fMRIflows: a consortium of fully automatic univariate and multivariate fMRI processing pipelines

Authors: **Michael P. Notter**[1]*, Peer Herholz[2,3]*, Sandra Da Costa[4], Omer F. Gulban[5,6], Ayse Ilkay Isik[7] and Micah M. Murray[1,4,8-9]

1. The Laboratory for Investigative Neurophysiology (The LINE), Department of Radiology, University Hospital Center and University of Lausanne, Switzerland
2. International Laboratory for Brain, Music and Sound Research, Université de Montréal & McGill University, Montréal, Canada
3. McConnell Brain Imaging Centre, Montréal Neurological Institute, McGill University, Montréal, Canada
4. CIBM Center for Biomedical Imaging, Lausanne, Switzerland
5. Department of Cognitive Neuroscience, Maastricht University, The Netherlands
6. Brain Innovation B.V., Maastricht, The Netherlands
7. Department of Neuroscience, Max Planck Institute for Empirical Aesthetics, Frankfurt am Main, Germany
8. Department of Ophthalmology, Fondation Asile des aveugles and University of Lausanne, Lausanne, Switzerland
9. Department of Hearing and Speech Sciences, Vanderbilt University, Nashville, TN, USA

** shared first co-authorship*

Address correspondence to:

Michael P. Notter

Chemin de la Milliere 1

1040 Villars-le-Terroir

Switzerland

Michael.Notter@epfl.ch

Abstract

How functional MRI (fMRI) data are analyzed depends on the researcher and the toolbox used. It is not uncommon that the processing pipeline is rewritten for each new dataset. Consequently, code transparency, quality control and objective analysis pipelines are important for improving reproducibility in neuroimaging studies. Toolboxes, such as Nipype and fMRIPrep, have documented the need for and interest in automated analysis pipelines. Here, we introduce fMRIflows: a consortium of fully automatic neuroimaging pipelines for fMRI analysis, which performs standard preprocessing, as well as 1st- and 2nd-level univariate and multivariate analysis. In addition to the standardized processing pipelines, fMRIflows also provides flexible temporal and spatial filtering to account for datasets with increasingly high temporal resolution and to help appropriately prepare data for multivariate analysis and improve signal decoding accuracy. This paper *first* describes fMRIflows' structure and functionality, *then* explains its infrastructure and access, and *lastly* validates the toolbox by comparing it to other neuroimaging processing pipelines such as fMRIPrep, FSL and SPM. This validation was performed on three datasets with varying temporal resolution to ensure flexibility and robustness, as well as to showcase the improved capability of fMRIflows. The outcome of the validation analysis shows that fMRIflows preprocessing pipeline performs comparably to the ones obtained from other toolboxes. Importantly, fMRIflows' flexible temporal filtering approach leads to improved signal-to-noise-ratio after preprocessing and increased statistical sensitivity, particularly in datasets with high temporal resolution. fMRIflows is a fully automatic fMRI processing pipeline which uniquely offers univariate and multivariate single-subject and group analyses as well as preprocessing. It offers flexible spatial and temporal filtering and thereby leads to more pronounced results for datasets with temporal resolutions at or below 1000ms.

Keywords: Python, neuroimaging, data processing, pipeline, reproducible research

1 Introduction

Functional magnetic resonance imaging (fMRI) is a well-established neuroimaging method used to analyze activation patterns in order to understand brain function. A full fMRI analysis includes preprocessing of the data, followed by statistical analysis and inference on the results, usually separated into 1st-level analysis (the statistical analysis within subjects) and 2nd-level analysis (the group analysis between subjects). The goal of preprocessing is to identify and remove nuisance sources, measure confounds, apply temporal and spatial filters and to spatially realign and normalize images to make them spatially conform (Caballero-Gaudes & Reynolds, 2017). A good preprocessing pipeline should improve the signal-to-noise ratio (SNR) of the data, ensure validity of inference and interpretability of results (Ashburner, 2009), reduce false positive and false negative errors in the statistical analysis and therefore improve the statistical power.

Even though the consequences of inappropriate preprocessing and statistical inference are well documented (Power, Plitt, Laumann, & Martin, 2017; Strother, 2006), most fMRI analysis pipelines are still established ad-hoc, subjectively customized by researchers to each new dataset (Carp, 2012). This usage can be explained by the circumstance that most researchers, by habit or lack of time, stick with the neuroimaging software at-hand or reuse and modify scripts and code snippets from colleagues and previous projects, and do not always adapt their processing pipelines to the newest standard in neuroimaging processing. Rehashing processing pipelines is associated with problems like persisting bugs in the code and delays in updating individual analysis steps to the most recent standards. This can lead to far reaching consequences. Of course, the constant updating of pipelines to newest standards and softwares also bears the risk of introducing new bugs and might lead to the pitfall of blindly trusting new untested procedures.

One solution to tackle this issue will require code transparency, good quality control and a collective development of well-tested objective analysis pipelines (Gorgolewski et al., 2016). Recent years have brought some important reformations to the neuroimaging community that go in this direction.

First, the introduction of Nipype (Gorgolewski et al., 2011) made it easier for researchers to switch between different neuroimaging toolboxes, such as AFNI (Cox & Hyde, 1997), ANTs (Avants et al., 2011), FreeSurfer (Fischl, 2012), FSL (Jenkinson, Beckmann, Behrens, Woolrich, & Smith, 2012), and SPM (Friston, Penny, Ashburner, Kiebel, & Nichols, 2006). Nipype together with other software packages such as Nibabel (Brett et al., 2018) and Nilearn (Abraham et al., 2014) opened up the whole Python ecosystem to the neuroimaging community. Code can be shared between researchers via online services such as GitHub (https://github.com), and the whole neuroimaging software ecosystem can be run on any machine or server through the use of container software such as Docker (https://www.docker.com) or Singularity (https://www.sylabs.io). Combined with a continuous integration service such as CircleCI (https://circleci.com) or TravisCI (https://travis-ci.org), this allows the creation of easy-to-read, transparent, shareable and continuously tested open-source neuroimaging processing pipelines.

Second, the next major advancement in the neuroimaging field was the introduction of a common dataset standard, such as the NIfTI standard (https://nifti.nimh.nih.gov/). This was important for the formatting of neuroimaging data. The neuroimaging community gathered together in a consortium to define a standard format for the storage of neuroimaging datasets, the so-called Brain Imaging Data Structure (BIDS) (Gorgolewski et al., 2016). A common data structure format facilitates the sharing of datasets and makes it possible to create universal neuroimaging toolboxes that work out-of-the-box on any BIDS conforming dataset. Additionally, through services like OpenNeuro (Gorgolewski, Esteban, Schaefer, Wandell, & Poldrack, 2017), a free online platform for sharing neuroimaging data, one can test the robustness and flexibility of a new neuroimaging toolbox on hundreds of different datasets.

Software toolboxes like MRIQC (Esteban et al., 2017) and fMRIPrep (Esteban et al., 2019) have shown how fruitful this new neuroimaging ecosystem can be and have highlighted the importance and need of good quality control and high-quality preprocessing workflows with consistent results from diverse datasets. Thus, the present software package, called fMRIflows, will be based on this new ecosystem and toolboxes, and will expand it to fully automated pipelines for univariate and multivariate individual and group analyses.

Moreover, fMRIflows provides flexible temporal and spatial filtering, to account for two recent findings in the field. *First*, flexible spatial filtering can become of importance when performing multivariate analysis, as it has been shown that the correct spatial band-pass filtering can improve signal decoding accuracy (Sengupta, Pollmann, & Hanke, 2018). *Second*, correct temporal filtering during preprocessing is important and can lead to improved signal-to-noise ratio (SNR), especially for fMRI datasets with a temporal resolution rate below one second (Viessmann, Möller, & Jezzard, 2018), but only if the filter is applied orthogonally to the other filters during preprocessing to ensure that previously removed noise is not again reintroduced into the data (Lindquist, Geuter, Wager, & Caffo, 2019). Due to technical improvements in imaging recording through acceleration techniques such as GRAPPA (Griswold et al., 2002) and simultaneous multi-slice/multiband acquisitions (Feinberg et al., 2010; Feinberg & Setsompop, 2013; Moeller et al., 2010), faster sampling rates became possible, to the point that respiratory and cardiac signals can be sufficiently sampled in the BOLD signal. This creates new challenges for the preprocessing of functional images, especially when the external recording of those physiological sources cannot be achieved. fMRIflows tackles this challenge in an innovative way by leveraging notebook format.

fMRIflows presents a consortium of fully automatic neuroimaging pipelines for fMRI analysis, performing standardized preprocessing, as well as 1st- and 2nd-level analyses for univariate and multivariate analysis, with the additional creation of informative quality control figures. fMRIflows is predicated on the insights and code base of MRIQC (Esteban et al., 2017) and fMRIPrep (Esteban et al., 2019), extending their functionality with regard to the following aspects: (a) flexible temporal and spatial filtering of fMRI data, i.e. low- or band-pass filtering allowing for the exclusion of high-frequency oscillations introduced through physiological noise (Viessmann et al., 2018); (b) accessible and modifiable

code base; (c) automatic computation of 1st-level contrasts for univariate and multivariate analysis; and (d) automatic computation of 2nd-level contrasts for univariate and multivariate analysis.

In this paper, we (1) describe the different pipelines included in fMRIflows and illustrate the different processing steps involved, (2) explain the software structure and setup, and (3) validate fMRIflows' performance by comparing it to other widely used neuroimaging toolboxes, such as fMRIPrep (Esteban et al., 2019), FSL (Jenkinson et al., 2012) and SPM (Friston et al., 2006).

2 Materials and Methods

2.1.1 fMRIflows' processing pipelines

The complete code base of fMRIflows is open access and stored conveniently in six different Jupyter notebooks on https://github.com/miykael/fmriflows. The first notebook does not contain any processing pipeline, but rather serves as a user input document that helps to create JSON files, which will contain the execution specific parameters for the five processing pipelines contained in fMRIflows: (1) anatomical preprocessing, (2) functional preprocessing, (3) 1st-level analysis, (4) 2nd-level univariate analysis and (5) 2nd-level multivariate analysis. Each of these five pipelines stores its results in a sub hierarchical folder, specified as an output folder by the user. In the following section, we explain the content of those six Jupyter notebooks.

Specification preparation

Each fMRIflows processing pipeline needs specific input parameters to run. Those parameters range from subject ID and, number of functional runs per subject, to requested voxel resolution after image normalization, etc. Each notebook will read the relevant specification parameters from a predefined JSON file that starts with the prefix "`fmriflows_spec`". There is one specification file for the anatomical and functional preprocessing, one for the 1st and 2nd level univariate analysis, and one for the 2nd-level multivariate analysis. For an example of these three JSON files, see Supplementary Note 1. The first notebook contained in fMRIflows, called `01_spec_preparation.ipynb`, can be used to create those JSON files, based on the provided dataset and some standard default parameters. It does so by using Nibabel v2.3.0 (Brett et al., 2018), PyBIDS v0.8 (Yarkoni et al., 2019) and other standard Python libraries. It is up to the user to change any potential processing parameter should they be different from the used default values.

Anatomical preprocessing

The anatomical preprocessing pipeline is contained within the notebook `02_preproc_anat.ipynb` and uses the JSON file `fmriflows_spec_preproc.json` for parameter specification such as voxel resolution. If a specific value is not set, fMRIflows normalizes to an isometric voxel resolution of 1 mm^3 by default. However, the user can also choose an anisometric voxel resolution that varies in all three dimensions. Additionally, the user can decide to have a fast or precise normalization. The precise normalization can take up to eight times as long as the fast approach but guarantees a normalization of high precision. For an example of the JSON file content, see Supplementary Note 1.

The anatomical preprocessing pipeline only depends on the subject specific T1-weighted (T1w) anatomical images as input files. The individual processing steps are visualized in Figure 1 and consist of: (1) image reorientation, (2) cropping of field of view (FOV), (3) correction of intensity non-uniformity (INU), (4) image segmentation, (5) brain extraction and (6) image normalization. For a more detailed description of the steps involved in this processing pipeline, see Supplementary Note 2.

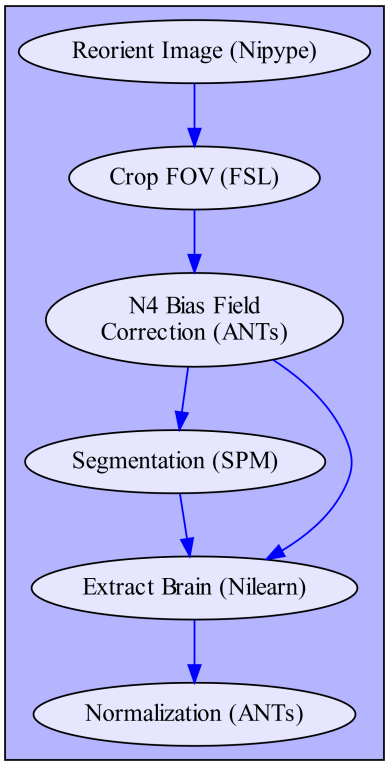


Figure 1: Depiction of fMRIflows' anatomical preprocessing pipeline. Arrows indicate dependency between the different processing steps and data flow. Name of each node describes functionality, with the corresponding software dependency mentioned in brackets.

Functional preprocessing

The functional preprocessing pipeline is contained within the notebook `03_preproc_func.ipynb` and uses the JSON file `fmriflows_spec_preproc.json` for parameter specification. As specification parameters, users can indicate if slice-time correction should be applied or not, and if so which reference timepoint should be used. The user can also indicate to which isometric or anisometric voxel resolution functional images should be sampled to, and if the sampling is into subject or template space. For the template space, the ICBM 2009c nonlinear asymmetric brain template is used (Fonov et al., 2011). Furthermore, users can specify possible values for low-, high- or band-pass filters in the temporal or spatial domain. Additionally, to investigate nuisance regressors, users can specify the number of CompCor (Behzadi, Restom, Liau, & Liu, 2007) or independent component analysis (ICA) components they want to extract and which threshold values they want to use to detect outlier volumes. The implications of those parameters will be explained in more details in the following sections. For an example of the JSON file content, see Supplementary Note 1.

The functional preprocessing pipeline depends as inputs on the output files from the anatomical preprocessing pipeline, as well as the subject-specific functional images and accompanying descriptive JSON file that contains information about the temporal resolution (TR) and slice order of the functional image recording. This JSON file is part of the BIDS standard and therefore should be available in the BIDS conform dataset. The individual processing steps are schematized in Figure 2 and consist of: (1) image reorientation, (2) non-steady-state detection, (3) creation of functional brain mask, (4) slice time correction, (5) estimation of motion parameters, (6) two-step estimation of coregistration parameters between functional and anatomical image, (7) finalization of motion parameters, (8) single-shot spatial interpolation applying motion correction, coregistration and if specified normalizing images to the template image, (9) construction and application of brain masks, (10) temporal filtering and (11) spatial filtering. It is important to mention that the functional preprocessing is done for each functional run separately to prevent inter-run contaminations. For a more detailed description of the steps involved in this processing pipeline, see Supplementary Note 3.

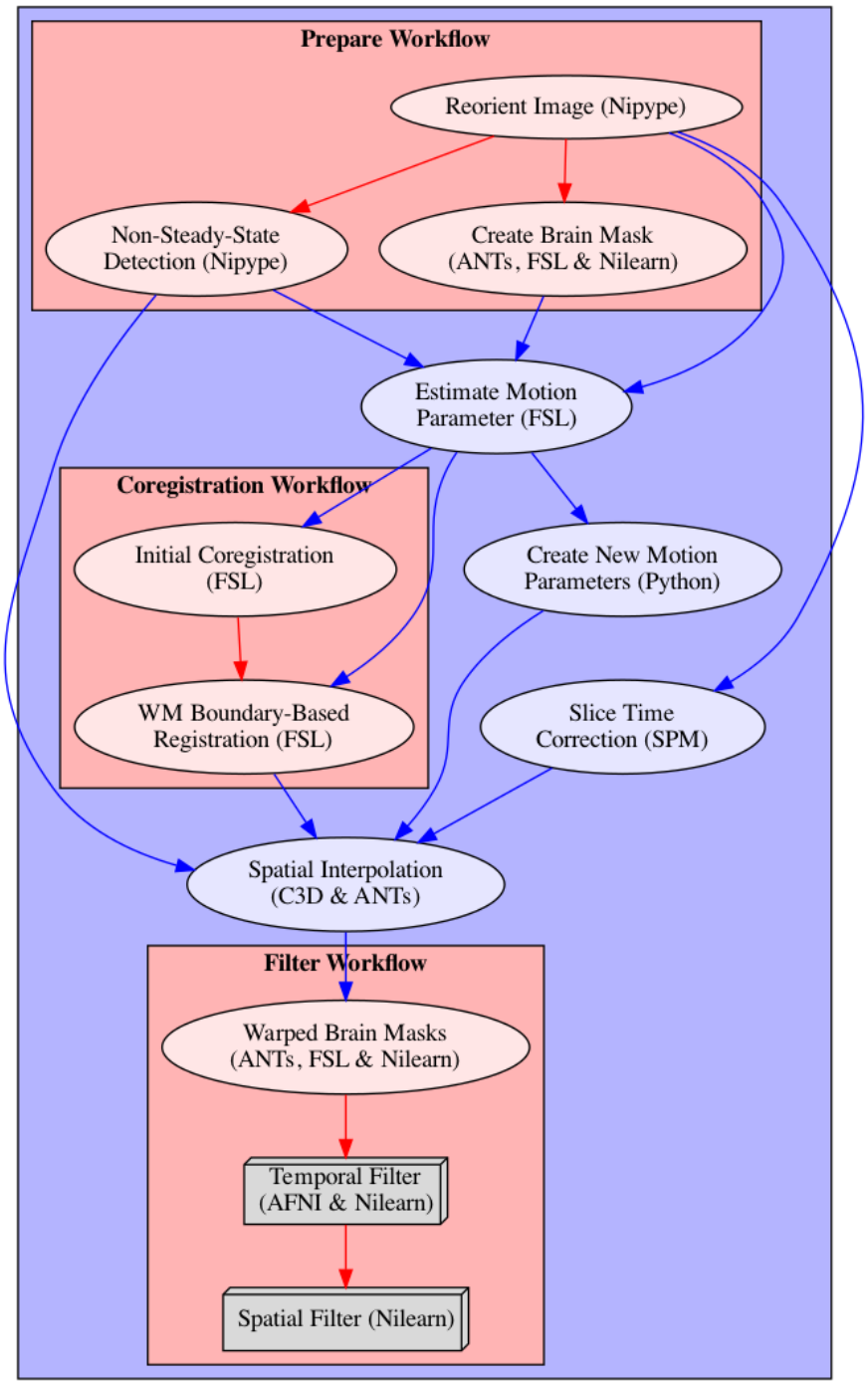


Figure 2: Depiction of fMRIflows' functional preprocessing pipeline. Arrows indicate dependency between the different processing steps and data flow. Name of each node describes functionality, with the corresponding software dependency mentioned in brackets. Steps that can be grouped into specific sections are contained within a red box to facilitate understanding of the pipeline. Color of arrows indicated if connection stays within a section (red) or not (blue). Nodes depicted as gray boxes indicate that they can be run multiple times with iterating input values, i.e. performing a spatial smoothing with an FWHM value of 4 and 8mm.

