configuration file to update environment prior to installation.

2.70.77 Version 2.0.2 (6/10/21)

- (rapidtide) Did you know that in python 3.8 and above, the default multiprocessing method is "spawn" rather than "fork"? Did you know the subtle differences? Do you know that that breaks rapidtide? I didn't, now I do, and now it doesn't.
- (rapidtide) Made some tweaks to the timing logger to improve output formatting.
- (rapidtide, happy) Tested on M1. The tests run more than twice as fast on an M1 mac mini with 8GB of RAM as on a 2017 MBP with a 2.9 GHz Quad-Core Intel Core i7. Emulated. Yow. When I get a native anaconda installation going, watch out.
- (happy, Docker) Now require tensorflow 2.4.0 or above to address a security issue.

2.70.78 Version 2.0.1 (6/8/21)

- (showxcorrx, plethquality, resamp1tc, simdata, happy, rapidtide) Cleaned up, improved, and unified text file reading and writing.
- (showxcorrx) Various functionality improvements.
- (package) Added options for timecourse normalization and zeropadding correlations.
- (documentation) Further cleanup.
- (Docker) Various fixes to versioning and other internals.

2.70.79 Version 2.0 (6/2/21)

Much thanks to Taylor Salo for his continuing contributions, with several substantive improvements to code, documentation, and automatic testing, and generally helping devise a sensible release roadmap that made this version possible.

This release is a big one - there are many new programs, new capabilities in existing programs, and workflow breaking syntax changes. However, this was all with the purpose of making a beter package with much more consistent interfaces that allow you to figure out pretty quickly how to get the programs to do exactly what you want. The biggest change is to rapidtide itself. For several years, there have been two versions of rapidtide; rapidtide2 (the traditional version), and rapidtide2x (the experimental version for testing new features). When features became stable, I migrated them back to rapidtide2, more and more quickly as time went by, so they became pretty much the same. I took the 2.0 release as an opportunity to do some cleanup. As of now, there is only one version of rapidtide, with two parsers. If you call "rapidtide", you get the spiffy new option parser and much more rational and consistent option naming and specification. This is a substantial, but simple, change. For compatibility with old workflows, I preserved the old parser, which is

called "rapidtide2x_legacy". This accepts options just as rapidtide2 and rapidtide2x did in version 1.9.6. There is only one rapidtide routine. Once the arguments are all read in and processed, "rapidtide" and "rapidtide2x_legacy" call the same processing workflow. However, in addition to the new parser, there are completely new options and capabilities in rapidtide, but you can only get to them using the new parser. This is my way of subtly forcing you to change your workflows if you want the new shiny, without pulling the rug out from under you. "rapidtide2x_legacy" WILL be going away though, so change over when you can. Also - all outputs now conform to BIDS naming conventions to improve compatibility with other packages. Use the "–legacyoutput" flag to get the old output file names.

- (rapidtide2x_legacy): Added deprecation warning.
- (rapidtide): The correlation function has been replaced by a more flexible "similarity function". There are currently 3 options: "correlation" (the old method), "mutualinfo", which uses a cross mutual information function, and "hybrid", which uses the correlation function, but disambiguates which peak to use by comparing the mutual information for each peak.
- (rapidtide) Pulled a number of default values into global variables so that defaults and help strings will stay in sync.
- (rapidtide) Fixed text file (nirs) processing.
- (rapidtide) Fixed a search range setting error.
- (rapidtide) Fixed the default method for global mean signal generation.
- (rapidtide) Added the '–negativegradient' option in response to https://github.com/bbfrederick/rapidtide/issues/67
- (rapidtide) Added flexibility to regressor input (can use multicolumn and BIDS text files).
- (rapidtide, tidepool) Fixed reading and writing the globalmean mask.
- (rapidtide) Gracefully handle refinement failure.
- (rapidtide) Added a new method for generating global signal using PCA.
- (rapidtide) Did some prep work to implement echo cancellation.
- (rapidtide) Added workaround for occasional MLE PCA component estimation failure (this seems to be an unresolved scikit-learn problem as of 0.23.2)
- (rapidtide) Significant enhancement to PCA refinement options.
- (rapidtide) Rapidtide can now run refinement passes until the change in the probe regressor falls below a specified mean square difference. Set –convergencethresh to a positive number to invoke this (0.0005 is good). Rapidtide will refine until the M.S.D. falls below this value, or you hit maxpasses (use –maxpasses NUM to set - default is 15). This implements the procedure used in Champagne, A. A., et al., NeuroImage 187, 154–165 (2019).
- (rapidtide) The PCA refinement algorithm has been improved to match the method described in Champagne, et al., and is now the default.
- (rapidtide, io) Significant improvement to CIFTI handling - now properly read and write parcellated scalars and time series.
- (rapidtide) Completely revamped CIFTI I/O. Should now read and write native CIFTI2 files (do not need to convert to NIFTI-2 in workbench).
- (rapidtide) Better handling of motion files.
- (rapidtide) Added coherence calculation. Not quite working right yet.
- (rapidtide, happy) Switched to using nilearn's mask generator for automatic mask generation, since it's much more sophisticated. It seems to be a big improvement, and handles data processed by fmriprep and SPM with no fiddling.
- (rapidtide, happy) General improvement of output of floating point numbers. Limit to 3 decimal places.

- (rapidtide) Use logging module for output.
- (rapidtide, rapidtide_legacy) Options file is now always saved as a json.
- (rapidtide) Added ability to autochoose an appropriate spatial filter by setting –spatialfilt to a negative value.
- (rapidtide, rapidtide2x_legacy) The options file is now always saved in .json format.
- (rapidtide) BIDS format output naming and file structures have been updated to be more compliant with the standard.
- (rapidtide) Fixed a longstanding bug which used an unnecessarily stringent amplitude threshold for selecting voxels to use for refinement.
- (rapidtide) Improvements to processing in "bipolar" mode.
- (rapidtide): The getopt argument parser has been completely rewritten using argparse. The way you specify many (most?) options has changed.
- (rapidtide): Any option that takes additional values (numbers, file names, etc.) is now specified as '–option VALUE [VALUE [VALUE...]]' rather than as '–option=VALUE[,VALUE[,VALUE...]]'.
- (rapidtide): After a lot of use over the years, I've reset a lot of defaults to reflect typical usage. You can still do any analysis you were doing before, but it may now require changes to scripts and workflows to get the old default behavior. For most cases you can get good analyses with a minimum set of command line options now.
- (rapidtide): There are two new macros, –denoise and –delaymapping, which will set defaults to good values for those use cases in subjects without vascular pathology. Any of the preset values for these macros can be overridden with command line options.
- (rapidtide, rapidtide2x_legacy): Regressor and data filtering has been changed significantly. While the nominal filter passbands are the same, the transitions to the stopbands have been tightened up quite a bit. This is most noticable in the LFO band. The pasband is still from 0.01-0.15Hz with a trapezoidal rolloff, but the upper stopband now starts at 0.1575Hz instead of 0.20Hz. The wide transition band was letting in a significant amount of respiratory signal for subjects with low respiratory rates (about half of my subjects seem to breath slower than the nominal adult minimum rate of 12 breaths/minute).
- (rapidtide): The -V, -L, -R and -C filter band specifiers have been retired. Filter bands are now specified with '–filterband XXX', where XXX is vlf, lfo, lfo_legacy, resp, cardiac, or None. 'lfo' is selected by default (LFO band with sharp transition bands). To skip filtering, use '–filterband None'. '–filterband lfo_legacy' will filter to the LFO band with the old, wide transition bands.
- (rapidtide): To specify an arbitrary filter, specify the pass freqs with –filterfreqs, and then optionally the stop freqs with –filterstopfreqs (otherwise the stop freqs will be calculated automatically from the pass freqs).
- (rapidtide): The method for specifying the lag search range has changed. '-r LAGMIN,LAGMAX' has been removed. You now use '–searchrange LAGMIN LAGMAX'
- (rapidtide): The method for specifying bipolar correlation search has changed. '-B' is replaced by '–bipolar'.
- (rapidtide): The method for specifying a fixed delay (no correlation lag search) has changed. '-Z DELAYVAL' is replaced by '–fixdelay DELAYVAL'.
- (rapidtide,rapidtide2x_legacy): The 'timerange' option is now handled properly. This can be used to restrict processing to a portion of the datafile. This is useful to get past initial transients if you didn't remove them in preprocessing, or to see if parameters change over the course of a long acquisition.
- (rapidtide): The multiprocessing code path can be forced on, even on a single processor.
- (rapidtide): Multiprocessing can be disabled on a per-routine basis.

Happy also got a new parser and BIDS outputs. You can call happy with the old interface by calling "happy_legacy".

- (happy) Output files now follow BIDS naming convention.