1st-level analysis

The first level analysis pipeline is contained within the notebook `04_analysis_1st-level.ipynb` and uses the JSON file `fmriflows_spec_analysis.json` for parameter specification. As specification parameters, users can indicate which nuisance regressors to include in the GLM, if outliers should be considered, and if the data is already in template space or if this normalization should be done after the estimation of the contrasts. Users can also specify other GLM model parameters, such as the high-pass filter value and the type of basis function that should be used to model the haemodynamic response function (HRF). Additionally, the users will also specify a list of contrasts they want to be estimated, or if they want to create specific contrasts for each stimulus column in the design matrix, and/or for each session separately, which then later might also be used for multivariate analysis. For an example of the JSON file content, see Supplementary Note 1.

The 1st-level analysis pipeline depends on a number of outputs from the previous anatomical and functional preprocessing pipelines, i.e. the TSV (tab separated value) file containing motion parameters and confound regressors, a text file indicating the number of non-steady-state volumes removed from the functional image, and a text file containing a list of indexes identifying outlier volumes. Additionally, the 1st-level analysis pipeline also requires BIDS conform events files containing information on the applied experimental design, including types of conditions and their respective onsets and durations. The individual processing steps included in the 1st-level analysis consist of: (1) collecting preprocessed files and model relevant information, (2) model specification and estimation, (3) univariate contrast estimation, (4) optional preparation for multivariate analysis, (5) optional spatial normalization of contrasts (Figure 3). All of the relevant steps, that is model creation, estimation and contrast computation are performed with SPM version 12. For a more detailed description of the steps involved in this processing pipeline, see Supplementary Note 4.

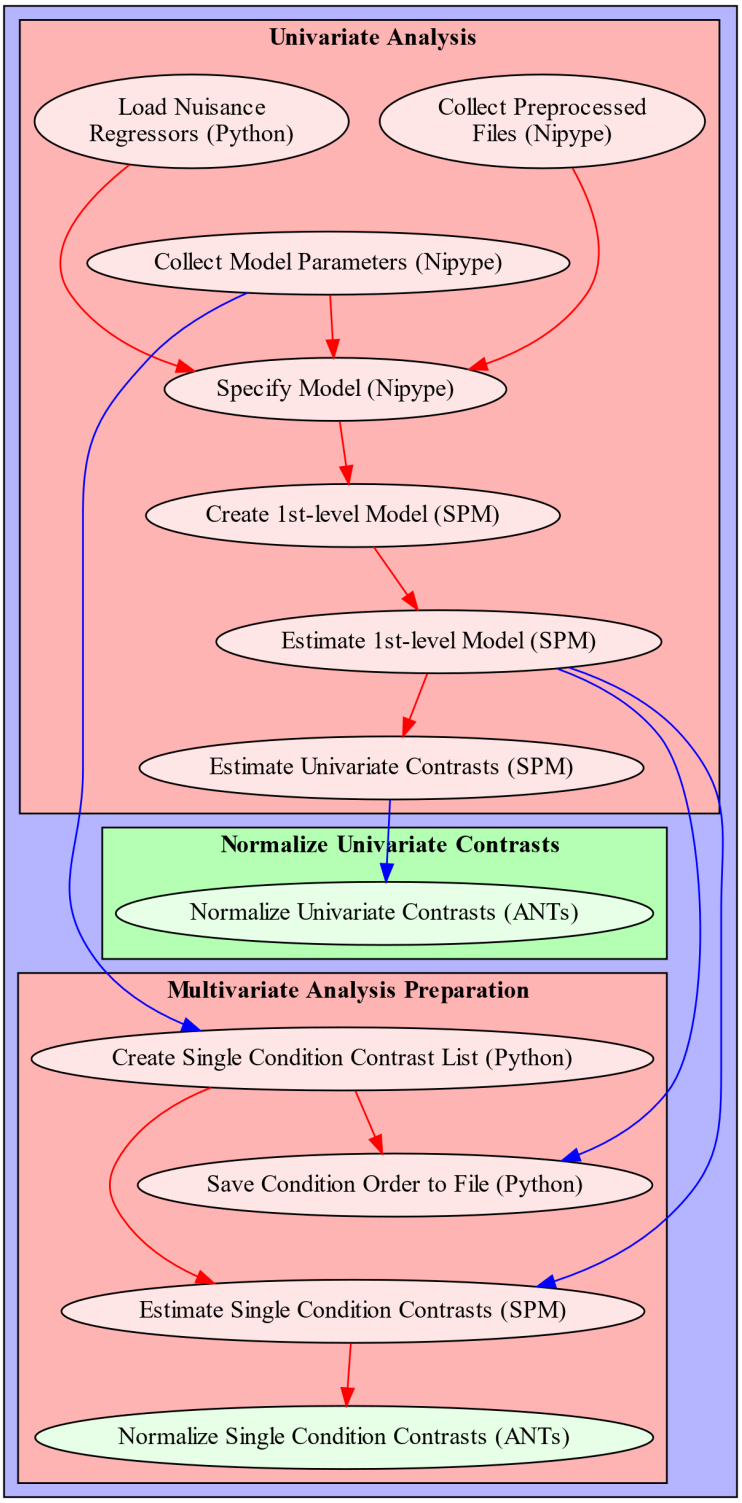


Figure 3: Depiction of fMRIflows' 1st-level analysis pipeline. Arrows indicate dependency between the different processing steps and data flow. Name of each node describes functionality, with the corresponding software dependency mentioned in brackets. Sections that can be grouped into specific sections are contained within a red box to facilitate understanding of the pipeline. Color of arrows indicated if connection stays within a section (red) or not (blue). Nodes depicted in green are optional and can be run if spatial normalization was not yet performed during preprocessing.

2nd-level univariate analysis

The second level univariate analysis pipeline is contained within the notebook `05_analysis_2nd-level.ipynb` and uses the JSON file `fmriflows_spec_analysis.json` for parameter specification. Users can specify the probability value used as a cutoff for the threshold of the GM probability tissue map in template space that is later used during the model estimation. Additionally, users can specify voxel- and cluster-threshold topological thresholding of the statistical contrast, as well as relevant AtlasReader (Notter et al., 2019) parameters for the creation of the output tables and figures.

The 2nd-level univariate analysis pipeline depends only on the estimated contrasts from the 1st-level univariate analysis. No further contrast specification is required as fMRIflows currently only implements a simple one-sample T-test. The individual processing steps included in the 2nd-level univariate analysis consist of: (1) gathering of the 1st-level contrasts, (2) creation and estimation of 2nd-level model, (3) estimation of contrast estimation, (4) topological thresholding of contrasts, (5) results creation with AtlasReader. As for the 1st-level analysis, all of the relevant model creation, estimation and contrast computation are performed with SPM12. For a more detailed description of the steps involved in this processing pipeline, see Supplementary Note 5.

2nd-level multivariate analysis

The second level multivariate analysis pipeline is contained within the notebook `06_analysis_multivariate.ipynb` and uses the JSON file `fmriflows_spec_multivariate.json` for parameter specification. Users can define a list of classifiers to use for the multivariate analysis, the sphere radius and step size of the searchlight approach. To perform a 2nd-level analysis of searchlight results users can decide between a classical GLM approach testing against chance level and a more recommended permutation based method as described in Stelzer, Chen, & Turner (2013) with the option of determining the number of permutations. Additionally, users can specify voxel- and cluster-threshold topological thresholding of the statistical contrast, as well as relevant AtlasReader parameters for the creation of the output tables and figures.

The 2nd-level multivariate analysis pipeline depends on the estimated contrasts from the 1st-level multivariate analysis, the associated CSV file containing a list of the corresponding contrast labels and a list of binary classification identifiers. In contrast to the other notebooks, this notebook uses Python 2.7 to accommodate the requirements of PyMVPA v2.6.5 (Hanke et al., 2009). The individual processing steps included in the 2nd-level multivariate analysis consist of: (1) data preparation for the analysis with PyMVPA, (2) searchlight classification, (3) computation of group analysis using a T-test, (4) computation of group analysis according to Stelzer et al. (2013), and (5) results creation with AtlasReader. For a more detailed description of the steps involved in this processing pipeline, see Supplementary Note 6.

2.1.2 Infrastructure and access to fMRIflows

The source code of fMRIflows is available at GitHub (https://github.com/miykael/fmriflows) and is licensed under the BSD 3-Clause "New" or "Revised" License. The code is written in Python v3.7.2 (https://www.python.org), stored in Jupyter Notebooks v4.4.0 (Kluyver et al., 2016) and distributed via Docker v18.09.2 (https://docker.com) containers that are publicly available via Docker Hub (https://hub.docker.com). The usage of Docker allows the user to run fMRIflows on any major operating system, with the following command:

```
docker run -it -p 9999:8888 -v /home/user/ds001:/data miykael/fmriflows
```

The first flag -it indicates that the docker container should be run in interactive mode, while the second flag `-p 9999:8888` defines the port (here 9999) that we want to use to access the Jupyter Notebooks via the web-browser. The third flag, `-v /home/user/ds001:/data` tells fMRIflows the location of the BIDS conform dataset that should be mounted in the docker container, here located at `/home/user/ds001`. Once the docker container is launched, the interactive Jupyter Notebooks can be accessed through the web-browser.

fMRIflows uses many different software packages for the individual processing steps. The neuroimaging software that are used are: Nipype v1.1.9 (Gorgolewski et al., 2011), FSL v5.0.9 (Smith et al., 2004), ANTs v2.2.0 (Avants et al., 2011), SPM12 v7219 (Penny, Friston, Ashburner, Kiebel, & Nichols, 2011), AFNI v18.0.5 (Cox & Hyde, 1997), Nilearn v0.5 (Abraham et al., 2014), Nibabel v2.3.0 (Brett et al., 2018), PyMVPA v2.6.5 (Hanke et al., 2009), Convert3D v1.1 (https://sourceforge.net/p/c3d), AtlasReader v0.1 (Notter et al., 2019) and PyBIDS v0.8 (Yarkoni et al., 2019). In addition to some standard Python libraries, fMRIflows also uses Numpy (Oliphant, 2007), Scipy (Jones, Oliphant, Peterson, & others, 2001), Matplotlib (Hunter, 2007), Pandas (McKinney & others, 2010) and Seaborn (http://seaborn.pydata.org).

With every new pull request pushed to the GitHub repository of fMRIflows, a test instance on CircleCI (https://circleci.com) is deployed to test the complete code base for execution errors. This framework allows the continuous integration of new code to fMRIflows, and guarantees the general functionality of the software package. Outputs are not controlled for their correctness.

2.1.3 Validation of fMRIflows

fMRIflows was validated in two phases. In *Phase 1*, we validated the proficiency of the toolbox by applying it on different kinds of fMRI datasets conforming to the BIDS standard (Gorgolewski et al., 2016) available via OpenNeuro.org (Gorgolewski et al., 2017). Insights during this phase allowed us to improve the code base and make fMRIflows robust to a diverse set of datasets. In *Phase 2*, we compared the performance of the toolbox to similar neuroimaging preprocessing pipelines such as fMRIPrep, FSL, and

SPM. To better understand where fMRIflows overlaps or diverges from comparable processing pipelines, we investigated the preprocessing, subject-level and group-level outcomes for all four toolboxes, run on three different datasets.

Phase 1: Proficiency validation

To investigate the capabilities and flaws of the initial implementation of the toolbox, fMRIflows was run on different datasets, either available publicly via OpenNeuro.org or available privately to the authors. Such an approach allowed the exploration of datasets with different temporal and spatial resolutions, SNRs, FOVs, numbers of slices, scanner characteristics, and other sequence parameters, such as acceleration factors and flip angles.

Phase 2: Performance validation

To validate the performance of fMRIflows, we used three different task-based fMRI datasets and compared its preprocessing to the three neuroimaging processing pipelines fMRIPrep, FSL and SPM. Comparison was done on preprocessing, subject-level and group-level outputs. Because of differences in how FSL and SPM perform subject- and group-level analyses and due to the lack of such routines in fMRIPrep, all subject- and group-level analyses for the performance validation were performed using identical Nistats (Abraham et al., 2014) routines.

The three datasets (see Table 1) were all acquired on scanners with a magnetic field strength of 3 Tesla and differ in many sequence parameters, most notably in the temporal resolution with which they were recorded. This is especially important as we aim to highlight that the right handling of temporal filtering is crucial for datasets with a temporal resolution below 1000ms.

Table 1: **Overview of the datasets used to validate fMRIflows.**

Dataset	**TR2000**	**TR1000**	**TR600**
Temporal resolution	2000ms	1000ms	600ms
Spatial resolution	3.5 x 3.5 x 3.3	2.0 x 2.0 x 2.4	3.0 x 3.0 x 3.0
Number of slices	36	64	24
Slice Order	Descending	Unknown	Interleaved
Coverage	Whole brain	Whole brain	Slab
Volumes per run	275	453	600
Number of runs	4	4	6
Acceleration Factor	None	4	3
Magnetic strength	3 Tesla	3 Tesla	3 Tesla
Number of subjects	12	20	17
Sequence type	2D-EPI	Multi-Band	SMS
Task	Audio-visual memory task	Mixed gamble task	Audio-visual observation task
Data Availability	OpenNeuro.org (ds001345, v 1.0.1)	OpenNeuro.org (ds001734, v.1.0.4)	OpenNeuro.org (will be made available after publication of experimental work)

Dataset TR2000 has a comparably low temporal and spatial resolution. It serves as a standard dataset, recorded with a standard EPI scan sequence. The dataset and paradigm are described in more details in Notter et al. (under review). In short, participants performed a continuous recognition task and indicated for each image whether it is old or new. When the image was presented for the first time (new) it was either presented with no sound (unisensory visual context) or together with a sound (multisensory context).

Dataset TR1000 has a rather high temporal and spatial resolution and serves as an advanced dataset, recorded with a scan sequence using a multiband acceleration technique. The dataset and paradigm are described in more detail in Botvinik-Nezer et al. (2019). In short, participants performed a mixed gambling task in which they were asked to either accept or reject a possible monetary gain or loss.

Dataset TR600 has a very high temporal resolution with a moderate spatial resolution and serves as an extreme dataset, recorded with scan sequences using a simultaneous multi-slice (SMS) acceleration technique (Feinberg et al., 2010). This dataset was collected for another project. In short, participants were shown auditory, visual or audiovisual stimuli containing either an animal (as an image or sound), pure noise or both together. Participants performed a discrimination task in which they had to indicate if they perceived a stimuli with an animal in it or not, independent of the stimuli modality. The stimuli were either presented in a unisensory or multisensory context.

All participants of the Datasets TR2000 and TR600 have been included in the performance validation, while only the first 20 out of the 120 total participants of the Dataset TR1000 was used in order

to reduce computation time and make this dataset comparable to the other two. Datasets TR2000 and TR1000 are already publicly available through the OpenNeuro platform. Dataset TR600 is in preparation to be published on OpenNeuro as well. Until then, this dataset is available upon request.

The **preprocessing pipelines** with fMRIflows, fMRIPrep, FSL and SPM were based on the default parameters and only differed in the following points from their standard implementations: (1) Functional images were resampled to an isometric voxel resolution according to the dominant resolution dimension within a dataset; (2) Spatial smoothing of the functional images is applied after preprocessing of the images, using a Nilearn routine and a smoothing kernel with a Full Width at Half Maximum (FWHM) of 6mm, in order to keep the approaches comparable, as fMRIPrep does not allow spatial smoothing of functional images; (3) Anatomical images in the FSL pipeline were first cropped to a standard FOV, followed by brain extraction using FSL's BET before FSL's FEAT was launched; (4) In the case of FSL, the normalization from structural to standard space was done using a non-linear warping approach with 12 degrees of freedom and a spline interpolation model; (5) In the case of SPM, the template brain for the normalization was its standard tissue probability brain TPM, while for fMRIflows, fMRIPrep and FSL, the ICBM 2009c nonlinear asymmetric brain template was used.

The statistical inference was not performed on any of the investigated toolboxes to prevent the introduction of a software specific bias. The 1st- and 2nd-level analysis was performed using Nistats, Nilearn and other Python toolboxes and only differed between the toolboxes in the following ways: (1) the estimated motion parameters added to the design matrix during the 1st-level analysis differed for each toolbox as they were based on the software-specific preprocessing routine; (2) the number of volumes per run used during the 1st-level analysis of fMRIflows might differ slightly from the other approaches, as the fMRIflows routine removes non-steady state volumes during the preprocessing; (3) SPM used its own tissue probability map to create a binary mask restricted to gray matter voxels during the group analysis, while the other three toolboxes used the ICBM 2009c gray matter probability map instead.

To compare the unthresholded group statistic maps between the toolboxes, we created for each pairwise combination of preprocessing approach a Bland-Altman 2D histogram plot, as described by Bowring, Maumet, & Nichols (2018). These plots show the difference between the statistic value (y-axis), against the mean statistic value (x-axis) for all voxels within the intersection of the respective brain mask. In other words, it summarized in a 2D histogram plot, for each voxel how much higher the statistical value in toolbox B is (y-axis), in comparison to toolbox A's statistical value (x-axis).

The complete lists of parameters, the scripts to perform preprocessing, 1st- and 2nd-level analysis and the scripts to create individual figures can be found on fMRIflows GitHub page (https://github.com/miykael/fmriflows/tree/master/paper). Derivatives generated for the validation in phase 2 can be inspected and downloaded on NeuroVault (Gorgolewski et al., 2015) under the following links: (1) Standard deviation maps of temporal averages after preprocessing (https://identifiers.org/neurovault.collection:5645), (2) temporal SNR maps after preprocessing (https://identifiers.org/neurovault.collection:5713), (3) binarized 1st-level activation count maps

(https://identifiers.org/neurovault.collection:5647), (4) 2nd-level activation maps (https://identifiers.org/neurovault.collection:5646).

2.2 Results

2.2.1 Summary of outputs obtained by fMRIflows' processing pipelines

Output generated after executing the anatomical preprocessing pipeline

After the execution of the anatomical preprocessing pipeline, the following files are generated for each subject: (1) image of the inhomogeneity-corrected full head image, (2) image of the extracted brain, (3) binary mask used for the brain extraction, (4) individual tissue probability maps for gray matter (GM), white matter (WM), cerebrospinal fluid (CSF), skull and head, (5) normalized anatomical image in template space (6) reverse-normalized template image in subject space, (7) plus the corresponding transformation matrices used for output 5 and 6. Each anatomical preprocessing output folder also contains (8) the ICBM 2009c brain template used for the normalization, sampled to the requested voxel resolution.

In addition to these files, the following three informative figures are generated: (1) tissue segmentation, (2) brain extraction and (3) spatial normalization of the anatomical image. A shortened version of those three figures, as well as their explanation are shown in Figure 4.

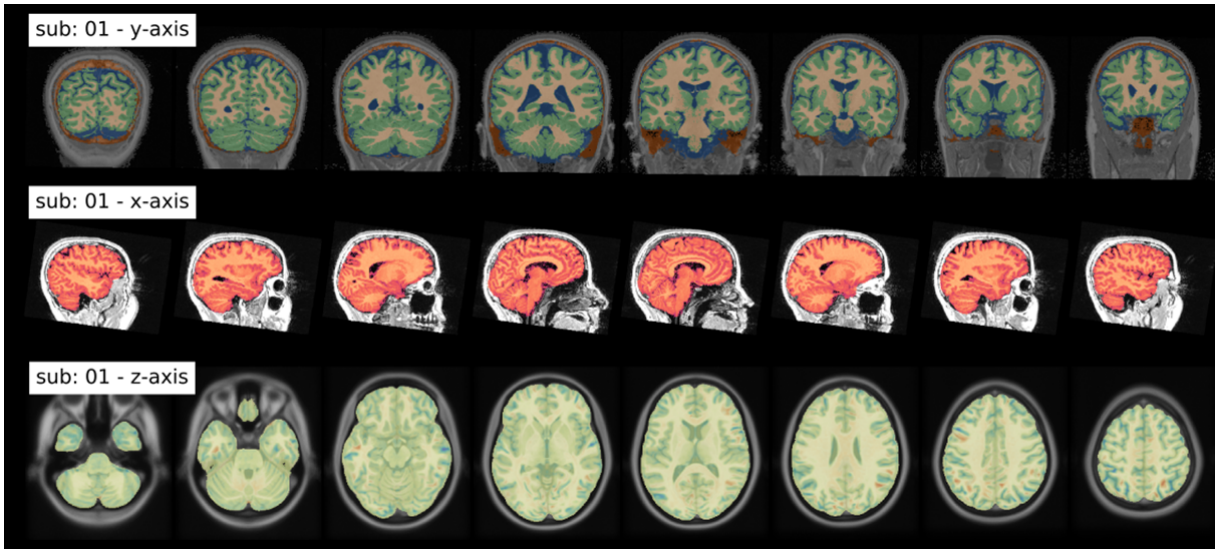


Figure 4: Summary of output figures generated by fMRIflows after executing the anatomical preprocessing pipeline. (***Top***) Coronal view of the image segmentation output, showing gray matter tissue in green, white matter tissue in beige, cerebrospinal fluid in blue. (***Middle***) Sagittal view of the brain extraction output, showing the extracted brain image in red, and the original anatomical image in gray. (***Bottom***) Axial view of the spatial normalization output, showing the normalized brain image highlighted in yellow, overlaid over the ICBM 2009c brain template in gray. Regions in red and blue show negative and positive deformation discrepancy between the normalized subject image and the template.

Output generated after executing the functional preprocessing pipeline

After the execution of the functional preprocessing pipeline, the following files are generated separately for each subject, each functional run and each temporal filtering: (1) text file indicating which volumes were detected as outliers, (2) tabular separated (TSV) file containing all extracted confound regressors, (3) text file containing the six motion parameter regressors according to FSL's output scheme, (4) binary masks for the brain, (5) masks for anatomical and functional component based noise correction, (6) functional mean image, and (7) completely preprocessed functional images, separated by spatial smoothing approaches. Each subject folder also contains (8) one text file per functional run indicating the number of non-steady-state volumes at the beginning of run.

The following is a more detailed description of the multiple confounds fMRIflows estimates during functional preprocessing:

Confounds based on motion parameters: In addition to the head motion parameters created during preprocessing, fMRIflows also computes (1) 24-parameter Volterra expansion of the motion parameters (Friston, Williams, Howard, Frackowiak, & Turner, 1996) using custom scripts and (2) Framewise Displacement (FD) component (Power, Barnes, Snyder, Schlaggar, & Petersen, 2012) using Nipype.

Confounds based on global signal: Functional images before spatial smoothing were used to compute confound regressors, such as (1) DVARS, which represents the spatial standard deviation of the signal after temporal differencing, to identify motion-affected frames (Power et al., 2012), using Nipype and (2) four global signal curves representing the average signal in the total brain volume (TV), GM, WM and CSF, using Nilearn.

Detection of outlier volumes: The user can specify which of the six signal curves for FD, DVARS and average signal in TV, GM, WM and CSF to use to identify outlier volumes (see Figure 5A). Those are volumes that have larger fluctuations in the signal values in a given volume, compared to the z-scored standard deviation throughout the time course. The exact threshold for each curve can be adapted by the user, but its default value is set to a z-value of 3.27, representing 99%, for the FD, DVARS and TV signal. The identification number of each outlier volume is stored in a text file that might be used in the 1st-level pipeline during the GLM model estimation to remove the effect of those volumes from the overall analysis, also known as censoring (Caballero-Gaudes & Reynolds, 2016).

Confounds based on signal components: Using the temporal filtered functional images, two different kinds of approaches are performed to extract components that could be used for denoising or dimensionality reduction of the data. The first approach is called CompCor (Behzadi et al., 2007) and uses principal component analysis (PCA) to estimate the main sources of noise within specific confound regions. Regions are either defined by their temporal or anatomical characteristics. The temporal CompCor approach (tCompCor) considers the 2% most variable voxels within the confound brain mask as sources of confounds. The anatomical CompCor approach (aCompCor), considers voxels within twice eroded WM and CSF brain masks as sources of confounds. The user can specify how many aCompCor and tCompCor components should be computed, but the default value is set to five each. The second approach uses

independent component analysis (ICA) to perform source separation in the signal. Using Nilearn's CanICA routine, fMRIflows computes by default the top ten independent components throughout the confound masks. The number of confounds to extract can be adjusted by the user.

Storage of confound information: All of the confound curves computed after functional preprocessing are stored in a TSV file to allow for easy access.

Diverse set of overview figures: To allow for visual inspection of the numerous outputs generated after the execution of the functional preprocessing pipeline, fMRIflows creates many informative overview figures. These overviews cover the motion parameters used for head motion correction, the anatomical and temporal CompCor components, FD, DVARS, average signal in TV, GM, WM and CSF, and the ICA components. fMRIflows also creates a brain overview figure showing the extent of the different masks applied during functional preprocessing, a spatial correlation map between the ICA components and the individual voxel signal, and a carpet plot according to Power (2017) and Esteban et al. (2019). To better visualize underlying structures in the carpet plot the time series traces are sorted by their correlation coefficients to the average signal within a given region, allowing for a positive or negative time lag of 2 volumes. A shortened version of all these figures, as well as their explanations are shown in Figures 5-7.

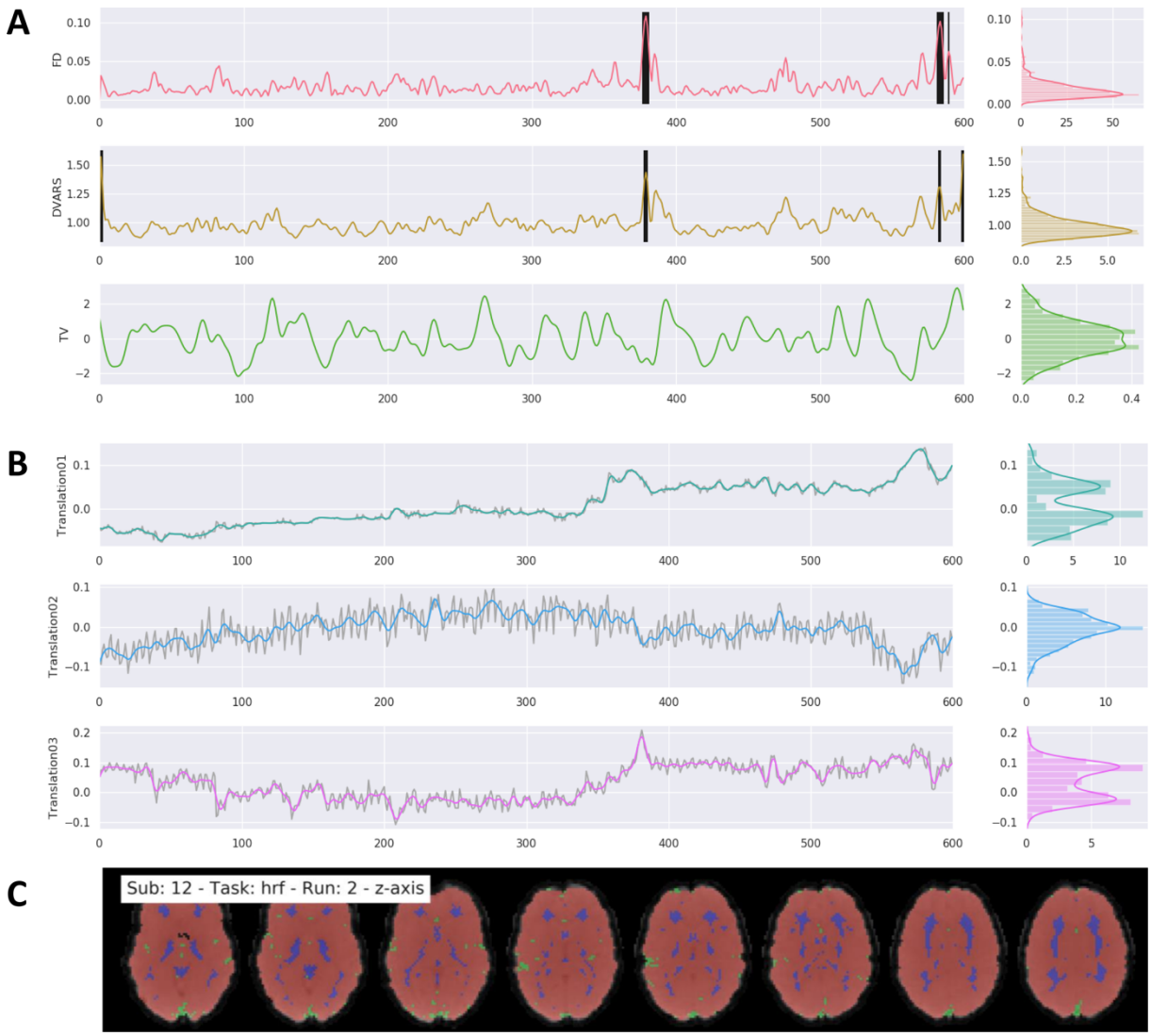


Figure 5: Example of general output figures generated by fMRIflows after executing the functional preprocessing pipeline. The dataset used to generate these figures was recorded with a TR of 600ms and had a total of 600 volumes per run. Preprocessing included a low-pass filter at 0.2 Hz. Distribution plots on the right side of the figures in part A and B represent value frequency in y-direction. **A)** Depiction of the nuisance confounds FD, DVARS and TV. Detected outlier volumes are highlighted with vertical black bars. **B)** Estimation of translation head motion after application of low-pass filtering at 0.2 Hz in color, and before temporal filtering in light gray. **C)** Depiction of brain masks used to compute DVARS (red), and temporal (green) and anatomical (blue) CompCor confounds, overlaid on the mean functional image (grey).

Output generated after executing the 1st-level analysis pipeline

After the execution of the 1st-level analysis, the following files are generated for the univariate analysis: (1) contrasts and statistical map of the specified contrasts, (2) SPM.mat file containing the information relevant for the model, (3) visualization of the design matrix used in the 1st-level model depicting the regressor for the stimuli, motion and confounds, and (4) glass brain plot for each estimated

contrast thresholded at the top 2% of positive and negative values created with AtlasReader (Notter et al., 2019) to provide a general overview of the quality of contrasts. The multivariate analysis part of this notebook creates: (1) one contrast image per condition and session which later can be used as samples for the multivariate analysis, and (2) a label file identifying the condition of each contrast.

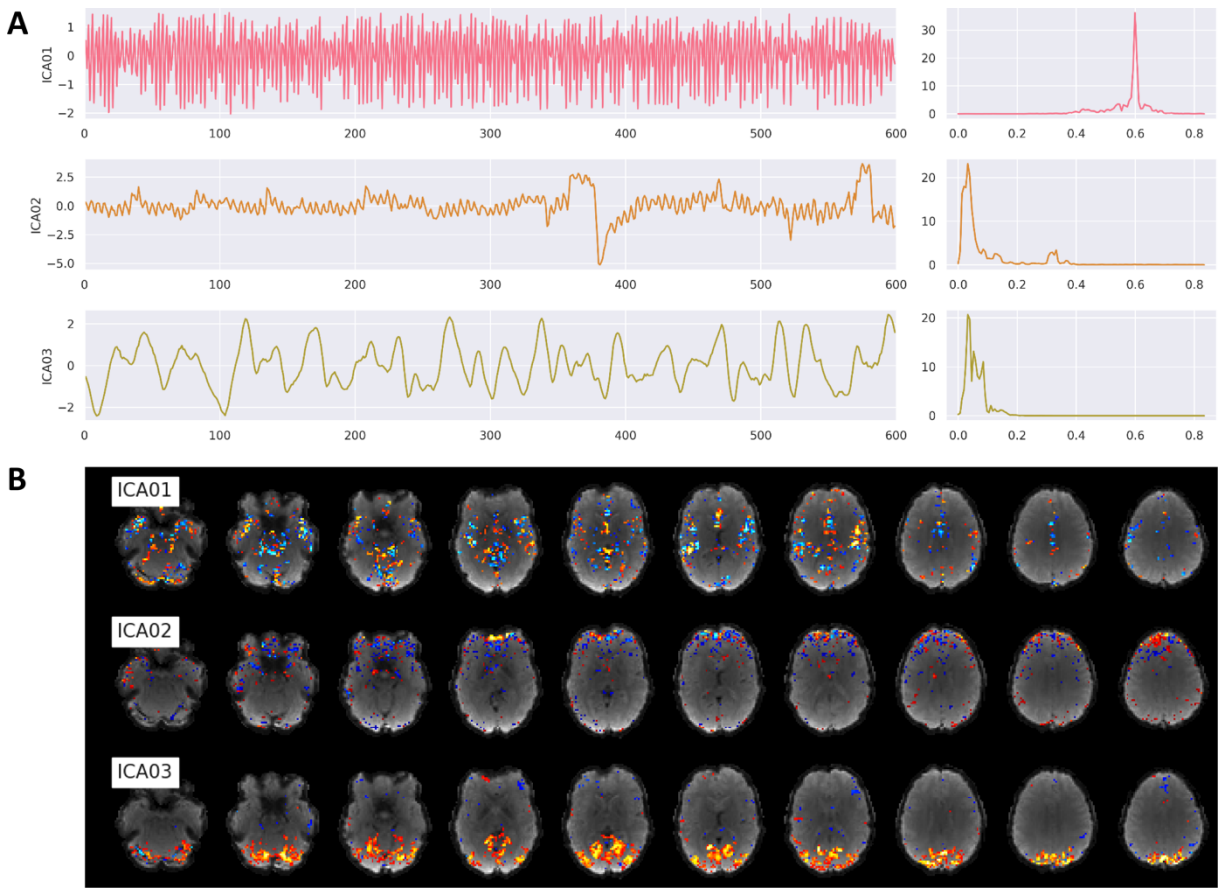


Figure 6: Example of ICA output figures generated by fMRIflows after executing the functional preprocessing pipeline. The dataset used to generate these figures was recorded with a TR of 600ms and had a total of 600 volumes per run. **A)** Correlation between the first three ICA components and the functional image over time (left) and the corresponding power density spectrum with frequency on the x-axis (right). First component most likely depicts respiration at 0.6 Hz, while third component is most likely visual activation induced by the visual stimulation task during data acquisition. B) Correlation strength between a given ICA component and the location in the brain volume for the first three ICA components.

Output generated after executing the 2nd-level analysis pipeline

After the execution of the 2nd-level univariate analysis, the following files are generated, individually for each contrast and spatial and temporal filter that was applied: (1) contrasts and statistical map of one-sample *t*-test contrast, (2) SPM.mat file containing the information relevant for the model, (3) thresholded statistical maps with corresponding AtlasReader outputs (i.e. glass brain plot to provide a

result overview, cross section plot showing each significant cluster individually, informative tables concerning the peak and cluster extent of each cluster).

After the execution of the 2nd-level multivariate analysis, the following files are generated, for each specified comparison individually: (1) subject-specific permutation files needed for correction according to Stelzer et al. (2013), (2) group-average prediction accuracy maps as well as corresponding feature-wise maps representing chance level acquired via bootstrapping approach (Stelzer et al., 2013), (3) group-average prediction accuracy maps after correction for multiple comparisons and (4) thresholded statistical result maps with corresponding AtlasReader outputs (i.e. glass brain plot to provide a result overview, cross section plot showing each significant cluster individually, informative tables concerning the peak and cluster extent of each cluster).

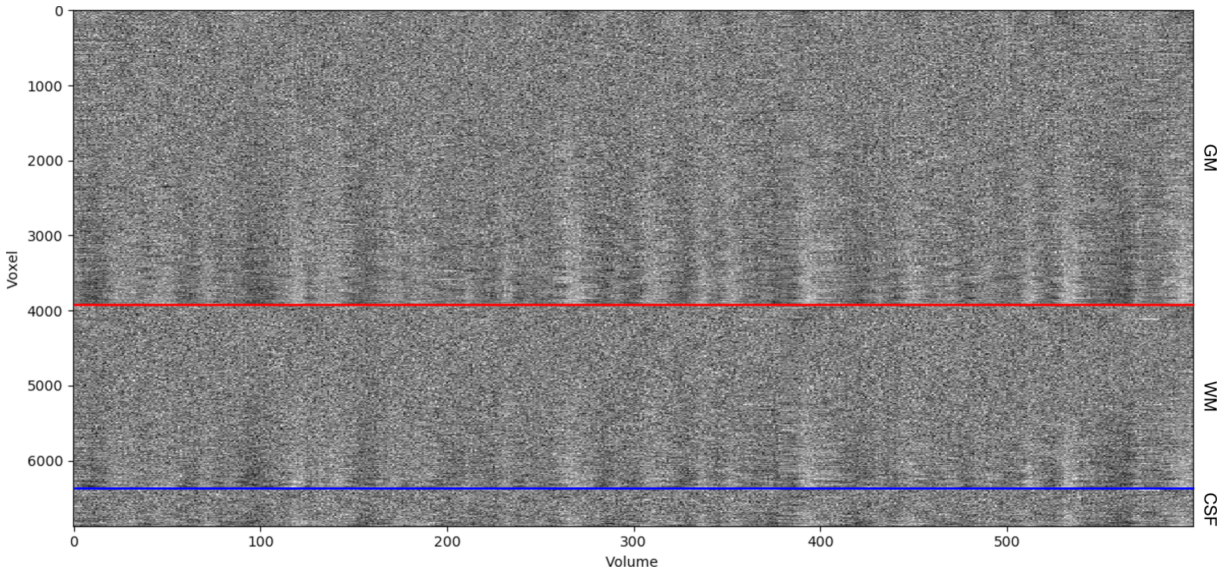


Figure 7: Example of a carpet plot figures generated by fMRIflows after executing the functional preprocessing pipeline. The dataset used to generate these figures was recorded with a TR of 600ms and had a total of 600 volumes per run. This panel shows the signal after preprocessing for every other voxel (y-axis), over time in volumes (x-axis). The panel shows voxels in the gray matter (top part), white matter (between blue and red line) and CSF (bottom section). The data was standardized to the average signal, and ordered within a given region according to the correlation coefficient between a voxel and to the average signal of this region.