- (happy) Code cleanup, improved tensorflow selection.
- (happy) Fixed logmem calls to work with new logging structure (and not crash immediately).
- (happy) Fixed a very subtle bug when an externally supplied pleth waveform doesn't start at time 0.0 (fix to issue #59).
- (happy) General formatting improvements.
- (happy) Added new tools for slice time generation.
- (happy) Added support for scans where there is circulating contrast.

General Changes to the entire package:

- (package) Python 2.7 support is now officially ended. Cleaned out compatiblity code.
- (package) Dropped support for python 3.3-3.5 and added 3.9.
- (package) Made pyfftw and numba requirements.
- (package) Significantly increased test coverage by including smoke tests (exercise as many code paths as possible to find crashers in neglected code - this is how the above bugs were found).
- (package) Automated consistent formatting. black now runs automatically on file updates.
- (package) General cleanup and rationalization of imports. isort now runs automatically on file updates.
- (package) Fixed a stupid bug that surfaced when reading in all columns of a text file as input.
- (package) Merged tsalo's PR starting transition to new logging output.
- (package) Started to phase out sys.exit() calls in favor of raising exceptions.
- (package) Updated all headers and copyright lines.
- (package) Trimmed the size of the installation bundle to allow deployment on pypi.
- (package) Copied Taylor Salo's improvements to build and deployment from the master branch.
- (package) Renamed some test data for consistency.
- (package) Began effort with T. Salo to address linter errors and generally improve PEP8 conformance - remove dead code, rationalize imports, improve docstrings, convert class names to CamelCase, use snake_case for functions.
- (package) Cleaned up imports and unresolved references
- (package) Addressed many linter issues, updated deprecated numpy and scipy calls.
- (package) readvectorsfromtextfile now handles noncompliant BIDS timecourse files.
- (package) Implemented new, generalized text/tsv/bids text file reader with column selection (readvectorsfromtextfile).
- (package) Significant internal changes to noncausalfilter.
- (package) CircleCI config files changed to keep tests from stepping on each other's caches (thanks to Taylor Salo).

Changes to the Docker container builds:

- (Docker) Fixed some problems with task dispatch and container versioning.
- (Docker) Improvements to version specification, entry point.
- (Docker) Update package versions.
- (Docker) Install python with mamba for ~10x speedup.

- (Docker) Switched to current method for specifying external mounts in the documentation.
- (Docker) Improved build scripts.
- (Docker) Containers are now automatically pushed to dockerhub after build.

Documentation changes:

- (documentation) Fixed errors in documentation files that caused errors building in readthedocs.
- (documentation) New "theory of operation" section for rapidtide. Still working on it.
- (documentation) The documentation has been expanded and revised to reflect the current state of rapidtide with significant reorganization, reformatting, cleanup and additions

Tidepool:

- (tidepool) Reorganized interface, ability to show dynamic correlation movies, much clearer histogram graphs.
- (tidepool) Tidepool now gracefully handles runs with more than 4 passes. The timecourses displayed are prefilt, postfilt, pass1, pass2, pass(N-1) and pass(N).
- (tidepool) Fixed to work with Big Sur (macOS 11).
- (tidepool) Corrected sample rate for regressor timecourses.
- (tidepool) Revised to properly handle new BIDS naming conventions for output files.
- (tidepool) You can now turn out-of-range map transparency on and off (it's off by default).
- (tidepool) Internal colortable code is now much cleaner.
- (tidepool): Now properly handles missing timecourses properly. Some cosmetic fixes.

Miscellaneous changes:

- (aligntcs, applydlfilter, pixelcomp, plethquality, resamp1tc, showstxcorr, showxcorrx) Fixed matplotlib backend initialization to allow headless operation.
- (glmfilt, linfit, temporaldecomp, spatialdecomp): Argument parsers were rewritten in argparse, main routines were moved into workflows.
- (applydlfilter, atlasaverage, ccorrica, filttc, happy2std, histnifti, histtc, pixelcomp, plethquality, rapidtide2std, resamp1tc, showarbcorr, showtc, showxcorrx, simdata, spatialfit) Argument parsers were rewritten in argparse.
- (rapidtide, showxcorrx, showarbcorr, showstxcorr, ccorrica, aligntcs, filttc) Changed argument handling for arbitrary filters. Specify pass freqs with –filterfreqs, stop freqs with –filterstopfreqs.
- (happy, rapidtide) Now properly handle negative mklthreads specification.
- (physiofreq): New program to get the average frequency of a physiological waveform.
- (showtc): Some cleanup in option specification.
- (showtc) Converted text inputs to standardized code.
- (atlastool) Major fixes to functionality with 4D template files.
- (atlastool) Fixed some import and syntax issues with numpy.
- (showxcorrx) Significant cleanup for maximum flexibility and utility.
- (showxcorr) Renamed to showxcorr_legacy
- (linfit) Renamed to polyfitim to better reflect it's function.
- (histnifti) Major upgrade to functionality.
- (showarbcorr) New program to do crosscorrelations on timecourses with different samplerates.

- (polyfitim) New program to fit a spatial template to a 3D or 4D NIFTI file.
- (endtidalproc) New program to extract end tidal CO2 or O2 waveforms from a gas exhalation recording.
- (dlfilter) Made diagnostics more informative to help get dlfilter enabled.
- (simdata) Complete overhaul - new parser better checks, more flexible input formats.
- (io.py) Vastly improved reading in arbitrarily large text files.
- (stats.py) Fixed a bug in getfracvals when you try to find the maximum value.
- (RapidtideDataset.py) Better handling of different file names.

2.70.80 Version 2.0alpha29 (6/1/21)

- (Docker) Fixed some problems with task dispatch and container versioning.
- (rapidtide) Pulled a number of default values into global variables so that defaults and help strings will stay in sync.
- (documentation) Reorganization and significant updates.
- (documentation) Fixed errors in documentation files that caused errors building in readthedocs.
- (package) Updated versioneer.

2.70.81 Version 1.9.6 (6/1/21)

- (Docker) ACTUALLY fixed some problems with task dispatch and container versioning.
- (package) Updated versioneer.

2.70.82 Version 2.0alpha28 (5/27/21)

- (testing) Synced function calls with some internal changes to Correlator to fix the tests crashing.

2.70.83 Version 1.9.5 (5/27/21)

- (Docker) Fixed some problems with task dispatch and container versioning.

2.70.84 Version 2.0alpha27 (5/27/21)

- (rapidtide, showxcorrx, showarbcorr, showstxcorr, ccorrica, aligntcs, filttc) Changed argument handling for arbitrary filters. Specify pass freqs with –filterfreqs, stop freqs with –filterstopfreqs.
- (happy) Code cleanup, improved tensorflow selection.
- (Docker) Improvements to version specification, entry point.
- (testing) Increased coverage.
- (packaging) Multiple corrections in support files.

2.70.85 Version 2.0alpha26 (5/5/21)

- (happy, rapidtide) Now properly handle negative mklthreads specification.
- (Dockerfile) Update package versions.
- (Docker container) Added happy test.
- (package) Further increased test coverage.

2.70.86 Version 2.0alpha25 (5/3/21)

- (rapidtide) Fixed text file (nirs) processing.
- (rapidtide) Fixed a search range setting error.
- (rapidtide) Fixed the default method for global mean signal generation.
- (rapidtide) Fixed a crash when using the mutualinfo similarity metric.
- (rapidtide, io) Significant improvement to CIFTI handling - now properly read and write parcellated scalars and time series.
- (io) Vastly improved reading in arbitrarily large text files.
- (stats) Fixed a bug in getfracvals when you try to find the maximum value.
- (package) Began aggressive implementation of smoke tests (excercise as many code paths as possible to find crashers in neglected code - this is how the above bugs were found).
- (package) More logging refinement.

2.70.87 Version 2.0alpha24 (4/14/21)

- (rapidtide) Added the '–negativegradient' option in response to https://github.com/bbfrederick/rapidtide/issues/67
- (rapidtide) Added flexibility to regressor input (can use multicolumn and BIDS text files).

2.70.88 Version 2.0alpha23 (4/14/21)

- (happy) Fixed logmem calls to work with new logging structure (and not crash immediately).

2.70.89 Version 2.0alpha22 (4/13/21)

- (rapidtide, tidepool) Fixed reading and writing the globalmean mask.
- (package) Fixed a stupid bug that surfaced when reading in all columns of a text file as input (really, this time).

2.70.90 Version 2.0alpha21 (4/12/21)

- (rapidtide) Gracefully handle refinement failure.
- (happy) Fixed some output timecourse naming.
- (atlastool) Major fixes to functionality with 4D template files.
- (aligntcs, applydlfilter, pixelcomp, plethquality, resamp1tc, showstxcorr, showxcorrx) Fixed matplotlib backend initialization to allow headless operation.
- (package) General cleanup and rationalization of imports. isort now used by default.
- (package) Dropped support for python 3.3-3.5
- (package) Fixed a stupid bug that surfaced when reading in all columns of a text file as input.
- (package) Merged tsalo's PR starting transition to new logging output.
- (package) Fixed environment for py39 testing.
- (package) Started to phase out sys.exit() calls in favor of raising exceptions.
- (rapidtide) Corrected BIDS naming of intermediate maps.