2.2.2 Results of phase 1: Proficiency validation

Due to differences in scanner hardware, scan protocols, research requirements and expertise of the person who records the images, fMRI datasets can come in many different shapes and forms. We ran fMRIflows on several datasets to make sure that it is capable of dealing with differences inherent to each of them. In this section, we summarize the main issues we encountered during this process and describe how we tackled each of them.

Image orientation

fMRIflows reorients all anatomical and functional images at the beginning of the preprocessing pipeline to the neurological convention RAS (right, anterior, superior) to prevent failures of coregistration between anatomical and functional images due to orientation mismatches within subjects.

Image extent

Some datasets have unusually large image coverage along the inferior-superior axis, which means that their anatomical images also often contain part of the participant's neck. This can lead to unwanted outcomes in certain neuroimaging routines, as they were not tested for such additional tissue coverage. This is most pronounced in the case of FSL's BET routine, which has difficulty finding the center and extent of the brain, or SPM's segmentation routine that depends on the distribution of the voxel intensities within the whole volume. To prevent these and other unforeseen behaviors, fMRIflows uses FSL's robustfov routine to restrict all anatomical images to the same spatial extent.

Image inhomogeneity

Depending on the scan sequence protocol or the scanner hardware itself, some datasets can contain strong image intensity inhomogeneities, caused by an inhomogeneous bias field during data acquisition. This can have a negative effect on many different neuroimaging routines, most pronounced in brain extraction and image segmentation. To tackle this issue, fMRIflows uses ANTs' N4BiasFieldCorrection routine, which allows the analysis of datasets with even low image quality and strong image inhomogeneity. In the anatomical preprocessing pipeline, inhomogeneity correction is applied to improve the final output image. In the functional preprocessing pipeline, inhomogeneity correction is only applied to improve the estimation and extraction of different tissue types, but does not directly change the values in the final output image.

Brain extraction

Different brain extraction routines were explored to ensure: 1) that the extraction is sufficiently robust to handle different kinds of datasets, 2) that it is neither too conservative nor liberal with the removal of non-brain tissues, and 3) that it has an overall reasonably fast computation time. The best and most consistent results were achieved using SPM's image segmentation routine, followed by a specific thresholding and merging of the GM, WM and CSF probability maps. FSL's BET routine was not robust enough to lead to stable results on all tested datasets. While ANTs' Atropos routine led to comparably good results, we went with SPM because of the much faster computation time.

Image interpolation

For the single-shot spatial interpolation during normalization, we used ANTs and explored NearestNeighbor, BSpline and LanczosWindowedSinc (Lanczos, 1964) interpolation. NearestNeighbor

interpolation led to unnatural looking voxel-to-voxel value transitions. BSpline led in general to good results, but had issues especially with datasets that did not have full brain coverage and introduced some rippling low value fluctuations at the borders of non-zero voxels. LanczosWindowedSinc interpolation led to the best outcome by minimizing the smoothing effects and preventing the introduction of additional confounds reaffirming the observations from fMRIPrep (Esteban et al., 2019).

2.2.3 Results of phase 2: Performance validation

The performance validation of fMRIflows was conducted on three different task-based fMRI datasets, as described in Table 1. The preprocessing of fMRIflows was compared to other neuroimaging processing pipelines such as fMRIPrep, FSL and SPM. We tested fMRIflows' preprocessing pipeline with and without a temporal low-pass filter of 0.2 Hz to better understand performance differences between toolboxes and to stress the importance of adequate temporal filtering when processing fMRI datasets with high temporal resolution.

Estimated spatial smoothness after functional preprocessing

Each preprocessing step that resamples a functional image, such as slice time correction, motion correction, spatial or temporal interpolation has the potential to increase the spatial smoothness in the data. The less smoothness is introduced during preprocessing, the closer the data are to their initial version. We used AFNI's 3dFWHMx to estimate the average spatial smoothness (FWHM) of each functional image after preprocessing to compare the amount of data manipulation that was applied to the raw data (see Figure 8). As this FWHM value depends on the voxel resolution of a given dataset, we normalized it by the volume of the voxel to achieve a common FWHM value per $1mm^3$.

Overall, the estimated spatial smoothness after preprocessing with fMRIflows (without low-pass filter) is comparable to the one with fMRIPrep, while SPM's is in general significantly lower and FSL's is slightly higher. The differences with respect to SPM are probably due to the fewer numbers of resampling steps involved in SPM's preprocessing pipeline. The differences with respect to FSL are probably due to the interpolation method used during image resampling. While the FSL preprocessing pipeline uses the spline interpolation, fMRIflows and fMRIPrep use the LanczosWindowedSinc interpolation, which is known to minimize the smoothing during interpolation. The application of a temporal low-pass filter at 0.2 Hz during fMRIflows' preprocessing leads to a significantly higher spatial smoothness for the TR600 dataset when compared with the other approaches. This effect might also be present for the TR1000 dataset. However, there the difference between the fMRIflows preprocessing with and without low-pass filtering is not significant. This increased spatial smoothness for the approach that uses a low-pass filter makes sense, as the goal of the temporal low-pass filter itself is to smooth the time series values. This temporal smoothing forcibly also increases the spatial smoothness at each individual time point.

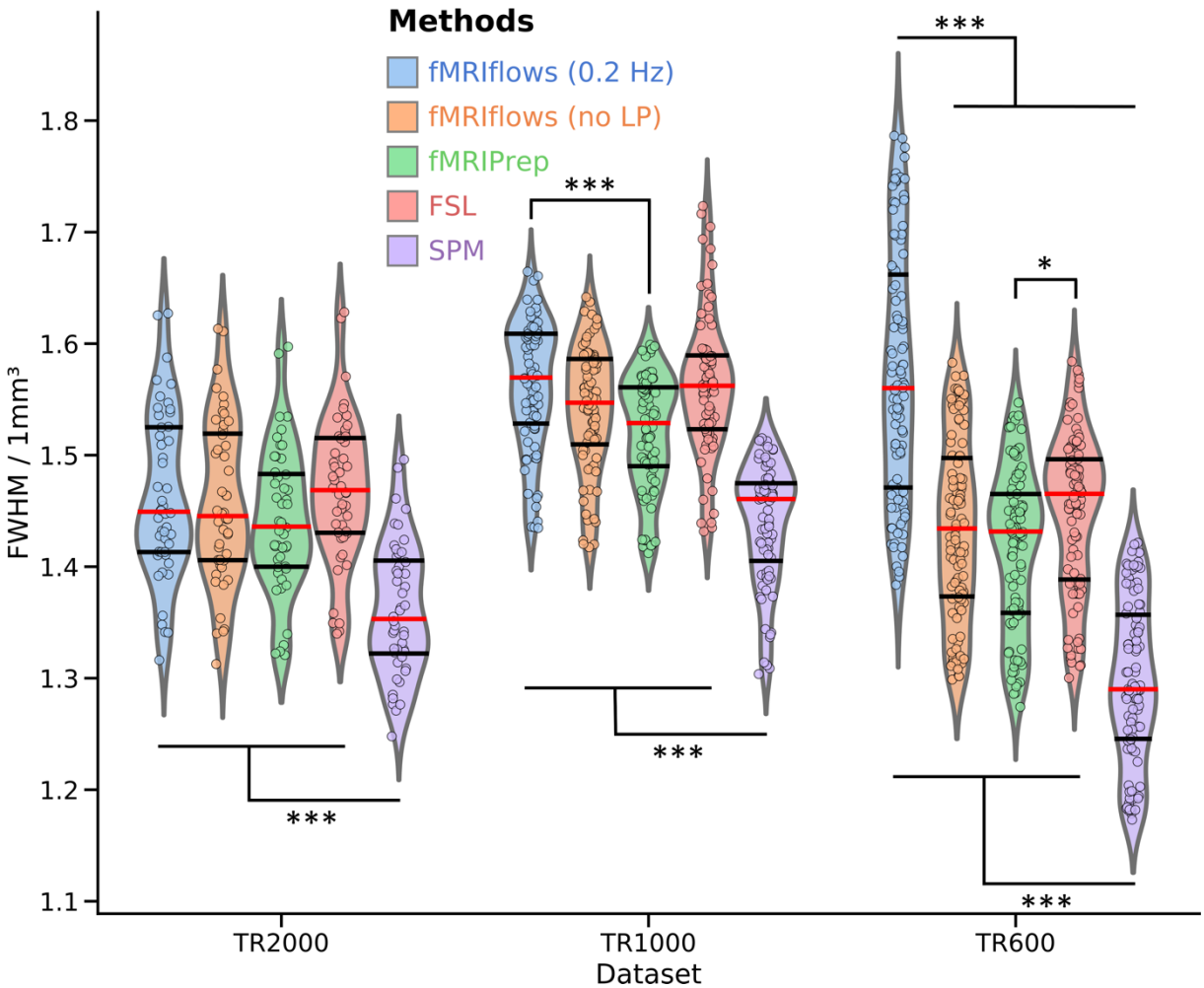


Figure 8: Investigation of estimated spatial smoothness after functional preprocessing of three different datasets, processed with varying approaches. The five different preprocessing approaches fMRIflows with (blue) and without (orange) a low-pass filter at 0.2 Hz, fMRIPrep (green), FSL (red) and SPM (violet) are plotted separately for the dataset TR2000 (left), TR1000 (middle) and TR600 (right). The violin plots indicate the overall distribution of the normalized smoothness estimates of each functional image (depicted in individual dots: TR2000=48 dots, TR1000=80 dots, TR600=102 dots). The red horizontal line represents the median value, while the horizontal black lines indicate the 25 and 75 percentiles of the value distribution respectively. Significant differences between groups are indicated with *: $p<0.05$ and ***: $p<0.001$.

Performance check of spatial normalization

We computed the standard deviation map for each population, based on the temporal average map of each preprocessed functional image, to compare the performance of spatial normalization of the different preprocessing methods on the three different datasets (see Figure 9).

The averaged standard deviation maps after fMRIflows' and fMRIPrep's preprocessing are very similar, which is not surprising as fMRIflows uses the same ANTs normalization routine with very similar parameters. The main difference lies in the fact that fMRIflows applies a brain extraction on the functional

images as well, which is not performed with fMRIPrep. Closer inspection reveals that the variability around the brain outline is slightly increased and more spread after normalization with fMRIflows or FSL, than with fMRIPrep. This effect was already observed in the original fMRIPrep paper, but seemed less pronounced in the current study. SPM performed comparably well, but a direct comparison was not possible as the spatial normalization with SPM is performed using SPM's own tissue probability map template, while the other three methods used the normalization to the ICBM's 2009c brain template. No clear performance differences have been observed between the three datasets.

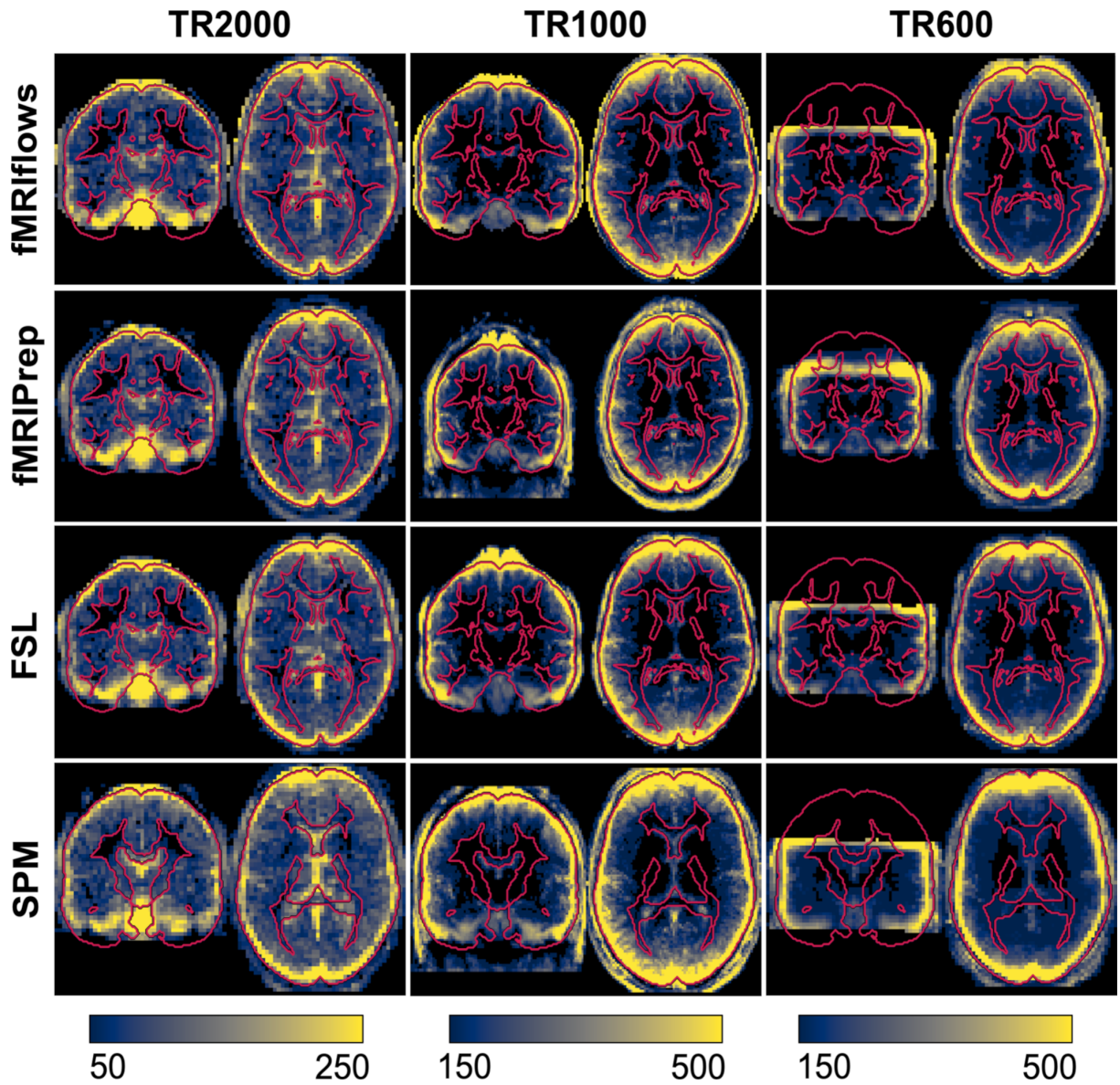


Figure 9: Depiction of standard deviation maps of the temporal averages of three different datasets, after multiple functional preprocessing approaches. Preprocessing was done with fMRIflows (with a temporal low-pass filter at 0.2 Hz; without low-pass filter looks identical), fMRIPrep, FSL and SPM (from top to bottom) separated for the TR2000 (left), TR1000 (middle) and TR600 (right) dataset. Color value represents the standard deviation value over all subjects. Color scale is the same within a dataset and was

set manually to highlight the border effects in gray matter regions. Regions with high inter-subject variability are shown in yellow, while regions with low inter-subject variability are shown in blue. Outline of the brain and subcortical white matter regions is delineated in red and is based on the ICBM 2009c brain template, except for the analysis with SPM where it is based on SPM's tissue probability map template.

Temporal signal-to-noise ratio (tSNR) after preprocessing

We computed the voxel-wise temporal SNR according to Smith et al. (2013) to assess the amount of informative signal contained in the data after preprocessing. This measurement serves as a rough estimate to compare different preprocessing methods, but did not allow a direct comparison between datasets, as the tSNR value is a relative measurement that depends highly on the paradigm presented, the initial spatial and temporal resolution of the functional images, as well as the MRI scan sequence specific parameters such as acceleration factors (Smith et al., 2013). Using Nipype's TSNR routine, we first removed 2nd-degree polynomial drifts in each functional image, and estimated tSNR maps by computing each voxel's temporal mean, dividing it by its temporal standard deviation, and multiplying it by the square root of the number of time points recorded in a given run. By averaging the tSNR maps over the population, we get a general tSNR map per preprocessing method for each dataset (see Figure 10).

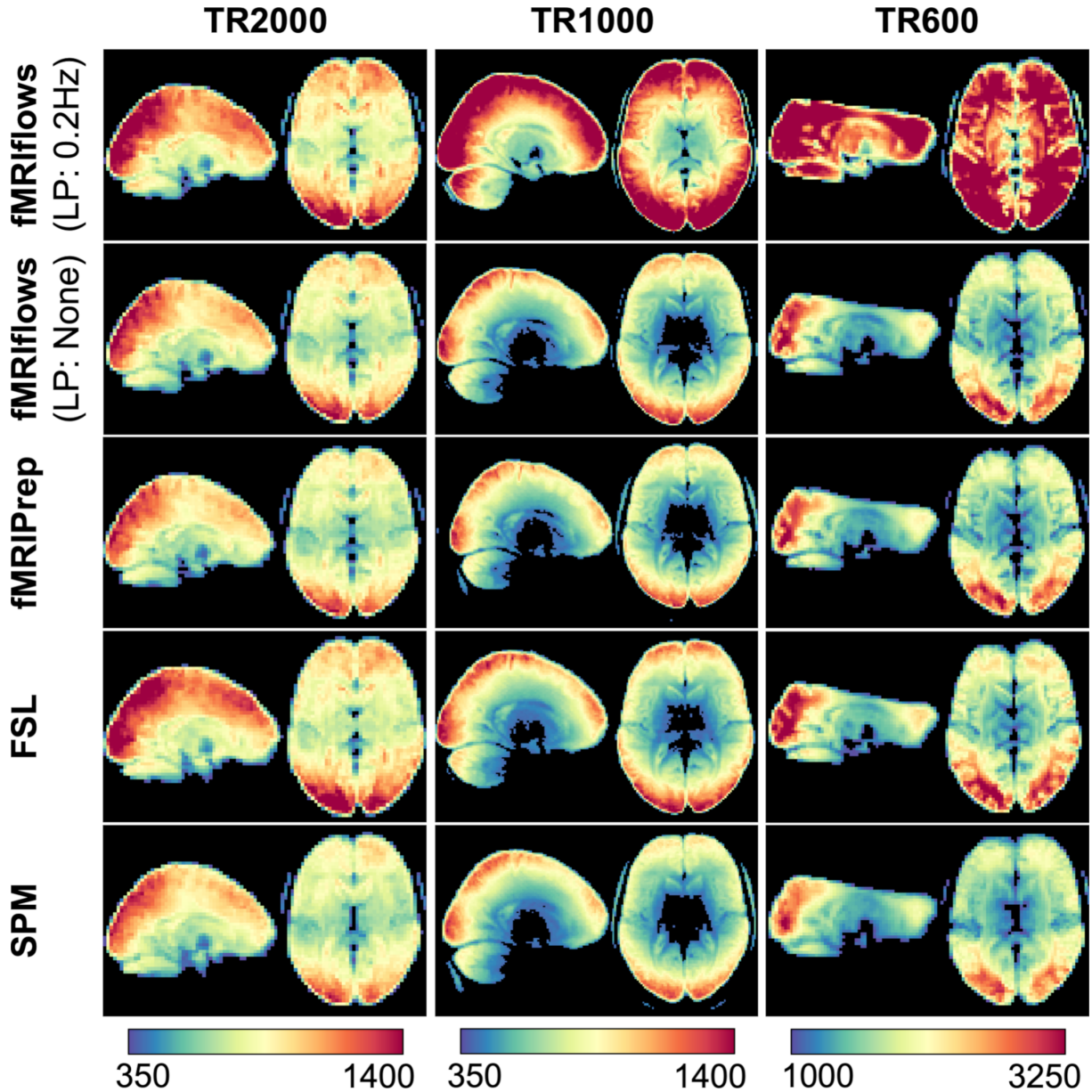


Figure 10: Depiction of temporal signal-to-noise ratio maps of three different datasets, after multiple functional preprocessing approaches. Preprocessing was done with fMRIflows (with and without a temporal low-pass filter at 0.2 Hz), fMRIPrep, FSL and SPM (from top to bottom) separated for the TR2000 (left), TR1000 (middle) and TR600 (right) dataset. Color value represents the tSNR value as computed with the Nipype routine TSNR. Color scale was set manually and differs between datasets, but is held constant between different preprocessing methods.

In general, preprocessing with fMRIflows without temporal low-pass filter led to similar average tSNR maps as preprocessing with fMRIPrep. Overall, preprocessing with FSL led to slightly increased average tSNR values, while preprocessing with SPM led to slightly decreased average tSNR maps. The additional application of a low-pass filter at 0.2 Hz in all three datasets led to increased tSNR values after preprocessing with fMRIflows. This effect was more pronounced for higher temporal resolution (as in

Dataset TR1000 and TR600). The color scales in Figure 10 were set manually so that the fMRIflows (without low-pass filter) approach shows comparable intensities for the three datasets.

Performance check after 1st-level analysis

To investigate the effect of the different preprocessing methods on the 1st-level analysis, we carried out within-subject statistical analysis using Nistats. The activation maps were estimated using a general linear model (GLM). The GLM included a constant term, the stimuli regressors convolved with a double-gamma canonical hemodynamic response function, six motion parameters (three translation and three rotation), and a high pass filter at 100Hz, represented by a set of cosine functions, and no temporal derivatives. The input data were smoothed using a kernel with a FWHM of 6mm, using a Nilearn routine. The analysis pipelines between the preprocessing methods and datasets were kept as identical as possible, and differed only in the number of time points contained in the dataset and the estimated motion parameters. The statistical map for each participant was binarized at $z = 3.09$, which corresponds to a one-sided test value of $p < 0.001$. The population average of these maps is shown in Figure 11.

The results show that the thresholded activation count maps between the fMRIflows approach without a low-pass filter, fMRIPrep, FSL and SPM do not differ too much between each other, for all three datasets. In contrast to the other preprocessing methods, however, the preprocessing with fMRIflows with a low-pass filter at 0.2 Hz drastically increased the size and fraction value of the thresholded activation count maps, for the datasets TR1000 and TR600. Thus, appropriate temporal filtering increased the statistics for datasets with higher temporal resolution remarkably. For a more detailed comparison between all the toolboxes, see Supplementary Note 7.

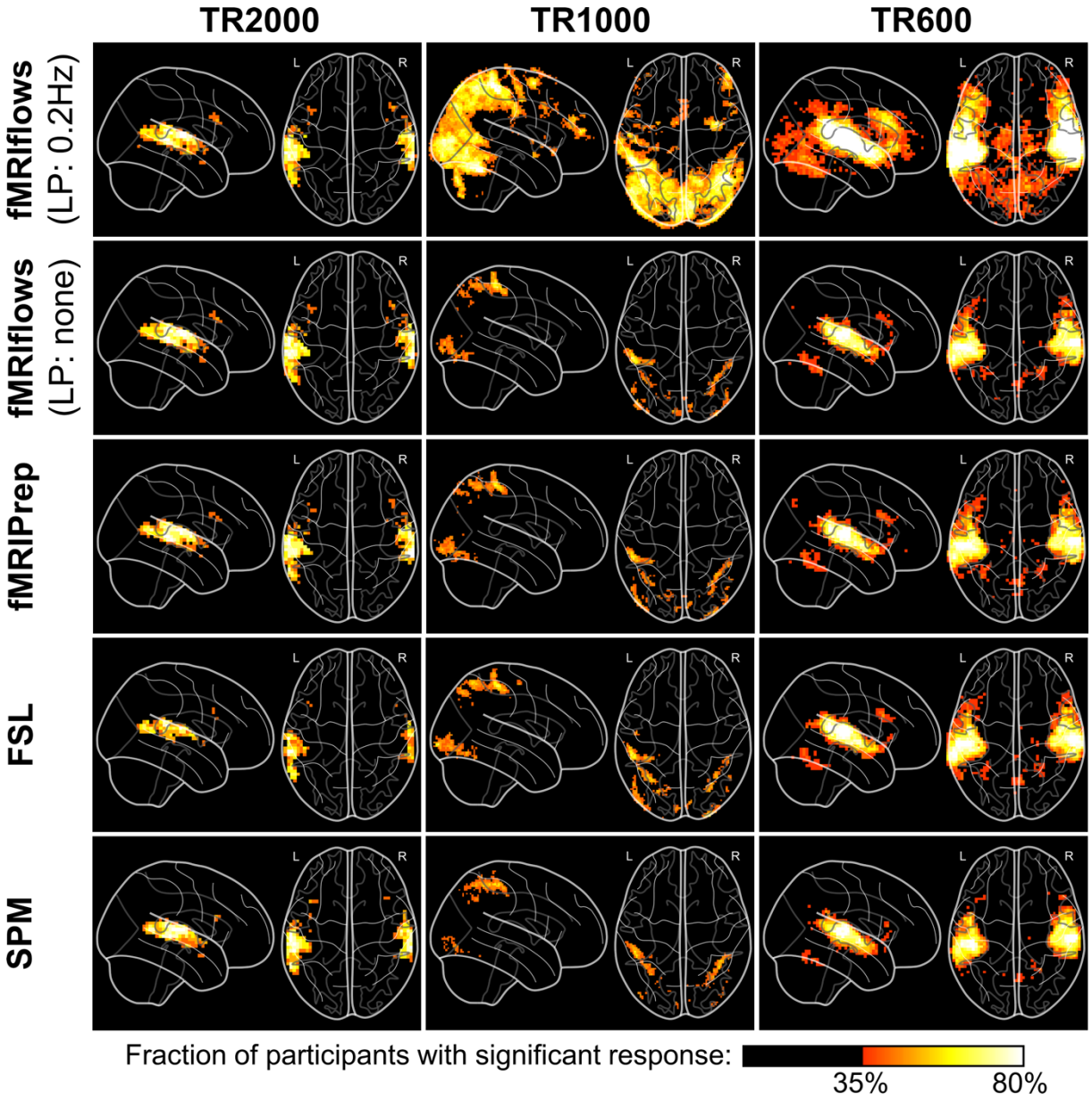


Figure 11: Depiction of binarized 1st-level activation count maps, thresholded at *p*<0.001, after multiple functional preprocessing approaches. Preprocessing was done with fMRIflows (with and without a temporal low-pass filter at 0.2 Hz), fMRIPrep, FSL and SPM (from top to bottom) separated for the TR2000 (left), TR1000 (middle) and TR600 (right) dataset. Activation count maps were normalized to the ICBM 2009c brain template. Color code represents the fraction of participants that show significant activation above a *p*-value threshold at 0.001 and corrected for false positive rate (FPR).

Performance check after 2nd-level analysis

To investigate the effect of the different preprocessing methods on the 2nd-level analysis, we carried out between-subject statistical analysis using Nistats and computed one-sample t-test for each preprocessing method and dataset. The unthresholded group-level T-statistic maps of each analysis was

then compared to each other on a voxel-by-voxel level using Bland-Altman 2D histograms (Bowring et al., 2018), see Figure 12.

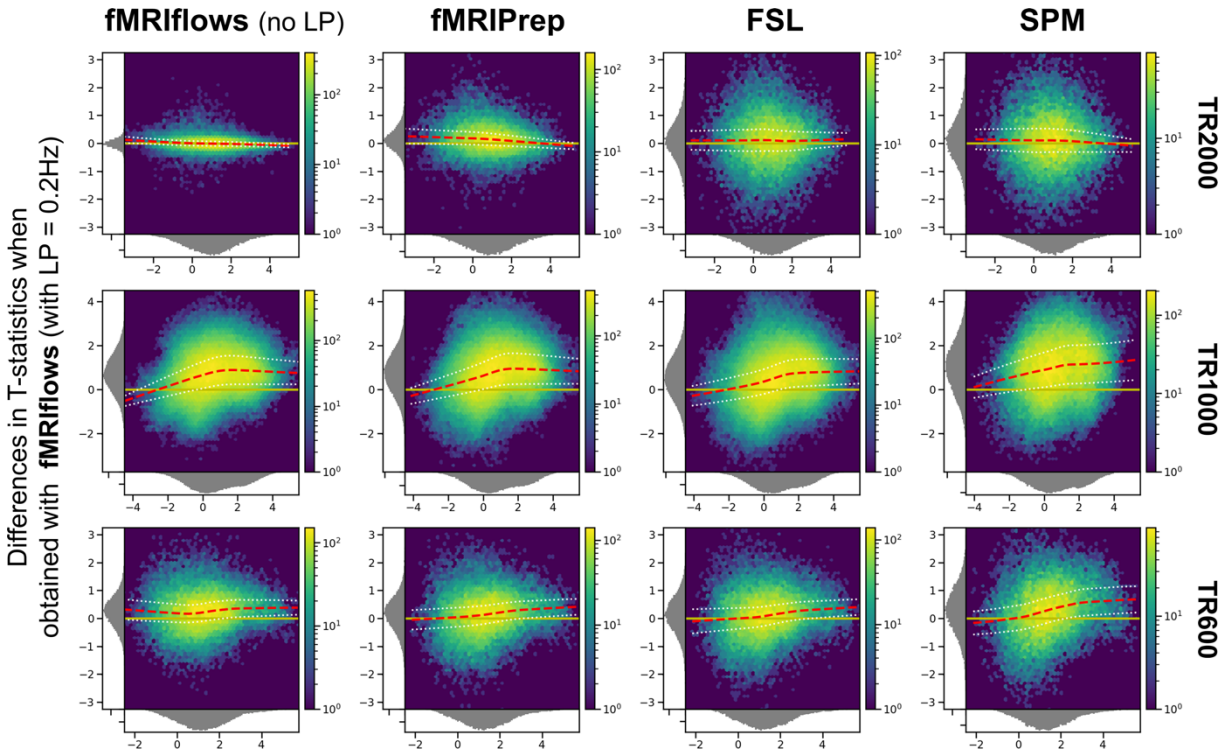


Figure 12: Bland-Altman 2D histograms of three different datasets, comparing unthresholded group-level T-statistic maps between multiple processing approaches. Datasets TR2000 (top), TR1000 (middle) and TR600 (bottom) were used for the comparison. Density plots show the relationship between average T-statistic value (horizontal) and difference of T-statistic values (vertical) at corresponding voxels for different pairwise combinations of toolboxes. The difference of T-statistics was always computed in contrast to a preprocessing with fMRIflows using a low-pass filter at 0.2 Hz, while the average T-statistics in horizontal direction investigated the preprocessing with (from left to right) fMRIflows without a low-pass filter, fMRIPrep, FSL and SPM. Distribution plots next to x- and y- axis depict occurrence of a given value in this domain. Color code within the figure indicates the number of voxels at this given overlap, from a few (blue) to many (yellow). Yellow horizontal line at zero indicates no value differences between corresponding voxels. Red dashed line depicts horizontal density average.

The results shown in Figure 12 indicate no pronounced differences between the preprocessing with fMRIflows with a low-pass filter at 0.2 Hz and the other four approaches for the analysis of the TR2000 dataset. An increased variability in the y-direction indicated a decrease in voxel-to-voxel correspondence, which might be explained by different spatial normalization implementations. The fact that the average horizontal density value (red dashed line) is close to the zero line (in yellow) indicated that the different preprocessing methods led to comparable group-level results with the TR2000 dataset. The Bland-Altmann plots for the TR1000 and TR600 datasets showed a clear increase of t-statistic when the preprocessing was

done with fMRIflows with a low-pass filter at 0.2 Hz, compared to any other method. This effect was stronger for higher t-values. For a more detailed comparison between all the toolboxes, see Supplementary Note 8.

2.3 Discussion

fMRIflows is a fully automatic fMRI analysis pipeline, which can perform state-of-the-art preprocessing, 1st-level and 2nd-level univariate analyses, and multivariate analysis. The goal of such an autonomous approach is to improve objectiveness of the analysis, maximize transparency, facilitate ease of use, and provide newest analysis approaches to every researcher, including users outside the field of neuroimaging. While the predefined analysis pipelines help to reduce the number of error-prone manual interventions to a minimum, it also has the advantage of decreasing the number of analytical degrees of freedom available to a user to its minimum (Carp, 2012). This constraint in flexibility is important as it helps to control the variability in data processing and analysis (Botvinik-Nezer et al., 2020). fMRIPrep showed a clear need for such analysis-agnostic approaches and was therefore chosen to provide much of the groundwork for fMRIflows.

In comparison with other neuroimaging software like fMRIPrep, FSL and SPM, fMRIflows achieved comparable or improved results in (1) SNR after preprocessing, (2) within-subject t-statistics, and (3) between-subject t-statistics. These results were more obvious in the context of datasets that had a temporal resolution equal to or below 1000ms, and if a low-pass filter at 0.2 Hz was applied. The latter might also influence the overall outcome of the other software packages, but so far fMRIflows is the only such neuroimaging software that performs orthogonal filtering between the motion correction and temporal filtering, as proposed by Lindquist et al. (2019).

Being an open-source project, shared via GitHub, facilitates the transparency in the development of fMRIflows. Users will be able to inspect the complete history of the changes and have access to all discussion connected to the software. Code adaptations and additional support to new usage will be proposed by the user community, which will make the adaptation to newest standards easy and straightforward. In addition to the version-controlled system used on GitHub, a continuous integration scheme with CircleCi will ensure continuous functionality.

fMRIflows also improved the overall computation time needed to perform preprocessing and 1st and 2nd-level analysis. Indeed, Nipype provides a parallel execution feature of processing pipelines, which is not yet possible with FSL or SPM. fMRIPrep uses the same boost of parallelism, but is overall much slower if the default execution of FreeSurfer's recon-all routine is performed. However, fMRIflows does not yet support parallel computation via a job scheduler on a computation cluster, which is currently possible with fMRIPrep.

The inclusion of many informative visual reports allows a direct quality control and verification of the performed processing steps, as fMRIflows' outputs provide a general quality assessment even though it is not as detailed and rigorous as MRIQC (Esteban et al., 2017). In contrast to other software packages, fMRIflows uses an adapted visualization of the carpet plot proposed by Power (2017) to highlight underlying temporal structure and voxel-to-voxel correlations within different brain tissue regions and/or throughout the brain. Such approaches help to observe general signal trends and sudden abrupt signal

changes throughout the brain, but the exact implications of these modified carpet plots need to be further investigated.

Results of fMRIflows' validation phase 1 suggests that the software is capable of analyzing different types of datasets, independently of the extent of head coverage, original image orientation, spatial or temporal resolution. By increasing the user base and testing fMRIflows on many more datasets, new adaptations might be required and hidden bugs could emerge. Users can observe any changes done to the software in the future directly on GitHub and are encouraged to state any questions or comments in connection with the software on the community driven neuroinformatics forum NeuroStars (https://neurostars.org).

Further development of the software will involve (1) moving away from an SPM dependency for the 1st and 2nd-level modeling, (2) using the more flexible FitLins toolbox (https://github.com/poldracklab/fitlins), and (3) implementing an fMRIflows BIDS-App to further improve the toolbox's accessibility.

Acknowledgments

We thank the creator of fMRIPrep for providing an excellent starting point and inspiration in the development of fMRIflows. We also thank the developers of PyMVPA, Nilearn, Nibabel and Nistats for bringing the neuroimaging domain into the Python universe, and the developer of Nipype and BIDS for creating a clear framework to execute processing pipelines, as well as the whole neuroimaging open source and science community with its numerous contributors. The authors have no conflicts of interest to disclose. CRediT author statement: M.P.N.: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing, Visualization, Project administration; P.H.: Conceptualization, Methodology, Software, Validation, Writing; S.D.C.: Methodology, Supervision, Writing - Review & Editing, Validation; O.F.G.: Writing- Reviewing and Editing, Validation; A.I.I.: Writing- Reviewing and Editing, Validation; M.M.M.: Supervision, Writing- Reviewing and Editing, Funding acquisition

Funding

This work was supported by the Swiss National Science Foundation (grant numbers 320030-169206 to M.M.M.) and research funds from the radiology service at the University Hospital in Lausanne (CHUV). This work was supported by the Centre d'Imagerie BioMédicale (CIBM) of the University of Lausanne (UNIL) and the Swiss Federal Institute of Technology Lausanne (EPFL). P.H. was supported in parts by funding from the Canada First Research Excellence Fund, awarded to McGill University for the Healthy Brains for Healthy Lives initiative, the National Institutes of Health (NIH) NIH-NIBIB P41 EB019936 (ReproNim), the National Institute Of Mental Health of the NIH under Award Number R01MH096906, as well as by a research scholar award from Brain Canada, in partnership with Health Canada, for the Canadian Open Neuroscience Platform initiative.