2.70.91 Version 2.0alpha20 (3/28/21)

- (package) Python 2.7 support is now officially ended. Cleaned out compatiblity code.
- (package) Made pyfftw and numba requirements.
- (docs) Wrote general description of text input functions, enhanced description of happy, include examples.
- (style) Began effort with T. Salo to address linter errors and generally improve PEP8 conformance - remove dead code, rationalize imports, improve docstrings, convert class names to CamelCase, use snake_case for functions.
- (showtc) Converted text inputs to standardized code.

2.70.92 Version 2.0alpha19 (3/26/21)

- (showxcorrx) Significant cleanup for maximum flexibility and utility.
- (showxcorr) Renamed to showxcorr_legacy
- (linfit) Renamed to polyfitim to better reflect it's function.
- (histnifti) Major upgrade to functionality.
- (showxcorrx, atlasaverage, happy2std, rapidtide2std, spatialfit, applydlfilter, plethquality, histnifti) Moved to argparse.
- (package) Updated all headers and copyright lines.

2.70.93 Version 2.0alpha18 (3/17/21)

- (package) Trimmed the size of the installation bundle (really, truly, correctly this time).

2.70.94 Version 1.9.4 (3/17/21)

- (package) T. Salo made a number of changes to allow pypi deployment.
- (package) Fixed a problem in setup.py that causes installation to fail.
- (rapidtide2x) Backported a critical fix from the dev version so that the refinement threshold is properly set with null correlations (the result is that the refine mask rejects fewer voxels, and gives a better regressor estimate).

2.70.95 Version 2.0alpha17 (3/17/21)

- (package) Trimmed the size of the installation bundle (maybe even correctly this time).

2.70.96 Version 2.0alpha16 (3/17/21)

- (package) Trimmed the size of the installation bundle.
- (various) Cleaned up imports and unresolved references

2.70.97 Version 2.0alpha15 (3/17/21)

- (rapidtide) Added a new method for generating global signal using PCA.
- (all) Further imports from master branch to improve deployment.
- (simdata) Complete overhaul - new parser better checks, more flexible input formats.
- (io.py) Improvements to readvecs to handle very large files.
- (ccorrica, filttc, histtc, pixelcomp, resamp1tc) Facelift, cleanup, new parsers.
- (testing) Removed python 2.7 CI build.
- (all) Addressed many linter issues, updated deprecated numpy and scipy calls.

2.70.98 Version 2.0alpha14 (3/1/21)

- (all) readvectorsfromtextfile now handles noncompliant BIDS timecourse files.
- (happy) Fixed a very subtle bug when an externally supplied pleth waveform doesn't start at time 0.0 (fix to issue #59).
- (filttc, histtc, showarbcorr) parser improvements.

2.70.99 Version 2.0alpha13 (2/22/21)

- (package) Copied Taylor Salo's improvements to build and deployment from the master branch.
- (all) Ran all python files through Black to give consistent formatting (really, truly this time).
- (all) Implemented new, generalized text/tsv/bids text file reader with column selection (readvectorsfromtextfile).
- (atlastool) Fixed some import and syntax issues with numpy.
- (showarbcorr) New program to do crosscorrelations on timecourses with different samplerates.
- (happy) Fixed column selection bug with BIDS files.
- (happy) General formatting improvements.
- (dlfilter) Made diagnostics more informative to help get dlfilter enabled.

2.70.100 Version 2.0alpha12 (2/5/21)

- (all) Fixed readbidstsv calls.
- (all) Beginning a rethink of a universal text timecourse reader.
- (happy) Added new tools for slice time generation.

2.70.101 Version 2.0alpha11 (1/5/21)

- (rapidtide) Rolled back default similarity metric to 'correlation' from 'hybrid'. 'hybrid' works very well most of the time, and fails strangely occasionally. When 'correlation' fails, it does so in more predictable and explicable ways.
- (happy) Restored functionality and options for motion regression that I broke when separating out the command parser.
- (tests) CircleCI config files changed to keep tests from stepping on each other's caches (thanks to Taylor Salo).

2.70.102 Version 2.0alpha10 (12/21/20)

- (package) Ran all python files through Black to give consistent formatting.
- (rapidtide) Did some prep work to implement echo cancellation.

2.70.103 Version 2.0alpha9 (12/9/20)

- (rapidtide) Added workaround for occasional MLE PCA component estimation failure (this seems to be an unresolved scikit-learn problem as of 0.23.2)

2.70.104 Version 2.0alpha8 (12/9/20)

- (rapidtide) Significant enhancement to PCA refinement options.
- (tidepool) Tidepool now gracefully handles runs with more than 4 passes. The timecourses displayed are prefilt, postfilt, pass1, pass2, pass(N-1) and pass(N).
- (happy) Added support for scans where there is circulating contrast.
- (happy, rapidtide2x, rapidtide) The parsers are now being properly installed during setup.
- (package) Renamed some test data for consistency.

2.70.105 Version 2.0alpha7 (12/1/20)

- (rapidtide) Rapidtide can now run refinement passes until the change in the probe regressor falls below a specified mean square difference. Set –convergencethresh to a positive number to invoke this (0.0005 is good). Rapidtide will refine until the M.S.D. falls below this value, or you hit maxpasses (use –maxpasses NUM to set - default is 15). This implements the procedure used in Champagne, A. A., et al., NeuroImage 187, 154–165 (2019).
- (rapidtide) The PCA refinement algorithm has been improved to match the method described in Champagne, et al., and is now the default.

2.70.106 Version 2.0alpha6 (11/30/20)

- (rapidtide) Completely revamped CIFTI I/O. Should now read and write native CIFTI2 files (do not need to convert to NIFTI-2 in workbench).
- (rapidtide) Better handling of motion files.
- (rapidtide) Added coherence calculation. Not quite working right yet.
- (happy) Started adding BIDS output.
- (tidepool) Fixed to work with Big Sur (macOS 11).

2.70.107 Version 2.0alpha5 (10/29/20)

Much thanks to Taylor Salo for his continuing contributions, with several substantive improvements to code, documentation, and automatic testing, and generally helping devise a sensible release roadmap.

- (rapidtide, happy) Switched to using nilearn's mask generator for automatic mask generation, since it's much more sophisticated. It seems to be a big improvement, and handles data processed by fmriprep and SPM with no fiddling.
- (rapidtide, happy) General improvement of output of floating point numbers. Limit to 3 decimal places.
- (rapidtide) Use logging module for output.
- (rapidtide, rapidtide_legacy) Options file is now always saved as a json.
- (rapidtide) Added ability to autochoose an appropriate spatial filter by setting –spatialfilt to a negative value.
- (rapidtide_parser) Code cleanup and formatting fixes.
- (documentation) Much reorganization, reformatting and cleanup.
- (documentation) New "theory of operation" section for rapidtide. Still working on it.

2.70.108 Version 2.0alpha4 (10/26/20)

- rapidtide2x has been renamed to rapidtide2x_legacy
- (rapidtide, rapidtide2x) The options file is now always saved in .json format.
- (rapidtide) BIDS format output naming and file structures have been updated to be more compliant with the standard.
- (RapidtideDataset.py) Better handling of different file names.
- (documentation) The documentation has been expanded and revised to reflect the current state of rapidtide.
- (all) Code cleanup.

2.70.109 Version 2.0alpha3 (10/19/20)

- (rapidtide) Fixed sample rate on BIDS regressor outputs.
- (tidepool) Corrected sample rate for regressor timecourses.
- (Docker) Switched to current method for specifying external mounts in the documentation.
- (tests) Fixed test_filter.py to remove bad test.
- (tests) Added test_delayestimation.py to try to get end to end validation on the core of rapidtide.

2.70.110 Version 2.0alpha2 (10/19/20)

- (all) Significant internal changes to noncausalfilter.
- (rapidtide) Fixed a longstanding bug which used an unnecessarily stringent amplitude threshold for selecting voxels to use for refinement.
- (rapidtide) Improvements to processing in "bipolar" mode.
- (rapidtide) Internal improvements to mutual information normalization.
- (rapidtide) New –bidsoutput option to make all output files BIDS compatible in naming and format.
- (tidepool) Revised to properly handle new naming conventions for output files.
- (tidepool) You can now turn out-of-range map transparency on and off (it's off by default).
- (tidepool) Internal colortable code is now much cleaner.
- (Docker) Improved build scripts.