2.4 Supplementary Material

Supplementary Note 1: Content examples of parameter specification files

Content example of file fmriflows_spec_preproc.json

```
{
    "subject_list_anat": ["01", "02", "03", "04", "05"],
    "session_list_anat": [],
    "T1w_id": "T1w",
    "res_norm": [1.0, 1.0, 1.0],
    "norm_accuracy": "precise",
    "subject_list_func": ["01", "02", "03", "04", "05"],
    "session_list_func": [],
    "task_list": ["multi"],
    "run_list": [1, 2, 3],
    "ref_timepoint": 500,
    "res_func": 2.0,
    "filters_spatial": [["LP", 6.0]],
    "filters_temporal": [[null, 100.0 ], [5.0, 100.0]],
    "n_compcor_confounds": 5,
    "outlier_thresholds": [3.27, 3.27, 3.27, null, null, null],
    "n_independent_components": 10,
    "n_parallel_jobs": 7
}
```

Content example of file fmriflows_spec_analysis.json


```
{
    "tasks": {
        "multi": {
            "condition_names": ["cond_01", "cond_02", "cond_03", "cond_04"],
            "contrasts": [
                ["cond_01", [1.0, 0.0, 0.0, 0.0], "T"],
                ["cond_02", [0.0, 1.0, 0.0, 0.0], "T"],
                ["cond_03", [0.0, 0.0, 1.0, 0.0], "T"],
                ["cond_04", [0.0, 0.0, 0.0, 1.0], "T"],
                ["cond_01 > cond_02", [1.0, -1.0, 0.0, 0.0], "T"]
            ]
        }
    },
    "subject_list": ["01", "02", "03", "04", "05"],
    "session_list": [],
    "filters_spatial": [["LP", 6.0]],
    "filters_temporal": [[null, 100.0], [5.0, 100.0]],
    "nuisance_regressors": ["Rotation", "Translation", "FD", "DVARS", "TV"],
    "use_outliers": true,
    "model_serial_correlations": "AR(1)",
    "model_bases": {"hrf": {"derivs": [0, 0]}},
    "estimation_method": {"Classical": 1},
    "normalize": true,
    "norm_res": [1, 1, 1],
    "con_per_run": true,
    "norm_res_multi": [3.0, 3.0, 3.0],
    "analysis_postfix": "",
    "gm_mask_thr": 0.1,
    "height_threshold": 0.001,
    "use_fwe_correction": false,
    "extent_threshold": 5,
    "use_topo_fdr": true,
    "extent_fdr_p_threshold": 0.05,
    "atlasreader_names": "default",
    "atlasreader_prob_thresh": 5,
    "n_parallel_jobs": 7
}
```

Content example of file fmriflows_spec_multivariate.json

```
{
    "subject_list": [ "01", "02", "03", "04", "05"],
    "session_list": [],
    "filters_spatial": [["LP", 6.0]],
    "filters_temporal": [[null, 100.0], [5.0, 100.0]],
    "multivariate_postfix": "",
    "clf_names": ["LinearNuSVMC"],
    "sphere_radius": 3,
    "sphere_steps": 3,
    "n_chunks": 6,
    "tasks": {
        "hrf": [
            [["cond_01", "cond_02"],["cond_01", "cond_02"]],
            [["cond_01", "cond_02"],["cond_03", "cond_04"]]
        ]
    },
    "n_perm": 100,
    "n_bootstrap": 100000,
    "block_size": 1000,
    "threshold": 0.001,
    "multicomp_correction": "fdr_bh",
    "fwe_rate": 0.05,
    "atlasreader_names": "default",
    "atlasreader_prob_thresh": 5,
    "n_parallel_jobs": 7
}
```

Supplementary Note 2: Description of anatomical preprocessing pipeline steps

Image reorientation: To make sure that all images throughout the processing have the same orientation, images are first reoriented with Nipype according to the neurological convention RAS (right, anterior, superior).

Image cropping: To make sure that the focus in the anatomical image is on the brain, we use FSL's robustfov function to remove irrelevant portions of the neck. This is particularly relevant for the later brain extraction step, and helps to ensure that the segmentation algorithm focuses on the brain and not on additional body sections.

Image inhomogeneity correction: To correct for intensity non-uniformities caused by the inhomogeneity of the bias field during data acquisition, we use ANTs' N4BiasFieldCorrection algorithm. This step improves the quality of the following image segmentation and is crucial for anatomical images of lower image quality, as they would otherwise fail during the image segmentation.

Anatomy segmentation: The image segmentation uses SPM12's standard image segmentation and provides probability maps for five tissue segments: gray matter (GM), white matter (WM), cerebrospinal fluid (CSF), skull and head.

Brain extraction: The GM, WM and CSF probability maps are combined using Nilearn to create a binary mask which is used to extract the brain. We chose this approach over others as it proved to be more robust and provided the best balance between restriction and inclusion than other algorithms, especially in the context of low image quality.

Spatial normalization: As a final step, the extracted brain image is spatially normalized to the ICBM 152 Nonlinear Asymmetrical template version 2009c (Fonov et al., 2011) with ANTs' antsRegistration algorithm, using nonlinear image registration with a b-spline interpolation.

Supplementary Note 3: Description of functional preprocessing pipeline steps

Image reorientation: As a first preprocessing step, functional images are reoriented to the neurological convention RAS, using Nipype, to make sure that they have the same orientation as the anatomically preprocessed images.

Non-steady-state detection: Afterwards, the first few volumes of each functional run are investigated for non-steady-state volumes using Nipype. If non-steady-state volumes are detected, they are removed before the motion correction is applied.

Creation of brain masks: To remove unwanted tissue from functional images a three-step approach was chosen. First, the mean image of the functional run is corrected for intensity inhomogeneity using ANTs' N4BiasFieldCorrection algorithm. Second, FSL's BET algorithm is applied to create a binary brain mask. Third, the binary brain mask is dilated by two iterations and holes are filled. This procedure has proven to be optimal in removing non-brain tissue in almost all cases that it was tested on, while ensuring that all brain tissue types are included within the mask. This brain mask is then applied to the functional images before the motion correction step, to make sure that the estimation of the motion parameter is in relation to the brain and not the whole head. The mask is additionally used to restrict the coregistration of the functional images to the anatomical images to the brain tissue and not to the whole head. However, this binary mask is not used to mask the functional images at any time.

Slice-timing correction: If specified, slice time correction (Sladky et al., 2011) is performed on the reoriented functional images using SPM, according to the slice onset parameters specified in the BIDS information file and the reference time point mentioned in the fMRIflows JSON specification file. Slice-time correction is applied after the estimation of the head motion as recommended by (Power, Plitt, Kundu, Bandettini, & Martin, 2017).

Head motion estimation: Estimation of the motion parameters is performed using FSL's MCFLIRT algorithm, on the reoriented functional images, after non-steady-states volumes are removed and the brain is masked with the binary mask computed during the previous step. If the user specified a low-pass filter, an additional step is included in the estimation of the motion parameters. This step takes the estimated motion parameters (three rotation and three translation) and applies a Butterworth (Stephen Butterworth, 1930) low-pass filter to each of the six components individually. This step is crucial to guarantee that the motion correction and the temporal filter are orthogonal to each other. Otherwise, previously filtered confounds might be reintroduced with a later step (Lindquist et al., 2019). We are using custom-written Python code to perform this step, using routines from Nilearn, output files from FSL's MCFLIRT routine and FSL's avscale routine. To our knowledge, our implementation is the first openly available routine that transforms the affine motion parameters for each volume (here applying a low-pass filter) before they are applied to the functional images.

Intra-subject registration: The coregistration of the functional image to the anatomical image is based on FSL's FEAT pipeline and fMRIPrep and uses a two-step co-registration. Both steps use FSL's FLIRT algorithm. The first step uses the anatomical image to pre-align the mean image from the estimation

of the motion parameters, followed by the second coregistration step where the white matter probabilistic image computed with SPM's segmentation routine is used together with the anatomical image in a boundary-based registration (BBR) approach. The first step uses six degrees of freedom (three rotations and three translations), while the second step uses nine degrees of freedom, adding three scaling degrees. The addition of these scaling degrees was copied from fMRIPrep's approach (Esteban et al., 2019) and allows the image to be stretched in the direction of the recording. For example, functional images that are recorded in the A-P axis are often squeezed a bit in this direction. Keep in mind that this scaling/stretching is only used for an optimal image coregistration, so that the functional images overlay optimally with the white matter boundaries.

Spatial interpolation: Once we have the slice-time corrected functional images, computed the correct motion parameters that we want to apply, have the coregistration matrix between the functional and anatomical images, and have the transformation matrix to normalize the anatomical image onto the ICBM 2009c template brain, we can go to the next step and apply all spatial interpolations in a single-shot. The spatial normalization is optional and can also be applied in the later 1st-level notebook. But we recommend doing all of those spatial transformations at once to keep the number of spatial interpolations as low as possible. First we use C3 to combine all the different transformation matrices and apply them to each volume individually with ANTs' ApplyTransforms routine, using the LanczosWindowedSinc interpolation (Lanczos, 1964). We chose this interpolation over others, as it minimizes the smoothing effect of the interpolation and creates the least artifacts outside the brain volume.

Creation and application of warped brain masks: Once the functional images are spatially transformed either into each subject's structural space using a user-defined voxel resolution, or directly normalized into template space, also using a user-specified voxel resolution, two masks are created to remove irrelevant voxels outside of the brain, such as skull, eyes and head, as well as voxels that do not have values in most volumes. Those voxels are mostly seen in functional images that only cover part of the brain (slabs) and are introduced through motion at top or bottom slices of the slab. The first mask is later used to mask the functional image before the application of the temporal filters, while the second mask (confound brain mask) is applied before the extraction of confound signals. For this reason, the first mask should be sensitive enough to keep all voxels within the brain, while the second mask should be specific enough to only keep brain voxels, so that extracted confound curves are only based on nuisance sources within the brain, i.e. the region that we want to clean. The starting point of both masks is again the binary brain mask. For this we first compute the functional mean image using Nilearn, second correct for intensity non-uniformities using ANTs N4BiasFieldCorrection routine and third apply FSL's BET routine to create a binary brain mask. This mask is then dilated by two iteration steps, holes are filled up and the mask is then applied to the spatially transformed functional image. The steps to create the first brain mask are as follows: the initial brain mask is first dilated by two voxels, holes are filled up and any voxels that have no signal in more than 1% of volumes are removed. The second brain mask is created as follows: the

initial brain mask is first dilated by one voxel, holes are filled, the binary image is again eroded by two iteration steps and afterwards, voxels with no signal in more than 5% of volumes are removed.

Temporal filtering: The next preprocessing step applies temporal filters if they were specified. The user can decide if they want to apply low-pass, high-pass or band-pass filters, or none. The temporal filtering is performed with AFNI's 3dBandpass routine. In this step we also apply the first mask from the previous step to remove voxels that are clearly non-brain tissues. After temporal filtering of the data, the image intensity is normalized in such a way that the white matter distribution peak throughout the functional image is at a value of 10'000.

Spatial filtering: The final preprocessing step applies a spatial filter to the functional images, if the user specified this. Contrary to other toolboxes, our software allows the application of spatial low-pass, high-pass and band-pass filters. The last one can be especially useful for the preprocessing of data that is later used in a multivariate analysis. Sengupta, Pollmann, & Hanke (2018) have shown that the correct application of a band-pass filter can drastically improve the prediction accuracy in a multivariate approach. Independently of whether or not a spatial filter was applied, functional images are checked for absolute values above 30'000. If this is the case, images are rescaled to have a maximum absolute value at 30'000. After this, images are stored in integer16 data format to reduce their footprint on the database.

Supplementary Note 4: Description of 1st-level analysis pipeline steps

Data collection: Preprocessed functional images and model relevant parameters are collected and prepared for the 1st-level analysis.

Model Specification and estimation: Model relevant parameters are combined to create the design matrix needed for the general linear model (GLM) analysis.

Univariate contrast estimation: 1st-level contrasts are computed for each subject individually, according to the input parameters specified in the fMRIflows JSON parameter file.

Optional multivariate contrast estimation: To create multiple beta contrasts that then can be used for multivariate analysis, fMRIflows computes one beta contrast per condition per run. These beta contrasts are based on the same design matrix as the one for the univariate analysis. This step also creates a CSV-file containing a list of condition identifiers which can be used in the multivariate analysis to label the contrast maps.

Optional spatial normalization: The user can spatially normalize the estimated contrasts, if they have not been normalized during functional preprocessing. Contrasts need to be normalized, should the user want to use them in a 2nd-level analysis.

Supplementary Note 5: Description of 2nd-level univariate analysis pipeline steps

Model Specification and estimation: See description in Supplementary Note 4.

Univariate contrast estimation: See description in Supplementary Note 4.

Topological thresholding: Once the 2nd-level contrast is estimated, fMRIflows uses Nipype's Thresholding routine to apply first, a voxel-wise threshold, followed by a cluster-wise topological False Discovery Rate (FDR) correction. The user can decide which parameters to use and if topological thresholding should be applied or not.

Supplementary Note 6: Description of 2nd-level multivariate analysis pipeline steps

Data preparation: To prepare the data for the multivariate pattern analysis (MVPA) with PyMVPA the β-maps from the first level analysis are normalized by voxel-wise z-scoring them. A binary mask is then created to only include voxels in the searchlight analysis that have a value in at least one of the β-maps. This binary mask is then dilated by two voxels and eroded by one to make sure that the mask does not include single voxel holes. The z-scored and masked images are then saved together with a list of corresponding labels in a PyMVPA conform dataset.

Searchlight classification: The searchlight classification is performed for each subject individually and is based on the beta contrasts created during the 1st-level analysis. The user can specify which binary classification identifiers to use for the training and testing of the classifier. This approach allows the user to look for patterns that distinguish two classes (i.e. when the two classes are the same in training and testing) or to look for patterns that are identifying class differences (i.e. when the two classes are different during training and testing). The second approach allows the investigation of recurrent patterns, even if the stimuli are not the same. For example, training on the differences between visual stimuli of cats and dogs and testing on auditory stimuli of cats and dogs would reveal regions with distinct brain patterns for cats and dogs, independent of the modality of the stimuli. Natively, fMRIflows supports the following classifiers: C- and Nu-SVM classifiers with a linear or radial basis function kernel, Sparse Multinomial Logistic Regression (SMLR) classifier, Gaussian Naive Bayes classifier and k-Nearest-Neighbor classifier. The addition of any other classifiers from PyMVPA or Scikit-Learn is straightforward and can be done very quickly, if needed. Searchlight specific parameters, such as radius of the sphere and step size can be specified by the user. For each sphere, an *N*-fold leave-one-out cross-validation is performed, where *N* is set by the user, but usually represents the number of runs per subject. The result is one accuracy value per sphere. To account for holes caused by a searchlight step size bigger than one, results between different searchlight spheres are aggregated so that each voxel in the brain mask represents the average accuracy of all searchlight spheres that contained this voxel in their classification. This approach has the additional beneficial effect of increasing the SNR by smoothing the results before the group analysis.

Group analysis using T-test: The application of a simple T-test to test the classification accuracies against chance level is not recommended (Stelzer et al., 2013), and users should choose the appropriate group analysis as proposed by Stelzer et al. (2013). Nonetheless, fMRIflows contains this approach, as it can give some preliminary insights into the group results, using extremely reduced computation time.

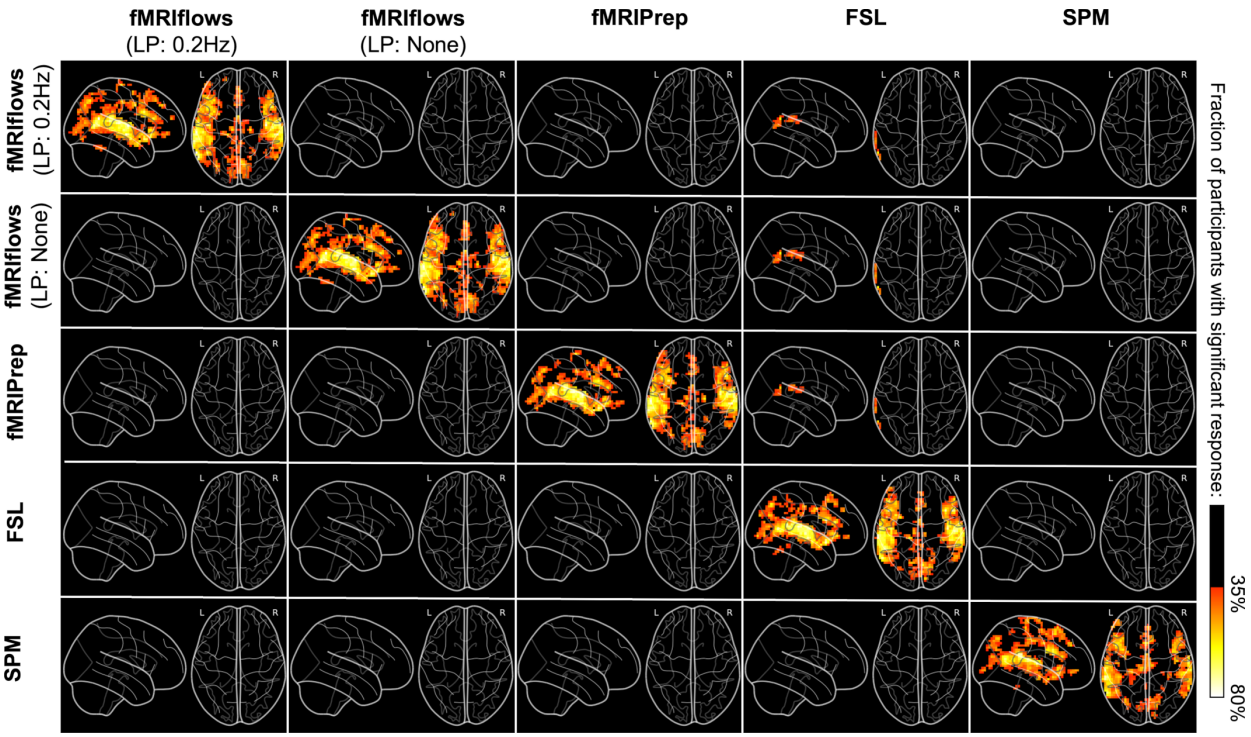
Group analysis using random permutations and cluster size control: The group analysis approach according to Stelzer et al. (2013) includes a permutation step where subject-specific classifications are rerun up to 100 times with randomized labels. In a later step, those randomized control accuracy maps are then combined up to 100’000 times into “false” group averages and help to estimate a voxel-wise null distribution and expected cluster size of the classification accuracy maps, given the dataset and specified classes. The subject specific searchlight accuracy maps are averaged over all subjects to

obtain a group classification accuracy map, which then is tested against the estimated null distribution. The user can additionally specify the strategy for the multiple comparison correction.