2.70.111 Version 2.0alpha1 (8/24/20)

- (all): Python 2.x is no longer supported. To be fair, I've done nothing to break 2.x compatibility on purpose, so it probably still works, but I'm expending no effort to keep it working.
- (documentation): General updates and cleanups.
- (rapidtide2): rapidtide2 has been eliminated. If you used it before, you can use rapidtide2x as a dropin replacement (but you really should start moving to using rapidtide, the new version that is actively being developed).
- (rapidtide2x): rapidtide2x has been deprecated and replaced by rapidtide (which is basically rapidtide2x v1.9.3 with a different argument parser and default option values).
- (rapidtide2x): Added deprecation warning.

- (rapidtide): The correlation function has been replaced by a more flexible "similarity function". There are currently 3 options: "correlation" (the old method), "mutualinfo", which uses a cross mutual information function, and "hybrid", the new default, which uses the correlation function, but disambiguates which peak to use by comparing the mutual information for each peak.
- (rapidtide): Changed the default peak fit type to "fastquad", which does a parabolic fit to the peaks to refine location.
- (rapidtide): The getopt argument parser has been completely rewritten using argparse. The way you specify many (most?) options has changed.
- (rapidtide): Any option that takes additional values (numbers, file names, etc.) is now specified as '–option VALUE [VALUE [VALUE...]]' rather than as '–option=VALUE[,VALUE[,VALUE...]]'.
- (rapidtide): After a lot of use over the years, I've reset a lot of defaults to reflect typical usage. You can still do any analysis you were doing before, but it may now require changes to scripts and workflows to get the old default behavior. For most cases you can get good analyses with a minimum set of command line options now.
- (rapidtide): There are two new macros, –denoise and –delaymapping, which will set defaults to good values for those use cases in subjects without vascular pathology. Any of the preset values for these macros can be overridden with command line options.
- (rapidtide, rapidtide2x): Regressor and data filtering has been changed significantly. While the nominal filter passbands are the same, the transitions to the stopbands have been tightened up quite a bit. This is most noticable in the LFO band. The pasband is still from 0.01-0.15Hz with a trapezoidal rolloff, but the upper stopband now starts at 0.1575Hz instead of 0.20Hz. The wide transition band was letting in a significant amount of respiratory signal for subjects with low respiratory rates (about half of my subjects seem to breath slower than the nominal adult minimum rate of 12 breaths/minute).
- (rapidtide): The -V, -L, -R and -C filter band specifiers have been retired. Filter bands are now specified with '–filterband XXX', where XXX is vlf, lfo, lfo_legacy, resp, cardiac, or None. 'lfo' is selected by default (LFO band with sharp transition bands). To skip filtering, use '–filterband None'. '–filterband lfo_legacy' will filter to the LFO band with the old, wide transition bands.
- (rapidtide): To specify an arbitrary filter, use '–filterfreqs LOWERPASS UPPERPASS [LOWERSTOP UPPERSTOP]'. If you don't specify the stop bands, the stop frequencies are set to 5% below and above LOWERPASS and UPPERPASS, respectively.
- (rapidtide): The method for specifying the lag search range has changed. '-r LAGMIN,LAGMAX' has been removed. You now use '–searchrange LAGMIN LAGMAX'
- (rapidtide): The method for specifying bipolar correlation search has changed. '-B' is replaced by '–bipolar'.
- (rapidtide): The method for specifying a fixed delay (no correlation lag search) has changed. '-Z DELAYVAL' is replaced by '–fixdelay DELAYVAL'.
- (rapidtide): Options file is saved in json by default now.
- (rapidtide,rapidtide2x): The 'timerange' option is now handled properly. This can be used to restrict processing to a portion of the datafile. This is useful to get past initial transients if you didn't remove them in preprocessing, or to see if parameters change over the course of a long acquisition.
- (physiofreq): New program to get the average frequency of a physiological waveform.
- (tidepool): Now properly handles missing timecourses properly. Some cosmetic fixes.
- (showtc): Converted to argparse, some cleanup in option specification.
- (glmfilt, linfit, temporaldecomp, spatialdecomp): Argument parsers were rewritten, main routines were moved into workflows.
- (docker container): Fixed some build errors, now pushes container to dockerhub.

- (rapidtide): Multiprocessing can be forced on, even on a single processor.
- (rapidtide): Multiprocessing can be disabled on a per-routine basis.

2.70.112 Version 1.9.3 (7/30/20)

- Bumped version number because I forgot to commit a file

2.70.113 Version 1.9.2 (7/30/20)

- (all): Changed over to using versioneer to handle version numbers.
- (rapidtide2, rapidtide2x, rapidtide_2x_trans, rapidtideX) Runtimings file now has additional version information.

2.70.114 Version 1.9.1 (6/17/20)

- (all): Documentation improvements.
- (all): Many internal changes to support future argument specifications.
- (all): Backported bugfixes from the development version.
- (rapidtide2x) Fixed specification of timerange.
- (docker): Fixed an incompatibility in versions between pyfftw and scipy (thank you to Niranjana Shashikumar for reporting the bug and providing the solution!)
- (docker): Improved container labelling.
- (docker): Cleaned up container build.
- (tidepool): Various fixes and improvements backported from the development version.

2.70.115 Version 1.9 (1/6/20)

- (all): Now compatible with nibabel 3.x
- (all): Significantly expanded test suite. Code coverage is now at 47%.
- (documentation): Added instructions for installing the deep learning filter code
- (documentation): Numerous tweaks and fixes
- (docker): There is now a containerized version of the rapidtide package, which avoids a lot of installation woes
- (rapidtide2x, showxcorrx): Completely replaced correlation and fitting routines with faster, more robust (and more rigorously tested) versions
- (rapidtide2x, showxcorrx): Enhancements to the permutation methods
- (rapidtide2x, showxcorrx): Revised internals to guarantee xcorr scale matches values
- (rapidtide2x, showxcorrx): Improved fitter performance in edge cases (thin peaks, symmetric around max)
- (rapidtide2x, showxcorrx): Changed limits to avoid crash when peak is at edge of range
- (rapidtide2x, showxcorrx): Fixed some (but apparantly not all) dumb errors in calls to null correlation calculations.
- (rapidtide2x): Implemented workaround for unknown crasher in GLM filtering when nprocs != 1

- (rapidtide2x): Added experimental respiration processing
- (rapidtide2x): Fixed an uncaught bug in bipolar processing.
- (rapidtide2x): Setting ampthresh to a negative number between 0 and 1 sets the percentile of voxels to use for refinement
- (rapidtide2x): Support for new minimum sigma limit in correlation fit
- (rapidtide2x): Fixed a bug that caused fit fails on very narrow peaks, added diagnostic info
- (rapidtide2x): Putting in scaffolding to support phase randomization for null correlation calculation.
- (rapidtide2x): Properly specify correlation limits in null correlation and accheck
- (rapidtide2x): Slight modifications to pickleft routine to avoid a rare bug
- (rapidtide2x): Masks should now be properly generated for zero mean and non-positive definite data.
- (rapidtide2x): Tweaked the autocorrelation correction
- (rapidtide2x): Added an error check to avoid crashing when no significance tests are nonzero
- (rapidtide2x): Added upper and lower sigma limits to peak fitting uss to match new class
- (showxcorrx): Updated to match rapidtide2x fitting
- (showxcorrx): Enhanced error reporting
- (showxcorrx): Added width range tests
- (showxcorrx): Added norefine option, output more debugging info
- (showxcorrx): Set validity limits for gaussian fit
- (happy, rapidtide2x): Fixed a bug in output nifti header parameters (copy headers by value, not reference!)
- (happy): New multipass architecture allows much better results - initial passes set estimation masks and find potential arterial voxels.
- (happy): Aliased correlation integrated into happy as experimental feature
- (happy): Fixed a conceptual error in how I normalized projected timecourses
- (happy): Added brain cine output
- (happy): If estmask is supplied, us it. If not, generate a vessel mask and repeat final steps.
- (happy): Added option to detect and invert arterial signals, improved time output.
- (happy): Shorten pulse recon step size to 10 ms
- (happy): Better envelope handling, fixed bug in timecourse phase projection.
- (happy): Some changes to improve performance with long TRs
- (happy): Added happy paper reference
- (happy): Increased gridbins (used in phase projection): default to 2 after testing showed lower noise than 1.5 bins
- (happy): Added ability to pad cyclically rather than reflecting around the ends
- (happy): Added ability to smooth projection in the phase direction (on by default)
- (happy): Significant improvements to GLM processing (spatial and temporal versions, aliased temporal correlation)
- (happy): Added "checkpoint" option to dump more intermediate data for debugging.

- (happy): Added more progress bars, and the ability to turn them off.
- (happy): Print out version info at runtime.
- (tidepool): Major update with new functionality
- (tidepool): The probe regressor, it's spectrum, and how it was filtered are now shown in the main window
- (tidepool): Properly disable atlas buttons when no atlas is loaded, avoiding crashes
- (tidepool): Removed support for pyqt4
- (tidepool): Some UI tweaks
- (tidepool): Added some infrastructure for future support for loading multiple runs
- (tidepool): New atlases to suport fmriprep default coordinates
- (tidepool): Numerous bug fixes
- (ccorrica): Added the ability to oversample the data prior to crosscorrelation
- (showtc): Added ability to select a column from a multicolumn file as input.
- (showtc): Can now select a column from multicolumn input text files for each vector.
- (showtc): Changed the specification of colors and legends. Internal code cleanup.
- (happy2std): New tool to align happy maps
- (happywarp): Improvements in filtering
- (aligntcs): You can now specify single columns out of multicolumn files
- (showxy): Initial support for specifying color names
- (spectrogram): Cleanup, added some configuration options
- (simdata): Some reformatting, updates, and improvements
- (simdata): Put some data in the example directory to use with simdata
- (fingerprint): New addition to the library to decompose delay maps using vascular territory templates
- (fingerprint): Added canonical HCP template maps to the distribution
- (helper_classes): Added freqtrack class
- (correlate.py): Added rudimentary mutual information calculation
- (correlate.py): Multiple aliased correlation methods added, depending on demeaning.
- (io.py): Fixed support for named columns in BIDS tsvs
- (io.py): Relaxed requirements for required fields in BIDS jsons
- (io.py): Added a few convenience routines for dealing with NIFTI files
- (io.py): Fixed import of parser_funcs

2.70.116 Version 1.8 (5/10/19)

- (fit.py): The following fixes to both variants of findmaxlag_gauss affect rapidtide2, rapidtide2x, showxcorr, showxcorrx, happy, and happyx.
- (fit.py): CRITICAL FIX - edge behavior in both versions of findmaxlag_gauss was very broken.
- (fit.py): Fixed a rare failure in findmaxlag_gauss when the correlation peak is very narrow.
- (fit.py): Constrain initial gaussian fit values to be rational.
- (fit.py): Always return rational (if wrong) values when zerooutbadfit is False.
- (fit.py): Fixed a rare problem where no peaks were found in autocorrcheck, causing crash.
- (fit.py): Fixed a pernicious bug that sometimes caused mayhem when –nofitfilt was set.
- (rapidtide2, rapidtide2x): There is now sanity checking on lagmin and lagmax input using -r.
- (rapidtide2x): Masking logic has been completely redone, with numerous bugfixes, error checks, and new capabilities.
- (rapidtide2x): Added option to refine offset on leftmost lag peak (–pickleft). This helps a lot with people with vascular pathology.
- (rapidtide2x): Added ability to save options file in json.
- (rapidtide2x): Fixed a bug when timerange was used in conjunction with glm filtering.
- (rapidtide2x): Added fixes to crash where there were bad significance estimates.
- (showtc): Allow multiple linewidths.
- (showtc): Added ability to set x and y axis labels.
- (showtc): Added DPI option to set resolution of file output.
- (showxy): Changed Bland-Altman plot to use open circles.
- (showhist): Add bar histograms.
- (showhist): Added the option to set binsize to makehistogram.
- (happy, happyx): All changes in happyx have now been synced to happy - at this moment, they are identical. New changes will be tested in happyx.
- (happy, happyx): Fixed starttime and endtime selection.
- (happy, happyx): Lowered maximum heart rate to 140 by default.
- (happy, happyx): Output of info as json is optional, not default.
- (happy, happyx): Save info file as json rather than text.
- (happy, happyx): writedictasjson now supports numpy objects, added readdict function.
- (happy, happyx): Cardiac filter and search range are now specified independently.
- (happy, happyx): Removed lowerpass from cardiac estimation.
- (happy, happyx): Retrained and optimized model after revising happy paper.
- (happy, happyx): Use explicit copies to avoid variable changing out from under me.
- (happy, happyx): Added pleth/filtpleth correlation.
- (happy, happyx): Turn off variance masking by default, correlate raw and filtered waveforms.
- (happy, happyx): Numerous tweaks to resampling to perform better in edge cases.

- (happy, happyx): Fixed problem reading json physio files.
- (happy, happyx): Added ability to force the use of raw cardiac waveform for phase projection.
- (happy, happyx): Allow varmasking by volume or slice, filter prior to cardiac correlation.
- (happy, happyx): Resolved problem with definition of notch filter width.
- (happy, happyx): Corrected a type coercion error for estimation masks.
- (happy, happyx): Reduced verbosity of notch filter.
- (happy, happyx): Altered estimation mask logic.
- (happy, happyx): Added sanity checks to lag range.
- (happy, happyx): Now properly handle uncompressed bids tsv files.
- (happy, happyx): Variance and projection masks are separate, can set variance thresh percent.
- (happy, happyx): Changes in response to paper review.
- (happy, happyx): Filter cardiac waveform to slice samplerate Nyquist frequency when upsampling.
- (happy, happyx): Also added choice of centric or noncentric phase reconstruction..
- (happy, happyx): Fixed implementation of Hilbert transform phase analysis.
- (happy, happyx): Made robust to missing anatomics, started adding ants support.
- (happy, happyx): harmonic notch filter notch pct was not properly scaled.
- (happy, happyx): Now align pleth regressor with cardfromfmri.
- (happy, happyx): Fixed convolution gridding.
- (happy, happyx): Changed default gridding kernel to 3.0 wide Kaiser-Bessel.
- (happy, happyx): Reordered usage to functionally separate flags.
- (happy, happyx): Implemented workaround for strange interaction of tensorflow and MKL.
- (happy, happyx): Lots of little bugs fixed, print statements cleaned up.
- (tidepool): Added support for files in MNI152NLin2009cAsym space (fmriprep output).
- (tidepool): Fixed a crash when no atlas exists.
- (ccorrica): Modernized ccorrica to use new library calls.
- (atlasaverage, filttc, histtc, aligntcs, highresmotion): Added to the distro.
- (tests): Numerous maintenance fixes. test_findmaxlag is much more sophisticated now.
- (whole project): Code cleanup, reformatting.

2.70.117 Version 1.7 (12/5/18)

- (whole project) Stopped pretending happy doesn't exist - adding to the changelog and will start writing docs.
- (whole project) Tried to generate some workflows.
- (whole project) Update issue templates.
- (whole project) Put back some critical information that got lost in the docs reorganization.
- (happyx) Changed default minhr to 40 BPM, cleaned up specification of min/max HR.
- (happyx) Put a lower limit on the envelope function in cleancardiac to limit gain..

- (happyx) Can use weighted masks, calculate envelop normalized cardiac waveforms..
- (happyx) Fixed notch filter to always filter at least one frequency bin.
- (happyx) Added ability to skip trs in fmrifile and motion file.
- (happyx) Mad normalize slice timecourses, refactor code, add some test data.
- (happy, happyx) Moved some routines out of happy(x) into libraries, added trendfilter.
- (happy, happyx, rapidtide2, rapidtide2x) Added motion regressor filtering.
- (happyx, rapidtide2, rapidtide2x) Add high order polynomial detrending.
- (happyx) Added deep learning filter for refining cardiac waveform (off by default).
- (rapidtide2, rapidtide2x) Oversample factor was erroneously set to 0 if TR <=0.5 seconds.
- (showxcorrx) Added file output capability.
- (showxcorrx) Set verbose to False by default.
- (showxcorrx) Trimming extraneous output.
- (tidepool) First crack at fixing atlas averaging.
- (tidepool) Initialize atlasniftiname.
- (showxy) Added Bland-Altman plots with annotations, range specifications, font scaling.
- (showxy) Updated for new matplotlib interface, enabled legends.
- (showtc) Now can specify legend location.
- (showtc) Added fontscalefac option.
- (resample.py) Fixed cutoff frequency on upsample filter.
- (resample.py) Lowpass filter after upsampling.
- (fit.py) Limit peakstart and peakend to stay within legal range.
- (io.py) New additions to readvecs to specify columns.
- (dlfilter.py) added.

2.70.118 Version 1.6 (9/19/18)

- (whole project) Cleanup and reorganization (tsalo).
- (documentation) Major revisions to clean things up (tsalo).
- (workflows) Initial creation (work in progress) (tsalo).
- (testing) Reorganized and fixed - now it actually works! (tsalo).
- (coverage) Code coverage for testing is now tracked (21% - we can improve significantly with workflows) (tsalo).
- (rapidtide2, 2x, happy) Finally found (and fixed) the reason for a range of random stalls and slowdowns when running on a cluster. MKL extensions were silently distributing some numpy calculations over all cores (which means running N jobs running on a cluster tried to use N^2 cores - not good at all…). The maxiumum number of MKL threads is now settable on the command line, and defaults to 1 (no multiprocessor numpy). Strangely, this makes everything a little faster in single processor mode, and A LOT faster in multiprocessor mode.
- (tide_funcs.py) tide_funcs.py has been split into filter.py, fit.py, io.py, miscmath.py, resample.py, stats.py, and util.py. All executables fixed to match.