Topological thresholding: This step is identical to the one used in the 2nd-level univariate analysis.

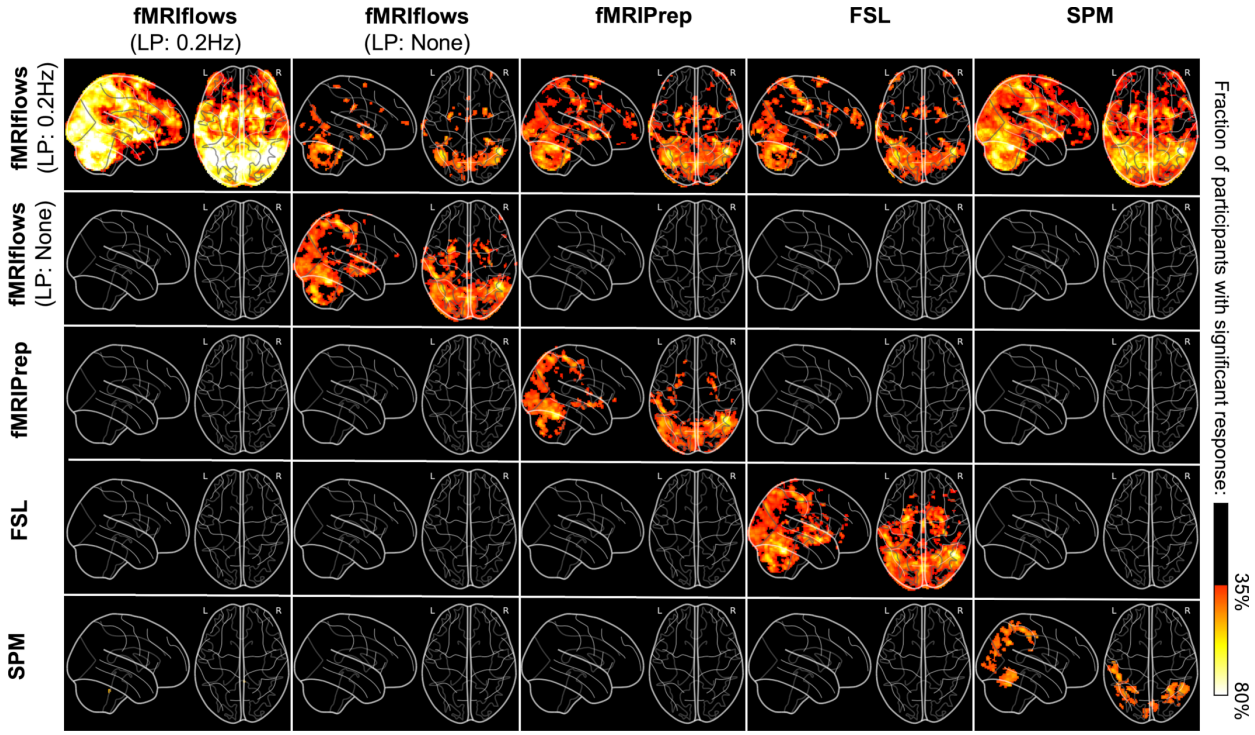
Supplementary Note 7: Complete 1st-level activation count maps results comparison between toolboxes, separated by dataset.

Dataset TR2000: The comparison between the binarized 1st-level activation count maps computed on the TR2000 dataset shows no clear differences between the toolboxes. The images on the diagonal all seem comparable. The diagonal offset images only show significantly increased activation counts for maps generated with fMRIflows and fMRIPrep, over activation count maps generated with FSL.



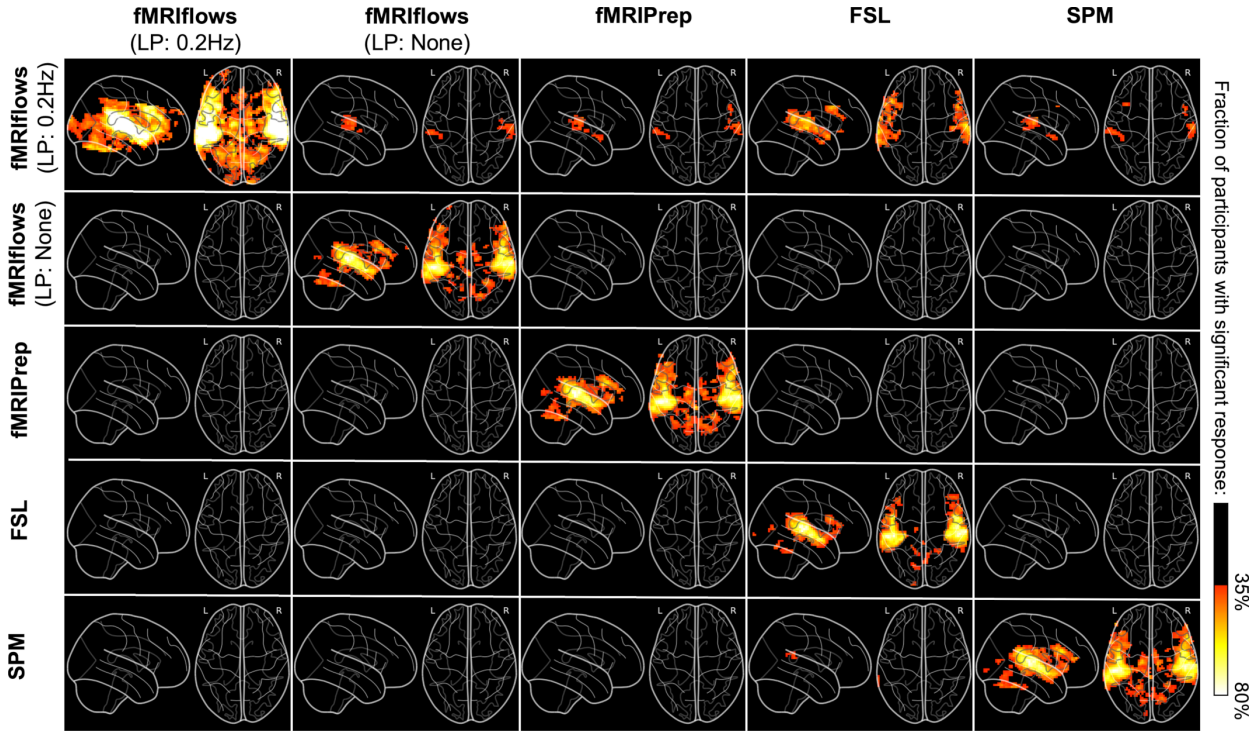
Supplementary Figure 1. Investigation of differences between binarized 1st-level activation count maps, thresholded at *p*<0.001, after multiple functional preprocessing approaches analyzing dataset TR2000. Preprocessing was done with fMRIflows (with and without a temporal low-pass filter at 0.2 Hz), fMRIPrep, FSL and SPM (from top to bottom). The diagonal images represent the original activation count map of the toolbox. The diagonal offset images represent the difference between the horizontal and vertical toolbox. Activation count maps were normalized to the ICBM 2009c brain template. Color code represents the fraction of participants that show significant activation above a p-value threshold at 0.001 and corrected for false positive rate (FPR).

Dataset TR1000: The comparison between the binarized 1st-level activation count maps computed on the TR1000 dataset shows clear differences between fMRIflows with a low-pass filter at 0.2 Hz and the other approaches. The images on the diagonal seem overall similar. However, fMRIflows with a low-pass filter at 0.2 Hz shows a clearly increased (and SPM shows a clearly decreased) activation count map. The significantly increased activation count map values between fMRIflows with a low-pass filter at 0.2 Hz and the other approaches, seem to be centered around locations with already increased overlap, indicating that the low-pass filtering improves the overall statistical sensitivity.



Supplementary Figure 2: Investigation of differences between binarized 1st-level activation count maps, thresholded at *p*<0.001, after multiple functional preprocessing approaches analyzing dataset TR1000. Preprocessing was done with fMRIflows (with and without a temporal low-pass filter at 0.2 Hz), fMRIPrep, FSL and SPM (from top to bottom). The diagonal images represent the original activation count map of the toolbox. The diagonal offset images represent the difference between the horizontal and vertical toolbox. Activation count maps were normalized to the ICBM 2009c brain template. Color code represents the fraction of participants that show significant activation above a p-value threshold at 0.001 and corrected for false positive rate (FPR).

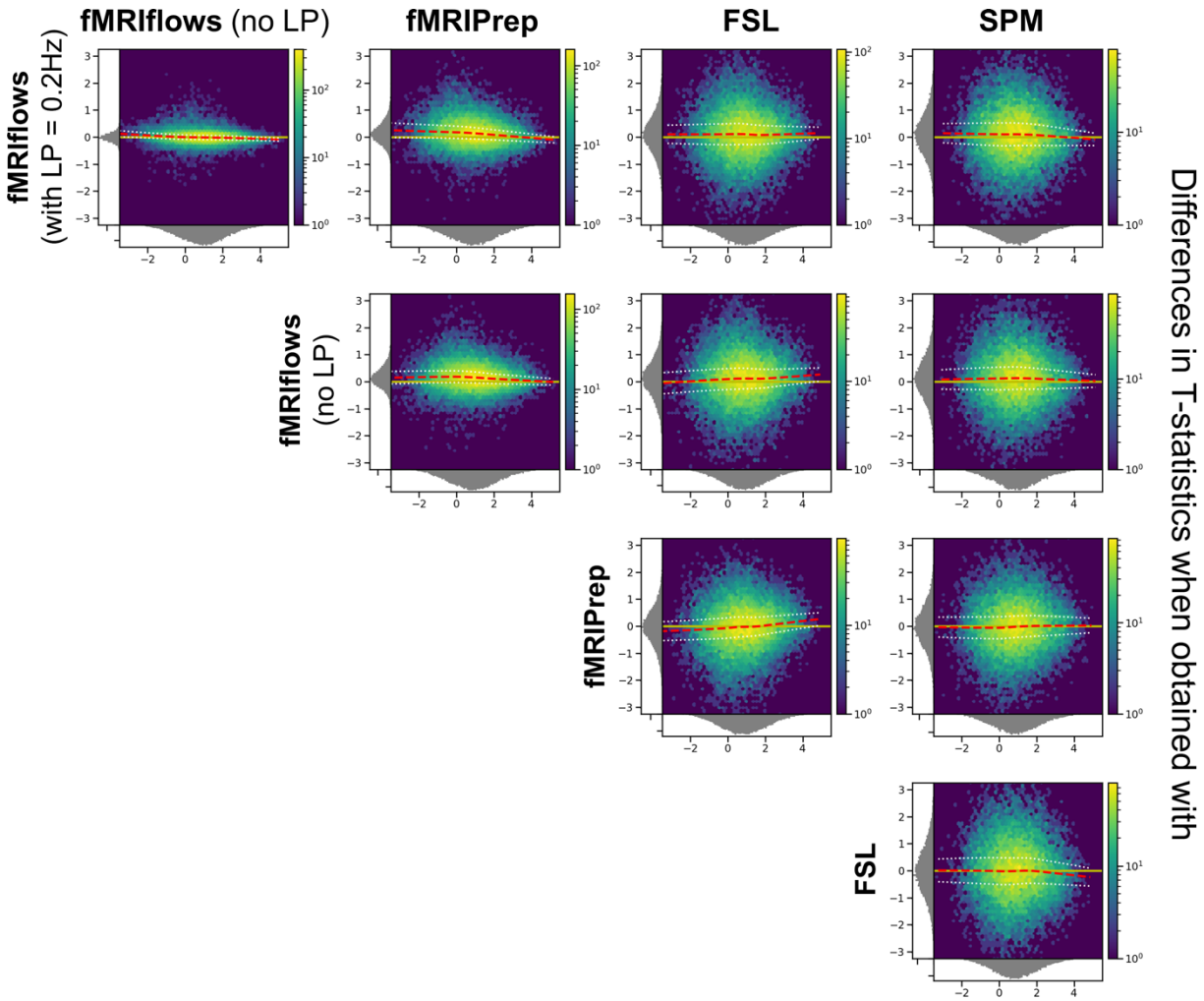
Dataset TR600: The comparison between the binarized 1st-level activation count maps computed on the TR600 dataset show clear differences between fMRIflows with a low-pass filter at 0.2 Hz and the other approaches. The images on the diagonal seem overall comparable. However, fMRIflows with a low-pass filter at 0.2 Hz shows a clearly increased activation count map. The significantly increased activation count map values between fMRIflows with a low-pass filter at 0.2 Hz and the other approaches, seem to be centered around locations with already increased overlap, indicating that the low-pass filtering improves the overall statistical sensitivity.



Supplementary Figure 3: Investigation of differences between binarized 1st-level activation count maps, thresholded at *p*<0.001, after multiple functional preprocessing approaches analyzing dataset TR600. Preprocessing was done with fMRIflows (with and without a temporal low-pass filter at 0.2 Hz), fMRIPrep, FSL and SPM (from top to bottom). The diagonal images represent the original activation count map of the toolbox. The diagonal offset images represent the difference between the horizontal and vertical toolbox. Activation count maps were normalized to the ICBM 2009c brain template. Color code represents the fraction of participants that show significant activation above a p-value threshold at 0.001 and corrected for false positive rate (FPR).

Supplementary Note 8: Complete group-level T-statistic maps comparison between toolboxes, separated by dataset.

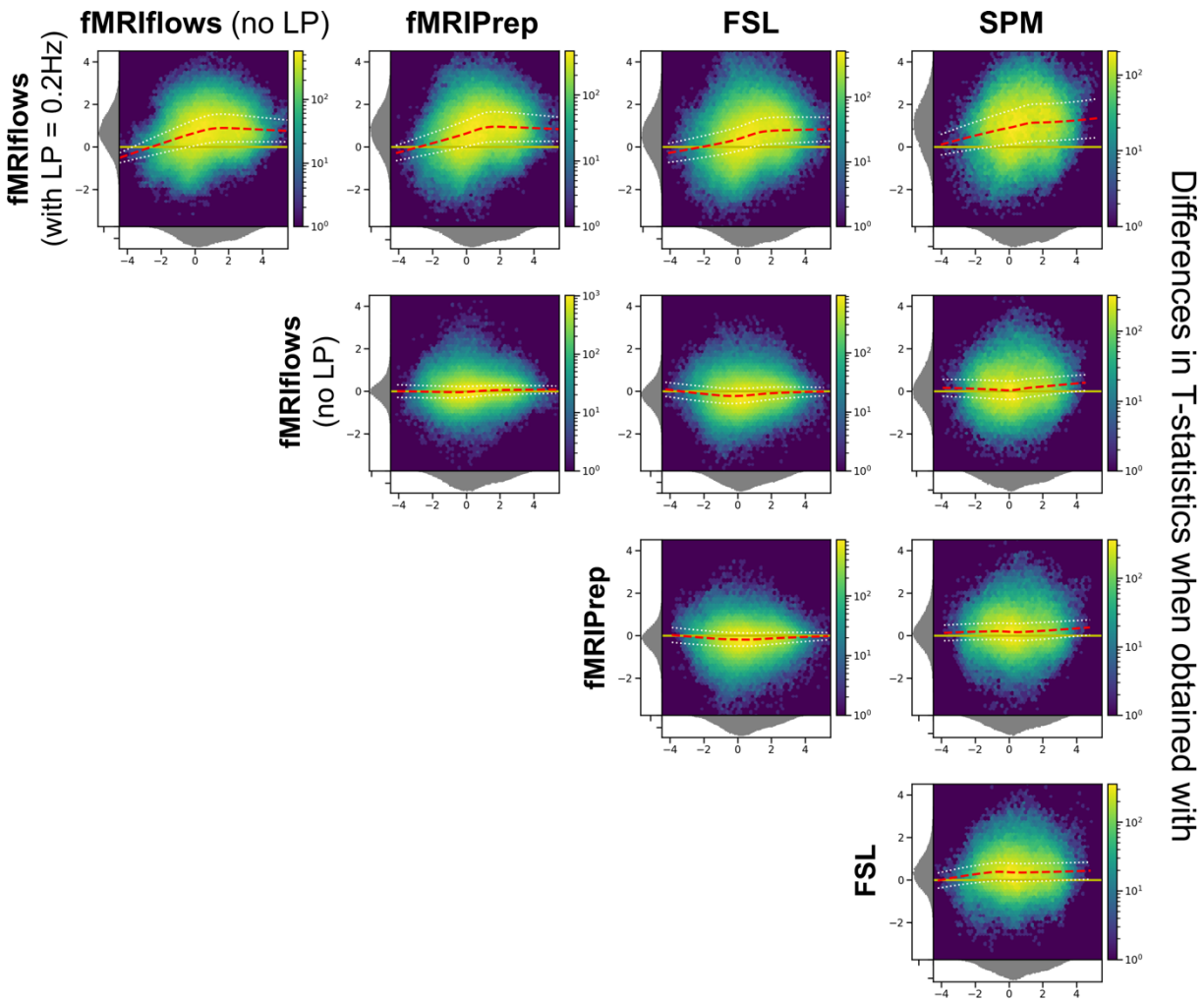
Dataset TR2000: The investigation of group-level T-statistic maps differences due to toolbox specific preprocessing pipelines on the TR2000 dataset does not show any clear differences between the results. The Bland-Altman 2D histograms show no particular trend in the x-direction and are centered around the horizontal zero line.