- (rapidtide2, 2x) Oversample factor is now set automatically by default to make the correlation timestep 0.5 or less. This dramatically improves fits for longer TRs (> 1.5 seconds).
- (rapidtide2, 2x) Moved the major passes (null correlation, correlation, correlation fit, refine, wiener filter and glm) into separate modules for maintainability and to simplify tinkering.
- (rapidtide2, 2x) Isolated multiprocessing code to make speeding up new routines easier and avoid massive code duplication.
- (rapidtide2, 2x) Fixed some bugs in correlation mask reading and saving include and exclude masks.
- (rapidtide2, 2x) Improved tmask, fixed a bug.
- (rapidtide2, 2x, glmfilt) Made glmpass more general so it could be used in other scripts)
- (resamp1tc, resample.py) Added arbresample, modified dotwostepresample.
- (fit.py) Added Kaiser Bessel window function.
- (io.py) savetonifti now properly sets output data type in header.
- (io.py) Added routine to read in motion timecourses.
- (filter.py) Consolidated doprecalcfftfilt and xfunc into transferfuncfilt.
- (filter.py) Added new "complex" spectrum option.
- (filter.py) Added docstrings, code cleanup and regularization.
- (filter.py) Added new 'spectrum' routine.
- (filter.py) Initial support for precalculated arb filtering.
- (resample.py) Adjusted gridding to be symmetric around output value.
- (util.py) Reimplemented valtoindex to make it faster.
- (showtc) Updated to use new spectrum routine.
- (spectrogram) added to distro.
- (rapidtide2, 2x, resamp1tc showxcorr, showxcorrx, showstxcorr) eliminated all use of Butterworth filters by default.
- (spatialfit) Fixed numpy imports

2.70.119 Version 1.5 (6/11/18)

- (documentation) Added description of rapidtide output files.
- (tide_funcs) Fixed a VERY old bug in detrend that added an offset to the detrended timecourse. The effect of the bug on rapidtide2(x) is probably small, because we almost always use data that has already been detrended. The effect in some very particular use cases, though, was large enough that I finally noticed it after 3 years.
- (rapidtide2, 2x) Added option to disable progress bars (good when saving output to a file).
- (rapidtide2, 2x) Improved output to memusage file.
- (rapidtide2, 2x) Report fit errors with better granularity.
- (rapidtide2, 2x) Allow specification of external correlation mask.
- (rapidtide2, 2x) Added "MTT" map to hopefully remove the effect of autocorrelation.
- (rapidtide2, 2x) Added some additional diagnostic information to significance estimation.

- (rapidtide2x, tide_funcs) Major changes to peak fitting to try to improve stability using new findmaxlag_gauss_rev function.
- (rapidtide2x, tide_funcs) Moved logmem function to tide_funcs so that it's available for other programs.
- (rapidtide2x) Fixed bug when running despeckling on a single processor.
- (rapidtide2, 2x) Attempting to stabilize the lagsigma measurement with better initial parameter estimates.
- (tide_funcs) Cast timecourse index as a long to avoid an overflow in NIRS timecourses.
- (rapidtide2, 2x, tide_funcs) Added ability to set searchfrac in fits.
- (rapidtide2, 2x) Disable threshold during significance estimation, simplify internal logic for turning on estimation.
- (rapitdide2) Fixed bug in mask generation
- (showtc) Added the ability to select columns to plot, and to read BIDS style json/tsv.gz physiological files
- (showtc, showxy) Updated colormap names for compatibility with matplotlib 2.2+
- (rapidtide2std) Initial support for warping with ANTs (does not work yet)
- (rapidtide2std) Added the ability to align a single file.
- (tidepool) Support for MTT map.
- (ccorrica, showstxcorr) PEP 8 reformatting.
- (testing) Added test for findmaxlag versions.
- (testing) Added test for stxcorr functions.
- (temporaldecomp) Allow 3D masks.
- (atlastool) Changed method for generating 3D files.
- (atlastool) Various bug fixes in 3D atlas generation.
- (resamp1tc) Modernized option selection, Added nodisplay option.
- (showstxcorr) Explicit integer cast of indices.
- (showstxcorr) Removed initial Hamming window.
- (showstxcorr) Added csv output of matrices.
- (linfit) Added to distribution.
- (tide_funcs) Changed value of rcond in leastsq to be compatible over multiple versions of bumpy (least. comprehensible. note. ever.)
- (tide_funcs) Added findexecutable and isexecutable functions
- (tide_funcs) Added a convolution gridding function
- (tide_funcs) Fixed readvec(s) so that is now works properly if a text file ends with an empty line.
- (tide_funcs) Added function to read slice times from a BIDS sidecar file.
- (spatialdecomp, temporaldecomp) Command line is now saved.
- (threeD) Added

2.70.120 Version 1.4.2 (2/21/18)

- (documentation) Fixed some formatting.
- (showxcorrx) Cleaned up usage statement.

2.70.121 Version 1.4.0 (2/21/18)

- (rapidtide2, 2x) Added macros to support setting multiple options at once.
- (rapidtide2, 2x) –nirs macro sets a number of parameters to be appropriate for NIRS data processing.
- (rapidtide2, 2x) –venousrefine macro sets refinement parameters to use data only from large draining veins (only works reliably in healthy subjects ATM).
- (rapidtide2, 2x) Now tabulate maximum correlation times without range limit for my own secret, diabolical purposes.
- (rapidtide2, 2x) Fixed a bug that was not shifting all of the timecourses if they were not in the refinement mask.
- (rapidtide2, 2x) Improved memory usage tracking.
- (rapidtide2, 2x) Reduced memory footprint.
- (rapidtide2, 2x) Move large arrays into shared memory for multiprocessor jobs to avoid duplicating RAM.
- (rapidtide2, 2x) You can now set the number of processors used (rather than always using all of them) when multiprocessing.
- (rapidtide2, 2x) Properly shut down worker procedures to free RAM earlier.
- (rapidtide2, 2x) Fixed a bug in the initialization of the dispersion calculation.
- (rapidtide2, 2x) Fixed a maddening bug in output of the refinement mask.
- (rapidtide2, 2x) Fixed the range on the Gaussian filtering progress bar.
- (rapidtide2, 2x) Made some improvements to the despeckling procedure.
- (rapidtide2, 2x) –refinepasses is now deprecated - use –passes instead.
- (rapidtide2, 2x) Added new methods to specify the sample rate (or sample time) of the input data file.
- (rapidtide2, 2x) Revised usage statement to make parameter names better reflect their function.
- (rapidtide2, 2x) A lot of internal code cleanup and dead code removal.
- (rapidtide2, 2x, showxcorrx) Allow specification of the correlation window function (hamming (default), hann, blackmanharris, or None).
- (showtc) Cleaned up some bugs introduced during the last overhaul.
- (tcfrom3col) Added to package (generates a timecourse from an FSL style 3 column regressor file.
- (tidepool) Default to displaying using the valid mask rather than the p<0.05 mask.
- (tidepool) Enabled usage of the refine mask.

2.70.122 Version 1.3.0 (12/15/17)

- (rapidtide2, 2x) Added new option, '–despeckle', which uses a spatial median filter to find and correct points where the correlation fit picked the wrong autocorrelation lobe. This dramatically improves the quality of the output maps. This will probably be turned on by default in the next release.
- (tidepool) FINALLY fixed the click positioning bug. Worth the update just for this. That was driving me crazy.
- (tidepool) Formatting improvements.
- (tidepool) Preliminary support for multiple territory atlases and averaging modes in tidepool.
- (tidepool) Atlas averaging is now (mostly) working.
- (rapidtide2, 2x) Now support text format NIRS datasets (2D text files) in addition to NIFTI fMRI files.
- (rapidtide2, 2x) Substantial internal changes to reduce memory footprint, improve speed.
- (rapidtide2x, showxcorrx) Initial support added for Choudry's cepstral analysis method for delay calculation.
- (showtc) Substantial improvements (improved formatting, ability to specify separate subplots, transpose input, line colors, waterfall plots, offsets between lines, titles, etc).
- (rapidtide2std) Internal code cleanup.
- (showxy) Now supports multiple input files.
- Added glmfilt to package to filter 1D or 4D data out of 4D datasets.
- Added spatialdecomp to do spatial PCA decomposition of 4D NIFTI files.
- Added temporaldecomp to do temporal PCA decomposition of 4D NIFTI files.
- Added ccorrica to the distribution to provide cross correlation matrices between all timeseries in a 2D text files.
- Added atlastool to aid in preparation of atlas files for tidepool.

2.70.123 Version 1.2.0 (6/20/17)

- New release to trigger a Zenodo DOI.
- Fully tested for python 3.6 compatibility.
- Added linfit to the distribution.
- Set a limit of 25 dispersion regressors.
- Reformatted the documentation somewhat.
- Added some recipes to the documentation for common use cases.
- Cleaned up and fixed the resampling code.
- Minor quality and speed improvement to timeshift.
- No longer output "datatoremove" to save space.
- Removed some redundant screen refreshes from tidepool.
- Reorganized and removed dead code.
- Changed default mode for calculating refined regressors to "unweighted_average".
- Synced changes in rapidtide2x to rapidtide2

2.70.124 Version 1.1.0 (4/3/17)

- I have now synced all of the changes in rapidtide2x back to rapidtide2.
- rapidtide now has multiprocessing support using the –multiproc flag. This can cause a dramatic increase in processing speed on multicore/processor machines.
- Dispersion calculation is now off by default.
- Tidepool is now compatible with both PyQt4 and 5.
- Reordered some memory allocation and deallocation to keep the RAM footprint down.
- Some additional significant speedups (support for fftw if present, caching hamming windows).
- Added an optional cross-spectral density filter for adaptive timecourse filtering. This is a work in progress - not really ready for general use.
- Skeleton of support for Wiener deconvolution to sharpen the correlation function (off by default, not ready for general use).

2.70.125 Version 1.0.0 (2/13/17)

- I decided to stop hedging and actually commit myself - this is version 1.0.0 - out of beta!
- To promote stability, new features will be put into test versions (the name of the program will have an "x" appended). This way I can do major internal changes and have them available to users without breaking something they might rely on. The "x" versions will sync with the "normal" versions after extensive testing.
- Major new feature (rapidtide2x only for now). Multiprocessing! Significant speedup when using the –multiproc option on machines with multiple cores.
- showxcorrx has new features and defaults.
- Memory allocation has been reorganized to reduce footprint (rapidtide2x).
- Changed imports for better compatibility when running in the NITRC-CE environment (rapidtide2x).
- rapidtide2std now supports nonlinear alignment.
- histnifti is added to the distribution.
- I've added some additional outputs to rapidtide2 and rapidtide2x during refinement to help figure out if the brain is a dispersive filter. This doesn't change how rapidtide2 does the refinement - it's just informational at this point.
- Added spatialfit to the distribution. I use this to analyze delay maps. More on this later.
- Fully implemented samplerate and sampletime options (rapidtide2)
- Corrected and enhanced the use of alternative correlation weighting functions (PHAT, Liang, and Eckart weighting) (rapidtide).
- Updated all scripts for compatibility with matplotlib 2.0.
- Fixed tidepool for compatibility with the new version of pyqtgraph.
- Significant enhancements to showstxcorr (this is a work in progress).
- Example data is no longer installed in the python directory (this never made sense to do).
- Added code to do matched filtering of regressors with mean PSD and/or cross-spectral density. It works, but doesn't do much (rapidtide2x).

2.70.126 Version 0.1.9 (12/19/16)

- Added code to allow runtime memory profiling if memory_profile library is present.
- Extensive casting of variables to lower memory footprint and allow future optimizations.
- Added explicit garbage collection to reduce memory usage.
- Added optional single precision calculation mode to lower memory footprint.
- Added a second script, "rapidtide2x" where I can distribute and test new features without breaking the main code branch.
- Did some speed optimizations in findmaxlag, including faster gaussian fitting and new MUCH faster parabolic fitting (still experimental).
- Minor bug fixes, code reorganization and cleanup.

2.70.127 Version 0.1.8 (11/30/16)

- Fixed a bug in the GLM filtering code - if spatial filtering was applied in rapidtide2, the smoothed data was filtered rather than the original data.
- Added an option in rapidtide2 ("–glmsourcefile=FILE") to apply GLM filter to a different dataset than the one used to estimate the delays (this is used for HCP data - the "hp2000_clean" data has had LFO signals partially removed and may compromise delay estimation, so that should be done on the un-"FIX"ed data).
- Added the ability to detect autocorrelation properties of the test regressor that may cause delay estimation to fail with the "–accheck" flag.
- Added an option "–acfix" to try to correct for bad test regressor autocorrelation properties. This is not yet working correctly.
- Added the ability to specify a slicetimes file ("–slicetimes=FILE") if slicetime correction has not yet been applied to the dataset. Not fully tested.
- (rapidtide2std, tidepool) Added the ability to transform and display functional data to highres anatomic space in addition to MNI152 space.
- Various small bugfixes and format cleanups.

2.70.128 Version 0.1.7 (11/15/16)

- I think I've resolved the issue of crashes due to numba functioning differently on machines other than mine.
- Fixed a masking bug in tidepool that was due to numbers being very close to, but not exactly, 1.
- Made a number of internal changes to rapidtide2 and tidepool to allow dynamic significance masking (not yet working - actually failing spectacularly for the most part, but it's currently commented out).
- Added showstxcorr to the distribution.
- Added the ability to set the mask threshold for correlation and global mask inclusion (this turns out to be needed if you use non-skull-stripped data.)
- Put in some code to start to figure out how to account for dispersion in the delay function.
- Moved the "start movie" button in tidepool to better align with the numerical spin boxes.
- showtc has gotten a significant upgrade in functionality, adding the ability to display power spectra, phase spectra, and set the sample rate to make the x-axis correct.

- Lots of internal formatting/style fixes, and fixed some formatting in the usage statements and documentation.

2.70.129 Version 0.1.6 (10/15/16)

- Fixed a critical bug that had been introduced in the last round of changes to findmaxlag.
- Disabled numba for findmaxlag (it seems to cause problems for some users).
- New option –skipsighistfit to omit fitting a Johnson SB function to the significance histogram.
- Fixed the usage statement.
- Fixed a bug that set ampthresh to zero when not doing significance estimation.

2.70.130 Version 0.1.5 (10/11/16)

- Fixed a bug that made it impossible to specify –regressortstep.
- Added undocumented option –nonumba to turn off just in time compilation if there's a problem with it.
- Print rapidtide version on launch.
- Made pandas import explicit (sklearn requires it).

2.70.131 Version 0.1.4 (10/10/16)

- Some fixes to usage output.
- Added functions for fitting trapezoids (for risetime calculations).
- Changed argument parsing and option output to avoid conflicts
- Added an option to not zero out bad fits (so as not to skew lag statistics)
- Improved fitting of probability distributions.
- Better handling of failed correlation peak fits.
- Now installations should work properly if not installed using git (fixed _gittag import problem).

2.70.132 Version 0.1.3 (9/2/16)

- Added a tool (rapidtide2std) to register all output maps to MNI152 coordinates (requires FSL).
- Made a 3mm resolution ASPECTS map for use in tidepool.
- Reference data is now properly installed, and tidepool can find it reliably.
- Redid the version information. Rapidtide now records both the release version and the git hash in in the output data to help with data provenance.
- Reorganized the distribution into what seems to be a more canonical layout.
- Resolved the issues I seem to have introduced with Python 3 compatibility.
- Significantly cleaned up resampling and filtering code and improved reliability.
- Added some unit tests for critical routines. Strangely, they all fail on Travis-CI, but work on my local machine. It seems to be a numerical precision issue. The answers are rightish, just not right on Travis.

2.70.133 Version 0.1.2 (8/5/16)

- Some bug fixes in filtering and resampling code.
- Beginning to add automated tests.
- Biphasic mode is now fully implemented, including two-tailed significance calculation.

2.70.134 Version 0.1.1 (7/8/16)

- First release

2.70.135 Version 1.3.0 (12/15/17)

- (rapidtide2, 2x) Added new option, '–despeckle', which uses a spatial median filter to find and correct points where the correlation fit picked the wrong autocorrelation lobe. This dramatically improves the quality of the output maps. This will probably be turned on by default in the next release.
- (tidepool) FINALLY fixed the click positioning bug. Worth the update just for this. That was driving me crazy.
- (tidepool) Formatting improvements.
- (tidepool) Preliminary support for multiple territory atlases and averaging modes in tidepool.
- (tidepool) Atlas averaging is now (mostly) working.
- (rapidtide2, 2x) Now support text format NIRS datasets (2D text files) in addition to NIFTI fMRI files.
- (rapidtide2, 2x) Substantial internal changes to reduce memory footprint, improve speed.
- (rapidtide2x, showxcorrx) Initial support added for Choudry's cepstral analysis method for delay calculation.
- (showtc) Substantial improvements (improved formatting, ability to specify separate subplots, transpose input, line colors, waterfall plots, offsets between lines, titles, etc).
- (rapidtide2std) Internal code cleanup.
- (showxy) Now supports multiple input files.
- Added glmfilt to package to filter 1D or 4D data out of 4D datasets.
- Added spatialdecomp to do spatial PCA decomposition of 4D NIFTI files.
- Added temporaldecomp to do temporal PCA decomposition of 4D NIFTI files.
- Added ccorrica to the distribution to provide cross correlation matrices between all timeseries in a 2D text files.
- Added atlastool to aid in preparation of atlas files for tidepool.