Supplementary Figure 4: Bland-Altman 2D histograms of datasets TR2000, comparing unthresholded group-level T-statistic maps between multiple processing approaches. Density plots show the relationship between average T-statistic value (horizontal) and difference of T-statistic values (vertical) at corresponding voxels for different pairwise combinations of toolboxes. The difference in T-statistics was computed in contrast to a preprocessing with (from top to bottom) fMRIflows with and without a low-pass filter at 0.2 Hz, fMRIPrep and FSL in respect to a preprocessing with (from left to right) fMRIflows without a low-pass filter at 0.2 Hz, fMRIPrep, FSL and SPM. Distribution plots next to x- and y- axis depict occurrence of a given value in this domain. Color code within figure indicates number

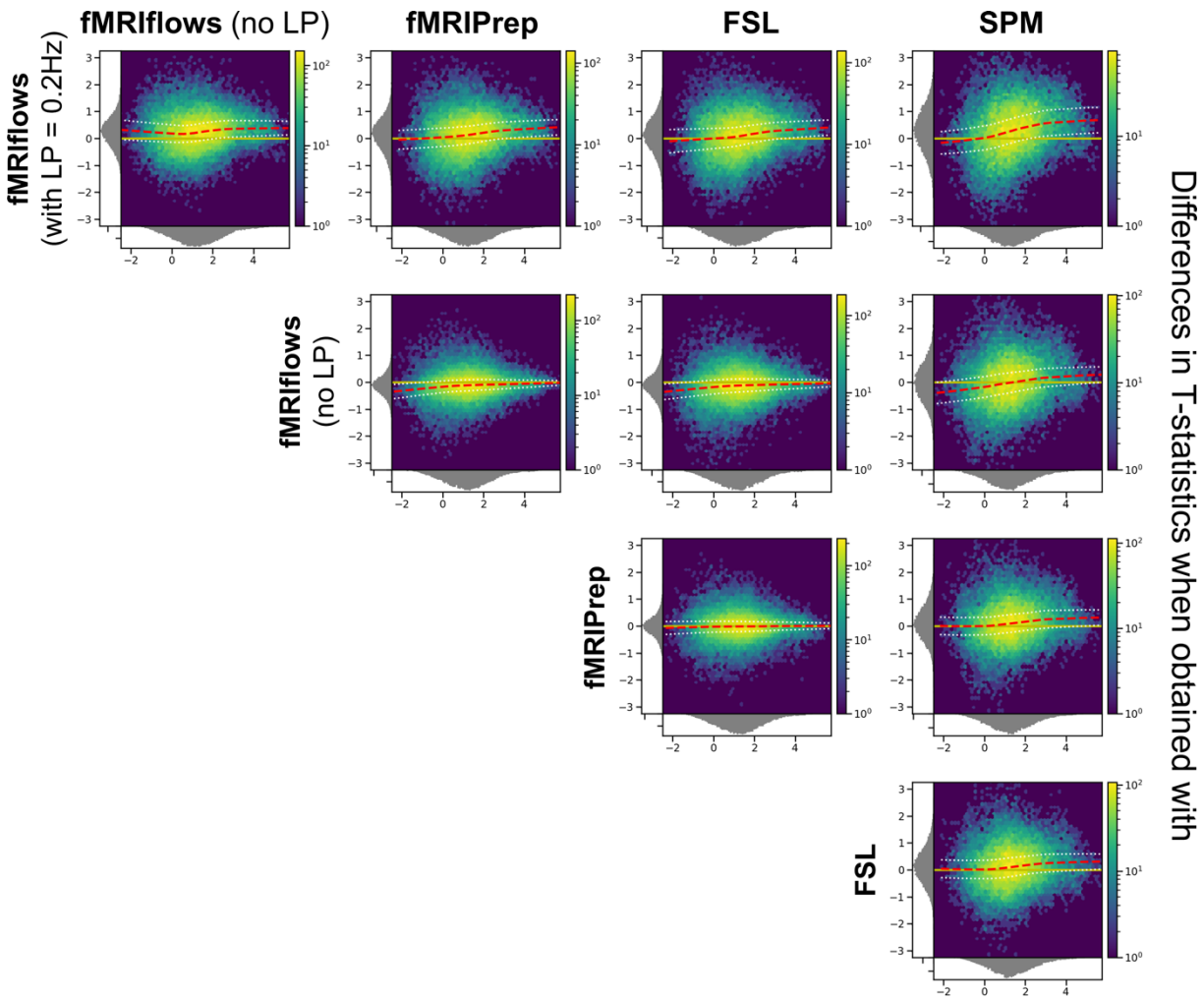
of voxels at this given overlap, from a few (blue) to many (yellow). Yellow horizontal line at zero indicates no value differences between corresponding voxels. Red dashed line depicts horizontal density average.

Dataset TR1000: The investigation of group-level T-statistic maps differences due to toolbox specific preprocessing pipelines on the TR1000 dataset shows clear differences between the results obtained with fMRIflows with a low-pass filter at 0.2 Hz and all the other approaches. The Bland-Altman 2D histograms show clearly increased t-statistic value differences in the first column of the following figure, especially for higher t-values. No clear differences can be seen between the analysis approach fMRIflows without a low-pass filter, fMRIPrep, FSL and SPM.



Supplementary Figure 5: Bland-Altman 2D histograms of datasets TR1000, comparing unthresholded group-level T-statistic maps between multiple processing approaches. Density plots show the relationship between average T-statistic value (horizontal) and difference of T-statistic values (vertical) at corresponding voxels for different pairwise combinations of toolboxes. The difference in T-statistics was computed in contrast to a preprocessing with (from top to bottom) fMRIflows with and without a low-pass filter at 0.2 Hz, fMRIPrep and FSL in respect to a preprocessing with (from left to right) fMRIflows without a low-pass filter at 0.2 Hz, fMRIPrep, FSL and SPM. Distribution plots next to x- and y- axis depict occurrence of a given value in this domain. Color code within figure indicates number of voxels at this given overlap, from a few (blue) to many (yellow). Yellow horizontal line at zero indicates no value differences between corresponding voxels. Red dashed line depicts horizontal density average.

Dataset TR600: The investigation of group-level T-statistic maps differences due to toolbox specific preprocessing pipelines on the TR600 dataset shows clear differences between the results obtained with fMRIflows with a low-pass filter at 0.2 Hz and all the other approaches. The Bland-Altman 2D histograms show increased t-statistic value differences in the first column of the following figure, especially for high t-values. No clear differences can be seen between the analysis approach fMRIflows without a low-pass filter, fMRIPrep, FSL and SPM.



Supplementary Figure 6: Bland-Altman 2D histograms of datasets TR600, comparing unthresholded group-level T-statistic maps between multiple processing approaches. Density plots show the relationship between average T-statistic value (horizontal) and difference of T-statistic values (vertical) at corresponding voxels for different pairwise combinations of toolboxes. The difference in T-statistics was computed in contrast to a preprocessing with (from top to bottom) fMRIflows with and without a low-pass filter at 0.2 Hz, fMRIPrep and FSL in respect to a preprocessing with (from left to right) fMRIflows without a low-pass filter at 0.2 Hz, fMRIPrep, FSL and SPM. Distribution plots next to x- and y- axis depict occurrence of a given value in this domain. Color code within figure indicates number of voxels at this given overlap, from a few (blue) to many (yellow). Yellow horizontal line at zero indicates no value differences between corresponding voxels. Red dashed line depicts horizontal density average.

References

Abraham, A., Pedregosa, F., Eickenberg, M., Gervais, P., Mueller, A., Kossaifi, J., … Varoquaux, G. (2014). Machine learning for neuroimaging with scikit-learn. *Frontiers in Neuroinformatics*, *8*(February), 14. https://doi.org/10.3389/fninf.2014.00014

Ashburner, J. (2009). Preparing fMRI data for statistical analysis. In *fMRI techniques and protocols* (pp. 151–178). Springer.

Avants, B. B., Tustison, N. J., Song, G., Cook, P. A., Klein, A., & Gee, J. C. (2011). A reproducible evaluation of ANTs similarity metric performance in brain image registration. *NeuroImage*, *54*(3), 2033–2044. https://doi.org/10.1016/j.neuroimage.2010.09.025

Behzadi, Y., Restom, K., Liau, J., & Liu, T. T. (2007). A component based noise correction method (CompCor) for BOLD and perfusion based fMRI. *NeuroImage*, *37*(1), 90–101. https://doi.org/10.1016/j.neuroimage.2007.04.042

Botvinik-Nezer, R., Holzmeister, F., Camerer, C. F., Dreber, A., Huber, J., Johannesson, M., … Schonberg, T. (2020). Variability in the analysis of a single neuroimaging dataset by many teams. *Nature*, *582*(7810), 84–88. https://doi.org/10.1038/s41586-020-2314-9

Botvinik-Nezer, R., Iwanir, R., Holzmeister, F., Huber, J., Johannesson, M., Kirchler, M., … Schonberg, T. (2019). fMRI data of mixed gambles from the Neuroimaging Analysis Replication and Prediction Study. *Scientific Data*, *6*(1), 106. https://doi.org/10.1038/s41597-019-0113-7

Bowring, A., Maumet, C., & Nichols, T. (2018). Exploring the impact of analysis software on task fMRI results. *BioRxiv*, 285585.

Brett, M., Hanke, M., Markiewicz, C., Côté, M.-A., McCarthy, P., Ghosh, S., … Basile. (2018). *nibabel: Access a cacophony of neuro-imaging file formats, version 2.3.0*. https://doi.org/10.5281/zenodo.1287921

Caballero-Gaudes, C., & Reynolds, R. C. (2016). Methods for cleaning the BOLD fMRI signal. *NeuroImage*, (December), 0–1. https://doi.org/10.1016/j.neuroimage.2016.12.018

Caballero-Gaudes, C., & Reynolds, R. C. (2017). Methods for cleaning the BOLD fMRI signal. *NeuroImage*, *154*(December 2016), 128–149. https://doi.org/10.1016/j.neuroimage.2016.12.018

Carp, J. (2012). The secret lives of experiments: methods reporting in the fMRI literature. *NeuroImage*, *63*(1), 289–300. https://doi.org/10.1016/j.neuroimage.2012.07.004

Cox, R. W., & Hyde, J. S. (1997). Software tools for analysis and visualization of fMRI data. *NMR in Biomedicine*, *10*(4–5), 171–178. https://doi.org/10.1002/(SICI)1099-1492(199706/08)10:4/5<171::AID-NBM453>3.0.CO;2-L

Esteban, O., Birman, D., Schaer, M., Koyejo, O. O., Poldrack, R. A., & Gorgolewski, K. J. (2017). MRIQC: Advancing the automatic prediction of image quality in MRI from unseen sites. *PloS One*, *12*(9), e0184661. https://doi.org/10.1371/journal.pone.0184661

Esteban, O., Markiewicz, C. J., Blair, R. W., Moodie, C. A., Isik, A. I., Erramuzpe, A., … Gorgolewski,

K. J. (2019). fMRIPrep: a robust preprocessing pipeline for functional MRI. *Nature Methods*, *16*(1), 111–116. https://doi.org/10.1038/s41592-018-0235-4

Feinberg, D. A., Moeller, S., Smith, S. M., Auerbach, E., Ramanna, S., Gunther, M., … Yacoub, E. (2010). Multiplexed echo planar imaging for sub-second whole brain FMRI and fast diffusion imaging. *PloS One*, *5*(12), e15710. https://doi.org/10.1371/journal.pone.0015710

Feinberg, D. A., & Setsompop, K. (2013). Ultra-fast MRI of the human brain with simultaneous multi-slice imaging. *Journal of Magnetic Resonance (San Diego, Calif. : 1997)*, *229*, 90–100. https://doi.org/10.1016/j.jmr.2013.02.002

Fischl, B. (2012). FreeSurfer. *NeuroImage*, *62*(2), 774–781. https://doi.org/10.1016/j.neuroimage.2012.01.021

Fonov, V., Evans, A. C., Botteron, K., Almli, C. R., McKinstry, R. C., & Collins, D. L. (2011). Unbiased average age-appropriate atlases for pediatric studies. *NeuroImage*, *54*(1), 313–327. https://doi.org/10.1016/j.neuroimage.2010.07.033

Friston, K. J., Williams, S., Howard, R., Frackowiak, R. S., & Turner, R. (1996). Movement-related effects in fMRI time-series. *Magnetic Resonance in Medicine*, *35*(3), 346–355. https://doi.org/10.1002/mrm.1910350312

Friston, K., Penny, W., Ashburner, J., Kiebel, S., & Nichols, T. (2006). *Statistical Parametric Mapping: The Analysis of Functional Brain Images* (Vol. 8). https://doi.org/10.1016/B978-012372560-8/50002-4

Gorgolewski, K., Burns, C. D., Madison, C., Clark, D., Halchenko, Y. O., Waskom, M. L., & Ghosh, S. S. (2011). Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python. *Frontiers in Neuroinformatics*, *5*(August), 13. https://doi.org/10.3389/fninf.2011.00013

Gorgolewski, K., Esteban, O., Schaefer, G., Wandell, B., & Poldrack, R. (2017). OpenNeuro - a free online platform for sharing and analysis of neuroimaging data. *Organization for Human Brain Mapping. Vancouver, Canada*, 1677. https://doi.org/10.1038/sdata.2016.44.3.

Gorgolewski, K. J., Auer, T., Calhoun, V. D., Craddock, R. C., Das, S., Duff, E. P., … Poldrack, R. A. (2016). The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. *Scientific Data*, *3*, 160044. https://doi.org/10.1038/sdata.2016.44

Gorgolewski, K. J., Varoquaux, G., Rivera, G., Schwarz, Y., Ghosh, S. S., Maumet, C., … Margulies, D. S. (2015). NeuroVault.org: a web-based repository for collecting and sharing unthresholded statistical maps of the human brain. *Frontiers in Neuroinformatics*, *9*(April), 8. https://doi.org/10.3389/fninf.2015.00008

Griswold, M. A., Jakob, P. M., Heidemann, R. M., Nittka, M., Jellus, V., Wang, J., … Haase, A. (2002). Generalized autocalibrating partially parallel acquisitions (GRAPPA). *Magnetic Resonance in Medicine*, *47*(6), 1202–1210. https://doi.org/10.1002/mrm.10171

Hanke, M., Halchenko, Y. O., Sederberg, P. B., Hanson, S. J., Haxby, J. V., & Pollmann, S. (2009). PyMVPA: A python toolbox for multivariate pattern analysis of fMRI data. *Neuroinformatics*, *7*(1),

37–53. https://doi.org/10.1007/s12021-008-9041-y

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

Jenkinson, M., Beckmann, C. F., Behrens, T. E. J., Woolrich, M. W., & Smith, S. M. (2012). Fsl. *Neuroimage*, *62*(2), 782–790.

Jones, E., Oliphant, T., Peterson, P., & others. (2001). *{SciPy}: Open source scientific tools for {Python}*. Retrieved from http://www.scipy.org/

Kluyver, T., Ragan-kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., … Willing, C. (2016). Jupyter Notebooks—a publishing format for reproducible computational workflows. *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 87–90. https://doi.org/10.3233/978-1-61499-649-1-87

Lanczos, C. (1964). Evaluation of Noisy Data. *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis*, *1*(1), 76–85. https://doi.org/10.1137/0701007

Lindquist, M. A., Geuter, S., Wager, T. D., & Caffo, B. S. (2019). Modular preprocessing pipelines can reintroduce artifacts into fMRI data. *Human Brain Mapping*, (January), 407676. https://doi.org/10.1002/hbm.24528

McKinney, W., & others. (2010). Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, *445*, 51–56. https://doi.org/10.3389/fninf.2014.00014

Moeller, S., Yacoub, E., Olman, C. A., Auerbach, E., Strupp, J., Harel, N., & Uğurbil, K. (2010). Multiband multislice GE-EPI at 7 tesla, with 16-fold acceleration using partial parallel imaging with application to high spatial and temporal whole-brain fMRI. *Magnetic Resonance in Medicine*, *63*(5), 1144–1153. https://doi.org/10.1002/mrm.22361

Notter, M., Gale, D., Herholz, P., Markello, R., Notter-Bielser, M.-L., & Whitaker, K. (2019). AtlasReader: A Python package to generate coordinate tables, region labels, and informative figures from statistical MRI images. *Journal of Open Source Software*, *4*(34), 1257. https://doi.org/10.21105/joss.01257

Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science & Engineering*, *9*(3). https://doi.org/10.1109/MCSE.2007.58

Penny, W. D., Friston, K. J., Ashburner, J. T., Kiebel, S. J., & Nichols, T. E. (2011). *Statistical parametric mapping: the analysis of functional brain images*. Elsevier.

Power, J. D. (2017). A simple but useful way to assess fMRI scan qualities. *NeuroImage*, *154*(August 2016), 150–158. https://doi.org/10.1016/j.neuroimage.2016.08.009

Power, J. D., Barnes, K. A., Snyder, A. Z., Schlaggar, B. L., & Petersen, S. E. (2012). Spurious but systematic correlations in functional connectivity MRI networks arise from subject motion. *NeuroImage*, *59*(3), 2142–2154. https://doi.org/10.1016/j.neuroimage.2011.10.018

Power, J. D., Plitt, M., Kundu, P., Bandettini, P. A., & Martin, A. (2017). Temporal interpolation alters motion in fMRI scans: Magnitudes and consequences for artifact detection. *PloS One*, *12*(9), e0182939. https://doi.org/10.1371/journal.pone.0182939

Power, J. D., Plitt, M., Laumann, T. O., & Martin, A. (2017). Sources and implications of whole-brain fMRI signals in humans. *NeuroImage*, *146*(May 2016), 609–625. https://doi.org/10.1016/j.neuroimage.2016.09.038

Sengupta, A., Pollmann, S., & Hanke, M. (2018). Spatial band-pass filtering aids decoding musical genres from auditory cortex 7T fMRI. *F1000Research*, *7*(0), 142. https://doi.org/10.12688/f1000research.13689.2

Sladky, R., Friston, K. J., Tröstl, J., Cunnington, R., Moser, E., & Windischberger, C. (2011). Slice-timing effects and their correction in functional MRI. *NeuroImage*, *58*(2), 588–594. https://doi.org/10.1016/j.neuroimage.2011.06.078

Smith, S. M., Beckmann, C. F., Andersson, J., Auerbach, E. J., Bijsterbosch, J., Douaud, G., … WU-Minn HCP Consortium. (2013). Resting-state fMRI in the Human Connectome Project. *NeuroImage*, *80*, 144–168. https://doi.org/10.1016/j.neuroimage.2013.05.039

Smith, S. M., Jenkinson, M., Woolrich, M. W., Beckmann, C. F., Behrens, T. E. J., Johansen-Berg, H., … Matthews, P. M. (2004). Advances in functional and structural MR image analysis and implementation as FSL. *NeuroImage*, *23 Suppl 1*(SUPPL. 1), S208-19. https://doi.org/10.1016/j.neuroimage.2004.07.051

Stelzer, J., Chen, Y., & Turner, R. (2013). Statistical inference and multiple testing correction in classification-based multi-voxel pattern analysis (MVPA): random permutations and cluster size control. *NeuroImage*, *65*, 69–82. https://doi.org/10.1016/j.neuroimage.2012.09.063

Stephen Butterworth. (1930). On the Theory of Filter Amplifiers. *Experimental Wireless and the Wireless Engineer*, Vol. 7, pp. 536–541.

Strother, S. C. (2006). Evaluating fMRI preprocessing pipelines. *IEEE Engineering in Medicine and Biology Magazine : The Quarterly Magazine of the Engineering in Medicine & Biology Society*, *25*(2), 27–41. https://doi.org/10.1109/MEMB.2006.1607667

Viessmann, O., Möller, H. E., & Jezzard, P. (2018). Dual regression physiological modeling of resting-state EPI power spectra: Effects of healthy aging. *NeuroImage*, *187*(December 2017), 68–76. https://doi.org/10.1016/j.neuroimage.2018.01.011

Yarkoni, T., Markiewicz, C., de la Vega, A., Gorgolewski, K., Salo, T., Halchenko, Y., … Blair, R. (2019). PyBIDS: Python tools for BIDS datasets. *Journal of Open Source Software*, *4*(40), 1294. https://doi.org/10.21105/joss.01294