2.70.136 Version 1.2.0 (6/20/17)

- New release to trigger a Zenodo DOI.
- Fully tested for python 3.6 compatibility.
- Added linfit to the distribution.
- Set a limit of 25 dispersion regressors.
- Reformatted the documentation somewhat.
- Added some recipes to the documentation for common use cases.

- Cleaned up and fixed the resampling code.
- Minor quality and speed improvement to timeshift.
- No longer output "datatoremove" to save space.
- Removed some redundant screen refreshes from tidepool.
- Reorganized and removed dead code.
- Changed default mode for calculating refined regressors to "unweighted_average".
- Synced changes in rapidtide2x to rapidtide2

2.70.137 Version 1.1.0 (4/3/17)

- I have now synced all of the changes in rapidtide2x back to rapidtide2.
- rapidtide now has multiprocessing support using the –multiproc flag. This can cause a dramatic increase in processing speed on multicore/processor machines.
- Dispersion calculation is now off by default.
- Tidepool is now compatible with both PyQt4 and 5.
- Reordered some memory allocation and deallocation to keep the RAM footprint down.
- Some additional significant speedups (support for fftw if present, caching hamming windows).
- Added an optional cross-spectral density filter for adaptive timecourse filtering. This is a work in progress - not really ready for general use.
- Skeleton of support for Wiener deconvolution to sharpen the correlation function (off by default, not ready for general use).

2.70.138 Version 1.0.0 (2/13/17)

- I decided to stop hedging and actually commit myself - this is version 1.0.0 - out of beta!
- To promote stability, new features will be put into test versions (the name of the program will have an "x" appended). This way I can do major internal changes and have them available to users without breaking something they might rely on. The "x" versions will sync with the "normal" versions after extensive testing.
- Major new feature (rapidtide2x only for now). Multiprocessing! Significant speedup when using the –multiproc option on machines with multiple cores.
- showxcorrx has new features and defaults.
- Memory allocation has been reorganized to reduce footprint (rapidtide2x).
- Changed imports for better compatibility when running in the NITRC-CE environment (rapidtide2x).
- rapidtide2std now supports nonlinear alignment.
- histnifti is added to the distribution.
- I've added some additional outputs to rapidtide2 and rapidtide2x during refinement to help figure out if the brain is a dispersive filter. This doesn't change how rapidtide2 does the refinement - it's just informational at this point.
- Added spatialfit to the distribution. I use this to analyze delay maps. More on this later.
- Fully implemented samplerate and sampletime options (rapidtide2)

- Corrected and enhanced the use of alternative correlation weighting functions (PHAT, Liang, and Eckart weighting) (rapidtide).
- Updated all scripts for compatibility with matplotlib 2.0.
- Fixed tidepool for compatibility with the new version of pyqtgraph.
- Significant enhancements to showstxcorr (this is a work in progress).
- Example data is no longer installed in the python directory (this never made sense to do).
- Added code to do matched filtering of regressors with mean PSD and/or cross-spectral density. It works, but doesn't do much (rapidtide2x).

2.70.139 Version 0.1.9 (12/19/16)

- Added code to allow runtime memory profiling if memory_profile library is present.
- Extensive casting of variables to lower memory footprint and allow future optimizations.
- Added explicit garbage collection to reduce memory usage.
- Added optional single precision calculation mode to lower memory footprint.
- Added a second script, "rapidtide2x" where I can distribute and test new features without breaking the main code branch.
- Did some speed optimizations in findmaxlag, including faster gaussian fitting and new MUCH faster parabolic fitting (still experimental).
- Minor bug fixes, code reorganization and cleanup.

2.70.140 Version 0.1.8 (11/30/16)

- Fixed a bug in the GLM filtering code - if spatial filtering was applied in rapidtide2, the smoothed data was filtered rather than the original data.
- Added an option in rapidtide2 ("–glmsourcefile=FILE") to apply GLM filter to a different dataset than the one used to estimate the delays (this is used for HCP data - the "hp2000_clean" data has had LFO signals partially removed and may compromise delay estimation, so that should be done on the un-"FIX"ed data).
- Added the ability to detect autocorrelation properties of the test regressor that may cause delay estimation to fail with the "–accheck" flag.
- Added an option "–acfix" to try to correct for bad test regressor autocorrelation properties. This is not yet working correctly.
- Added the ability to specify a slicetimes file ("–slicetimes=FILE") if slicetime correction has not yet been applied to the dataset. Not fully tested.
- (rapidtide2std, tidepool) Added the ability to transform and display functional data to highres anatomic space in addition to MNI152 space.
- Various small bugfixes and format cleanups.

2.70.141 Version 0.1.7 (11/15/16)

- I think I've resolved the issue of crashes due to numba functioning differently on machines other than mine.
- Fixed a masking bug in tidepool that was due to numbers being very close to, but not exactly, 1.
- Made a number of internal changes to rapidtide2 and tidepool to allow dynamic significance masking (not yet working - actually failing spectacularly for the most part, but it's currently commented out).
- Added showstxcorr to the distribution.
- Added the ability to set the mask threshold for correlation and global mask inclusion (this turns out to be needed if you use non-skull-stripped data.)
- Put in some code to start to figure out how to account for dispersion in the delay function.
- Moved the "start movie" button in tidepool to better align with the numerical spin boxes.
- showtc has gotten a significant upgrade in functionality, adding the ability to display power spectra, phase spectra, and set the sample rate to make the x-axis correct.
- Lots of internal formatting/style fixes, and fixed some formatting in the usage statements and documentation.

2.70.142 Version 0.1.6 (10/15/16)

- Fixed a critical bug that had been introduced in the last round of changes to findmaxlag.
- Disabled numba for findmaxlag (it seems to cause problems for some users).
- New option –skipsighistfit to omit fitting a Johnson SB function to the significance histogram.
- Fixed the usage statement.
- Fixed a bug that set ampthresh to zero when not doing significance estimation.

2.70.143 Version 0.1.5 (10/11/16)

- Fixed a bug that made it impossible to specify –regressortstep.
- Added undocumented option –nonumba to turn off just in time compilation if there's a problem with it.
- Print rapidtide version on launch.
- Made pandas import explicit (sklearn requires it).

2.70.144 Version 0.1.4 (10/10/16)

- Some fixes to usage output.
- Added functions for fitting trapezoids (for risetime calculations).
- Changed argument parsing and option output to avoid conflicts
- Added an option to not zero out bad fits (so as not to skew lag statistics)
- Improved fitting of probability distributions.
- Better handling of failed correlation peak fits.
- Now installations should work properly if not installed using git (fixed _gittag import problem).

2.70.145 Version 0.1.3 (9/2/16)

- Added a tool (rapidtide2std) to register all output maps to MNI152 coordinates (requires FSL).
- Made a 3mm resolution ASPECTS map for use in tidepool.
- Reference data is now properly installed, and tidepool can find it reliably.
- Redid the version information. Rapidtide now records both the release version and the git hash in in the output data to help with data provenance.
- Reorganized the distribution into what seems to be a more canonical layout.
- Resolved the issues I seem to have introduced with Python 3 compatibility.
- Significantly cleaned up resampling and filtering code and improved reliability.
- Added some unit tests for critical routines. Strangely, they all fail on Travis-CI, but work on my local machine. It seems to be a numerical precision issue. The answers are rightish, just not right on Travis.

2.70.146 Version 0.1.2 (8/5/16)

- Some bug fixes in filtering and resampling code.
- Beginning to add automated tests.
- Biphasic mode is now fully implemented, including two-tailed significance calculation.

2.70.147 Version 0.1.1 (7/8/16)

- First release

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[Tong2019] Tong, Y., Hocke, L.M., and Frederick, B.B., Low Frequency Systemic Hemodynamic "Noise" in Resting State BOLD fMRI: Characteristics, Causes, Implications, Mitigation Strategies, and Applications. Front Neurosci, 2019. 13: p. 787. | http://dx.doi.org/10.3389/fnins.2019.00787

[Erdogan2016] Erdoğan S, Tong Y, Hocke L, Lindsey K, Frederick B. Correcting resting state fMRI-BOLD signals for blood arrival time enhances functional connectivity analysis. Front. Hum. Neurosci., 28 June 2016 | http://dx.doi.org/10.3389/fnhum.2016.00311

[Hocke2016] Hocke LM, Tong Y, Lindsey KP, Frederick BB (2016). Comparison of peripheral near-infrared spectroscopy low-frequency oscillations to other denoising methods in resting state functional MRI with ultrahigh temporal resolution. Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine. 2016. | http://dx.doi.org/10.1002/mrm.26038. PubMed PMID: 26854203.

PYTHON MODULE INDEX

r

INDEX

M

R